# Optimizing One-Way Car Sharing Systems

by

**Hedayat Zarkoob**

B.Sc., Isfahan University of Technology, 2013

Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences


**SIMON FRASER UNIVERSITY**
**Summer 2015**

# Approval

| | |
|---|---|
| **Name:** | Hedayat Zarkoob |
| **Degree:** | Master of Science (Computing Science) |
| **Title:** | *Optimizing One-Way Car Sharing Systems* |
| **Examining Committee:** | **Dr. Binay Bhattacharya** (chair)<br>Professor |

**Dr. Andrei Bulatov**
Senior Supervisor
Professor _____

**Dr. Ramesh Krishnamurti**
Co-Supervisor
Professor _____

**Dr. Tim Huh**
Supervisor
Professor
Sauder School of Business
University of British Columbia _____

**Dr. Qianping Gu**
Internal Examiner
Professor _____

**Date Defended:** 4 August 2015

# Abstract

In one-way car sharing systems picking up and returning the rental cars can be done at different stations. In these systems, since the customer demand is asymmetric, operators need to hire some staff to manually relocate the cars between stations to keep the system balanced. In this thesis, we address the problem of designing optimal relocation strategies for the one-way car sharing operators both in deterministic and stochastic settings. For the deterministic case, we give a minimum cost network flow formulation. To model the stochastic one, we use stochastic dynamic programming. Our theoretical results show that the exact optimal policy to relocate the cars in a two-station case is a threshold type policy. Based on this result, a heuristic algorithm is proposed to handle the m-station case. Our heuristic significantly decreases the computational complexity of the problem.

**Keywords:** one-way car sharing systems; mathematical programming; threshold type optimal policy

# Dedication

To my family!

# Acknowledgements

I would like to express the deepest appreciation and thanks to my supervisor, Dr. Andrei Bulatov, for all his kindness, compassion, and support through the process of my master studies. Anytime I needed help, he made his time available for me and helped me with valuable advice and guidance whilst allowing me the room to work in my own way.

Furthermore, I would like to express my appreciation to my external supervisors, Dr. Tim Huh and Dr. Steven Shechter, for all their helpful remarks and advice during my thesis process. I also want to thank my committee members, Dr. Qianping Gu and Dr. Ramesh Krishnamurti, for their useful comments on my thesis. In addition, a thank you to Dr. Binay Bhattacharya for serving as my thesis examination committee chair.

I also want to use this opportunity to express my gratitude to my brother, Hadi Zarkoob, who supported me a lot in all steps of my research. Moreover, I owe a sincere thanks to Erfan Sadeqi Azer for his useful supports during this research.

A warm and special appreciation to Mohammad Akbari for his kind and remarkable helps on writing this thesis. I also want to acknowledge my friends, Abdollah Saffari, Ehsan Haghshenas, Sajjad Gholami, and Mehran Khodabandeh.

Let me also thank my lab-mates, Kamyar Khodamoradi and Ehsan Iranmanesh, for the stimulating discussions, for the times that we worked together, and for all the fun we have had in the last two years.

Last but not the least, I would like to thank my family. Words cannot express how grateful I am to them for all of the sacrifices that they have made on my behalf.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Motivation

Car sharing or car rental is a type of transportation system offering a shared vehicle service to the customers. If someone needs a vehicle temporarily, they can rent one from a car sharing system and return it when the rental period is over. The origin of car sharing systems goes back to the 1940s in Europe when there were only very limited services offered by car sharing operators [29]. However, the popularity of these systems has recently grown resulting in a number of different car sharing systems all around the world [27, 28].

Car sharing systems can generally be divided into two main groups: (*i*) round-trip or traditional and (*ii*) one-way car sharing systems [17]. Traditional car sharing systems often offer different car sizes with affordable prices to their customers. In these systems, picking up and returning the rental cars should be done at the same station. The customers in traditional car sharing system are usually those who need a car for a few days or want to go shopping in several places during the rental period. As an example, a study done in Toronto [12] on the behaviour of customers in a traditional car sharing system shows that most of the customers rent cars in order to do their grocery or other household shopping materials.

There are three important weaknesses associated with traditional car sharing systems:

- They usually have a limited number of stations in each city.
- The availability of service is highly subject to making reservations well ahead of time.
- They significantly charge their customers, if they do not return the cars to the initial rental station.

The second type of car sharing systems is one-way car sharing system. When compared to traditional systems, these newer ones have many stations in their region and allow their customers to return the cars to any arbitrary station. Moreover, customers are ideally not

required to do reservations in advance in these systems any more. Although the rental rates in one-way car sharing systems are usually higher than in the traditional ones, the flexibility of service of these systems make them mostly appropriate for frequent but rather short distance trips. As a result, these systems can be very popular in small regions such as city centres.

The simultaneous presence of these two types of car sharing systems in the market has increased the demand of renting cars rather than owning them because of the following:

- The quality and variety of rental services are increasing every day. For instance, in some cities (e.g., Vancouver), one can often find an available car to rent either for a short or a long trip from a nearby station.

- The costs of ownig a car such as oil price, monthly insurance, and car maintenance are high. Consequently, having a reliable car rental system with reasonable prices encourages people to use alternatives such as public transportation to do their daily activities and rent a vehicle for their specific purposes [32].

Having more customers results in more expenses for car sharing systems. Car maintenance and depreciation costs are two examples of these growing costs. In addition to the extra expenses that all rental systems have in common, the freedom that one-way car sharing systems give to their customers causes many operational issues for the system operators. One of the most important challenges that the system operators face is to ensure the availability of cars at each station to serve their upcoming customers. This difficulty is because (i) the customers do not make a reservation beforehand and (ii) the demand is usually asymmetric in these systems. As a result, if the operators leave their systems without any supervision, the stations with higher demand will become empty after some time, while some other stations will be full of unnecessary cars.

To overcome the availability issue, various mechanisms have been proposed. One is to use incentive mechanisms such as price flexibility to increase the number of trips from low demand stations to high demand ones [2, 13, 20, 33]. Although incentive mechanisms seem to help the operators balance the system, they do not guarantee reaching a convenient car distribution in the system in general. Another way of keeping the system balanced is to grant a central control unit the authority of accepting or rejecting the customer requests [17]. This unit makes rental decisions in order to help the system keep itself balanced. However, this mechanism may not perform well when the number of customer requests is insufficient. In addition, some customers may find their requests rejected even though there might be available cars at some station.

Another mechanism that one-way car sharing operators use is to hire a number of staff for manually relocating some of the cars between the stations. When a car sharing system is operating, some of the useless cars are sometimes left in a station, while the operator has

run out of them at some other stations. When this happens, the operator manually moves some of the cars to the empty station so that they can satisfy more requests and gain more profit. The staff based mechanism can be an effective way to keep the system balanced. However, it is very important to do the relocations in an optimal way. This is because:

1. Doing these relocations is very costly for the system operators. Note that beside the regular costs of moving cars, such as oil price and depreciation costs, system operators also have to pay the staff to perform these relocations.

2. If cars are moved between the stations in a wrong way, it is possible that the operator misses some profitable requests because of the relocations.

In this thesis, we study the optimal ways of doing relocations. Our goal is to first come up with a good mathematical model of the one-way car sharing systems. Then, based on our model, the best choices for relocations are found and suggested to the operators in one-way car sharing systems.

## 1.2 Problem Description and Contributions

In this thesis, we address the problem of designing optimal relocation strategies for the one-way car sharing operators. In our analysis, a one-way car sharing system has a finite operating time and region. The operating region consists of a number of stations with a number of cars. The operating time of the system consists of several periods in which customers request for cars. In addition, the system has some staff who can relocate cars between these stations. In a one-way car sharing system, the system operator does two main jobs at each period: (i) rentals and (ii) relocations. While renting the cars results in revenue for the system, doing relocations incurs a cost to it. The goal of the system operator is to get the optimal pay-off from the system during the whole operating time.

We consider two types of customer behaviour in the car sharing systems: (i) deterministic and (ii) stochastic. In the deterministic case, the operator knows the total demand of customers for the whole operating time, while in the stochastic case, the operator knows only some probabilistic information about the system's upcoming demand such as the distribution function. In this thesis, we look at the one-way car sharing relocation problem both in the deterministic and stochastic settings. In order to do that, we first model each setting, and then, study (i) how to rent cars and (ii) how relocate them between the stations to obtain the minimum cost.

Throughout this thesis, three main assumptions are made in the models:

1. Operating periods consist of two stages: (i) days during which customers travel and (ii) nights during which the staff do relocations.

2. The total number of customer requests in each day is no more than the total number of cars in the system.

3. Each car in the system can only be used once per day. In other words, each trip in our car sharing system takes exactly one day.

While one would think that these assumptions make the model far from reality, they reasonably match many of the real world scenarios. Consequently, their models can give us a very good insight on the optimal way of performing relocations in a one-way car sharing system [11, 14]. In this thesis, the problem of studying optimal strategies in the one-way car sharing systems under these three assumptions is named day/night one-way car sharing problem.

We analyze the deterministic framework with a new network flow formulation for our problem. In our formulation, we calculate the optimal way of doing the rentals and relocations in the system. This formulation is an oversimplification of the problem. This is because we want to extend our analysis to the stochastic case in which too many details of the real world scenarios cannot be included due to the computational issues.

Assuming that customer demand is known before making the decisions for renting and relocating the cars does not really seem to be realistic in one-way car sharing systems. This is because, as mentioned before, customers are ideally not required to reserve cars beforehand in these systems. Consequently, it is important to study different aspects of one-way car sharing systems in a stochastic framework, even though computational tools may be more limited in this case.

The main contribution of this thesis is in the modeling of the problem when the assumption of knowing the customer demand in advance is relaxed. We use stochastic dynamic programming to model this problem. To the best of our knowledge, we are the first who have tried to solve the problem of optimizing the relocations in one-way car sharing systems using dynamic programming and optimal control theory. In the stochastic analysis, we first calculate the optimal way of doing rentals and relocations, given a distribution for the customer demand over time. Next, we focus on finding the optimal ways for doing the relocations. In order to do that, we assume that there is an oracle in the system, which always accepts customer requests in order to get the best expected result until the end of the operating time from the system. Having this oracle, we do not need to make rentals decisions anymore. As is usually the case in stochastic dynamic programming, finding the optimal solution for all possible states and demand realizations of the system is computationally intractable. In order to deal with this issue, we first prove that the objective functions in our formulation are convex. Following that, a threshold type policy is obtained for the problem . Our theoretical results show that when we have two stations, the exact optimal policy for performing the relocations is a threshold type policy. Based on this result, we propose a heuristic algorithm in order to handle the case with more than two

stations. Our heuristic algorithm significantly decreases the computational complexity of the problem.

## 1.3 Related Work

One-way car sharing systems have been studied in the literature from various perspectives. Based on the assumptions on the customer behaviour, the previous research studies can be divided into two main groups: deterministic and stochastic. Researchers mainly modeled the one-way car sharing system in the deterministic frameworks with mixed integer programming [6, 11, 17, 18]. They consider two main groups of decisions in their models: (i) structural decisions such as the location and size of the stations, fleet size, etc. and (ii) operational decisions such as rentals and relocations.

Correia and Antunes [11] studied the optimal way to select locations for some certain depots and to perform relocations between them in the one-way car sharing system. They assumed relocations do not happen during the day and vehicles need to be available at their original positions for the next day. They considered three schemes for the car sharing organizations. In the first one, system operator had the total control over satisfying customer requests. In the second one, all requests had to be satisfied by the system. In their last scheme, there was no need for satisfying all the requests, but they could be rejected only if there were no vehicles available at the pick-up stations.

Boyaci, Geroliminis and Zografos [6] studied the optimal fleet size, the number, and location of the stations as well as the optimal way of performing relocations simultaneously. They also conducted sensitivity analysis for different parameters to understand the effect of them on the system model.

Although deterministic analysis results in a good insight about how one-way car sharing systems should be controlled in details, a significant weakness in all of them is that they do not consider the uncertainty of demand in the real world scenarios.

The research on the stochastic setting is mainly focused on finding the optimal strategies for performing the relocations. These studies follow two different approaches: (i) simulation-based [8, 22, 24] and (ii) mathematical optimization methods. The simulation-based works usually tried to simulate a certain real world scenario of a one-way car sharing systems in order to find the optimal ways of doing relocations empirically.

Apart from the simulation-based techniques, people have also tried to use mathematical optimization methods to solve the optimal relocations problem for one-way car sharing systems.

Fan et al. [14] modeled the one-way car sharing problem using multi-stage stochastic programming. In their model, they assumed that the relocation costs and rental revenues are given and the travels take exactly one day for both customers and staff. They supposed that at the beginning of each day, the car sharing operator knows the inventory in each station,

the demand for the upcoming day, and the distribution for the future demand. According to these assumptions, the operator decided which customer requests should be satisfied and which relocations need to be performed during the operating time to achieve a higher profit from the system. Note that in their model, they assumed that relocations/rentals happened simultaneously which was not proved to be a profitable assumption.

Nair and Miller-Hooks [21] looked at the problem as a stochastic mixed integer program with joint chance constraints. Their model generated partial plans for relocating cars between stations throughout the day. In their work, they defined a level-of-service constraint which was a threshold for the probability of satisfying a demand. They assumed that the goal of the operator in the system was to serve all of the demand, so he/she had no control over the rentals.

It is also worth mentioning that in the literature, there are similar operational problems studied for bike sharing systems [1, 3, 9, 25, 26, 30]. As in car sharing systems, bike sharing systems have several stations from which customers can pick up the bikes and return them to any station they want. The asymmetry in customer demand of these systems causes similar operational problems for the bike sharing systems.

Shu et al. [30] is an example of one of the works in which bike sharing systems are studied. In their work, they tried to develop a stochastic network flow model for the problem. They focused on finding the optimal number of needed bikes and also the optimal way to distribute them among the stations in order to maximize the number of satisfied requests. They assumed that the costumers come to the stations randomly and bicycles are allocated to them on a first come first serve basis. They also studied the impact of relocation strategies on the number of bikes and docks required in the system. However, they did not consider any kind of cost such as relocation cost. Similar to [14], they also assumed that any travel between two stations took exactly one period of time.

In spite of the fact that there are some similarities between bike sharing and one-way car sharing systems, there is a number of differences between them making it essential to study their problems separately. One of the major differences is that in bike sharing systems, the operators can use trucks to move many of the bikes among the stations at the same time. Consequently, a single truck can perform a number of relocations, while in the case of one-way car sharing systems, only one car at a time can be moved. Furthermore, because of the nature of the bike sharing systems, the total inventory and the length of the trips are very different from the similar parameters in the car sharing systems. As a result, different assumptions are required in order to model the car and bike sharing systems and we cannot use the same strategies for these systems.

## 1.4 Thesis Organization

The main methodologies used in this research are introduced in Chapter 2. In Chapter 3, we go over the deterministic model of the day/night one-way car sharing problem. Chapter 4 is dedicated to the explanation of the stochastic model of the problem and related theoretical results. In order to evaluate the efficiency of our model, the numerical results will be presented in Chapter 5. The thesis will be concluded in Chapter 6 where the future works will also be discussed.

# Chapter 2

# Preliminaries

In this chapter, we discuss the main methodologies used in this thesis. We start with defining the standard mathematical optimization problem and some of its special cases such as the convex optimization and the linear programming problems. Next, we review the concepts of dynamic programming and optimal policy.

## 2.1 Mathematical Optimization Problem

In this section, we define the **standard mathematical optimization problem** [7], and discuss some of its properties.

**Definition 2.1.1** *The standard form of the mathematical optimization problem is the following:*

$$\min f_0(x) \tag{2.1}$$

*subject to:*

$$f_i(x) \leq b_i, \quad i = 1, 2, ..., m \tag{2.2}$$

$$h_i(x) = 0. \quad i = 1, 2, 3, ..., p \tag{2.3}$$

*In this mathematical formulation, the function $f_0(x) : R^n \longrightarrow R$ in (2.1) is called the **objective function**. The functions $f_i(x) : R^n \longrightarrow R$ for $i \in \{1, 2, ..., m\}$ in (2.2) and $h_i(x) : R^n \longrightarrow R$ for $i \in \{1, 2, ..., p\}$ in (2.3) are known as the **constraint functions**. Also, the zeros in (2.3) and the parameter $b_i \in R$ are the **right hand side constraint** of the problem. Note that the relations (2.2) and (2.3) are limiting $R^n$ into a smaller region in which we need to solve the problem. The region specified by (2.2) and (2.3) is named the **feasible region** of the problem. The goal of all mathematical optimization problems is to find the optimal value of the **optimization variables**, i.e. a vector $x$, which is in the feasible region and gives the minimum value of the objective function.*

### 2.1.1 Duality

Let us start this section by the following definition:

**Definition 2.1.2 ([7])** *For every mathematical optimization problem given by* (2.1) *and* (2.2)*, the corresponding* ***Lagrange dual function*** $g(\lambda)$ *is defined as follows:*

$$g(\lambda,\nu) = \inf_{x \in D} \mathbf{L}(x,\lambda,\nu) = \inf_{x \in D} \left( f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \right), \tag{2.4}$$

*where* $D = (\bigcap_{i=0}^{m} dom(f_i)) \cap (\bigcap_{i=0}^{p} dom(h_i))$*. The function* $L : R^n \times R^m \longrightarrow R$ *is the* ***Lagrangian*** *associated with the problem, and* $\lambda_i s$ *and* $\nu_i s$ *are* ***Lagrange multipliers*** *of the dual function.*

**Theorem 2.1.3 ([7])** *For any given* $\lambda$ *and* $\nu$*, the function* $g(\lambda,\nu)$ *is a lower bound for the optimal value of the original optimization problem. In other words, for any* $\lambda \in R^m$ *and* $\nu \in R^p$*:* $g(\lambda,\nu) \leq f_0(x)$*.*

As a result, we can have the following definition:

**Definition 2.1.4 ([7])** *The maximization problem given below, derived based on the standard form given by* (2.1) *and* (2.2)*, is called the* ***Lagrangian dual problem****:*

$$\max g(\lambda,\nu) \tag{2.5}$$

*subject to:*

$$\lambda \geq 0. \tag{2.6}$$

Usually, the original and the Lagrangian dual problems are respectively called as the **primal** and **dual problems**.

**Definition 2.1.5 ([7])** *The difference between the value of the primal and dual problems is called the* ***duality gap****.* ***Strong duality*** *holds for a mathematical optimization problem, if the duality gap is zero.*

### 2.1.2 Convex Optimization Problems

In general, the mathematical optimization problems are NP-hard problems, except for the convex optimization problem [7] which can be solved more efficiently. This is because convex functions have some additional properties helping us find their optimal values expeditiously. In this section, we first give a formal definition of convexity of a set and a function. Then, we introduce the convex optimization problem accordingly.

**Definition 2.1.6 ([7])** *A set $S \subseteq R^n$ is a **convex set**, if for any $x, y \in S$ and $0 \leq \theta \leq 1$, the following relation is satisfied:*

$$\theta x + (1 - \theta)y \in S. \tag{2.7}$$

**Definition 2.1.7 ([7])** *A function $f : R^n \longrightarrow R$ is a **convex function**, if*

1. *its domain, $dom(f)$, is a convex set, and*
2. *for any $x, y \in dom(f)$ and $0 \leq \theta \leq 1$, the following relation is satisfied:*

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y). \tag{2.8}$$

**Proposition 2.1.8 ([7])** *Zero function and linear functions are convex.*

**Proposition 2.1.9 ([7])** *Non-negative weighted sum preserves convexity.*

**Proposition 2.1.10 ([7])** *If $f : R^n \longrightarrow R$ is a convex function, $A \in R^{n \times m}$, and $b \in R^n$ the function $g : R^m \longrightarrow R$ defined as follows is a convex function:*

$$g(x) = f(Ax + b) \tag{2.9}$$

*where $dom(g) = \{x | Ax + b \in dom(f)\}$.*

**Definition 2.1.11 ([7])** *A mathematical optimization problem, is called a **convex optimization problem** if every $f_i(x)$ $\forall i \in 0, 1, ..., m$ is a convex function.*

Convex optimization problems have an important property. To specify this property, we first introduce the following definition of a **local optimum**:

**Definition 2.1.12 ([7])** *A point $x_1$ is local optimum for the mathematical optimization problem, if it is in the feasible region, and is the optimal point in its neighbourhood in the feasible region.*

**Theorem 2.1.13 ([7])** *In convex optimization problems, a local optimal in the feasible region is also the global optimum.*

This property reduces the convex optimization problem to the problem of finding a single local optimal solution.

### 2.1.3 Linear Programming

We present the linear programming problem [23] and some of its useful properties in this section. The linear programming problem is a special case of the convex optimization problem, which is defined as follows:

**Definition 2.1.14** *In a convex optimization problem, if every $f_i(x), i \in \{0, 1, ..., m\}$ is a linear function, the problem is called a linear programming problem.*

In terms of computational complexity, linear programming problems are efficient optimization problems. In fact, there are a couple of algorithms designed to solve them in polynomial time. Any arbitrary linear programming problem can be written in the canonical form as shown in below:

$$\min_x \ c^T x \tag{2.10}$$

subject to:

$$Ax = b, \tag{2.11}$$
$$x \geq 0. \tag{2.12}$$

**Dual of a Linear Program**

We describe the dual of a linear programming problem, and state some of its useful properties.

**Theorem 2.1.15 ([7])** *Given a linear programming problem in the canonical form, its Lagrangian dual problem is the following:*

$$\max_v \ b^T v \tag{2.13}$$

*subject to:*

$$A^T v \ \leq \ c^T, \tag{2.14}$$
$$v \ \geq \ 0, \tag{2.15}$$

*where $v_i s$ are the lagrange multipliers.*

**Theorem 2.1.16 ([7])** *For any given linear programming problem, strong duality always holds.*

### 2.1.4 Mathematical Optimization Problems with a Piecewise Linear Objective Function

In the following, we extend the result of the previous section to a more general case, where instead of a linear program, we have an optimization problem with a **piecewise linear** objective function and linear constraints. In order to do that, we first define a piecewise linear function:

**Definition 2.1.17 ([5])** *Function $f : R^n \longrightarrow R$ is a piecewise linear function if:*

- *it is a continuous function.*
- *there is a finite set $f_1, f_2, ..., f_m$ of linear functions $R^n \longrightarrow R$ such that for each $x \in R^n$, $f(x) = f_i(x)$ for some $i \leq m$.*

Based on this definition, the following three properties of piecewise linear functions are easy to see:

**Proposition 2.1.18** *Summation of a group of piecewise linear functions is a piecewise linear function.*

**Proposition 2.1.19** *If $f$ is a piecewise linear function, then $a.f$ is also a piecewise linear function for any $a \in R$.*

**Proposition 2.1.20** *If $f(y)$ is a piecewise function with respect to $y$ and $y = x_1 + x_2 + x_3 + ... + x_n$, then $g(x_1, x_2, ..., x_n) = f(x_1 + x_2 + x_3 + ... + x_n)$ is also a piecewise linear function with respect to $x_i$s.*

**Theorem 2.1.21 ([16])** *For any optimization problem with a piecewise linear objective function and linear constraints:*

- *There is a corresponding dual problem which is a linear program.*
- *Strong duality holds for the primal and dual problem.*

For a linear programming problem defined by (2.10), (2.11) and (2.12), one can define a function $y(b)$ in the following form:

$$y(b) = \min_x \ c^T x \tag{2.16}$$

subject to:

$$Ax = b, \tag{2.17}$$

$$x \geq 0. \tag{2.18}$$

The following theorem states a very useful property of the function $y(b)$, which will be used later in this thesis:

**Theorem 2.1.22 ([19])** *Given a linear programming problem in the canonical form, the relation $y(b)$ defined as a function of right hand side constraints, $b$ is a convex and piecewise linear function.*

The function $y(b)$ can also be defined for an optimization problem with a piecewise linear objective function and linear constraints. The following theorem shows that similar to the previous case, $y(b)$ is a convex and piecewise linear function when the objective function is piecewise linear.

**Theorem 2.1.23** *Given an optimization problem with a piecewise linear objective function and linear constraints, the relation $y(b)$ defined as a function of right hand side constraints, $b$ is a convex and piecewise linear function.*

**Proof** According to Theorem 2.1.21, there is a dual linear program for the optimization problem having a piecewise linear objective function and linear constraints. Also, since by Theorem 2.1.21, strong duality holds for this problem, the optimum value of this dual problem is equal to the optimum value of the original problem. Therefore, by Theorem 2.1.22, we conclude that $y(b)$ is a convex and piecewise linear function. $\triangle$

### 2.1.5 Integer Linear Programming

An **integer programming problem** [34] can be defined as follows:

**Definition 2.1.24** *In a linear programming problem, if the decision variables only take integer values, the problem is called an integer linear programming problem.*

Unlike linear programming problems, the integer programming problem is NP-hard. However, many real world optimization problems force us use integer values. We will introduce the minimum cost network flow problem, which is an important example of the integer linear programming problems in the next Section. The minimum cost network flow problem is used later in this thesis. For further reading about integer linear programming, we refer the reader to [34].

### 2.1.6 Minimum Cost Network Flow Problem

In this section, we introduce the minimum cost network flow problem [10]. The minimum cost network flow problem can be considered as a special case of the integer linear programming problem, which can be used in order to formulate a number of real world problems such as the transportation problem. The linear programming formulation of a minimum cost network flow problem for a graph, $G(V, E)$, is the following:

$$\min \sum_{i,j} c_{ij} x_{ij} \tag{2.19}$$

subject to:

$$l_{ij} \leq x_{ij} \leq u_{ij} \qquad \forall i, j \in V, \tag{2.20}$$

$$\sum_j x_{ij} - \sum_k x_{ki} = b_i \qquad \forall i \in V. \tag{2.21}$$

In this formulation:

- The optimization variable $x_{ij}$ is the value of flow that can go through the arc $(i, j)$ of $G$.
- The parameter $c_{ij}$ in (2.19) stands for the cost per unit of flow through arc $(i, j)$ .
- The constraint (2.21) is known as the conservation flow constrains of the problem. In this constraint, if $b_j$ is a positive number, node $j$ is called a **source**, if it is a negative number, node $j$ is called a **sink**. Note that for the remaining nodes, $b_j$ must be zero.
- The parameters $l_{ij}$ and $u_{ij}$ in (2.20) are the limits on the arc $(i, j)$ forcing it to have an upper and a lower bound on the amount of its flow.
- The goal is to find a flow that minimizes the total cost going through the arcs which satisfies all the constraints of the problem such as conservation flow constraints.

The following theorem states a very useful property of the minimum cost network flow problem:

**Theorem 2.1.25 ([10])** *In the minimum cost flow problem defined by* (2.19)*,* (2.20) *and* (2.21)*, the optimal solution is always integer if the parameters $l_{ij}$ and $u_{ij}$ are integer.*

Therefore, if we formulate the minimum cost network flow problem using integer linear programming, it can be solved in polynomial time using the algorithms designed for solving linear programming problems.

## 2.2 Stochastic Dynamic Programming

The dynamic programming method [4, 31] can be used to study the problems requiring decisions to be made sequentially over time. Since making decision for short periods of time is usually easier than making long term decisions, subproblems of the dynamic programming method in this case are determined by looking at the problems with shorter operating periods. In the time-based dynamic programming problems, we have a set of states and several time periods. At each specific state during the operating period, a decision should be made. Based on this decision, the system goes to another state which results in a cost. The goal of a dynamic program is to minimize the whole resulting cost throughout the operating time of the system. The time-based dynamic programs can have an infinite or

a finite time horizon, and their timing can be either continuous or discrete. Here, we only consider finite horizon discrete time dynamic programs.

Time-based dynamic programming formulations can be divided into two different versions: deterministic and stochastic. In the deterministic case, the outcome of our decisions for every state and every possible decision is precisely known. However, in the stochastic case, there is a randomness associated with the outcome of our decisions. This randomness can result from either the cost that we expect from the system or the state we will reach after making a decision.

In general, a discrete time stochastic dynamic programming formulation can be characterized by the following parameters:

- $t \in \{1, 2, ..., N\}$ : the index of time,
- $w_t$ : the states of the system,
- $y_t$ : the decision variables,
- $\theta_t$ : a random parameter which specifies the randomness of the system,
- $g_t(w_t, y_t, \theta_t)$ : a cost function. Note that the cost incurred at each step accumulates over time, and
- $f_t(w_t, y_t, \theta_t)$ : a probabilistic function that describes how the states are updated in the system. For example, this function can be $w_{t+1} = f_t(w_t, y_t, \theta_t) = w_t + y_t - \theta_t$ where $\theta_t$ is a random variable.

### 2.2.1 Optimal Policies

As is mentioned above, the dynamic programming method can be used to solve the problems in which the goal is to optimize the resulting expected cost in the system by making decisions over time. In these problems, in order to achieve the optimal expected cost, the best decision for every possible state at every time period should be known. One proposed way to present an optimal solution of such problems is to specify some rules for decision making rather than assigning an exact optimal decision at each state and time period. Such a rule is called a policy for the problem. Accordingly, we give the following formal definition:

**Definition 2.2.1 ([4])** *A **policy** is a function, $\mu_t(w_t)$, which takes the state $w_t$ at the operating period $t$, and returns a value for the decision variable. An **optimal policy**, $\mu^*$, is a policy which optimizes the resulting expected cost of the system.*

One way to calculate the optimal policy is to use the dynamic programming method. In general, finding an optimal policy at every possible state of the system is a computationally intractable problem. However, there might be cases in which the optimal policy has some

additional properties. For instance, if one knows that the optimal policy for a problem is a well-known function (e.g., a linear function), finding the optimal policy can be done by finding the corresponding coefficients of the linear function, either theoretically or computationally. Knowing that the optimal policy has a structural property can significantly decrease the computational complexity of the problem of finding the optimal policy. One famous example is **piecewise linear** or **threshold type policy**. This kind of policy is very useful because it has a very simple structure and is easy to represent. In fact, we only need to find a small number of parameters to build an optimal policy of this kind.

# Chapter 3

# Deterministic Model

In this chapter, we discuss car relocations in the day/night one way car sharing problem in the deterministic framework. In our deterministic framework, it is assumed that the demand is known in advance for the whole operating time. We start our analysis by studying the optimal decisions for rentals and relocations assuming that the operating period only consists of one day and one night. For this simple case, it is assumed that the initial and final distributions of cars among stations are given. We model this case using integer linear programming. Next, we state and formulate a multi-period version of the problem using the minimum cost network flow problem. The network flow formulation is an extension of the single-period formulation, which admits time efficient solution algorithms.

## 3.1 Single-Period Model

### Problem Statement

In the day/night one way car sharing problem, the initial and final distribution of the cars, the number of stations, and the corresponding costs/revenues of rentals/relocations in the system are given. Based on this information, we want to decide which requests to satisfy and which relocations to perform in order to minimize the resulting cost in the system.

### Problem formulation

To formulate this problem, we need three groups of decision variables $x_{ij}$, $y_{ij}$, and $e_i$. The variable $x_{ij}$ indicates the number of cars given to the customers from the station $i$ to station $j$ during the day. The variable $y_{ij}$ denotes the number of relocations from the station $i$ to station $j$ at night. In addition, $e_i$ indicates the number of cars in the station $i$ which are not given to any customer. The reason that we need to separately define $e_i$ is that when we are deciding about rentals, two similar cases can happen. There might be a case where we decide not to give some of the cars at some station to any customer. This can happen

either because the customer requests are not beneficial for us or we may not receive enough requests. There also might be another case where we decide to give a car to a customer who will return the car to the same station. In both cases, the number of cars at the station remains unchanged after the rental period. However, keeping a car at a station might result in a cost for us, while renting it to a customer yields us revenue. Therefore, in our models we need to define two distinct groups of decision variables to be able to specify the differences between those cars that we keep at one station and those we give to customers who will return them to the same station.

We use the following notation in our formulation:

- Input parameters:
  - $m$: The number of stations.
  - $S = \{s_1, s_2, ..., s_m\}$: The initial distribution of cars among stations, where $s_k$ shows the number of cars at the station $k$ in the beginning.
  - $w = \{w_1, w_2, ..., w_m\}$: The final distribution of cars among stations, where $w_k$ shows the number of cars that are at the station $k$ at the end.
  - $c_{ij}$ : The cost of relocating a car from the station $i$ to station $j$.
  - $cr_i$ : The cost of keeping a car at the station $i$.
  - $r_{ij}$ : The revenue of renting a car from the station $i$ to station $j$.
  - $d_{ij}$ : The demand of customers from the station $i$ to station $j$.
- Decision variables:
  - $x_{ij}$ : The number of satisfied requests from the station $i$ to station $j$.
  - $y_{ij}$ : The number of relocations from the station $i$ to station $j$.

We can express the problem as the following linear program:

$$\min \sum_{i,j} c_{ij} y_{ij} + \sum_{i} cr_i e_i - \sum_{i,j} r_{ij} x_{ij} \tag{3.1}$$

subject to:

$$0 \leq x_{ij} \leq d_{ij} \qquad \forall i, j, \tag{3.2}$$

$$s_k = \sum_i x_{ki} + e_k \qquad \forall k, \tag{3.3}$$

$$w_k = \sum_i y_{ik} \qquad \forall k, \tag{3.4}$$

$$\sum_i y_{ki} = \sum_i x_{ik} + e_k \qquad \forall k, \tag{3.5}$$

$$e_i \geq 0, y_{ij} \geq 0. \tag{3.6}$$

In this formulation, the constraint (3.2) ensures that we do not give more cars to customers than the demand. The constraint (3.3) ensures that we will neither keep nor rent more cars than our inventory at each station. The constraint (3.4) together with (3.6) ensures that we put as many cars as required at each station. Note that $y_{ii}$ is the number of those cars relocated to the same station with zero cost(remaining cars). Finally, the constraint (3.5) together with (3.6) ensures that we do not relocate cars more than the inventory of each station at the beginning of the night. This inventory may be supplied from two sources: (i) those cars that we kept at each station and (ii) those cars brought to the station by the customers at the end of the day. This formulation fits a network flow problem whose more general form will be discussed in Section 3.2.

## 3.2 Multi-Period Model

In the multi-period problem, we have several time periods still divided into days and nights. With respect to the given information about initial and final points, different cases of the problem can be defined. In our formulation, it is assumed that the initial and the final distribution of the cars are not given. Based on this assumption, we also need to decide about the optimal initial and final distributions.

### Problem statement

In the multi-period day/night one way car sharing system, the following information is given: the total number of cars and stations, the demand in each period and all of the corresponding revenues/costs of rentals/relocations. In order to minimize the cost resulting from the car sharing system, at each time period, it is required to decide how many cars should be assigned to each station, which customer requests should be satisfied, and which relocations should be performed.

As mentioned before, two different approaches can be utilized in order to formulate the multi-period problem: minimum cost network flow and dynamic programming.

### Minimum Cost Network Flow Formulation

In this section, we first formulate our problem using integer linear programming. Similar to what we had for the single period case, we define:

- Input parameters:
  - $t \in \{0, 1, 2, 3, ...T\}$.
  - $m$: The number of stations.
  - $n$: The number of cars.
  - $d_{ijt}$ : The demand of customers from the station $i$ to station $j$ at the period $t$.
  - $c_{ijt}$ : The cost of relocating a car from the station $i$ to station $j$ at the period $t$.
  - $r_{i0}$ : The cost of keeping a car at the station $i$.
  - $r_{ij}$ : The revenue of renting a car from the station $i$ to station $j$.
- Decision variables:
  - $x_{ijt}$ : The number of satisfied demands from the station $i$ to station $j$ at the period $t$.
  - $y_{ijt}$ : The number of relocations from the station $i$ to station $j$ at the period $t$.
  - $e_{it}$ : The number of cars left at the station $i$ during the day at the period $t$. Note that $s_{it} \in \{s_{1t}, s_{2t}, ..., s_{mt}\}$ is the number of cars at the station $i$ at the beginning of the period $t$ and $s_{it} = e_{it} + \sum_k x_{kit}$.

Accordingly, the formulation is written as:

$$\min \sum_{i,j,t} c_{ijt} y_{ijt} + \sum_{i,t} r_{i0} e_{it} - \sum_{i,j,t} r_{ij} x_{ijt} \tag{3.7}$$

subject to:

$$0 \leq x_{ijt} \leq d_{ijt} \tag{3.8}$$

$$s_{kt} = \sum_i x_{kit} + e_{kt} \qquad \forall k, t \tag{3.9}$$

$$s_{kt} = \sum_i y_{ik(t-1)} \qquad \forall k, t \tag{3.10}$$

$$\sum_i y_{kit} = \sum_i x_{ikt} + e_{kt} \qquad \forall k, t \tag{3.11}$$

$$\sum_k s_{kt} \leq n \qquad \forall t \tag{3.12}$$

$$e_{it} \geq 0, y_{ijt} \geq 0. \tag{3.13}$$

The first four constraints in this formulation are the same as the ones defined for the single-period case. The only difference is the time indices showing the changes over time in

the system. For the multi-period case, we also need to have (3.12) in order to make sure the number of cars assigned to the stations at each period is not more than the total number of the cars.

**Theorem 3.2.1** *The integer linear programming formulation defined by* (3.7)-(3.13) *can be solved in polynomial time.*

**Proof** The integer linear programming formulation given above can be reduced to a minimum cost network flow problem, which will be characterized below. As a result, the problem can be solved in polynomial time.

- For every station at each time period we first assign three nodes in the network, one for the beginning of the day, one for the beginning of the night, and one for the end of the night. Then, the following connections are made:
  - all nodes at the beginning of the days to all nodes at the beginning of nights of the same time periods.
  - all nodes at the beginning of the nights to all nodes at the end of nights of the same time periods.
  - all nodes at the end of the nights to all nodes at the beginning of the next time periods.
- The flows are the number of cars moving between stations.
- We add a source to the network and connect it to the first group of nodes at the beginning of the day of the first time period. The supply of this source is the total number of cars in the system. In addition, a sink is added to the network and every node at the end of the last period is connected to this sink. The demand of this sink is also the total number of cars in the system.
- The cost per flow for each arc corresponds to the cost of relocating or the negation of the revenue of renting the car from the starting station to the final one.
- The lower bound of each edge is zero. The upper bound is the total number of cars and $d_{ijt}$ for the relocation and rental edges, respectively.

An example illustrating the above-mentioned process is shown in Figure 3.1. In this graph, the nodes specify the stations at each stage of the operating periods and the upper bound of the arcs are shown by their corresponding numbers. If we write the conservation flow constraints for the vertices (stations) at the end of night, at the beginning of day, and at the end of day, the relations (3.9), (3.10), and (3.11) can be inferred accordingly. Finally, if we set maximum outgoing flow of the source node equal to $n$, we get (3.12). After solving

the minimum cost flow problem, if the outgoing flow from this node was less than $n$, it can be concluded that the system could have a smaller number of cars. $\triangle$. It is also worth mentioning that since the graph in our network flow formulation is a bipartite graph, it can be reduced to the Hitchcock problem [15], which is a special case of minimum cost network flow problem and can be solved in a very effiecient time.

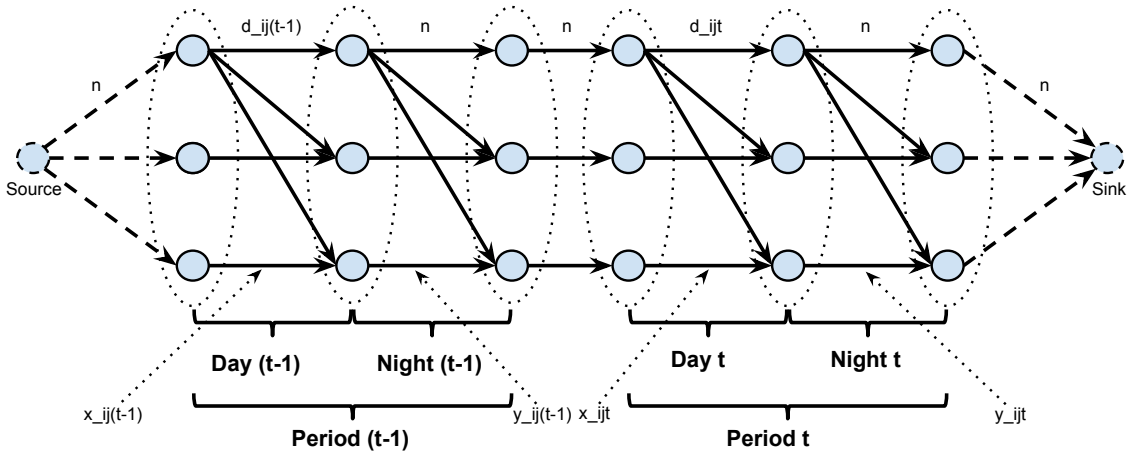


Figure 3.1: The network flow structure for the determinstic model at periods $t-1$ and $t$. The dashed lines show the periods before $t-1$ and after $t$.

# Chapter 4

# Stochastic Model

In this chapter, we develop a framework to analyze the day/night one way car-sharing problem when the customer demand is stochastic. Similar to the deterministic version, the modelling starts with a single-period case and then it is extended to the multi-period one. As before, it is assumed that (i) relocations only happen over night and (ii) each rental/relocation takes exactly one day/night.

As in the deterministic version, a time period in the stochastic model consists of two stages: rental and relocation. However, this time each period starts with the relocation stage, i.e., the beginning of night when the operator knows the current distribution of cars among the stations and makes decisions about relocations. These decisions are made to optimize the expected cost of the remaining periods until the end of the operating time. Then, in the morning, the rental stage starts in which the actual demand becomes known and the operator decides to either accept or deny the customer requests. Based on these decisions, cars are moved among stations. This is when a period ends in our model.

In order to model our stochastic multi-period problem, we use a dynamic programming approach. As mentioned in Section 2.2, dynamic programming is usually a good mathematical tool to model real world problems over time. However, if a dynamic programming formulation does not have the desirable properties such as convexity, it is often computationally intractable. In this chapter, after presenting the dynamic programming formulation of our problem, the following two important properties are discussed:

- The mathematical optimization problems which will be defined later are convex optimization problems for each stage.
- There can be simple policies for performing relocations that can obtain a resulting expected cost close to optimal.

Sections 4.1 and 4.2 of this chapter are dedicated to the mathematical formulations of the single- and multi-period day/night one way car sharing problem. In Section 4.3, it is shown that solving our dynamic programming formulation at every stage is a convex

optimization problem. Therefore, we can find the optimal values of our objective functions more efficiently. Although the convexity property helps us run our dynamic program much faster, finding the optimal values for a real world problem is still time consuming. In Section 4.4, to derive some theoretical results, it is assumed that the decisions in the rental stages will be done by an oracle. This oracle always performs rentals in order to get the best expected result until the end of operating time. Therefore, the system operator only need to decide about relocations. Following this assumption, a simplified two-station case is discussed. It is shown that a threshold type policy introduced in Section 2.2.1 is the optimal policy for relocations. Based on this result, a heuristic algorithm is proposed in Section 4.5 to approximate the optimal ways for performing the relocations in a general $m$-station case. Our heuristic algorithm significantly decreases the computational complexity of our problem.

## 4.1 Single-Period Problem

### 4.1.1 Problem Statement

In the day/night car sharing system, the initial and final distributions of cars, all corresponding costs/benefits of renting a car from the station $i$ to station $j$, keeping it at the station $i$, and relocating it from the station $i$ to station $j$ are given. Let $f_{ij}(x)$ be a probability mass function representing the probability that $x$ customers request for a car from station $i$ to station $j$. The question is, what is the optimum way of renting cars and performing relocations at the minimum expected cost for the system?

### 4.1.2 Problem Formulation

We use the following notation for the problem formulation:

- Input parameters:
  - $m$ : The number of stations.
  - $S = \{s_1, s_2, ..., s_m\}$ : The initial distribution of cars among stations.
  - $W = \{w_1, w_2, ..., w_m\}$ : The final distribution of cars among stations.
  - $c_{ij}$ : The cost of relocating a car from the station $i$ to station $j$, such that $i, j \in \{1, 2, ..., m\}$
  - $r_{ij}$ : The revenue of renting a car from the station $i \in \{1, 2, ..., m\}$ to $j \in \{0, 1, 2, ..., m\}$. The cost of keeping a car at the station $i$ during day is denoted by $r_{i0}$.
  - $d_{ij}$ : The random variable denoting the customer demand from the station $i$ to station $j$. Also, $\mathbf{D} = [d_{ij}]$ is a random matrix formed by the $d_{ij}$s.

- $f_{ij}(x)$ : The probability mass function of $d_{ij}$.
- $P[\mathbf{D} = D]$ : The probability that a specific realization of the demand, $D$, happens. It is assumed that the customer demand from different stations are independent from each other. So, $P[\mathbf{D} = D] = \prod_{i,j} f_{ij}(d_{ij})$.

- Decision variables:
  - $x_{ij}$ : The number of satisfied demands from the station $i \in \{1, 2, ..., m\}$ to station $j \in \{0, 1, ..., m\}$. Note that $x_{ii}$ denotes the number of accepted requests from the station $i$ to itself and $x_{i0}$ is the number of cars left at the station $i$ during the day.
  - $y_{ij}$ : The number of relocations from the station $i$ to station $j$.

According to the description of a time period in the stochastic model, decisions about relocations and rentals are made at two distinct times. Therefore, two different objective functions are needed for each of these decisions. In this section, a linear program is defined to find the optimal value of $x_{ij}$ in the rental stage. Then, an optimization problem is proposed in order to obtain the optimal value of $y_{ij}$, which minimizes the cost of the relocation stage.

**Rental Stage**

Let $K = \{k_1, k_2, ..., k_m\}$ and $W = \{w_1, w_2, ..., w_m\}$ be the distributions of cars among the stations after the relocation and rental periods, respectively. Based on the problem statement, when we want to decide about $x_{ij}$, $W$ and the actual demand, $d$, are known. Consequently, the following linear program is proposed to find the optimal value of $x_{ij}$:

$$U(K, W, D) = \min_{x_{ij}} \left( - \sum_{i,j} r_{ij} x_{ij} \right) \tag{4.1}$$

subject to:

$$0 \leq x_{ij} \leq d_{ij} \qquad \forall i, j, \tag{4.2}$$

$$\sum_{i=0}^{m} x_{li} = k_l \qquad \forall l, \tag{4.3}$$

$$\sum_{i=0}^{m} x_{il} = w_l \qquad \forall l. \tag{4.4}$$

In this formulation, the rental costs are simply defined as the the negated rental revenues in the objective function. The constraint (4.2) basically keeps the number of rentals less than the demand in each station. The constraint (4.3) ensures that the operator does not

rent more cars than its inventory at each station. Finally, the constraint (4.4) forces the system to have the given distribution of cars after the rental period.

**Relocation Stage**

For the relocation stage, we try to find an optimal way of performing the relocations based on what we have for the rental stage so far. According to the problem statement, the general form of the objective function is as follows:

$$\begin{aligned} \textit{Resulting cost of the relocation stage} &= (\textit{cost of relocating the cars}) \\ &+ (\textit{expected cost of the next day}), \end{aligned}$$

where cost of relocating cars is given by $\sum_{i,j} y_{ij} c_{ij}$. Regarding the expected costs of the next day, we note that:

- No matter what the customer demand is at the rental stage, decisions about rentals are made in order to minimize the cost of the system.
- Decisions about rentals highly depend on (i) the resulting distribution of cars among station after relocations and (ii) the actual realization of the demand.

Considering these points, we write the following mathematical program for the relocation stage:

$$\min_{y_{ij}} \left( \sum_{i,j} y_{ij} c_{ij} + \sum_{all\ possible\ D} P[\mathbf{D} = D] U(K, W, D) \right) \tag{4.5}$$

subject to:

$$\sum_{i=1}^{m} y_{li} = s_l \quad \forall l, \tag{4.6}$$

$$\sum_{i=1}^{m} y_{il} = w_l \quad \forall l, \tag{4.7}$$

where the objective function (4.5) is the summation of the costs of relocating cars and the expected cost of the rental stage over all possible realizations of the demand. Given the realization of the customer demand, $D$, the function $U(K, W, D)$ returns the resulting cost of the rental stage starting from $K$ to $W$. In addition, the constraint (4.6) ensures that the operator does not relocate more cars than the current inventory of each station. Finally, the constraint (4.7) gives the resulting distribution of cars among stations after performing the relocations.

While the expectation term in (4.5) may have a simple summation form, it requires a lot of computation. This is because we are summing over all possible realizations of customer

demand, and the number of these realizations is exponential. There are various approaches in the literature to deal with this issue. In this thesis, we use sampling methods in order to estimate the expected values. In these methods, one can only sample a limited number of possible realizations of demand, and estimate the value of the desired expectation using these samples.

## 4.2 Multi-Period Problem

### 4.2.1 Problem Statement

In the day/night car sharing system, there are $T$ operating periods. Each period starts with a night when the relocations are performed and ends with a day when customers rent cars. The distribution of cars among stations at the beginning of each period, all the corresponding costs/benefits of renting a car from the station $i$ to station $j$, keeping it at the station $i$, relocating it from the station $i$ to station $j$ are given. Let $f_{ij}^t(x)$ be a probability mass function representing the stochastic behaviour of the customer demand at the period $t$. The questions is, what is the optimal way of renting cars during the day and performing relocations at nights to have the minimum expected cost from the system?

### 4.2.2 Problem Formulation

We use dynamic programming to model our problem. In our model, it is assumed that as time goes on, decisions are sequentially made at each stage to minimize the expected cost of the system until the end of the operating time.

Let $W_t = \{w_{1t}, w_{2t}, w_{3t}, ..., w_{mt}\}$ and $K_t = \{k_{1t}, k_{2t}, k_{3t}, ..., k_{mt}\}$ be the states of the system before and after the relocation stage of the period $t$. Similar to the single-period case, we make decisions of two different kinds at each period:

- First, we look at the current state $W_t$, and make decisions about the relocations. Based on these decisions, the staff move cars between the stations and the system goes to $K_t$.
- Then, as the day begins, the actual demand becomes known and decisions about renting the cars are made. Following these decisions, customers move cars between the stations, and the system goes to the state $W_{t+1}$.

In order to develop our dynamic program, we define the following functions for each group of the decision variables:

- $V_t(W_t, y_{ijt})$ : The resulting cost of the system from period $t$ on, after the decisions about relocations, $y_{ijt}$, are made, where $W_t$ is the state of the system. In order to initialize the dynamic program the value of $V_{T+1}$ is set to zero.

- $V_t^*(W_t)$ : The optimal value of $V_t(W_t, y_{ijt})$ over all possible $y_{ijt}$.
- $U_t(K_t, D, x_{ijt})$ : The resulting cost of the system from period $t$ on, after the decisions about renting cars, $x_{ijt}$, are made, where $K_t$ is the current distribution of cars among stations and $D$ is a realization of customers request.
- $U_t^*(K_t, D)$ : The optimal value of $U_t(K_t, D, x_{ijt})$ over all possible $x_{ijt}$.

The above defined values can be calculated as follows:

$$V_t(W_t, y_{ijt}) = \sum_{i,j} y_{ijt} c_{ijt} + \sum_{all\ possible\ D} P(\mathbf{D} = D) U_t^*(K_t, D) \tag{4.8}$$

subject to:

$$\sum_{i=1}^{m} y_{lit} = w_{lt} \quad \forall l, t, \tag{4.9}$$

$$\sum_{i=1}^{m} y_{ilt} = k_{lt} \quad \forall l, t. \tag{4.10}$$

And for each $D = [d_{ij}]$,

$$U_t(K_t, D, x_{ijt}) = \left( -\sum_{i,j} x_{ijt} r_{ijt} \right) + V_{t+1}^*(W_{t+1}) \tag{4.11}$$

subject to:

$$0 \leq x_{ijt} \leq d_{ij} \quad \forall i, j, \tag{4.12}$$

$$\sum_{i=0}^{m} x_{lit} = k_{lt} \quad \forall l, t, \tag{4.13}$$

$$\sum_{i=0}^{m} x_{ilt} = w_{l(t+1)} \quad \forall l, t, \tag{4.14}$$

where $V_t^*$ and $U_t^*$ are obtained by solving the following optimization problems with the same constraints:

$$V_t^*(W_t) = \min_{y_{ijt}} V_t(W_t, y_{ijt}) \tag{4.15}$$

$$U_t^*(K_t, D) = \min_{x_{ijt}} U_t(K_t, D, x_{ijt}) \tag{4.16}$$

The constraints (4.9), (4.10), (4.12), (4.13), (4.14) are defined similar to the constraints in the single-period formulation. However, the objective functions (4.8) and (4.11) are slightly different from the single-period case. The function (4.8) is the summation of the cost of relocations and the optimal expected cost after relocations until the end of the operating

time. The function (4.11) is the cost of renting cars and the optimal cost after the rental period.

## 4.3 Convexity Analysis of The Dynamic Programming Formulation

In this section, we prove the convexity of the following functions:

- $V_t(W_t, y_{ijt})$ with respect to $y_{ijt}$.
- $U_t(K_t, D, x_{ijt})$ with respect to $x_{ijt}$
- $V_t^*(W_t)$ with respect to $W_t$.
- $U_t^*(K_t, D)$ with respect to $K_t$.

As mentioned before, it is important to know that we only require to solve a convex optimization problem for finding the optimal value of the objective function at each stage. This is because without having the additional property of convexity, it is almost impossible to find the optimal values of the functions $V_t$ and $U_t$ defined by (4.8) and (4.11), in the previous section.

In order to show that relations (4.15) and (4.16) are convex optimization problems for each $t$, we need to check the convexity of (i) the constraints of our mathematical formulations and (ii) the objective functions. The constraints in our formulation are linear, so, they are convex. For the objective functions, we first show their piecewise linearity in Theorem 4.3.1. Then, using this property, the convexity of these functions is proved in Theorem 4.3.2.

**Theorem 4.3.1** *The objective functions $V_t(W_t, y_{ijt})$ and $U_t(K_t, D, x_{ijt})$ are piecewise linear functions with respect to $y_{ijt}$ and $x_{ijt}$.*

**Proof** We prove this theorem by induction on $t$. The functions $U_T$ and $V_{T+1}$ are considered as the base cases. The function $V_{T+1}$ is a zero function, so it is piecewise linear. Moreover, since $V_{T+1}^*(W_{t+1})$ equals zero, $U_T$ is a (piecewise) linear function.

For the induction step, we assume that $V_t$ is a piecewise linear function. Having this, we first prove the piecewise linearity of $U_{t-1}$ and then extend the argument to show that $V_{t-1}$ is piecewise linear as well. For $U_{t-1}$, we have:

$$U_{t-1}(K_{t-1}, D, x_{ij(t-1)}) \quad = \quad (-\sum_{i,j} x_{ij(t-1)} r_{ij(t-1)}) + \min_{y_{ijt}} (V_t(W_t, y_{ijt})), \qquad (4.17)$$

According to the constraint (4.14), the set $W_t$ plays the same role as the vector $b$ in Theorem 2.1.23. By this theorem, the minimization term in (4.17) is a piecewise linear function

with respect to $W_t$. In addition, $w_{kt}$ is a linear function of $x_{ij(t-1)}$. Thus, by Propositions 2.1.18 and 2.1.20, $U_{t-1}$ is also a piecewise linear function with respect to $x_{ij(t-1)}$.

For $V_{t-1}$, we have:

$$\begin{aligned} V_{t-1}(W_{t-1}, y_{ijt-1}) &= \sum_{i,j} y_{ij(t-1)} c_{ij(t-1)} \\ &+ \sum_{all\ possible\ D} P(\mathbf{D} = D) \min_{x_{ij(t-1)}} U_{t-1}(K_{t-1}, D, x_{ij(t-1)}), \quad (4.18) \end{aligned}$$

Similar to the previous case, based on the constraint (4.10) and our knowledge that $U_{t-1}$ is a piecewise linear function, we understand that $K_{t-1}$ plays the same role as the vector $b$ in Theorem 2.1.23. Consequently, with the same argument as for $U_{t-1}$, and since $P[\mathbf{D} = D]$ is a real number, and by Proposition 2.1.19, we conclude that $V_{t-1}$ is also a piecewise linear function with respect to $y_{ij(t-1)}$. $\triangle$

**Theorem 4.3.2** *The objective functions $V_t(W_t, y_{ijt})$ and $U_t(K_t, D, x_{ijt})$ are convex functions with respect to $y_{ijt}$ and $x_{ijt}$.*

**Proof** We prove the convexity of $V_t$ and $U_t$ inductively. As the base cases, we have $V_{T+1}$ and $U_T$. Function $V_{T+1}$ always equals zero, so it is a convex function. Furthermore, from the previous proof, we know that $U_T$ is a linear function and thus, is convex.

For the induction step, we assume that functions $V_t$ and $U_t$ are convex functions and show the convexity of $V_{t-1}$ and $U_{t-1}$, respectively. In order to show that $U_{t-1}$ is convex, it suffices to prove that the minimization function in (4.17) is a convex function with respect to $x_{ij(t-1)}$. By Theorem 4.3.1, we know that $V_t$ is a piecewise linear function. Also, the set $W_t$ is the right hand side constraint vector of the minimization function in (4.17). Thus, by Theorem 2.1.21, it can be inferred that the objective function of the dual of the minimization term can be written as follows:

$$\phi(X_{t-1}) = \max_v \ [IX_{t-1}v] \quad (4.19)$$

where $X_{t-1} = [x_{ij(t-1)}]$ is a $m \times m$ matrix, $I$ is a $1 \times m$ unity vector and $v$ is a $m \times 1$ vector indicating the dual's optimization variables. Observe that for any two functions $f(x)$ and $g(x)$, $\max_x(f(x) + g(x)) \leq \max_x f(x) + \max_x g(x)$. Therefore, we write:

$$\begin{aligned} \theta\phi(X_1) + (1-\theta)\phi(X_2) &= \theta \max_v[\ IX_1v] + (1-\theta) \max_v[IX_2v] && (4.20) \\ &= \max_v[\theta IX_1v] + \max_v[(1-\theta)IX_2v] && (4.21) \\ &\geq \max_v[I(\theta X_1 + (1-\theta)X_2)v] && (4.22) \\ &= \phi(\theta X_1 + (1-\theta)X_2) && (4.23) \end{aligned}$$

As a result, $\phi(X_{t-1})$ is a convex with respect to $x_{ij(t-1)}$. Finally, by Proposition 2.1.9, the sum of this convex and piecewise linear function with the linear term $\sum_{i,j} x_{ij(t-1)} r_{ij(t-1)}$ is also a convex function, which results in the convexity of $U_{t-1}$ with respect to $x_{ij(t-1)}$.

For $V_{t-1}$, we already know that $U_{t-1}$ is both convex and piecewise linear. Furthermore, based on the constraint (4.10) $K_{t-1}$ is the right hand side constraint of the minimization term in (4.18). Therefore, with a similar argument as for $U_{t-1}$, we conclude that this minimization term is a convex function. Moreover, since $P[\mathbf{D} = D]$ is always nonnegative and $\sum_{i,j} y_{ij(t-1)} c_{ij(t-1)}$ is a convex function, by Proposition 2.1.9, we conclude that $V_{t-1}$ is a convex function with respect to $y_{ij(t-1)}$. Consequently, the proof is complete. $\triangle$

As mentioned in Chapter 2, knowing that $V_t(W_t, y_{ijt})$ and $U_t(K_t, D, x_{ijt})$ are convex functions helps us solve the minimization problems at each stage, efficiently. Without this additional property, running the dynamic program even for one period of time is basically not possible.

**Theorem 4.3.3** *The objective functions $V_t^*(W_t)$ and $U_t^*(K_t, D)$ are convex functions with respect to $W_t$ and $K_t$.*

**Proof** According to (4.9) and (4.13), $W_t$ and $K_t$ are the right hand side constraints of the minimization problems defined by (4.15) and (4.16), respectively. In addition, by Theorem 4.3.1, the functions $V_t(W_t, y_{ijt})$ and $U_t(K_t, D, x_{ijt})$ are piecewise linear functions for any $t$. As a result, by Theorem (2.1.23), we conclude that $V_t^*(W_t)$ and $U_t^*(K_t, D)$ are convex functions with respect to $W_t$ and $K_t$. $\triangle$

Theorem 4.3.3 will be used in the next section to find the optimal policy of the two station car sharing problem.

## 4.4 Verifying The Optimality of Two-Threshold Type Policy For The Two Station Car Sharing Problem

In this section, we address the problem of finding an optimal policy for performing the relocations in the two-station version of the day/night one way car sharing problem. We assume that there is an oracle in the system, who accepts customer requests in order to get the best expected result until the end of the operating time. So, $x_{ijt}$ is not a decision variable anymore. Having this assumption, it is shown that when there are only two stations in the system, characterizing the optimal policy for performing relocations is possible using a threshold type policy. Although having only two stations seems to be unrealistic, it is going to help propose a heuristic for a general $m$-station version of the problem.

For the two-station version, the following observations can be made:

1. Since there are only two stations and a constant number of cars in the system, each state can be uniquely determined by the number of cars in one of the stations at period $t$. Therefore, the state and decision spaces have only one dimension. In our formulation, the state space for the given number of cars, $n$, is specified by the number of cars at station 1 and is denoted by $w_t$, where $w_t \in \{0, ..., n\}$.

2. In the two station case, relocations at period $t$ can be specified by a single decision variable $y_t$ defined as the number of cars that we relocate from station 1 to station 2, i.e., $y_t = y_{12t} - y_{21t}$. Since at each period $t$, the number of cars at station 1 is $w_t$ and at station 2 is $n - w_t$, $y_t$ can vary between $-n + w_t$ and $w_t$.

We begin characterizing the optimal policy by stating the following theorem:

**Theorem 4.4.1** *In the two staion one way car sharing problem, the functions $V_t'^*(w_t)$ and $U_t'^*(w_t, D)$ defined below are convex functions with respect to $w_t$.*

$$V_t'^*(w_t) = V_t^*(w_t, n - w_t) \tag{4.24}$$

$$U_t'^*(w_t, D) = U_t^*(w_t, n - w_t, D) \tag{4.25}$$

*where $w_t$ is the number of cars at station 1 , $n - w_t$ is the number of cars at station 2, and $V^*$ and $U^*$ are defined by* (4.15) *and* (4.16)*,*

**Proof** By Theorem 4.3.3, $V_t^*(w_{1t}, w_{2t})$ and $U_t^*(w_{1t}, w_{2t}, D)$ defined by (4.15) and (4.16) are convex functions with respect to $(w_{1t}, w_{2t})$. By Proposition 2.1.10, the functions $V_t^*(w_{1t} - w_{2t}, -w_{1t} - w_{2t} + n)$ and $U_t^*(w_{1t} - w_{2t}, -w_{1t} - w_{2t} + n, D)$ are also convex. If we set $w_{2t} = 0$, we get:

$$V_t^*(w_{1t}, -w_{1t} + n) = V_t'^*(w_{1t}) \tag{4.26}$$

$$U_t^*(w_{1t}, -w_{1t} + n, D) = U_t'^*(w_{1t}, D) \tag{4.27}$$

So, the proof is complete. $\triangle$

We can characterize the optimal policy in the two-station case by the following theorem:

**Theorem 4.4.2** *In the day/night one way car sharing problem where we only have two stations, the total optimal policy can be characterized in the following two-threshold type structure:*

$$\mu_t^*(w_t) = \begin{cases} w_t - S_t', & \text{if } w_t < S_t' \\ 0, & \text{if } S_t' \le w_t \le S_t \\ w_t - S_t, & \text{if } w_t > S_t \end{cases} \tag{4.28}$$

*where $\mu_t^*(w_t)$ is the optimal policy of the problem.*

**Proof** Following the above assumptions, we can rewrite the recursion (4.8) to calculate $V_t'^*$ defined by (4.24) as in below:

$$V_t'^*(w_t) = \min_{y_t} (c|y_t| + E[U_t'^*(w_t - y_t, D))], \tag{4.29}$$

where $V_t'^*$ and $U_t'^*$ are defined by (4.24) and (4.25) and the values of $y_t$ can vary between $(-n + w_t, w_t)$. Note that a zero $y_t$ means that we do not relocate any cars, a positive one means that cars are being moved from the station 1 to 2 and a negative one means that cars are being moved from station 2 to 1. We assume that the problem is solved once for $y_t \in (0, w_t)$ and once for $y_t \in (-n + w_t, 0)$. Clearly, only one these strategies can be optimal at each time and the optimal strategy would be the one that results in the smaller cost at each time.

Let $z_t = w_t - y_t$, when $y_t$ is nonnegative, $|y_t|$ equals $y_t$. Therefore, we can define the following optimization problem based on the equation (4.29):

$$V_t'^*(w_t) = \min_{0 \le z_t \le w_t \le n} [G_t^1(z_t) + cw_t] \tag{4.30}$$

where

$$G_t^1(z_t) = -cz_t + E[U_t'^*(z_t, D)] \tag{4.31}$$

$$\tag{4.32}$$

By Theorem 4.4.1, and Propositions 2.1.9, $E[U_t'^*(z_t, D)]$ is a convex function. This implies that $G_t^1$ is also convex. Therefore, $S_t$ can be defined as below:

$$S_t = \arg \min_{0 \le z_t \le n} G_t^1(z_t) \tag{4.33}$$

When we have the constraint $z_t \le w_t$, based on the convexity of the function $G_t^1(z)$, the optimal value of the minimization problem (4.30) can be obtained by one of the following cases:

1. If $w_t \le S_t$, the feasible region only includes the values equal or smaller than $S_t$. By (4.33), the values of $G_t^1(z_t)$ for any $z_t$ in this feasible region is larger than $G_t^1(S_t)$. In addition, we can not have any local optimum in this region. Therefore, the more we decrease $z_t$, the larger $G_t^1(z_t)$ becomes. Thus, the optimal value for $z_t$ equals the biggest value in the feasible region which is $w_t$.

2. If $w_t > S_t$, since $S_t$ is in the feasible region of the minimization problem, based on (4.33), the optimal answer for $z_t$ equals $S_t$.

By substituting the optimal values of $z_t$ by $w_t - y_t$, we get:

$$y_t^* = \mu_t^*(w_t) = \begin{cases} 0, & \text{if } w_t \leq S_t \\ w_t - S_t, & \text{if } w_t > S_t \end{cases} \tag{4.34}$$

When $y_t$ is nonpositive, $|y_t|$ equals $-y_t$ and we can write:

$$V_t'^*(w_t) = \min_{0 \leq w_t \leq z_t \leq n} [G_t^2(z_t) - cw_t] \tag{4.35}$$

where

$$G_t^2(z_t) \quad = \quad cz_t + E[U_t'^*(z_t, D)] \tag{4.36}$$

Similar to $G_t^1$, $G_t^2(z)$ is a convex function. So, we can define:

$$S_t' = \arg \min_{0 \leq z_t \leq n} G_t^2(z_t). \tag{4.37}$$

Accordingly, the optimal value of the minimization problem defined by (4.35) can be obtained by one of the following two cases:

1. If $w_t \geq S_t'$, the values in the feasible region are equal or greater than $S_t'$. Therefore, the optimal value for $z_t$ is $w_t$.
2. If $w_t < S_t'$, then $S_t'$ is in the feasible region of the minimization problem, thus, the optimal $z_t$ equals $S_t'$.

By substituting the values of $z_t$ by $w_t - y_t$, we get:

$$y_t^* = \mu_t^*(w_t) = \begin{cases} 0, & \text{if } w_t > S_t' \\ w_t - S_t', & \text{if } w_t \leq S_t' \end{cases} \tag{4.38}$$

This equation could also be derived by assuming that the problem is being solved when station 2 indicates the state of the problem and $y_t' = -y_t$ is the decision variable. With this assumption, the optimal policy for this case has the similar structure and we can write:

$$y_t'^* = \mu_t^*(n - w_t) = \begin{cases} 0, & \text{if } n - w_t < S_t \\ (n - w_t) - S_t, & \text{if } n - w_t \geq S_t \end{cases} \tag{4.39}$$

If we set $S' = n - S_t$, we have:

$$y_t'^* = \mu_t^*(w_t) = \begin{cases} 0, & \text{if } w_t > S_t' \\ S_t' - w_t, & \text{if } w_t \leq S_t' \end{cases} \tag{4.40}$$

Therefore:

$$y_t^* = -y_t^{'*} = \mu_t^*(w_t) = \begin{cases} 0, & \text{if } w_t > S_t' \\ w_t - S_t', & \text{if } w_t \leq S_t' \end{cases} \tag{4.41}$$

which is the same as (4.38).

In order to derive (4.28), we observe that the value of $S_t'$ defined by (4.37) can not be bigger than the value of $S_t$ defined by (4.33) for any state, $w_t$, in the system. We prove this observation by contradiction. If $S' > S$, then by (4.31) and (4.36) we have:

$$cS' + E[U_t'^*(S', D)] < cS + E[U_t'^*(S, D)] \tag{4.42}$$

and

$$-cS + E[U_t'^*(S, D)] < cS' + E[U_t'^*(S', D)]. \tag{4.43}$$

If we sum them up, we get:

$$c \cdot (S' - S) < c \cdot (S - S') < 0 \tag{4.44}$$

which is impossible since $c > 0$ and $S' - S > 0$.

As a result, according to equations (4.34) and (4.41), the optimal value of the system can be obtained from one of the following cases:

- $w_t \leq S_t'$: In this case, the policy is zero when $y_t$ is nonnegative and is nonzero when $y_t$ is negative. Therefore, by (4.30) and (4.35), we have:

$$\begin{aligned} V_t'^*(w_t) &= \min\{E[U_t'^*(w_t, D)], c|w_t - S_t'| + E[U_t'^*(S_t', D)]\} && (4.45) \\ &= \min\{E[U_t'^*(w_t, D)], cS_t' + E[U_t'^*(S_t', D)] - cw_t\} && (4.46) \\ &= cS_t' + E[U_t'^*(S_t', D)] - cw_t && (4.47) \end{aligned}$$

  Thus, the optimal cost is obtained when $y_t$ is negative.

- $S't \leq w_t \leq S_t$: In this case, the policy for both a nonnegative $y_t$ and a negative one is zero. So, the optimal policy equals zero as well.

- $S_t \leq w_t$: In this case, the policy obtained by a nonpositive $y_t$ is zero, while the policy obtained by a positive $y_t$ is nonzero. By (4.30) and (4.35), we get:

$$\begin{aligned} V_t'^*(w_t) &= \min\{E[U_t'^*(w_t, D)], c|St - w_t| + E[U_t'^*(S_t, D)]\} && (4.48) \\ &= \min\{E[U_t'^*(w_t, D)], cS_t + E[U_t'^*(S_t, D)] + cw_t\} && (4.49) \\ &= cS_t + E[U_t'^*(S_t, D)] + cw_t && (4.50) \end{aligned}$$

Therefore, the cost resulting from a positive $y_t$ is the optimal cost.

Consequently, if we combine (4.34) and (4.41), we get (4.28). $\triangle$

## 4.5 The Heuristic Algorithm For The $m$-Station Case

In the previous sections, we came up with a convex dynamic program for the day/night car sharing problem. Also, it was mentioned that the additional convexity property makes finding the optimal solution of our formulation more efficient. However, due to the big size of input data in the real world problems, running our dynamic program still takes a very long time. In order to handle this difficulty, a heuristic algorithm can be used to find a way of performing the relocations much faster and yet obtains a solution close enough to the optimal resulting cost.

In this section, we propose a heuristic algorithm to find a way for performing the relocations for each state and time period. This heuristic algorithm is designed based on the previous section in which it was shown that when there were only two stations in the system, a threshold type policy was the optimal policy for moving the cars among the stations. For a general $m$-station case, since our objective functions are convex and piecewise linear, similar to the two station case, we expect to have a piecewise linear optimal policy. For this case, the exact policy will be too complicated to characterize because we have more dimentions. However, we claim that a simple two-threshold type policy can be proposed for every individual station in the $m$-station case such that relocating cars among the stations based on these thresholds will make the total resulting cost of the system very close to optimal. We anticipate that because the two-threshold type policy is optimal for the 2-station case, our heuristic algorithm gives a reasonable output for the general case. The accuracy of this heuristic algorithm will be tested in Chapter 5.

### The Algorithm

Our heuristic algorithm consists of two stages: stage $(i)$ in which a lower and an upper bound is calculated for each station using dynamic programming, and stage $(ii)$ in which the relocations are decided based on the calculated thresholds. Remember that the thresholds obtained by stage $(i)$ are assumed to be time independent. Therefore, they can be used for any arbitrary length of operating time in stage $(ii)$. In the heuristic algorithm, we assume:

- As before, $W_t = \{w_{1t}, w_{2t}, w_{3t}, ..., w_{mt}\}$ and $K_t = \{k_{1t}, k_{2t}, k_{3t}, ..., k_{mt}\}$ are the states of the system before and after the relocation stage of the period $t$.
- The initial distribution function for the states of the system in the first period is given.

### Stage $(i)$: Finding The Sets of Thresholds

The heuristic algorithm is designed to decide about relocations much faster than when solving the problem optimally. However, it is important to have reasonable sets of thresholds in the this algorithm. Let $\mathbf{l}$ and $\mathbf{u}$ be the sets of lower and upper bounds, in order to calculate the optimal values for $l_i$s and $u_i$s, the dynamic programming method can be used by defining the following functions:

- $V_t^{''}(W_t, \mathbf{l}, \mathbf{u})$ : The resulting cost of the system from period $t$ on, after the decisions about relocations are made for station $i$ based on $l_i \in \mathbf{l}$ and $u_i \in \mathbf{u}$. The initial value of $V_{T+1}^{''}$ is zero.
- $P(\mathbf{l}, \mathbf{u})$: The cost at period $t$ resulting from the defined minimum cost network flow problem where $l_i \in \mathbf{l}$ and $u_i \in \mathbf{u}$ are the thresholds for station $i$.
- $U_t^{''*}(K_t, D)$ : The resulting cost of the system from period $t$ on, after the decisions about renting cars, $x_{ijt}$s, are made optimally when the relocations are performed based on the threshold type policy. The variable $K_t$ is the current distribution of the cars among the stations and $D$ is a realization of the customer request.

Consequently, we rewrite (4.8) and (4.11) as in below:

$$V_t^{''}(W_t, \mathbf{l}, \mathbf{u}) \;=\; P(\mathbf{l}, \mathbf{u}) + \sum_{all\ possible\ D} P(\mathbf{D} = D) U_t^{''*}(K_t, D), \tag{4.51}$$

and

$$U_t^{''*}(K_t, D) \;=\; \min_{x_{ijt}} \left( (-\sum_{i,j} x_{ijt} r_{ijt}) + V_{t+1}^{''}(W_{t+1}, \mathbf{l}, \mathbf{u}) \right) \tag{4.52}$$

Assuming that the initial distribution function of the states in the system is given, we can evaluate each $\mathbf{l}$ and $\mathbf{u}$ as in below:

$$(\mathbf{l}^*, \mathbf{u}^*) = \arg \min_{0 \le l_i \le u_i \le n} E[V_1^{''}(W_{initial}, \mathbf{l}, \mathbf{u})] \tag{4.53}$$

where the expectation is taken over the initial distribution of the states denoted by $W_{initial}$. In Algorithm 1, the pseudocode for stage $(i)$ is presented for more clarification.

### Stage $(ii)$: Deciding About Relocations

After the sets of thresholds are found, to decide about relocations at the time period $t$, the algorithm calculates $p_{it}$ for the station $i$ based on $l_i$ and $u_i$ as follows:

$$p_{it} = \begin{cases} l_i - w_{it}, & \text{if } w_{it} < l_i \\ 0, & \text{if } l_i \leq w_{it} \leq u_i \\ w_{it} - u_i, & \text{if } w_{it} > u_i \end{cases} \tag{4.54}$$

Then, it neglects those stations with $p_{it} = 0$ and forms a minimum cost network flow problem for the remaining nodes as below:

- The nodes with $p_{it} > 0$ are source nodes.
- The nodes with $p_{it} < 0$ are sink nodes.
- Every source node is connected to every sink node with an arc.
- The cost per flow for arc $(i, j)$ is the cost of relocating a car from the station $i$ to station $j$.
- The lower and upper bounds for the flow of each arc are 0 and $p_{it}$ of the arc's source node, respectively.
- When $\sum_i p_{it}$ does not equal to zero, a phantom source for a negative sum or a phantom sink for a positive sum is added and connected by zero-cost arcs to every node in the network. This is because the conservation flow constraints must always hold in the minimum cost network flow problem.

After solving the minimum cost network flow problem, the cars are relocated based on the calculated flow of each arc. In reality, when we want to relocate the cars, the flow of the phantom source and sink including their arcs are neglected. Thus, the relocations corresponding to the neglected flow are not performed.

**Algorithm 1** The dynamic programming algorithm to find $\mathbf{l}^*$ and $\mathbf{u}^*$.

*Input*: number of cars($n$), number of periods($T$), number of stations($m$), cost matrix($C$), revenue matrix($R$), distribution of customer demand($P(\mathbf{D} = D)$), initial distribution of the states($P(W_{initial} = W)$).
*Output*: the sets of best possible thresholds. ($\mathbf{l}^*$,$\mathbf{u}^*$)
$V''_{T+1}(W_{T+1}, \mathbf{l}, \mathbf{u}) \leftarrow 0$
**for** $0 \leq l_i \leq u_i \leq n$ **do**
    **for** $t = T : -1 : 1$ **do**
        **for all** $W_t$ **do**
            **for all** $D$ **do**
                $U''^*_t(K_t, D) \leftarrow \min_{x_{ijt}} \left( (-\sum_{i,j} x_{ijt} r_{ijt}) + V''_{t+1}(W_{t+1}, \mathbf{l}, \mathbf{u}) \right)$
    **for all** $W_t$ **do**

$$p_{it} \leftarrow \begin{cases} l_i - w_{it}, & \text{if } w_{it} < l_i \\ 0, & \text{if } l_i \leq w_{it} \leq u_i \\ w_{it} - u_i, & \text{if } w_{it} > u_i \end{cases}$$

        Calculate $P(\mathbf{l}, \mathbf{u})$ by the minimum cost network flow problem using $p_{it}$ and $C$.
        $V''_t(W_t, l_i, u_i) \leftarrow P(\mathbf{l}, \mathbf{u}) + \sum_{all\ possible\ D} P(\mathbf{D} = D) U''^*_t(K_t, D)$
$(\mathbf{l}^*, \mathbf{u}^*) \leftarrow \arg\min_{0 \leq l_i \leq u_i \leq n} \sum_W P(W_{initial} = W) V''_1(W, \mathbf{l}, \mathbf{u})]$

# Chapter 5

# Numerical Results

In this chapter, we present the results of our numerical experiments. In Section 5.1, the running time of the deterministic model is analyzed. The stochastic model is studied in Section 5.2 in which we first evaluate the dynamic programming defined by (4.7)- (4.16) for a three-station case and then analyze the two-station case defined in Section 4.3. In conclusion, the proposed heuristic algorithm for a three-station case is evaluated in Section 5.2.3.

## 5.1 Deterministic Model

In this section, we measure the running time of finding the optimal values in the deterministic model proposed in Chapter 3 for different number of cars, stations, and operating periods. The model is implemented in MATLAB R2014b and tested on a laptop with an Intel Pentium(R) N3530 @ 2.16GHz $\times$ 4 Quadcore CPU and 3.7 GiB of RAM. In order to solve the optimization problem, the IBM ILOG CPLEX tool is utilized. The input revenue and cost values are chosen uniformly at random, which are assumed to be in the interval of $[1, 10]$. In addition, the customer demand is generated by taking samples from discrete uniform distribution on the interval of $[1, n]$. Each experiment is done 20 times and the reported time is the average over all running times obtained from the system.

In Table 5.1, the number of cars and stations is fixed, while the number of periods changes. Remember that by increasing the number of periods, we are actually increasing the number of decision variables in the linear programming formulation defined by the relations (3.7)–(3.13). The resulting average values show that $(i)$ it takes only a few milliseconds for the computer to solve the deterministic model and $(ii)$ the average running times slightly change with respect to the number of periods. These observations confirm that the running time of the model is polynomial in the number of stations. However, when the number of periods exceeds 80, our computer runs out of memory because it cannot initialize the network flow formulation vectors in the Cplex solver anymore. Consequently, the solving process fails to start.

| Number of cars=100, Number of stations=10 | |
|---|---|
| Number of periods | Running time(s) |
| 40 | 0.4852856 |
| 50 | 0.72859075 |
| 60 | 1.04222235 |
| 70 | 1.38080965 |
| 80 | 2.00929005 |
| 90 | Out of Memory |

Table 5.1: The average running time for different number of periods.

In Table 5.2, we fix the number of cars and periods, and test the effect of increasing the number of stations on the running time of the system. Similar to the previous case, it takes only a few milliseconds to solve the model and the running time of the system is also polynomial in the number of stations. This is because when the number of stations increases in the system, the number of constraints and decision variables increases in the network flow formulation. Again, due to the memory limitation of our computer, we could not continue increasing the number of stations, arbitrarily.

| Number of cars=100, Number of periods=30 | |
|---|---|
| Number of stations | Running time(s) |
| 1 | 0.03850645 |
| 5 | 0.08710755 |
| 10 | 0.2998784 |
| 15 | 0.8052173 |
| 20 | 1.7709849 |
| 25 | Out of Memory |

Table 5.2: The average running time for different number of stations.

In Table 5.3, it is shown that the running time of the system is almost independent of the number of cars. This is because the computational complexity of the linear programming formulation is not a function of the values of each parameter, but only depends on the number of variables and constraints in the problem.

## 5.2 Stochastic Model

The dynamic programming proposed in Section 4.2 is first evaluated in Section 5.2.1. Moreover, the two station case is tested and Theorems 4.3.2, 4.4.1, and 4.4.2 are verified through some numerical examples in Section 5.2.2. Finally, the heuristic algorithm is analyzed in Section 5.2.3.

| Number of stations=20, Number of periods=30 | |
|---|---|
| Number of cars | Running time(s) |
| 10 | 1.7754501 |
| 50 | 1.7910669 |
| 100 | 1.7709849 |
| 500 | 1.78206435 |
| 1000 | 1.82224765 |
| 10000 | 1.8017887 |

Table 5.3: The average running time for different number of cars.

### 5.2.1 Evaluating the Dynamic Programming Formulation

In order to evaluate our dynamic programming formulation and the sampling method, we use an example of the system with 6 cars, 3 stations and 4 periods where the distribution for customer demand is the following:

- The realization of demand at each station and period can be $\{0, 1, 2\}$.
- The realizations of demand are uniformly distributed.

Furthermore, we assume that the cost and revenue matrices are:

$$[c_{ij}] = \begin{bmatrix} 0 & 2 & 4 \\ 2 & 0 & 3 \\ 4 & 3 & 0 \end{bmatrix}.$$

$$[r_{ij}] = \begin{bmatrix} 10 & 30 & 40 \\ 20 & 50 & 40 \\ 10 & 20 & 50 \end{bmatrix}.$$

For this example, we first calculate the exact value of the optimal expected cost resulting from the system. Remember that based on the defined distribution, we can have $3^9 = 19683$ different realizations of customer demand. Next, we try to use a number of samples from the defined distribution to estimate the value of the expected cost.

Figure 5.1 is the plot of the resulting cost of the system for different number of samples taken uniformly at random from the distribution of the customer demand. In this figure, when we have small numbers of samples (500-5000), the optimal cost of the system changes with high variance and does not necessarily give a good approximation of the exact expected cost. However, as the number of samples is increased (5000-15000), the estimation becomes closer to the exact cost of the system. Note that in general, the samples are important to guarantee an accurate estimation of the expected values in the sampling methods. We leave the study of proper sampling from different distributions in the one way car sharing problem as a future work.

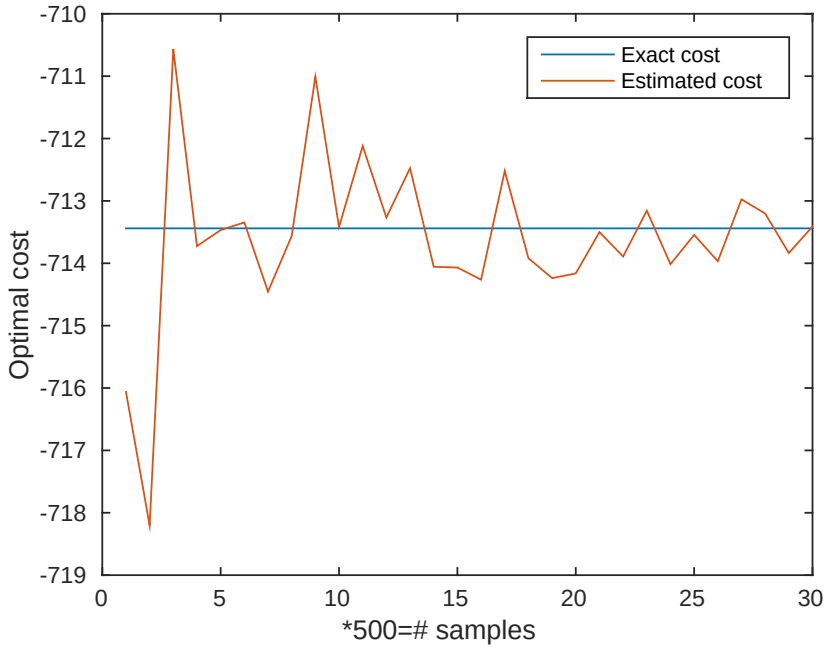


Figure 5.1: Plot of the optimal cost resulting from the system versus the number of samples.

### 5.2.2 Two-Station Case

We first check the properties of the dynamic programming model proposed in Chapter 4 for the two-station case of the car sharing problem. Then, we also study the effect of changing the relocation cost on the structure of the optimal policy. We show that although for deriving our theoretical results, we assume that the decision variables are continuous in the models, our results still hold when the variables are discrete.

#### -Convexity of Objective Functions and the Structure of the Optimal Policy

The convexity of function $V_t$ in Theorem 4.3.2 and functions $U_t^*$ and $V_t^*$ in Theorem 4.4.1 is tested when the variables are discrete. Unfortunately, the function $U_t$ in Theorem 4.3.2 cannot be visualized due to the large number of its input variables. So, plotting its structure is skipped. We also analyze the threshold type structure of the optimal policy (derived in Theorem 4.4.2) for performing the relocations in the two-station case. In Section 4.4, in order to derive the theoretical results, it was assumed that there was an oracle performing the rentals. However, here, this oracle is simulated by manually solving the problem of finding the optimal way of renting the cars at each stage defined by (4.11) and (4.16).

In order to run the experiments, it is assumed that there are 11 cars and 4 periods in the system. Furthermore, with respect to the customer demand distribution, the cost, and revenues, we design the following three test cases. Then, we plot the functions $V_t$, $U_t^*$, $V_t^*$, and the optimal policy in each these test cases.

- **Test case 1**: The customer demand has a Poisson distribution function and the cost and revenues are assumed to be as given below:

$$c = [10]$$

and

$$[r_{ij}] = \begin{bmatrix} 10 & 30 \\ 20 & 35 \end{bmatrix}.$$

- **Test case 2**: The customer demand has a uniform distribution function and the cost and revenues are similar to the first test case.

- **Test case 3**: In this test case, the distribution function for the customer demand is either Poisson or uniform at each period and the cost and revenues in the system are generated uniformly at random at each period.

Moreover, similar to the previous section, we use 1000 two by two matrices as samples, generated randomly from each of the above-mentioned distribution functions for the realizations of the demand.

**Test case 1**

The data used in this test case as the customer demand matrices is generated by independently sampling from a Poisson distribution function with $\lambda = 5$ at each period of time. In Figure 5.2, the function $V_1(y_1)$ is plotted for every state as an example of $V_t(y_t)$. In this figure, the state number is denoted by $i$. Therefore, $i = i_0$ means that there are $i_0$ cars at station 1 and $n - i_0$ cars at station 2. As it can be seen from this figure, even though $V_1(y_1)$ is a discrete function, it has a convex behaviour at all of states and Theorem 4.3.2 holds.

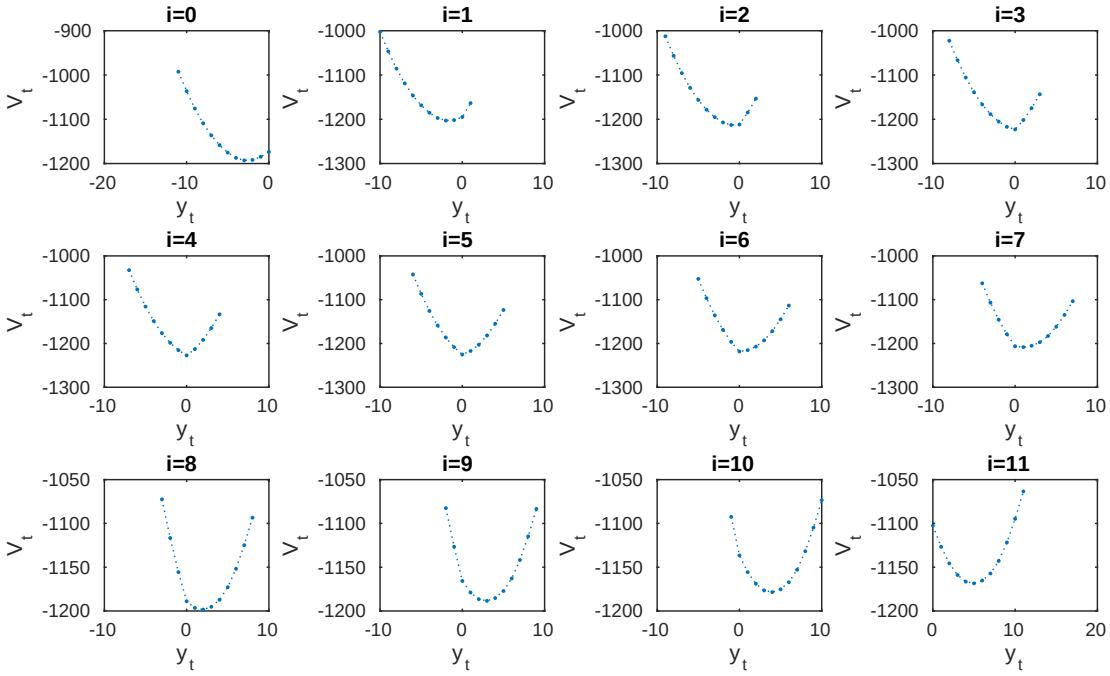


Figure 5.2: Plot of the function $V_t(y_t)$ at the first period, when the customer demand has a Poisson distribution with $\lambda = 5$.

Figure 5.3 and 5.4 are the plots of $V_t^*(w_t)$ and $U_t^*(w_t)$ for every period of time. In these figures, as $w_t$ changes, $V_t^*$ and $U_t^*$ form convex functions, which verifies Theorem 4.4.1.

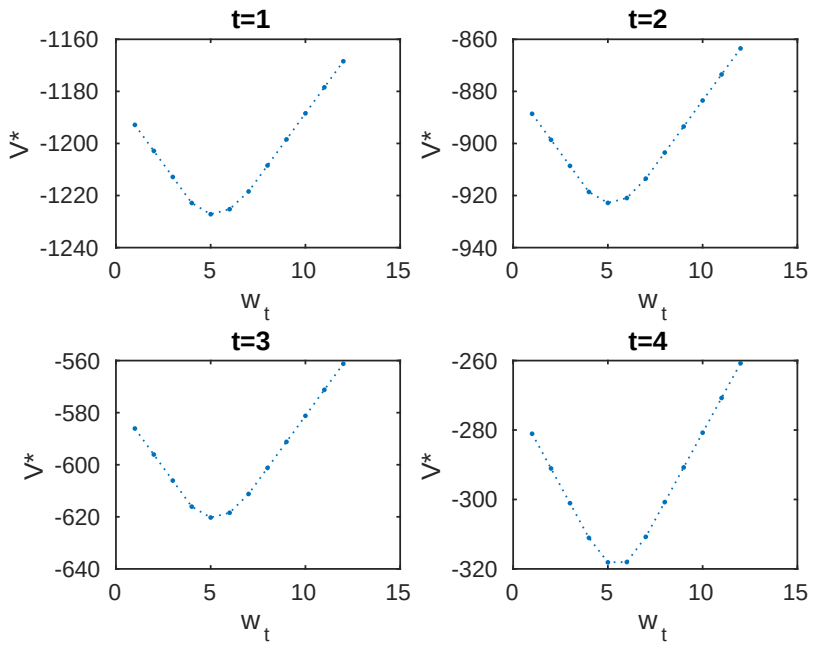


Figure 5.3: Plot of the function $V_t^*(w_t)$ when there are four periods in the system and the customer demand has a Poisson distribution with $\lambda = 5$.

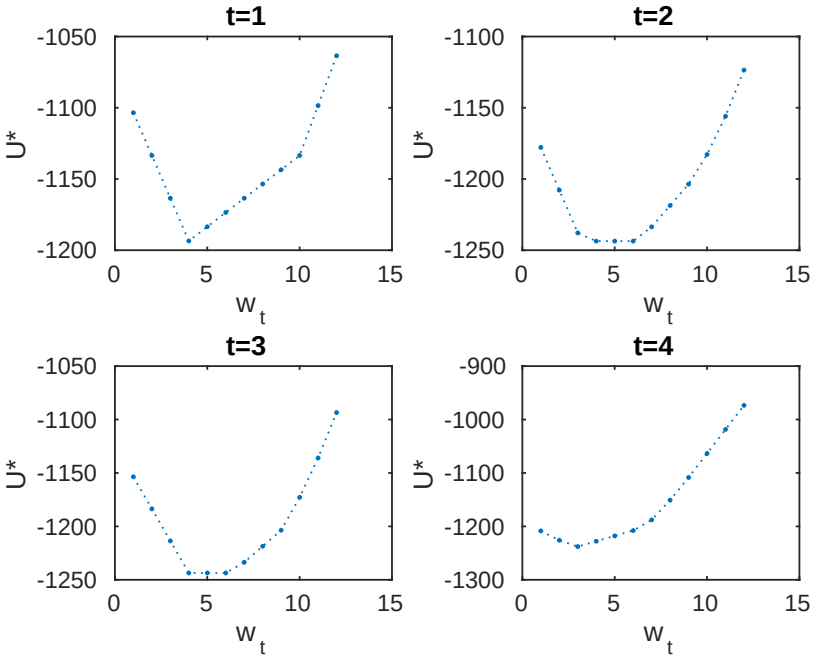


Figure 5.4: Plot of the functions $U_t^*(w_t)$ when there are four periods in the system and the customer demand has a Poisson distribution with $\lambda = 5$.

Finally, Figure 5.5 shows the optimal policy at each period. The threshold type structure that the optimal policy has in this figures at every period verifies Theorem 4.4.2. We can also observe since the size of problem is small, the structure of the optimal policy at every period happened to be the same in Figure 5.5 which does not need to be true for a general case.

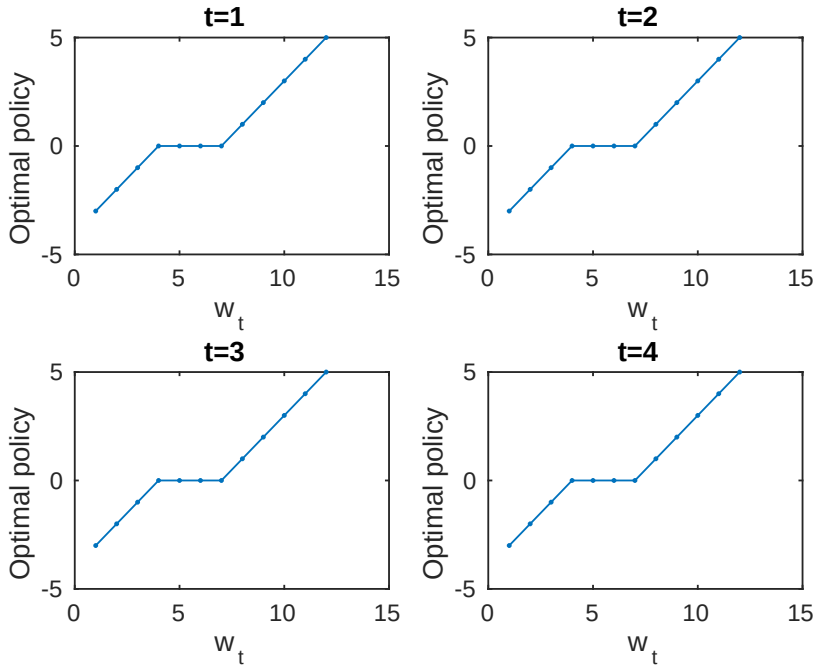


Figure 5.5: Plot of the optimal policy when there are four periods in the system and the customer demand has a Poisson distribution with $\lambda = 5$.

**Test case 2**

We repeat the above experiment with the assumption that the customer demand has a uniform distribution on the interval of $[1, n]$. Similarly, the samples for the customer demand have been taken independently at each period. In Figure 5.6, it is observed that although comparing to the previous test case, the values of $V_1(y_1)$ slightly change, the structure of this function still verifies Theorem 4.3.2.

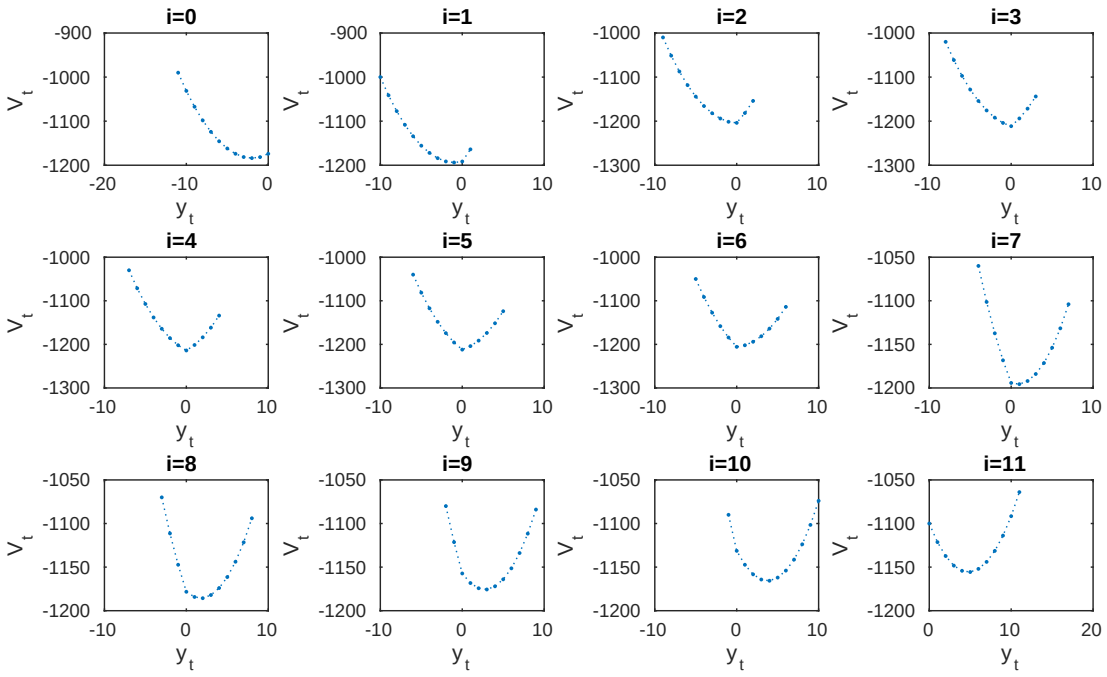


Figure 5.6: Plot of the function $V_t(y_t)$ at the first period, when the customer demand has a Uniform distribution.

In Figures 5.7 and 5.8, we plot the functions $V_t^*(w_t)$ and $U_t^*(w_t)$ for test case 2. Here again, the convex structure of these functions verifies Theorem 4.4.1.

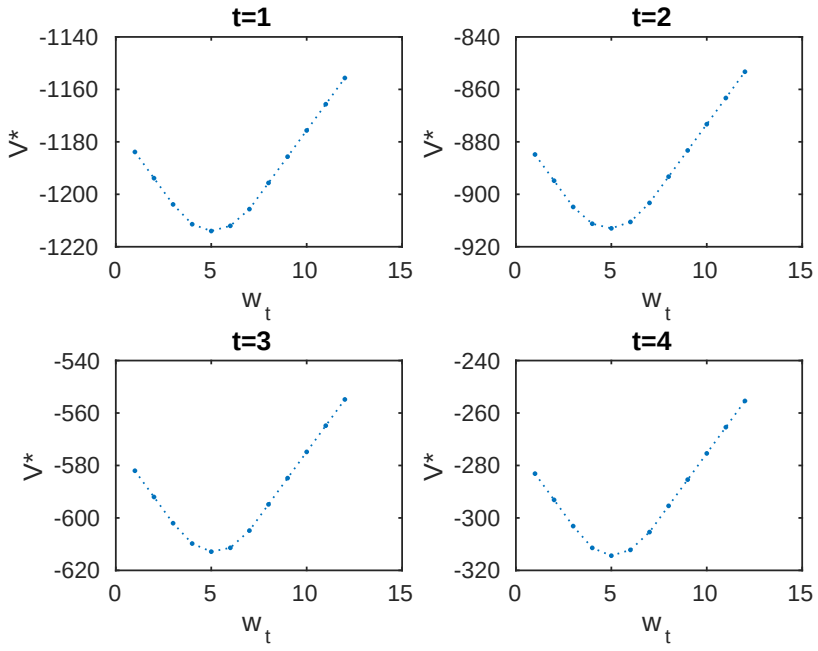


Figure 5.7: Plot of the function $V_t^*(w_t)$ when there are four periods in the system and the customer demand has a Uniform distribution.

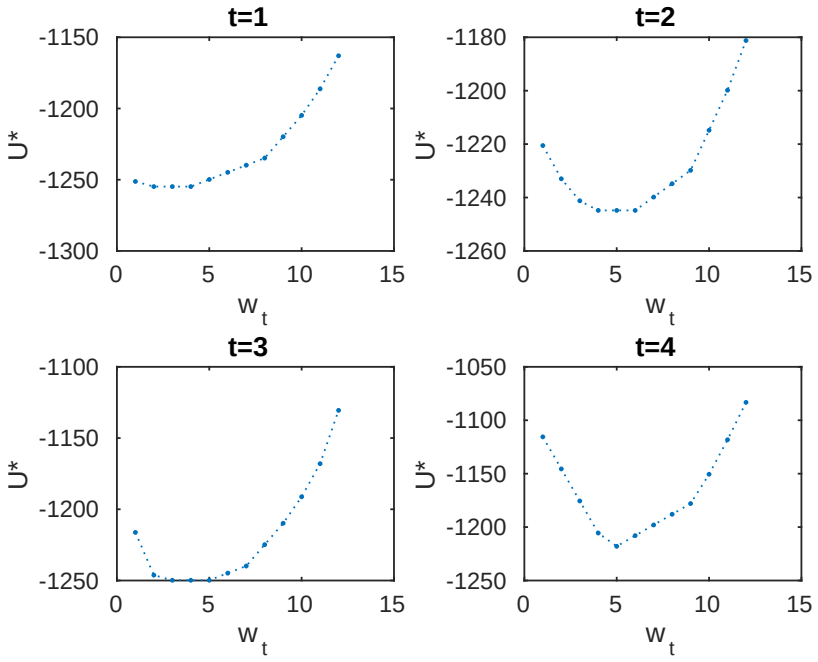


Figure 5.8: Plot of the function $U_t^*(w_t)$ when there are four periods in the system and the customer demand has a Uniform distribution.

Figure 5.9 shows the optimal policy for doing the relocations in test case 2. Similar to Figure 5.5, this figure also confirms Theorem 4.4.2. Similar to the test case 1, due to the small size of the problem, the optimal policy happens to be the same at every period.

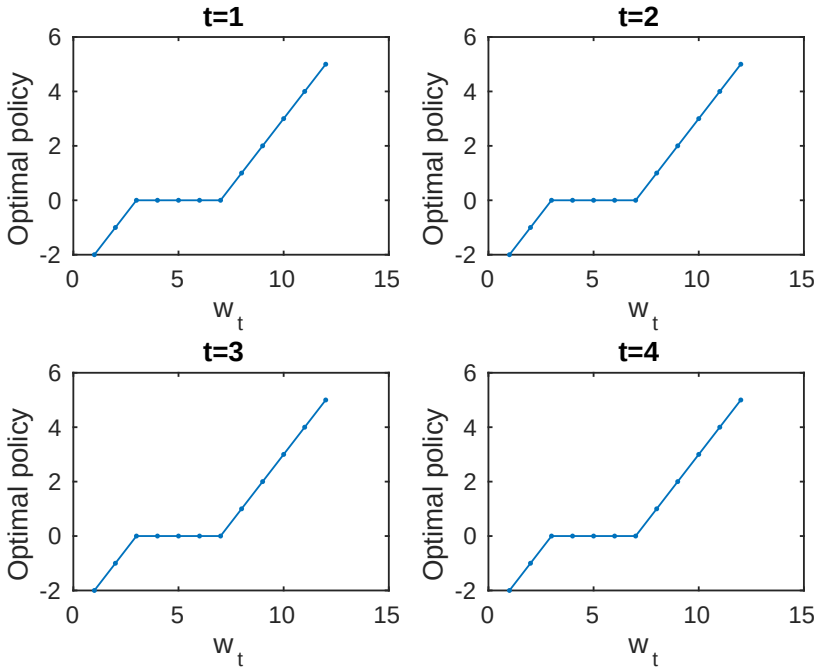


Figure 5.9: Plot of optimal policy when there are four periods in the system and the customer demand has a Uniform distribution.

**Test case 3**

The third experiment is done assuming that the demand distribution function is Poisson with $\lambda = 5$ at period 1, uniform on the interval of $[1, n]$ at period 2, Poisson with $\lambda = 2$ at period 3 and again uniform on the interval of $[1, n]$ at the last period. Furthermore, the cost and revenues in the system are also generated uniformly at random. Figures 5.10, 5.11, and 5.12 show that the changes in the distribution functions, the cost, and the revenue of the system do not influence the structure of the functions $V_t$, $U_t^*$, and $V_t^*$. Therefore, these figures verify Theorems 4.3.2, 4.4.1 as well.

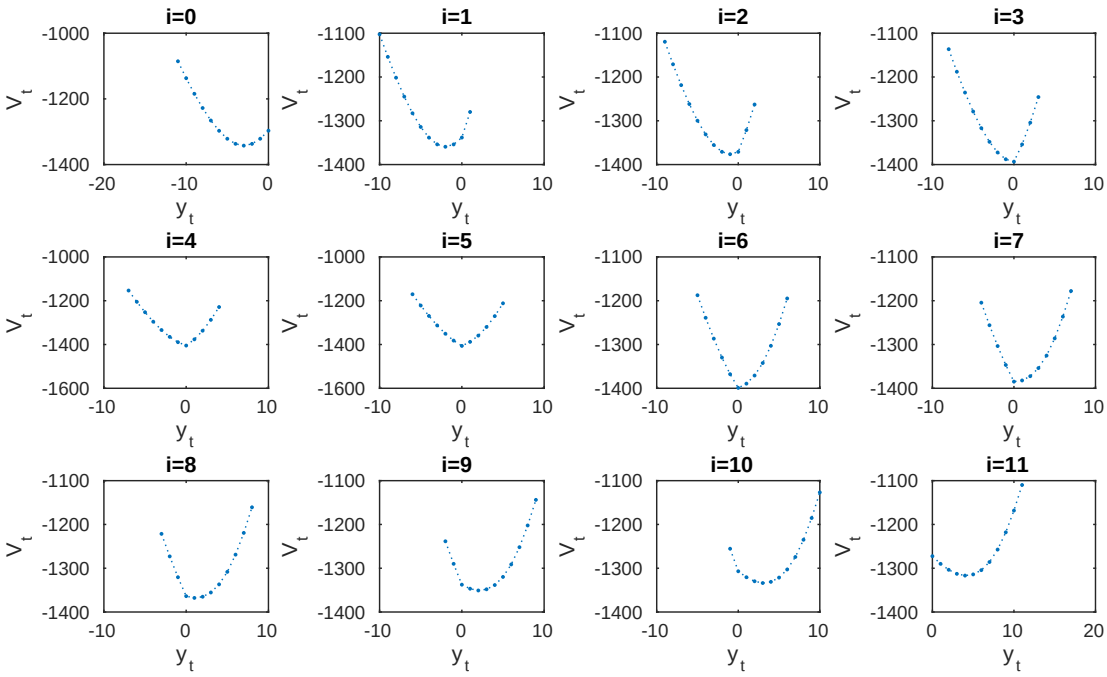


Figure 5.10: Plot of the function $V_t(y_t)$ at the first period, when the customer demand has a different distribution at each period.

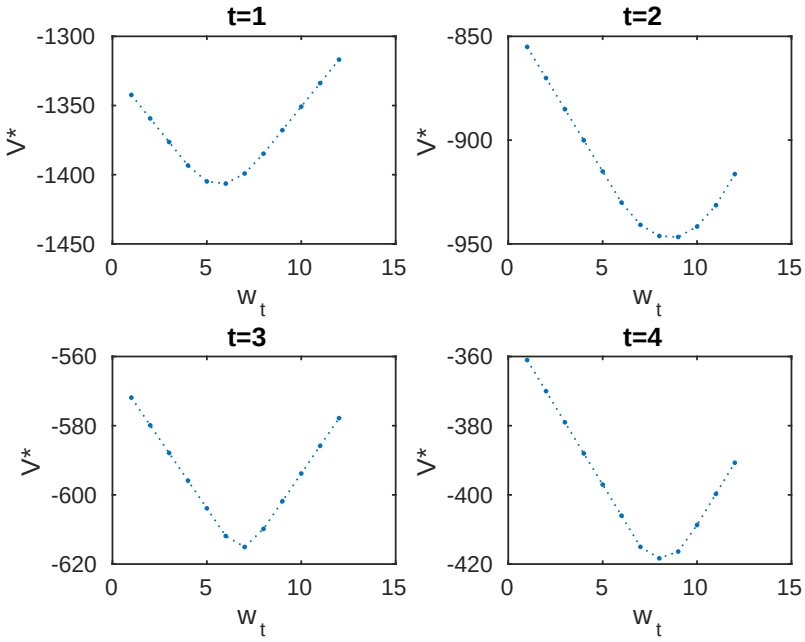


Figure 5.11: Plot of the function $V^*(w_t)$ when there are four periods in the system and the customer demand has a different distribution at each period.

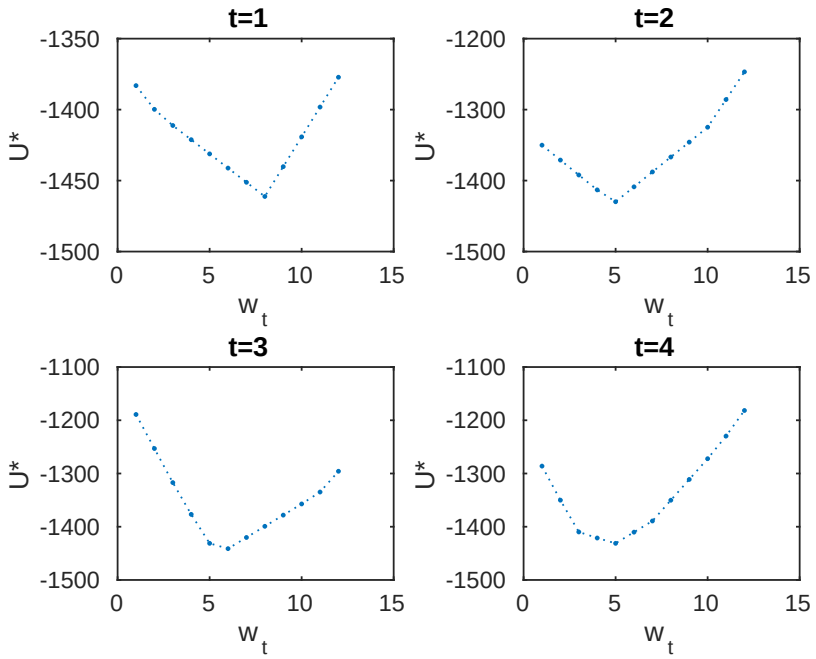


Figure 5.12: Plot of the function $U^*(w_t)$ when there are four periods in the system and the customer demand has a different distribution at each period.

Figure 5.13 indicates the optimal policy at each period in the test case 3. Unlike the previous cases, in this figure, even though the size of problem is still small, since we also change the cost and revenues of the system at each period, the optimal policy slightly changes at each period. Note that the threshold type structure of the optimal policy can still be observed at every period.

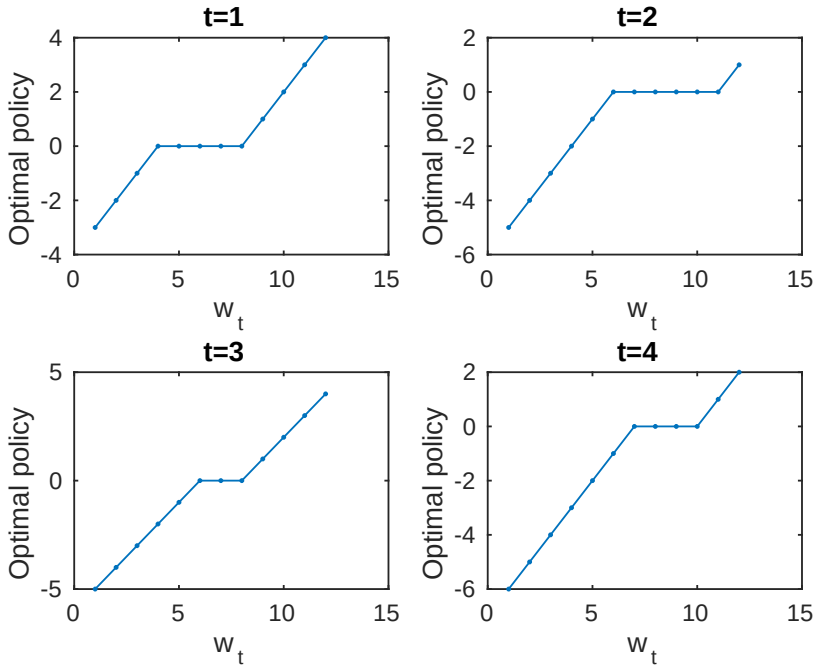


Figure 5.13: Plot of optimal policy when there are four periods in the system and the customer demand has a different distribution at each period.

**-Effect of the Relocation Cost**

In this section, we monitor the effect of changing relocation cost on the structure of the optimal policy in two-station car sharing problem when the are 10 cars in the system and the optimal cost is calculated by the dynamic programming defined in Section 4.2. The cost and revenues are the same as in test case 1, and the data for customer demand is generated by sampling the Poisson distribution with $\lambda = 5$ in this experiment.

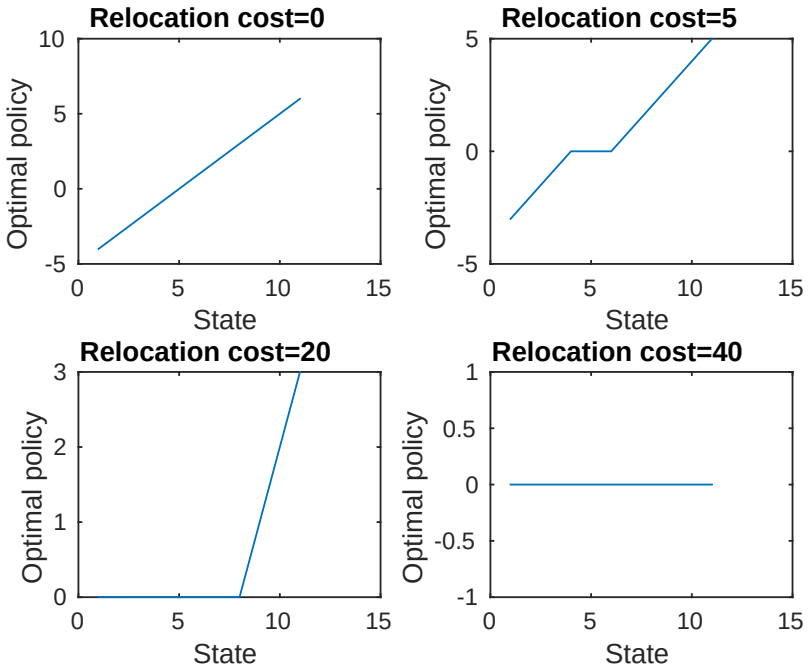


Figure 5.14: Plot of different optimal policies at the first period when the relocation cost changes in the system.

In Figure 5.14, the optimal policy at the first period is plotted for 4 different relocation costs. In this figure, when the relocation cost is zero, the system naturally tends to move some of the cars at every state to a more profitable state. However, as the cost of relocations increases (e.g., the top right plot), the system still tends to relocate cars. However, it also prefers to keep cars at some specific states. When the cost of relocations equals to 20, it can be seen that it is only optimal if we move the cars from the station 1 to 2 because the station 2 is more profitable. Finally, when the cost of relocation becomes larger, the model does not perform any relocation at any state because it does not improve the resulting cost(e.g., the bottom right plot).

### 5.2.3 Evaluating the Heuristic Algorithm

In this section, we examine the accuracy of the resulting cost of our heuristic algorithm compared to the optimal cost obtained by the dynamic programming given in Section 4.2. In addition, we study the improvements that performing relocations cause in the system by comparing the resulting cost of the heuristic algorithm and the optimal cost with the resulting cost of the system when no relocations happen.

The heuristic algorithm was implemented in MATLAB R2014a and tested on a computer with an Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz CPU and 15.46 GiB of RAM. The optimal sets of lower and upper bounds for the heuristic algorithm are calculated from

a system with 6 cars, 3 stations, and 4 periods as examples. These bounds can be later used for other problems with different inputs. Also, the initial distribution of the states is assumed to be uniform and the number of samples used to estimate the expected values in (4.51) is only 100. In order to calculate the best set of thresholds for our heuristic algorithm, every possible set of thresholds should be individually evaluated using the dynamic programming formulation defined in Section 4.5. In order to do that, we need to run the dynamic programming for each fixed set of thresholds and take the average of the resulting values for the first period, $V_1^*$, over the initial distribution of the states. Therefore, we need to run the dynamic program as many times as the number of possible sets of the thresholds to find the optimal one. In Section 4.4, we assumed that there was an oracle who wisely rented the cars. Since this process is done manually in this evaluation, we choose a small number of cars and samples to reduce the computational complexity of the algorithm.

Regarding to the distribution of the customer demand, we define 6 test cases in which the corresponding realizations at each period are sampled from:

- **Test case 1**: a Poisson distribution function with $\lambda = 2$,
- **Test case 2**: a Poisson distribution function with $\lambda = 3$,
- **Test case 3**: a Poisson distribution function with $\lambda = 4$,
- **Test case 4**: a uniform distribution function,
- **Test case 5**: a Poisson distribution function with a random $\lambda$ at odd periods and a uniform distribution function on interval of $[1, 6]$ at even periods.
- **Test case 6**: a Poisson distribution function at every period, where $\lambda$ is chosen uniformly at random from the interval of $[1, 4]$ at each period.

For each of these cases, the corresponding cost and revenue matrices are the following. Since we have a small size problem, to make sure that the relocations happen for several times during the process, the relocation costs are chosen to be small in comparison to the revenues of renting the cars.

$$[c_{ij}] = \begin{bmatrix} 0 & 2 & 4 \\ 2 & 0 & 3 \\ 4 & 3 & 0 \end{bmatrix}.$$

$$[r_{ij}] = \begin{bmatrix} 10 & 30 & 40 \\ 20 & 50 & 40 \\ 10 & 20 & 50 \end{bmatrix}.$$

Table 5.4 illustrates the result of comparing the expected costs of the optimal dynamic programming to the dynamic programming designed in Section 4.5 for calculating the

thresholds in the heuristic algorithm. The costs reported in this table are calculated by taking the average over the resulting cost for every state at period 1, assuming that the initial distribution of the states is uniform. This table shows that when there is only 6 cars and 4 periods in the system, our heuristic algorithm almost results in the optimal expected cost for each state. However, when we use different distribution functions for customer demand at each period, the difference between the accuracy of our heuristic algorithms becomes worse.

| Number of stations=3, Number of cars=6, Number of periods=4 | | | | |
|---|---|---|---|---|
| Distribution | $\lambda$ | Optimal cost | Heuristic cost | Difference |
| Poisson | 2 | -897.8 | -897.6 | 0.2 |
| Poisson | 3 | -1002.3 | -1002.1 | 0.2 |
| Poisson | 4 | -1057.0 | -1056.4 | 0.6 |
| Uniform | - | -1036.2 | -1036.1 | 0.1 |
| Different1 | - | -1026.4 | -1025.3 | 1.1 |
| Different2 | - | -1019.1 | -1007.9 | 11.2 |
| Average | - | -1006.5 | -1004.2 | 2.3 |

Table 5.4: Comparison between the resulting cost of the heuristic dynamic programming and the optimal cost.

Table 5.5 shows the optimal thresholds for each of the stations for all 6 test cases. In this table, it can be observed that the lower and upper bounds for each state are usually the same, which means the system rarely decides to neglect a station in the relocation stages. This is because the costs in the system are smaller than the revenues. So, the systems tend to do more relocations. Also, the number of cars in the system is only 6 such that the range in which the thresholds can change is small.

| Number of stations=3, Number of cars=6, Number of periods=4 | | | | | | |
|---|---|---|---|---|---|---|
| Test case | $l_1$ | $u_1$ | $l_2$ | $u_2$ | $l_3$ | $u_3$ |
| 1 | 2 | 2 | 3 | 3 | 0 | 1 |
| 2 | 2 | 2 | 4 | 5 | 0 | 0 |
| 3 | 0 | 1 | 6 | 6 | 0 | 0 |
| 4 | 1 | 1 | 5 | 5 | 0 | 0 |
| 5 | 1 | 2 | 5 | 5 | 0 | 0 |
| 6 | 1 | 2 | 5 | 5 | 0 | 0 |

Table 5.5: The optimal thresholds for the heuristic algorithm.

After finding the optimal lower and upper bounds for the stations, we run the heuristic algorithms with the calculated thresholds for the 6 test cases assuming that there are 30 periods in system. In Table 5.6, we compare the expected cost of each state for the optimal

dynamic programming, the heuristic algorithm with the calculated thresholds, and the system with no relocations. Our numerical experiments show that:

- Regardless of performing the relocations optimally or based on the heuristic algorithm, they strongly help the system reduce the resulting expected cost.
- Although the thresholds are calculated using a system with only 4 periods, the resulting cost of the heuristic algorithm is still reasonable.
- When the changes in customers demand increases, the performance of the heuristic algorithm decreases(Test case 6).

| Number of stations=3, Number of cars=6, Number of periods=30 | | | | | |
|---|---|---|---|---|---|
| Test case | Initial cost | Optimal cost | Improvement | Heuristic cost | Improvement |
| 1 | -6390.3 | -6732.5 | -342.2 | -6732.5 | -342.2 |
| 2 | -7084.3 | -7451.8 | -367.5 | -7451.4 | -367.1 |
| 3 | -7482.5 | -7941.1 | -458.6 | -7939.6 | -457.1 |
| 4 | -7338.2 | -7777.1 | -438.9 | -7777.1 | -438.9 |
| 5 | -7003.2 | -7415.0 | -411.8 | -7308.2 | -305.0 |
| 6 | -6495.7 | -6903.6 | -407.9 | -6641.4 | -145.7 |
| Average | -6965.7 | -7370.2 | -404.5 | -7308.4 | -342.7 |

Table 5.6: Comparison between the initial, the optimal, and the heuristic cost when there are 30 periods in the system

# Chapter 6

# Conclusions and Future Work

In this thesis, we modeled the day/night one way car sharing system in both deterministic and stochastic frameworks. In the deterministic framework, we studied the optimal way of renting the cars during the day, and relocating them during the nights to minimize the total cost resulting from the system during the operating time. This problem was modeled using integer linear programming. We showed that this problem can be solved in polynomial time because it matched a minimum cost network flow problem.

In the stochastic framework, the problem was modeled by the stochastic dynamic programming method. It was proved that when our decision variables were continuous, the objective functions defined for finding the optimal way of renting and relocating the cars were convex. Next, we proved that in a simple two-station case of the problem, if an oracle wisely rented the cars during the days, a threshold type policy could be used to find the optimal way of relocating the cars during the nights. Based on this result, we proposed a heuristic algorithm to find a way for relocating the cars during the nights for a general $m$-station case. Our numerical experiments showed that the heuristic algorithm resulted in a good approximation of the optimal cost in the system.

The followings are some potential directions that can be used to extend this work:

- One can relax the assumption that each travel takes exactly one day in the system and reformulate the system in order to handle multiple rentals in one day. As a result, the cars returned in the middle of the day can be rented to another customer for the rest of the day, which results in more profit for the system.

- The capacity of each station was not considered as a parameter in our model. In the real world situations, the capacity of stations is limited. Therefore, there might be a case in which the number of customers intending to return the cars to a specific station is more than its capacity. A future direction is to revise the model such that each station has limited capacity.

- In order to make the model closer to real world scenarios, the problem needs to be analyzed and modeled assuming that the operator does not have any control on accepting or rejecting the customer requests during the rental periods and customers are served on a first-come, first-serve basis.
- Our model was based on assuming that the number of periods in the system is finite. One future direction is to analyze the system in the case where the number of periods is infinite.

# Bibliography


[1] Panagiotis Angeloudis, Jun Hu, and Michael GH Bell. A strategic repositioning algorithm for bicycle-sharing schemes. *Transportmetrica A: Transport Science*, 10(8):759–774, 2014.

[2] Matthew Barth, Michael Todd, and Lei Xue. User-based vehicle relocation techniques for multiple-station shared-use vehicle systems. 2004.

[3] Mike Benchimol, Pascal Benchimol, Benoît Chappert, Arnaud De La Taille, Fabien Laroche, Frédéric Meunier, and Ludovic Robinet. Balancing the stations of a self service "bike hire" system. *RAIRO-Operations Research*, 45(01):37–61, 2011.

[4] Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, 1995.

[5] WM Beynon. Combinatorial aspects of piecewise linear functions. *Journal of the London Mathematical Society*, 2(4):719–727, 1974.

[6] Burak Boyacı, Konstantinos G Zografos, and Nikolas Geroliminis. An optimization framework for the development of efficient one-way car-sharing systems. *European Journal of Operational Research*, 240(3):718–733, 2015.

[7] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[8] Alfred Chellanthara. *Evaluating car sharing fleet management strategies using Discrete Event Simulation*. PhD thesis, Concordia University, 2013.

[9] Daniel Chemla, Frédéric Meunier, and Roberto Wolfler Calvo. Bike sharing systems: Solving the static rebalancing problem. *Discrete Optimization*, 10(2):120–146, 2013.

[10] John W Chinneck. Practical optimization: a gentle introduction. *Systems and Computer Engineering), Carleton University, Ottawa. http://www. sce. carleton. ca/faculty/chinneck/po. html*, 2006.

[11] Gonçalo Homem de Almeida Correia and António Pais Antunes. Optimization approach to depot location and trip selection in one-way carsharing systems. *Transportation Research Part E: Logistics and Transportation Review*, 48(1):233–247, 2012.

[12] Cindy Costain, Carolyn Ardron, and Khandker Nurul Habib. Synopsis of users' behaviour of a carsharing program: A case study in toronto. *Transportation Research Part A: Policy and Practice*, 46(3):421–434, 2012.

[13] Angela Di Febbraro, Nicola Sacco, and Mahnam Saeednia. One-way carsharing: Solving the relocation problem. *Transportation research record*, (2319):113–120, 2012.

[14] Wei (David) Fan, Randy B Machemehl, and Nicholas E Lownes. Carsharing: Dynamic decision-making problem for vehicle allocation. *Transportation Research Record: Journal of the Transportation Research Board*, 2063(1):97–104, 2008.

[15] Lester Randolph Ford and Delbert R Fulkerson. *A simple algorithm for finding maximal network flows and an application to the Hitchcock problem*. Citeseer, 1955.

[16] Robert Hlavatỳ. Interpretation of dual model for piecewise linear programming problem.

[17] Diana Jorge and Gonçalo Correia. Carsharing systems demand estimation and defined operations: a literature review. *EJTIR*, 13(3):201–220, 2013.

[18] Alvina GH Kek, Ruey Long Cheu, Qiang Meng, and Chau Ha Fung. A decision support system for vehicle relocation operations in carsharing systems. *Transportation Research Part E: Logistics and Transportation Review*, 45(1):149–158, 2009.

[19] Richard Kipp Martin. *Large scale linear and integer optimization: a unified approach*. Springer Science & Business Media, 2012.

[20] William J Mitchell. *Reinventing the automobile: Personal urban mobility for the 21st century*. MIT press, 2010.

[21] Rahul Nair and Elise Miller-Hooks. Fleet management for vehicle sharing operations. *Transportation Science*, 45(4):524–540, 2011.

[22] Mehdi Nourinejad and Matthew J Roorda. A dynamic carsharing decision support system. *Transportation Research Part E: Logistics and Transportation Review*, 66:36–50, 2014.

[23] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[24] Marco Pavone, Stephen L Smith, and Emilio Frazzoli Daniela Rus. Load balancing for mobility-on-demand systems. *Robotics: Science and Systems VII*, page 249, 2012.

[25] Tal Raviv and Ofer Kolka. Optimal inventory management of a bike-sharing station. *IIE Transactions*, 45(10):1077–1093, 2013.

[26] Tal Raviv, Michal Tzur, and Iris A Forma. Static repositioning in a bike-sharing system: models and solution approaches. *EURO Journal on Transportation and Logistics*, 2(3):187–229, 2013.

[27] Susan Shaheen, Daniel Sperling, and Conrad Wagner. Carsharing in europe and north american: past, present, and future. 1998.

[28] Susan A Shaheen and Adam P Cohen. Growth in worldwide carsharing: An international comparison. *Transportation Research Record: Journal of the Transportation Research Board*, 1992(1):81–89, 2007.

[29] Susan A Shaheen, Adam P Cohen, and Melissa S Chung. North american carsharing. *Transportation Research Record: Journal of the Transportation Research Board*, 2110(1):35–44, 2009.

[30] Jia Shu, Mabel C Chou, Qizhang Liu, Chung-Piaw Teo, and I-Lin Wang. Models for effective deployment and redistribution of bicycles within public bicycle-sharing systems. *Operations Research*, 61(6):1346–1359, 2013.

[31] David K Smith. *Dynamic programming: a practical introduction*. Prentice Hall, 1991.

[32] Tai Stillwater, Patricia L Mokhtarian, and Susan A Shaheen. Carsharing and the built environment. *Transportation Research Record: Journal of the Transportation Research Board*, 2110(1):27–34, 2009.

[33] Kentaro Uesugi, Naoto Mukai, and Toyohide Watanabe. Optimization of vehicle assignment for car sharing system. In *Knowledge-based intelligent information and engineering systems*, pages 1105–1111. Springer, 2007.

[34] Laurence A Wolsey. *Integer programming*, volume 42. Wiley New York, 1998.