ABSTRACT

Title of Dissertation: DECENTRALIZED TRANSPORTATION MODEL IN VEHICLE SHARING

Ying Li
Doctor of Philosophy, 2023

Dissertation Directed by: Professor Ilya Ryzhov
Department of Decision, Operations,
and Information Technologies

This dissertation introduces the concept of decentralization to address the rebalancing challenges in bike-sharing systems and proposes a model known as the Decentralized Route Assignment Problem (DRAP) under specific assumptions. The primary contributions of this research include the formulation of the DRAP and the derivation of theoretical results that facilitate its transformation into a lower-dimensional global optimization problem. This transformation enables efficient exploration using modern search methods. An extended version of DRAP, called DRAP-EA, is also proposed for further analysis by introducing more agents into the system.

Various solution approaches, such as branch-and-cut, hill-climbing, and simulated annealing, are explored and customized to enhance their performance in the context of rebalancing. Two simulated annealing methods, Gurobi with warm-start, and an extension of the local search algorithm are implemented on 24 instances derived from a comprehensive

case study for experimental evaluation. The experimental results consistently demonstrate the superior performance of the simulated annealing methods. Furthermore, a comparison between SA and SA-PS is conducted, and the obtained solutions are visualized to help further explore the spatial patterns and traffic flows within the bike-sharing system.

Decentralized Transportation Model In Vehicle Sharing

by

Ying Li

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2023

Advisory Committee:
Professor Ilya O. Ryzhov, Chair/Advisor
Professor Vincent Lyzinski
Professor Bruce Golden
Professor Kunpeng Zhang
Professor Alexander Estes

Preface

This dissertation represents a compilation of my PhD research, encompassing a diverse range of results and methods that could be seen as contributions to the field. While not all findings were ultimately utilized, I believe that even unused discoveries may hold value in the future. I consider this dissertation to be a resource, comparable to a dictionary, where various topics may be briefly mentioned but still offer guidance to future researchers. Even may be viewed as research failures, I believe these findings can serve as valuable lessons, assisting others in avoiding unproductive paths in the field.

And, ultimately, done is better than perfect.

Dedication

To my past self, who was once pessimistic about her future, who endured every painful point in life, and whose confusion seemed never-ending.

I dedicate this dissertation to you, for your resilience to stand up and pursue the light in every darkest moment, for your determination that lifted your spirits and finally found your lost heart. I dedicate all of this work to you, for your unwavering gratitude towards your sufferings, no matter how challenging they may have been.

Acknowledgments

I want to express my gratitude to everyone who has helped me, with or without awareness.

First and foremost, I am thankful to my advisor, Professor Ilya Ryzhov, for accepting me as his student. When I was searching for an advisor in business school, I had no knowledge of operations research and business. But he believed in me and my abilities, even with my different background. He gave me the freedom to explore topics without restrictions, and I appreciate his guidance.

I also want to thank Professor Ashish Kabra for providing the research topic and sharing valuable business insights along the way. He has always been kind and supportive, and some of his ideas have helped shape my research.

I would like to thank all the members of my committee, both the dissertation committee and the preliminary committee. Their support throughout my research years has been instrumental in ensuring a smooth path towards my graduation.

I am grateful to Jessica Sadler, the AMSC program coordinator, for making everything official easy for us. She has been attentive and knowledgeable, ensuring our smooth years in the program.

I want to acknowledge the AMSC committee for accepting me into the program and for their attention to detail in document submissions. Although my proposals were rejected

several times, I learned something from each rejection, without which the writing of my dissertation would be less smoother.

I would like to thank the math department, where I had a wonderful time. The staff was helpful and supportive, and I enjoyed all the courses I took. It is truly a remarkable department.

I owe my gratitude to the fellow students I befriended in the program. Our discussions about research helped me gain clarity, and their encouragement kept me going during challenging times.

Also, I would like to thank the birds singing in the woods, the winds that blow, the rains that fall from the clouds, the grass that grows out of the mud, and the sun that shines upon the ground. I am thankful for the flowing water, the glowing light, the tranquil rock, and the steady heartbeat within. Everything holds its unique beauty, and I am grateful for their presence.

Lastly, Thank you all for shaping me and the world.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

B&C	Branch-and-cut
DRAP	Decentralized Route Assignment Problem
ELS	Extended Local Search
GRB	Gurobi
MIP	Mixed-integer Program
MIQP	Mixed-integer Quadratic Program
SA	Simulated Annealing
SA-PS	Simulated Annealing with Probabilistic Shaking
QP	Quadratic Program

Chapter 1: Introduction

1.1 Business context: bike-sharing systems

Bike-sharing systems have become an increasingly popular mode of transportation in urban areas around the world, offering a low-cost, convenient, and eco-friendly alternative to traditional modes of transportation. As of August 2022, there were 1,914 bike-sharing systems in 1,590 cities worldwide, with over 26 million bikes available for rent [1].

There are two main types of bike-sharing systems: station-based and dockless. Station-based systems were the first to be introduced, requiring fixed locations and capacities for each station. While they provide an organized and predictable system that is easy for users to navigate, they also require significant investment in infrastructure and maintenance. Dockless systems, on the other hand, allow users to park bikes anywhere within a designated area, making them more flexible and convenient but also more challenging to manage and maintain.

To meet the needs of bike-sharing users, it is essential to effectively manage and maintain the supply of bikes at each station to match user demand. This process, known as rebalancing, involves redistributing bikes from stations with a surplus to those with a deficit. Fricker et al. [2] studied the dynamics by applying mean-field theory to some heterogeneous bike-sharing systems. Traditionally, bike-sharing companies rebalanced their

bikes by employing drivers to operate trucks that can accommodate a certain number of vehicles. However, with the rise of the gig economy in recent years, bike-sharing companies such as Bird and Lime have adopted a decentralized approach by hiring independent contractors, known as gig workers, to reposition bikes in real-time [3].

These companies hire a certain number of rebalancers and post rebalancing tasks on their map at a specific time of day, usually in the evening. The map shows the locations of bikes that need to be repositioned, along with a price label. Another map shows the locations with high demand for bikes. Rebalancers pick up bikes from supply locations and drop them off at demand locations according to the demand map, and they are paid the price shown in the supply map per bike. This approach allows bike-sharing companies to optimize the rebalancing process in real-time, reducing the need for costly and time-consuming truck-based operations. Additionally, gig workers benefit from the flexible and dynamic nature of the job, enabling them to earn money on a task-by-task basis. Overall, this decentralized approach has the potential to enhance the efficiency and sustainability of bike-sharing systems while creating new opportunities for gig workers in the sharing economy.

Motivated by this strategy, our research employs a stylized model where each agent is assigned the task of moving a single bike. Our rebalancing strategy aligns with the decentralized nature of the gig-based setting, where agents cannot be directly assigned specific tasks but can be influenced through pricing mechanisms. Although our model simplifies the scenario, it captures the essential decentralized aspects of the system.

1.1.1 Related research

Over the years, extensive research has been conducted on bike-sharing systems. Laporte et al. provide a comprehensive survey of the main operational research issues and corresponding methods in vehicle sharing systems in their works [4] and [5]. In the bike-sharing literature, we identify three main topics: demand analysis, strategic planning, and rebalancing operations. These topics are explored using various approaches, including empirical studies and theoretical modeling.

In the field of demand analysis, researchers examine the patterns of demand and investigate the factors that influence the service usage, aiming to provide guidelines for the operation of bike-sharing services. Zou et al. [6] analyze dockless e-scooter trip trajectories in the Washington DC area using real-time data, offering insights into route choices. Kabra et al. [7] conduct an analysis that reveals a numerical relationship between bike-sharing usage and station availability for customers. Goh et al. [8] develop a locational demand model with distance ranking parametrization to account for substitution behaviors based on proximity. Eren and Uz [9] provide a practical perspective and address the impact of factors such as weather conditions, urban environment, station-level design, and temporal considerations on service usage.

The literature on strategic planning explores optimal strategies for managing the system to maximize its benefits. Lin and Yang [10] address the station location problem by formulating a non-linear integer program that considers user flow, station locations, and bicycle lanes. Fricker and Gast [11] propose a queueing-based approach to determine the desirable fleet size, aiming to reduce the number of stations without bikes or available

slots. Datner et al. [12] focus on inventory level management and develop an approach that minimizes expected dissatisfaction over the planning horizon.

In terms of rebalancing, Faghih-Imani et al. [13] perform station-level analysis using data from Barcelona and Seville, Spain, and propose a logistic model to estimate the amount of rebalancing. As for operations, Various models for rebalancing are developed based on different strategies. These strategies are often categorized into truck-based routing and user incentives. Truck-based routing involves manually redistributing bikes between stations using trucks or vans and is commonly modeled as a one-commodity vehicle pickup-and-delivery vehicle routing problem, as proposed by Hernandez-Perez and Salazar-Gonzalez in [14]. Osorio et al. [15] provide a more specific modeling approach where e-scooters are allowed to be charged while being transported. On the other hand, the user-incentive strategy aims to encourage users to ride bikes from over-capacity stations to under-capacity stations. Fricker and Gast [11] study the pricing strategy by proposing a two-choice (of stations) model and show that the strategy helps improve the performance even if only a small proportion of users choose the less-loaded station. Furthermore, researchers have investigated special programs, such as volunteer-based rebalancing programs, as studied by Huang et al. [16].

To the best of our knowledge, the agent-based rebalancing strategy discussed in the previous subsection has not been explored in prior research. Different from the truck-based approach, the agent-based strategy considers the problem in a decentralized way. Unlike the truck-based approach, the agent-based strategy takes a decentralized approach to address the problem. While the user-incentives approach also adopts a decentralized perspective, it balances between daily usage and repositioning, whereas agent-based rebalancing focuses

solely on achieving rebalance objectives.

1.1.2 Pricing models

In our research, we introduce a decentralized pricing structure for rebalancing in bike-sharing systems. Pricing models commonly used in shared vehicle services include dynamic pricing and spatial pricing. Dynamic pricing has been proven beneficial in ride-sharing platforms, as confirmed by Cachon et al. [17]. It involves adjusting prices based on real-time demand and supply conditions. On the other hand, spatial pricing focuses on pricing strategies based on different regions. Bimpikis et al. [18] examine spatial pricing in ride-sharing systems while analyzing the demand pattern in the network. They demonstrate that the effectiveness of this strategy depends on the balance of demand across different regions. Banerjee et al. [19] propose an elevated flow relaxation of the non-convex pricing problem as a basis for a general approximation framework to design pricing policies in shared vehicle systems.

The pricing structure proposed in our research aims to optimize operations in decentralized settings, and we refer to this pricing as decentralized pricing. It is designed to incentivize efficient behavior and coordination among agents in the decentralized rebalancing operation in the bike-sharing system.

1.2 Decentralized models

Decentralized networks have gained significant attention in today's business landscape, particularly with the emergence of gig workers in the market over the past decades.

The idea of decentralization is closely connected to the principal-agent problem in microeconomics, where the decisions made by agents are considered by the principal decision-maker. Sternberg and Andersson [20] give a research review on decentralized intelligence in freight transport and measure its necessity in supply chain management by incorporating the literature. Demirag and Swann [21] examine decentralized logistics networks in the sea cargo industry and compare their performance with that of centralized networks. The authors recognize the NP-hard nature of the multi-route decentralized problem and propose several heuristic methods to obtain upper bounds. They conclude that the performance gap between the decentralized and centralized solutions relates to the performance gap between the upper bound and the centralized solution. Decentralized models are also discussed in the ride-sharing business where the matching between drivers and passengers is based on auctions, as proposed by Kleiner et. al in [22]. Nourinejad and Roorda [23] present an agent-based model that could match single or multiple drivers with single or multiple passengers.

It is worth noting that when optimization problems involve agent decisions, they are often formulated as bilevel problems, which can be difficult to solve. To address this challenge, a common approach is to reformulate the bilevel problem as a single-level optimization problem, as the approach adopted by Aslan et. al in [24]. Fortunately, our model can be transformed into a single-level problem, which simplifies the optimization process.

1.3 Dissertation outline

Chapter 2 of this dissertation presents the formulation of the model. Initially, it is formulated as a Mixed-Integer Quadratic Programming (MIQP) problem, but by exploiting structural properties that we identify, we can simply it into a Mixed-Integer Linear Program (MILP). Furthermore, we introduce and prove several theoretical results that transform the MILP into a combinatorial optimization problem. Furthermore, we develop an efficient algorithm to carry out a portion of the optimization process without the need for solvers. The proof of its effectiveness is provided in the chapter as well. Additionally, we extend the model with corresponding optimization problems by making slight modifications to the assumptions, allowing us to gain further insights into the formulation.

Chapter 3 focuses on exploring computational methods to address the Decentralized Rebalancing and Pricing (DRAP) problem introduced in Chapter 2. We discuss both exact and heuristic methods. The exact methods include branch-and-cut, which theoretically solve the MILP formulation precisely. On the other hand, the heuristic methods involve derivative-free searching techniques such as hill-climbing local search, simulated annealing, and genetic algorithms. We thoroughly examine these methods and propose potential refinements to their underlying formulations. Notably, we highlight certain methods that will be employed in the subsequent chapter for experimental purposes.

In Chapter 4, we conduct a comparative analysis of the most efficient algorithms identified in the previous chapter through a comprehensive case study. To do this, we generate numerical instances by making up a realistic business situation that utilizes real-world data. Subsequently, we conduct extensive numerical experiments using these methods to

evaluate their performance. Additionally, we present visualizations of the solutions on a map to gain a comprehensive understanding of the characteristics of the heuristic solutions.

Finally, in the concluding chapter, we summarize the key findings of the entire dissertation and provide potential directions for future research and discussion.

Chapter 2: Models and related results

2.1 Overview

In this chapter, we present the formulation of an optimization problem for agent-based rebalancing operations. We propose multiple versions of the problem with a gradual reduction in complexity. Additionally, we provide theoretical results that offer a new perspective, greatly simplifying the problem. Furthermore, we introduce extensions of the models that allow for further evaluations.

2.2 Model formulation

We investigate the model setup of our research by analyzing the dynamics that arise when a team of independent contractors, known as rebalancers, are hired by the platform to operate within the bike-sharing system. The platform assigns tasks by setting prices for each origin-destination pair and labeling the demand and supply for each location, which the rebalancers will act upon. We assume that each contractor can only carry one bike from their pick-up location, transporting it to the designated drop-off point and receiving the agreed-upon compensation. In this proposal, the assignment process occurs within a single time round, disregarding the time aspect. In this framework, rebalancers act au-

tonomously and the platform guides their behavior only indirectly by setting prices. This arrangement enables the rebalancers to determine their workflow, creating a decentralized system where the platform does not control the assignment of tasks. Notice that the model we are formulating is a one-time operation that does not include the factor of time.

2.2.1 MIQP formulation

Let $i, j \in \{1, ..., N\}$ represent regions, which can be either stations in the station-based system or spots where shared bikes accumulate in the dockless setting. The distribution of vehicles to be picked up and dropped off is denoted by the vectors $\boldsymbol{o} = (o_i)$ and $\boldsymbol{t} = (t_i)$, respectively. The conservation equation $\sum_i o_i = \sum_j t_j$ ensures that the number of vehicles to be picked up is equal to that to be dropped off. Let $\ell \in \{1, ..., L\}$ represent the rebalancers in the system. The model assumes that each rebalancer rebalances one bike, and all bikes will be rebalanced, which indicates that $L = \sum_i o_i$. Later, in Section 2.3, we will also consider the situation where there is a surplus of rebalancers, i.e., $L > \sum_i o_i$. Let $c_{\ell ij}$ denote the personalized cost associated with rebalancer ℓ moving a bike from location i to location j. This personalized cost can vary based on various factors that are specific to each rebalancer, such as their preferences for the time taken, travel distance, road conditions, and other relevant factors. The personalized cost can be determined using a personalized utility function that reflects the preferences of the rebalancers. Ultimately, the cost is expressed as a value associated with different origin and destination pairs.

The binary decision variable $x_{\ell ij}$ indicates whether or not the rebalancer ℓ chooses to take a route from region i to region j in the decentralized system. The vector $\boldsymbol{x}$ describes

the *assignment* of all available tasks.

Based on the aforementioned assumptions, the following equations can be derived:

$$\sum_i \sum_j x_{\ell ij} = 1 \quad \forall \ell$$

$$\sum_j \sum_\ell x_{\ell ij} = o_i \quad \forall i$$

$$\sum_i \sum_\ell x_{\ell ij} = t_j \quad \forall j$$

The first condition signifies the constraint that each rebalancer can transport only one bike at a time. The second and third conditions represent the supply and demand requirements for bikes, respectively. In addition, we assume that the personalized cost matrix $c_\ell = (c_{\ell ij})$ for each rebalancer ℓ is known to the platform.

Though the decision variables $x_{\ell ij}$ are explicitly modeled in our framework, these decisions are made by rebalancers. The decision made by the platform consists of prices $p_{i,j}$ that the platform is willing to pay rebalancers for moving one bike from region i to j. These prices are independent of ℓ, so each rebalancer sees the same set of prices. In addition, a rebalancer will only be willing to participate in the rebalancing system and undertake the corresponding task if the payment exceeds their personal cost. This condition can be expressed as:

$$p_{ij} - c_{\ell ij} \geq 0 \quad \text{when } x_{\ell ij} = 1 \quad \forall \ell, i, j \tag{2.2.1.1}$$

or, alternately,

$$p_{ij} \geq c_{\ell ij} x_{\ell ij} \quad \forall \ell, i, j$$

In the model, we assume that a rebalancer selects the most profitable route, where higher profits indicate greater desirability. Mathematically, if we define $(i(\ell), j(\ell))$ as the origin-destination pair chosen by rebalancer ℓ, the assumption can be expressed as:

$$p_{i(\ell)j(\ell)} - c_{\ell i(\ell)j(\ell)} = \max_{(i,j)}\{p_{ij} - c_{\ell ij}\} \quad \text{when } x_{\ell i(\ell)j(\ell)} = 1 \quad \forall \ell.$$

This condition resembles a type of principal-agent problem formulation in microeconomics, and unlike other formulations, our condition can be expressed as a set of inequalities implying that the chosen route yields the highest profit compared to all other options:

$$p_{ij} - c_{\ell ij} \geq p_{i'j'} - c_{\ell i'j'} \quad \text{when } x_{\ell ij} = 1 \quad \forall \ell, i, j, i', j'. \tag{2.2.1.2}$$

Then we use the big-M technique to eliminate the logical element:

$$p_{ij} - p_{i'j'} \geq (c_{\ell ij} - c_{\ell i'j'})x_{\ell ij} - M(1 - x_{\ell ij}) \quad \forall \ell, i, j, i', j',$$

where the constant M satisfies $M \geq \max_{i,j} p_{ij}$. The pricing that satisfies the aforementioned property for a given assignment $\boldsymbol{x} = (x_{\ell ij})$ is referred to as *decentralized pricing*.

Ultimately, the platform aims to minimize its cost by considering all feasible decentralized pricing options. This objective is represented by the function $\sum_{\ell,i,j} p_{ij}x_{\ell ij}$, which calculates the total cost incurred by the platform based on the chosen prices and assignment of rebalancers to bike movements.

With all the preparations described above, we can now outline the optimization prob-

lem as follows:

$$\min_{\mathbf{p},\mathbf{x}} \quad \sum_{\ell}\sum_{i}\sum_{j} p_{ij}x_{\ell ij} \tag{2.2.1.3a}$$

$$\text{s.t} \quad \sum_{j}\sum_{\ell} x_{\ell ij} = o_i \quad \forall i \tag{2.2.1.3b}$$

$$\sum_{i}\sum_{\ell} x_{\ell ij} = t_j \quad \forall j \tag{2.2.1.3c}$$

$$\sum_{i}\sum_{j} x_{\ell ij} = 1 \quad \forall \ell \tag{2.2.1.3d}$$

$$p_{ij} \geq c_{\ell ij}x_{\ell ij} \quad \forall \ell, i, j \tag{2.2.1.3e}$$

$$p_{ij} - p_{i'j'} \geq (c_{\ell ij} - c_{\ell i'j'})x_{\ell ij} - M(1 - x_{\ell ij}) \quad \forall \ell, i, j, i', j' \tag{2.2.1.3f}$$

$$x_{\ell ij} \in \{0, 1\} \quad \forall \ell, i, j \tag{2.2.1.3g}$$

$$p_{ij} \geq 0 \quad \forall i, j \tag{2.2.1.3h}$$

Since the objective function is quadratic (product of decision variables) and the constraints are linear, the formulation is a mixed-integer quadratic programming problem.

2.2.2 MIQP linearization

To proceed with the formulation, we can linearize the quadratic problem by employing a common technique known as variable substitution. In this approach, we introduce additional variables $z_{\ell ij}$ to replace the quadratic term $p_{ij}x_{\ell ij}$. Therefore, we have the following formulation:

$$\min_{\mathbf{p},\mathbf{x}} \quad \sum_{\ell}\sum_{i}\sum_{j} z_{\ell ij} \tag{2.2.2.1a}$$

s.t $(2.2.1.3b) - (2.2.1.3h)$

$$z_{\ell ij} \leq M x_{\ell ij} \quad \forall \ell, i, j \tag{2.2.2.1b}$$

$$z_{\ell ij} \leq p_{ij} \quad \forall \ell, i, j \tag{2.2.2.1c}$$

$$z_{\ell ij} \geq p_{ij} - M(1 - x_{\ell ij}) \quad \forall \ell, i, j \tag{2.2.2.1d}$$

$$z_{\ell ij} \geq 0 \quad \forall \ell, i, j \tag{2.2.2.1e}$$

The last four constraints enforce the relationship between the original variables p_{ij} and $x_{\ell ij}$ and the substitution variables $z_{\ell ij}$, ensuring that $z_{\ell ij} = p_{ij} x_{\ell ij}$. This substitution works in this case because it is the product of a binary variable and a continuous variable. The validity of this substitution can be justified by separating the case into two scenarios: $x_{\ell ij} = 0$ and $x_{\ell ij} = 1$. As a result, (2.2.1.3a) and (2.2.2.1a) are equivalent. Due to the maturity of linear programming methods compared to quadratic programming methods, the linearized formulation (2.2.2.1a) is considered more favorable than the original formulation (2.2.1.3a) in terms of computational efficiency, despite the increased number of variables and constraints.

2.2.3 MILP reformulation

The MIQP (2.2.1.3a) represents a natural formulation of the model and incorporates two key aspects: how rebalancers make job choices individually and how the platform globally sets prices on the map. However, when considering the perspective of rebalancers, their compensation can be divided into two components: the cost incurred and the surplus (profit) earned. From this, we can derive an alternative formulation.

If we define q_ℓ as the surplus obtained by rebalancer ℓ upon completing their task, we can express this relationship as follows:

$$\begin{aligned} q_\ell &= p_{ij} - c_{\ell ij}, \quad \text{when } x_{\ell ij} = 1 \\ &= p_{i(\ell)j(\ell)} - c_{\ell i(\ell)j(\ell)} \\ &= (p_{i(\ell)j(\ell)} - c_{\ell i(\ell)j(\ell)}) x_{\ell i(\ell)j(\ell)} \\ &= (p_{i(\ell)j(\ell)} - c_{\ell i(\ell)j(\ell)}) x_{\ell i(\ell)j(\ell)} + \sum_{(i,j)\neq(i\ell,j(\ell))} (p_{ij} - c_{\ell ij}) x_{\ell ij} \\ &= \sum_{i,j} (p_{ij} - c_{\ell ij}) x_{\ell ij} \end{aligned} \tag{2.2.3.1}$$

where $(i(\ell), j(\ell))$ is selected such that $x_{\ell i(\ell)j(\ell)} = 1$. The fourth line is a consequence of our model assumption, where only one route, specifically $(i(\ell), j(\ell))$, is chosen by rebalancer ℓ, and all other routes (i, j) invalidate their corresponding assignment variable, resulting in $x_{\ell ij} = 0$.

It is worth noting that if no rebalancer chooses route (i, j), then all the constraining conditions in (2.2.1.3a) for p_{ij} simplify to $p_{ij} \geq 0$. Consequently, the minimization process leads to the conclusion that $p_{ij} = 0$. Based on this observation, we can deduce that every pair (i, j) where $p_{ij} > 0$ is associated with at least one rebalancer, and specifically, $p_{ij} = c_{\ell ij} + q_\ell$ if rebalancer ℓ is one of those associated.

$$\begin{aligned} \sum_\ell \sum_i \sum_j p_{ij} x_{\ell ij} &= \sum_\ell \left(\sum_i \sum_j p_{ij} x_{\ell ij} \right) \\ &= \sum_\ell \left(\sum_i \sum_j (c_{\ell ij} + q_\ell) x_{\ell ij} \right) \end{aligned}$$

$$
\begin{aligned}
&= \sum_{\ell} \sum_{i} \sum_{j} c_{\ell ij} x_{\ell ij} + \sum_{\ell} \sum_{i} \sum_{j} q_{\ell} x_{\ell ij} \\
&= \sum_{\ell} \sum_{i} \sum_{j} c_{\ell ij} x_{\ell ij} + \sum_{\ell} \left(\sum_{i} \sum_{j} x_{\ell ij} \right) q_{\ell} \\
&= \sum_{\ell} \sum_{i} \sum_{j} c_{\ell ij} x_{\ell ij} + \sum_{\ell} q_{\ell}
\end{aligned}
$$

The first equation prompts us to consider the perspective of rebalancer ℓ, who always selects a task from the system, while the last equation directly follows from the original formulation.

Now let us re-evaluate constraints (2.2.1.2) as an equivalence to (2.2.1.3f). For (i', j') that is not chosen by any rebalancer, the condition $p_{i'j'} = 0$ implies $p_{ij} \geq c_{\ell ij} - c_{\ell i'j'}$, for ℓ such that $x_{\ell ij} = 1$, which is automatically true given the other constraint $p_{ij} \geq c_{\ell ij}$ when $x_{\ell ij} = 1$. For (i', j') chosen by rebalancer ℓ', we have

$$
\begin{aligned}
& p_{ij} - c_{\ell ij} \geq p_{i'j'} - c_{\ell i'j'} \quad \text{when } x_{\ell ij} = 1 \quad \forall \ell, i, j, i', j' \\
\iff \quad & (c_{\ell ij} + q_{\ell}) - c_{\ell ij} \geq (c_{\ell' i'j'} + q_{\ell'}) - c_{\ell i'j'} \quad \text{when } x_{\ell ij} = 1, \quad \forall \ell, \ell', i, j, i', j' \\
\iff \quad & q_{\ell} - q_{\ell'} \geq c_{\ell' i'j'} - c_{\ell i'j'} \quad \text{when } x_{\ell ij} = 1, x_{\ell' i'j'} = 1 \quad \forall \ell, \ell', i, j, i', j' \\
\iff \quad & q_{\ell} - q_{\ell'} \geq c_{\ell' i'j'} - c_{\ell i'j'} \quad \text{when } x_{\ell' i'j'} = 1 \quad \forall \ell, \ell', i', j' \\
\iff \quad & q_{\ell} - q_{\ell'} \geq (c_{\ell' i'j'} - c_{\ell i'j'}) x_{\ell' i'j'} \quad \text{when } x_{\ell' i'j'} = 1 \quad \forall \ell, \ell', i', j' \\
\iff \quad & q_{\ell} - q_{\ell'} \geq \sum_{i', j'} (c_{\ell' i'j'} - c_{\ell i'j'}) x_{\ell' i'j'} \quad \forall \ell, \ell'
\end{aligned}
$$

Constraints on (i, j) in the third equivalence are removed as the variable has already been eliminated in the previous deduction. The last equivalence is derived similarly to (2.2.3.1).

With these preparations, we can substitute the price variables p_{ij} with the new surplus variables q_ℓ in (2.2.1.3a) and reformulate the problem into the following mixed-integer linear program (MILP):

$$\underset{\mathbf{q},\mathbf{x}}{\text{minimize}} \quad \sum_{\ell}\sum_{i}\sum_{j} c_{\ell ij}x_{\ell ij} + \sum_{\ell} q_\ell \tag{2.2.3.2a}$$

$$\text{s.t} \quad \sum_{j}\sum_{\ell} x_{\ell ij} = o_i \quad \forall i \tag{2.2.3.2b}$$

$$\sum_{i}\sum_{\ell} x_{\ell ij} = t_j \quad \forall j \tag{2.2.3.2c}$$

$$\sum_{i}\sum_{j} x_{\ell ij} = 1 \quad \forall \ell \tag{2.2.3.2d}$$

$$x_{\ell ij} \in \{0,1\} \quad \forall \ell, i, j \tag{2.2.3.2e}$$

$$q_\ell - q_s \geq \sum_{i,j} x_{sij}(c_{sij} - c_{\ell ij}) \quad \forall \ell, s \tag{2.2.3.2f}$$

$$q_\ell \geq 0 \quad \forall \ell \tag{2.2.3.2g}$$

where the last constraint is justified by a straightforward reformulation of (2.2.1.1) which is equivalent to (2.2.1.3e). We observe that this MILP formulation not only simplifies the complexity of (2.2.1.3a) by converting it into a linear form, but also reduces the number of both variables and constraints involved. And the optimal prices can be recovered from the optimal surplus values using $p_{ij} = q_\ell + c_{\ell ij}$, when $x_{\ell ij} = 1$.

In the given formulation, let us consider the case where we have a parameter $c_{\ell ij}$, resulting in an optimal solution $x^*_{\ell ij}$, q^*_ℓ, and an objective value z. Now, if we modify the parameter $c_{\ell ij}$ to $\alpha c_{\ell ij}$, where α is a constant multiplier, we observe that the new optimal solution becomes $\alpha x^*_{\ell ij}$ and αq^*_ℓ, with the corresponding objective value being αz. This

indicates that both the parameter $c_{\ell ij}$ and the objective value exhibit the same scaling factor α. Therefore the scale of the parameter $c_{\ell ij}$ does not alter the fundamental aspects of the problem. If required, we can adjust the scale of the parameter during computation as needed.

The formulation is referred to as the Decentralized Route Assignment Problem (DRAP). The term *decentralized* emphasizes the autonomous decision-making of rebalancers in selecting routes, while *route assignment* highlights the task of assigning specific routes to individual rebalancers. The DRAP provides a comprehensive framework for addressing this problem and will be used throughout this dissertation to discuss the proposed methodologies and findings.

2.2.4 Minimal agent cost model

Another model is the minimal agent cost problem (MACP), which does not consider the decentralized pricing structure. The solution of this model will be used to construct a feasible solution to DRAP for computational purposes. In this model, we assume that the platform has control over the assignments, and all contractors will follow the assigned tasks. Prices are not generated from the optimization problem; instead, we only produce an assignment. Therefore the program is formulated with a single set of decision variables $\boldsymbol{x}$, and the objective is to minimize the overall cost of rebalancers, referred to as the agent cost:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \sum_{\ell,i,j} c_{\ell ij} x_{\ell ij} \\ \text{s.t.} \quad & (2.2.1.3b) - (2.2.1.3d). \end{aligned} \tag{2.2.4.1}$$

where only constraints on assignments in (2.2.3.2) are kept. We notice that the problem is a three-index transportation problem.

In later sections of the dissertation, we establish that the solution to MACP can be used to construct a feasible solution for DRAP. This conclusion implies that the solution obtained from MACP can serve as an initial point for the proposed algorithms in Chapter 3, enhancing their efficiency and effectiveness.

2.3 Theoretical results

We observe that, in some cases, the assignment variables $x_{\ell ij}$ satisfying constraints (2.2.3.2b)-(2.2.3.2e) may not be able to induce a set of surplus variables q_ℓ that make DRAP feasible. In this section, our focus is on investigating the pattern. In Section 2.2.1, we will demonstrate the conditions under which the assignment variables $x_{\ell ij}$ allow the existence of a feasible set of surplus q_ℓ. Section 2.2.2 presents an algorithm designed to determine the surplus q_ℓ when the aforementioned conditions for $x_{\ell ij}$ in the previous subsection are satisfied.

2.3.1 Existence of decentralized pricing

From the assignment variable $\boldsymbol{x} = (x_{\ell ij})$, we introduce a new variable $\boldsymbol{y}$, $y_{ij} = \sum_\ell x_{\ell ij}$. We interpret y_{ij} as the number of bikes to be moved from region i to region j and refer to it as the *flow* variable. It is worth noting that, as the bike routes are determined by fixing y_{ij}, we are now assigning L bikes to L rebalancers. Therefore, for a fixed flow $\boldsymbol{y}$, there are $L!$ feasible assignments $\boldsymbol{x}$ by bipartite matching. The variable $\boldsymbol{y}$ is subject to the

following constraints:

$$\begin{aligned} \sum_j y_{ij} &= o_i, \\ \sum_i y_{ij} &= t_j, \\ y_{ij} &\in \mathbb{Z}_+. \end{aligned} \tag{2.3.1.1}$$

In this section, we show that the flow uniquely determines the decentralized assignment, which allows us to focus on a lower dimensional problem in further computational approaches.

Theorem 2.3.1. *There exists a set of decentralized pricing $\boldsymbol{p}$ (hence $\boldsymbol{q}$) for the assignment $\boldsymbol{x}$ of given flow $\boldsymbol{y}$ if and only if the assignment achieves the minimal agent cost among all feasible assignments for the flow.*

The aforementioned finding establishes a connection between the existence of decentralized pricing and the assignment of rebalancers. The implication is that, when aiming to facilitate a decentralized pricing structure, there is no requirement to impose additional constraints on the flow of vehicles, as long as the flow remains valid by satisfying the demand and supply requirements. In simpler terms, as long as the flow of vehicles fulfills the necessary demand and supply conditions, the platform can promptly determine the appropriate assignment and prioritize the establishment of a decentralized pricing mechanism without necessitating further constraints on the flow itself. As a result, any flow configuration can generate a feasible assignment that corresponds to the minimum agent cost. This observation supports the following argument:

Corollary 2.3.1. *For any given flow $\boldsymbol{y}$, there exists an assignment $\boldsymbol{x}$ producing a decen-*

tralized pricing $\boldsymbol{p}$. Since (2.3.1.1) is always feasible, so is DRAP.

The corollary provides additional confirmation of DRAP's feasibility by demonstrating a feasible flow with valid parameters. Moreover, as implied by Theorem 2.3.1, it is indeed possible to obtain the feasible assignment that corresponds to the lowest agent cost for a given flow $\boldsymbol{y}$. Therefore, when the flow $\boldsymbol{y}$ is held constant, the computation of the assignment feasible for decentralized pricing can be achieved by solving the following problem:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \sum_{\ell,i,j} c_{\ell ij} x_{\ell ij} \\ \text{s.t.} \quad & \sum_{\ell} x_{\ell ij} = y_{ij} \\ & \sum_{i} \sum_{j} x_{\ell ij} = 1 \end{aligned} \tag{2.3.1.2}$$

Note that this problem can be formulated as a classical assignment problem, which can be efficiently solved using the Hungarian algorithm [25]. The Hungarian algorithm is widely available in modern programming language packages such as Scipy for Python.

It is worth noting that there might be more than one solution to (2.3.1.2). In order to guarantee the consistency of the problem, we need a result showing that all of them lead to the same objective value to DRAP:

Proposition 2.3.1. *If two assignments $\boldsymbol{x}$ and $\boldsymbol{x}'$ can both induce decentralized pricing and correspond to the same flow $\boldsymbol{y}$, i.e., $\sum c_{\ell ij} x_{\ell ij} = \sum c_{\ell ij} x'_{\ell ij}$ and $\sum_{\ell} x_{\ell ij} = \sum_{\ell} x'_{\ell ij}$ for all i, j, then they will also yield the same total surplus. As a result, the objective value for DRAP will be the same for both assignments as well.*

The proposition implies that the corresponding optimal assignment, in terms of cost,

will also yield the optimal objective value for DRAP, regardless of the specific assignment chosen within the set of minimal-agent-cost assignments. In other words, the solution to DRAP, given a specific flow, is unique in the sense that it yields the same objective value.

Furthermore, we observe that the assignment that minimizes the total cost of global rebalancers is automatically minimal for its corresponding flow. This implies that the solution to MACP not only allows for the enforcement of a decentralized pricing structure but also provides a feasible solution to DRAP.

2.3.1.1 Proofs

In this subsection, our goal is to provide comprehensive proof for Theorem 2.3.1. To achieve this, we will demonstrate the if direction and the only if direction separately. We will begin by proving the if direction, followed by the only if direction, within which we will establish a lemma that enhances the statement and contributes to the overall proof.

From the assignment variable $\boldsymbol{x} = (x_{\ell ij})$, we can derive an assignment of routes. Specifically, when $x_{\ell ij} = 1$, it signifies that the route (i, j) (denoted as $i(\ell)$ for convenience) is assigned to rebalancer ℓ. We define a *permutation* of an assignment as a rearrangement of the rebalancers while preserving the distribution of routes. For example, let ℓ_k, $k = 1, \ldots, n$, represent n rebalancers, and $i(\ell_k)$ denote the route assigned to rebalancer ℓ_k. An assignment can be represented as $(\ell_1, i(\ell_1)), (\ell_2, i(\ell_2)), \ldots, (\ell_n, i(\ell_n))$. A permutation of this assignment could be $(\ell_1, i(\ell_2)), (\ell_2, i(\ell_3)), \ldots, (\ell_{n-1}, i(\ell_n)), (\ell_n, i(\ell_1))$.

For convenience, we introduce the notation $m_{s\ell} = c_{\ell i(\ell)} - c_{si(\ell)}$ for any rebalancers s and ℓ. The quantity $m_{s\ell}$ can be interpreted as the switching cost for rebalancer s to take

the route assigned to rebalancer ℓ. As a direct result, we have $m_{\ell\ell} = 0$ for all ℓ.

Proposition 2.3.2. *If an assignment of routes allows for decentralized pricing, then the assignment achieves the minimum agent cost among all its permutations. In other words, it attains the minimum agent cost for the corresponding flow.*

Proof. Since the assignment $\boldsymbol{x}$ inducing $(\ell, i(\ell))$ makes a decentralized pricing structure exist, it implies the existence of a surplus vector $\boldsymbol{q} = (q_\ell)$ such that:

$$q_s - q_\ell \geq c_{\ell i(\ell)} - c_{si(\ell)} = m_{s\ell}.$$

Consider a permutation of the assignment for n rebalancers:

$$(\ell_1, i(\ell_2)), (\ell_2, i(\ell_3)), ..., (\ell_{n-1}, i(\ell_n)), (\ell_n, i(\ell_1)),$$

which is a permutation of

$$(\ell_1, i(\ell_1)), (\ell_2, i(\ell_2)), ..., (\ell_{n-1}, i(\ell_{n-1})), (\ell_n, i(\ell_n)).$$

We can observe the following inequalities:

$$q_{\ell_1} - q_{\ell_2} \geq m_{\ell_1\ell_2}$$

$$q_{\ell_2} - q_{\ell_3} \geq m_{\ell_2\ell_3}$$

$$...$$

$$q_{\ell_{n-1}} - q_{\ell_n} \geq m_{\ell_{n-1}\ell_n}$$

$$q_{\ell_n} - q_{\ell_1} \geq m_{\ell_n \ell_1}$$

Add them all up, and we obtain:

$$0 \geq m_{\ell_1 \ell_2} + m_{\ell_2 \ell_3} + ... + m_{\ell_{n-1} \ell_n} + m_{\ell_n \ell_1}.$$

By the definition of $m_{s\ell}$, we can rewrite the inequality as follows:

$$\begin{aligned} & c_{\ell_1 i(\ell_1)} + c_{\ell_2 i(\ell_2)} + ... + c_{\ell_{n-1} i(\ell_n)} + c_{\ell_n i(\ell_n)} \\ \leq & c_{\ell_1 i(\ell_2)} + c_{\ell_2 i(\ell_3)} + ... + c_{\ell_{n-1} i(\ell_n)} + c_{\ell_n i(\ell_1)} \end{aligned}$$

Notice that the left-hand side of the inequality represents the total agent cost of the original assignment, while the right-hand side represents the total agent cost of the permuted assignment. Since the inequality holds for any permutation of the rebalancers, it implies that the original assignment achieves the minimal agent total cost among all its permutations. Since all permutations of an assignment give rise to the same flow, it implies that the assignment also corresponds to the minimal agent cost for the given flow. □

Next, we prove the converse of Proposition 2.3.2:

Proposition 2.3.3. *If an assignment of routes achieves the minimum agent cost among all its permutations, then it enables a decentralized pricing structure.*

Proof. To prove the proposition, it suffices to show that there exists $(q_1, ..., q_n)$ such that

$q_s \geq 0, \forall s$ and

$$q_s - q_\ell \geq m_{s\ell} \quad \forall s, \ell, s \neq \ell.$$

Consider the following linear program:

$$\begin{aligned} \underset{\mathbf{y}}{\text{maximize}} \quad & \sum_{s,\ell} m_{s\ell} y_{s\ell} \\ \text{s.t.} \quad & \sum_{\ell} y_{s\ell} = \sum_{\ell} y_{\ell s} \quad \forall s \\ & y_{s\ell} \geq 0 \quad \forall s, \ell \end{aligned} \tag{2.3.1.3}$$

Let (β_s), $\beta_s \geq 0$ be dual variables associated with the first set of constraints. The objective value of the dual problem is 0 if $m_{s\ell} - \beta_s + \beta_\ell \leq 0$ for all s, ℓ, or ∞ otherwise. It is important to note that (β_s) satisfying $m_{s\ell} - \beta_s + \beta_\ell \leq 0$ is exactly a set of surplus (q_s) we desire.

By the strong duality of linear programs, we can claim that the existence of $\boldsymbol{q} = (q_s)$ is equivalent to the primal problem reaching an objective value of 0. In other words, if there exists a dual solution (β_s) that satisfies $m_{s\ell} - \beta_s + \beta_\ell \leq 0$ for all s, ℓ, then the primal problem achieves an optimal objective value of 0.

To prove that the objective value of the primal problem (2.3.1.3) is 0, we can introduce an additional lemma that shows under the given constraints, $\sum_{s,t} m_{st} y_{st}$ can be expressed as a sum of terms of the form $(m_{s_1 s_2} + m_{s_2 s_3} + ... + m_{s_n s_1})d$ for some $d \geq 0$ and $(s_1, s_2, ..., s_n)$.

Once this lemma is proven, we can utilize the assumption that the assignment achieves the smallest total agent cost, which implies that $(m_{s_1 s_2} + m_{s_2 s_3} + ... + m_{s_n s_1}) \leq 0$ following

the same derivation as in Proposition 2.3.2. Then we can conclude that $(m_{s_1 s_2} + m_{s_2 s_3} + ... + m_{s_n s_1})d \leq 0$. Consequently, the objective value of (2.3.1.3) is also 0, as desired. □

We will now proceed to prove the lemma indicated in the previous proof, which is necessary to conclude the overall proof:

Lemma 2.3.1. *Given constraints* $\sum_\ell y_{s\ell} = \sum_\ell y_{\ell s}$, $\forall s$, *and* $y_{s\ell} \geq 0$, $\sum_{s,t} m_{st} y_{st}$ *can be expressed as sum of terms in the form of* $(m_{s_1 s_2} + m_{s_2 s_3} + ... + m_{s_n s_1})d$ *for some* $d \geq 0$ *and* $s_1, ..., s_n \in \mathbb{Z}_+$.

Proof. To simplify the analysis, we can represent $\boldsymbol{m} = (m_{s\ell})$ and $\boldsymbol{y} = (y_{s\ell})$ as an $L \times L$ matrix, where each entry corresponds to a switching cost $m_{s\ell}$ and a flow value $y_{s\ell}$, respectively. The first constraint in (2.3.1.3) can be interpreted as the requirement that in the matrix $\boldsymbol{y} = (y_{s\ell})$, the sum of entries in each row is equal to the sum of entries in each column.

Now we proceed with the proof of the lemma using induction on the number of non-zero entries in the matrix $\boldsymbol{y} = (y_{s\ell})$.

The base case yields a zero matrix, which automatically satisfies the statement. Now we consider the case where $\boldsymbol{y}$ is a non-zero matrix with at least one non-zero entry. We will update $\boldsymbol{y}$ to a new matrix $\boldsymbol{y'}$ with fewer non-zero entries while satisfying both constraints in the (2.3.1.3). We start by defining $d > 0$ as the minimum value among all non-zero entries of $\boldsymbol{y}$, i.e., $d = \min_{s,\ell} y_{s\ell} = y_{s_1 s_2}$. Next, we construct the matrix $\boldsymbol{y'}$ by setting $y'_{s_1 s_2} = 0$, effectively removing the non-zero entry with the minimum value. Then we examine the entries of $\boldsymbol{y}$ row by row. For each row s_k (starting from s_2), we look for a non-zero entry $y_{s_k s_{k+1}} \geq d$. If such an entry exists, we subtract d from it and set $y'_{s_k s k+1} = y_{s_k s_{k+1}} - d$.

If $s_{k+1} = s_1$, we stop the process. Otherwise, we continue to the next row s_{k+1}. During this process, we have updated $\boldsymbol{y}$ to $\boldsymbol{y'}$ by subtracting d from certain entries. The first constraint, which requires the sum of entries in each row to be equal to the sum of entries in each column, is still satisfied for $\boldsymbol{y'}$. Additionally, the entries of $\boldsymbol{y'}$ remain non-negative. Therefore, we can express $\sum_{s,t} m_{st} y_{st}$ as:

$$\sum_{s,\ell} m_{s\ell} y_{s\ell} = d(m_{s_1 s_2} + ... + m_{s_n s_1}) + \sum_{s,\ell} m_{s\ell} y'_{s\ell}.$$

Since $y'_{s_1 \ell_1} = 0 < d = y_{s_1 \ell_1}$, there is less number of non-zero entries in $\boldsymbol{y'}$ than in $\boldsymbol{y}$. Then the principle of induction validates the lemma. □

Theorem 2.3.1 then follows as a direct consequence of Proposition 2.3.2 and Proposition 2.3.3.

2.3.2 An iterative algorithm for pricing

Once we have obtained a feasible assignment for DRAP that enables decentralized pricing, the next step is to determine the pricing structure and compute the exact prices for the corresponding flow and assignment. We will focus on the linear part of DRAP with the assignment variable $\boldsymbol{x}$ fixed, which corresponds to enforcing the decentralized pricing on the assignment determined by $\boldsymbol{x}$.

$$\begin{aligned} \underset{\mathbf{q}}{\text{minimize}} \quad & \sum_{\ell} q_\ell \\ \text{s.t.} \quad & (2.2.3.2f) - (2.2.3.2g). \end{aligned} \tag{2.3.2.1}$$

The following theorem highlights the key property of the optimal solution to the problem:

Theorem 2.3.2. *If* $\boldsymbol{q^*} = (q_\ell^*)$ *is optimal to (2.3.2.1), then we have* $q_\ell^* = \max_s\{m_{\ell s} + q_s^*\}$ *for all* ℓ.

The property states that $q_\ell^* = \max_s m_{\ell s} + q_s^* = m_{\ell s_0} + q_{s_0}^*$ for some s_0. This property implies the existence of *impact* chains between the rebalancers. Each rebalancer surplus is either directly influenced by itself (surplus of 0) or impacted by another rebalancer. The concept of *impact* can be seen as a one-to-one mapping from the set of rebalancers $\mathcal{L}$ to itself, indicating the influence between rebalancers. It is worth noticing that there may exist multiple possible one-to-one mappings for a solution, each yielding the same total surplus. In other words, different rebalancers may impact each other in different ways, but the overall surplus remains unchanged.

Building upon the theorem, we present an alternative approach to solve the optimization problem (2.3.2.1) without the need for an LP solver. This approach is advantageous due to the potentially high computational cost of LP solvers, particularly for large-scale problems where traditional techniques like simplex methods can have exponential worst-case time complexity. Meanwhile, our algorithms offer increased interpretability, allowing for better insights into the problem, and can be a cost-effective solution for companies seeking alternatives to LP solvers.

By considering $m_{\ell s}$ as a matrix, we observe that the algorithm employs matrix addition during each iteration until it converges to the desired equation outlined in Theorem 2.3.2. Each iteration can be viewed as a step towards achieving a balance between rebal-

Algorithm 1 Iterative Algorithm For Pricing

Initialization: set $k = 1$ and $q_\ell^k = 0$, $\ell = 1, 2, ..., L$
while True **do**
  $q_\ell^{k+1} = \max_s\{m_{\ell s} + q_s^k\}$, $\ell = 1, 2, ..., L$
  if $q_\ell^{k+1} = q_\ell^k$ **then**
    break
  end if
  $k \leftarrow k + 1$
end while
$d = \sum_{\ell=1}^{L} q_\ell^k$

ancers, and the iteration process terminates when a favorable trade-off is achieved. This iterative approach not only demonstrates good interpretability but also allows for a clear understanding of how the algorithm progressively solves the problem by iteratively refining the solution until reaching a desired equilibrium.

We now provide a result regarding the computational complexity of the algorithm.

Theorem 2.3.3. *For an assignment $\boldsymbol{x}$ that is feasible for decentralized pricing, Algorithm 1 requires at most $L - 1$ iterations. Consequently, the algorithm has the computational complexity of $O(L^3)$.*

Based on the previous analysis, we can develop a subroutine to find a unique solution to DRAP up to the same objective value, given a flow variable $\boldsymbol{y}$. The proposed outline for the subroutine is given as follows:

Algorithm 2 Subroutine

Initialization: flow variable $\boldsymbol{y}$;
Step 1 apply Hungarian algorithm to find a feasible assignment variable $\boldsymbol{x}$;
Step 2 apply Iterative algorithm to find the corresponding surplus variable $\boldsymbol{q}$;
Step 3 calculate objective value of DRAP with $\boldsymbol{x}$ and $\boldsymbol{q}$.

Now we claim that we can greatly reduce the complexity of the problem by transforming it from searching for feasible assignment $\boldsymbol{x}$ and surplus $\boldsymbol{q}$ to searching for an

optimal flow $\boldsymbol{y}$. This reduction in complexity is attributed to the fact that every feasible flow $\boldsymbol{y}$ satisfies the market-clearing condition and that the dimension of the flow variable $\boldsymbol{y}$ is much smaller compared to the combined dimensions of $\boldsymbol{x}$ and $\boldsymbol{q}$.

2.3.2.1 Proofs

In this section, we proceed to prove Theorem 2.3.2 and Theorem 2.3.3. Throughout the proofs, we will maintain the notation introduced in Algorithm 1. Additionally, we will provide proof for Proposition 2.3.1 using the results obtained from the previous theorems.

Lemma 2.3.2. *After every iteration k, the rebalancer surplus q_ℓ^k can be expressed as a sum of terms $m_{\ell\ell'}$. More precisely, $q_\ell^k = m_{\ell\ell_1} + m_{\ell_1\ell_2} + ... + m_{\ell_{n-1}\ell_n}$ for some $\ell_1, \ell_2, ..., \ell_n$.*

Proof. We prove the lemma by induction. The base case when $k = 1$ holds automatically since $q_\ell^1 = 0 = m_{\ell\ell}$. As for $k \geq 2$, by the construction of q_ℓ^k in the algorithm, $q_\ell^k = m_{\ell\ell_1} + q_{\ell_1}^{k-1}$ for some ℓ_1. By assumption of the induction, $q_{\ell_1}^{k-1} = m_{\ell_1\ell_2} + ... + m_{\ell_{n-1}\ell_n}$ for some $\ell_2, ..., \ell_n$. Therefore we obtain $q_\ell^k = m_{\ell\ell_1} + m_{\ell_1\ell_2} + ... + m_{\ell_{n-1}\ell_n}$ for some $\ell_1, \ell_2, ..., \ell_n$ as desired and complete the proof. □

Later in the section, for $k \geq 1$, we refer to q_ℓ^k as an *update* if $q_\ell^k > q_\ell^{k-1}$.

Lemma 2.3.3. *If q_ℓ updates in the k-th iteration, then q_ℓ^k can be expressed as the sum of exactly k terms from the set $m_{\ell,\ell'}$.*

Proof. Again we prove it by induction. Let $T(q_\ell^k)$ be the number of terms of $\{m_{\ell\ell'}\}$ in q_ℓ^k written as sum. For $k = 1$, if q_ℓ^k updates, then for some $\ell_0 \neq \ell$, we have $q_\ell^1 = m_{\ell\ell_0} = \max_{\ell' \neq \ell}\{m_{\ell\ell'}\}$, which indicates $T(q_\ell^1) = 1$.

For case when $k \geq 2$, q_ℓ^k updates, there exists $\ell_1 \neq \ell$ such that $q_\ell^k = m_{\ell\ell_1} + q_{\ell_1}^{k-1} > q_\ell^{k-1}$. Therefore we have $T(q_\ell^k) = 1 + T(q_{\ell_1}^{k-1})$.

Now we claim that $q_{\ell_1}^{k-1}$ must have updated in iteration $k-1$. If not, then $q_{\ell_1}^{k-1} = \tilde{p_1}_\ell^{k-2}$ and

$$q_\ell^{k-1} \geq m_{\ell\ell_1} + q_{\ell_1}^{k-2} = m_{\ell\ell_1} + q_{\ell_1}^{k-1} > q_\ell^{k-1}$$

which is a contradiction. Thus we conclude that $T(q_{\ell_1}^{k-1}) = k-1$ and $T(q_\ell^k) = (k-1)+1 = k$, i.e. q_ℓ^k has exactly k terms of $\{m_{\ell,\ell'}\}$ in the sum. □

With the lemmas stated above, we are now ready to prove Theorem 2.3.3.

Proof. We will prove the claim by contradiction.

Suppose the number of iterations reaches L, then at the L-th iteration, there exists ℓ such that q_ℓ^L updates, meaning $q_\ell^L > q_\ell^{L-1}$. We will have

$$q_\ell^L = m_{\ell_L\ell_{L-1}} + m_{\ell_{L-1}\ell_{L-2}} + ... + m_{\ell_1\ell_0}$$

where $\ell_L = \ell$. Among the values $\ell_L, \ell_{L-1}, \ldots, \ell_0$, where each element takes a value in the set $1, 2, \ldots, L$, there must exist indices i and j such that $i > j$ and $\ell_i = \ell_j$. From the construction of q_ℓ^k, we conclude that q_{ℓ_i} updates at iteration i and that:

$$q_{\ell_i}^i = m_{\ell_i\ell_{i-1}} + ... + m_{\ell_{j+1}\ell_j} + m_{\ell_j\ell_{j-1}} + ... + m_{\ell_1\ell_0}.$$

Since $m_{\ell_i\ell_{i-1}} + ... + m_{\ell_{j+1}\ell_j} = m_{\ell_i\ell_{i-1}} + ... + m_{\ell_{j+1}\ell_i} \leq 0$, we have

$$q^i_{\ell_i} \leq m_{\ell_j\ell_{j-1}} + q^{j-1}_{\ell_{j-1}} = m_{\ell_i\ell_{j-1}} + q^{j-1}_{\ell_{j-1}} \leq q^j_{\ell_i}$$

which means q_{ℓ_i} doesn't update during iteration $j+1, ..., i$, leading to a contradiction.

Since each iteration of the algorithm involves a matrix addition that requires L^2 calculations, the overall complexity of the algorithm is at most $O(L^3)$. □

Now we have established the proof for the computational complexity of Algorithm 1. To prove Theorem 2.3.2, our objective is to show that the final solution produced by the algorithm is optimal for the problem (2.3.2.1), as indicated by the termination condition of the algorithm.

Theorem 2.3.4. *Let $d = \sum_{\ell=1}^{L} q_\ell$ be the total surplus computed in the algorithm, and let z be the objective value of problem (2.3.2.1), then $d = z$.*

Proof. On the one hand, when the algorithm terminates, the following property holds:

$$q_\ell = \max_{\ell'}\{m_{\ell\ell'} + q_{\ell'}\} \geq m_{\ell s} + q_s, \quad \forall s$$

$$\Longrightarrow q_\ell - q_s \geq m_{\ell s}, \quad \forall \ell, s$$

And since we have $q^k_\ell \geq m_{\ell\ell} + q^{k-1}_\ell = q^{k-1}_\ell$, we can confirm that q^k_ℓ is non-decreasing with each iteration and that $q_\ell \geq 0$. Therefore, q_ℓ is a feasible solution of problem (2.3.2.1). As a result, we have $d \geq z$.

To prove the other side of the equation, we consider the dual problem of (2.3.2.1):

$$\begin{aligned} \underset{\mathbf{q}}{\text{maximize}} \quad & \sum_{\ell,s} \beta_{\ell s} m_{\ell s} \\ \text{s.t.} \quad & \sum_{s} \lambda_{\ell s} - \sum_{s} \lambda_{s\ell} \leq 1, \quad \ell \\ & \lambda_{s\ell} \geq 0, \quad \forall \ell, s \end{aligned} \tag{2.3.2.2}$$

Let $\lambda^*_{\ell s} = \#\{m_{\ell s} \text{ appeared in } d\}$. Then for each ℓ,

$$\begin{aligned} & \sum_{s} \lambda^*_{\ell s} - \sum_{s} \lambda^*_{s\ell} \\ =& \#\{m_{\ell s}, \forall s \text{ appeared in } d\} - \#\{m_{s\ell}, \forall s \text{ appeared in } d\} \\ =& \sum_{\ell'} \left[\#\{m_{\ell s}, \forall s \text{ appeared in } q_{\ell'}\} - \#\{m_{s\ell}, \forall s \text{ appeared in } q_{\ell'}\}\right] \\ =& \sum_{\ell' \neq \ell} \left[\#\{m_{\ell s}, \forall s \text{ appeared in } q_{\ell'}\} - \#\{m_{s\ell}, \forall s \text{ appeared in } q_{\ell'}\}\right] \\ & + \left[\#\{m_{\ell s}, \forall s \text{ appeared in } q_{\ell}\} - \#\{m_{s\ell}, \forall s \text{ appeared in } q_{\ell}\}\right] \end{aligned}$$

Note that for $\ell' \neq \ell$, every $m_{\ell s}$ appeared in $q_{\ell'}$ as the sum of terms follows a $m_{s'\ell}$, which means

$$\#\{m_{\ell s}, \forall s \text{ appeared in } q_{\ell'}\} - \#\{m_{s\ell}, \forall s \text{ appeared in } q_{\ell'}\} \leq 0.$$

While in q_ℓ,

$$\#\{m_{\ell s}, \forall s \text{ appeared in } q_{\ell}\} - \#\{m_{s\ell}, \forall s \text{ appeared in } q_{\ell}\} = 1 - 0 = 1.$$

Therefore, we obtain that $\sum_s \lambda^*_{\ell s} - \sum_s \lambda^*_{s\ell} \leq 1, \forall \ell$. Together with the fact that $\lambda^*_{\ell s} \geq 0$ for

all ℓ and s, we can conclude that $\lambda^*_{\ell s}$ is a feasible solution to the dual problem of (2.3.2.1). By weak duality, we have

$$d = \sum_{\ell} q_\ell = \sum_{\ell s} \lambda^*_{\ell,s} m_{\ell s} \leq z.$$

Now we can conclude that $d = z$. □

We will then prove Proposition 2.3.1 and begin by stating a lemma that is needed for the proof.

Lemma 2.3.4. *For a given flow, if $\boldsymbol{x}$ and $\boldsymbol{x}'$ both minimize the total agent cost and induce the switching costs $(m_{\ell s})$ and $(m'_{\ell s})$ respectively, then there exists a permutation of assignments from $(i_1, i(i_1)), (i_2, i(i_2)), ..., (i_k, i(i_k))$ to $(i_1, i(i_2)), ..., (i_{k-1}, i(i_k)), (i_k, i(i_1)))$. We claim that the following equations hold:*

$$m'_{\ell i_j} = m_{\ell i_{j+1}} - m_{i_j i_{j+1}}, \quad j = 1, ..., k$$

$$m'_{\ell s} = m_{\ell s}, \quad s \neq i_1, ..., i_k$$

$$m_{i_1 i_2} + ... + m_{i_{k-1} i_k} + m_{i_k i_1} = 0.$$

Proof. We recall that for a fixed flow, different assignments are permutations of each other, which justifies the first part of the lemma. From the construction of $m_{\ell s}$, we obtain that

$$\begin{aligned} m'_{\ell i_j} &= c_{i_j i(i_{j+1})} - c_{\ell i(i_{j+1})} \\ &= -c_{i_{j+1} i(i_{j+1})} + c_{i_j i(i_{j+1})} + c_{i_{j+1} i(i_{j+1})} - c_{\ell i(i_{j+1})} \end{aligned}$$

$$= m_{\ell i_{j+1}} - m_{i_j i_{j+1}} \quad j = 1, ..., k$$

and that when $s \neq i_1, ..., i_k$, $m'_{\ell s} = c_{si(s)} - c_{\ell i(s)} = m_{\ell s}$.

Since $(m'_{\ell s})$ and $m_{\ell s}$ are both induced by feasible assignment, the last equality can be justified by the following:

$$\begin{aligned} 0 \geq & m'_{i_1 i_k} + m'_{i_k i_{k-1}} ... + m'_{i_2 i_1} \\ = & m_{i_1 i_1} - m_{i_k i_1} + m_{i_k i_k} - m_{i_{k-1} i_k} ... + m_{i_2 i_2} - m_{i_1 i_2} \\ = & - m_{i_1 i_2} - ... - m_{i_{k-1} i_k} - m_{i_k i_1} \geq 0 \end{aligned}$$

where we used the fact that for any feasible $\boldsymbol{m} = (m_{\ell s})$, $m_{\ell\ell} = 0 \; \forall j$ and $m_{\ell_1 \ell_2} + ... + m_{\ell_{n-1} \ell_n} + m_{\ell n} \leq 0 \; \forall \ell_1, ..., \ell_n$. □

Now we prove the proposition.

Proof. We keep the notation in the proof of the lemma above. Suppose the algorithm outputs q'_ℓ for $\boldsymbol{x}'$. As discussed previously, q'_ℓ can be written as a sum of terms in $\boldsymbol{m'} = (m'_{\ell s})$. Now we construct q_ℓ as follows:

Consider $q'_\ell = m'_{\ell \ell_1} + ... + m'_{\ell_{n-1} \ell_n}$ for some $\ell_1, ..., \ell_n$. If there is an $m'_{\ell_t i_j}$ term in the sum, we substitute via the equation:

$$\begin{aligned} m'_{\ell_t i_j} = & m_{\ell_t i_{j+1}} - m_{i_j i_{j+1}} \\ = & m_{\ell_t i_{j+1}} - m_{i_j i_{j+1}} + (m_{i_1 i_2} + ... + m_{i_{k-1} i_k} + m_{i_k i_1}) \\ = & m_{\ell_t i_{j+1}} + m_{i_{j+1} i_{j+2}} + ... + m_{i_k i_1} + ... m_{i_{j-1} i_j} \end{aligned}$$

and for other terms, we can immediately substitute with its counterpart in the set $\{m_{\ell s}\}_{\ell,s}$. The new sum is denoted as q_ℓ and we have that $q_\ell = q'_\ell$.

By following the same process as in the proof of Theorem 2.3.4, we can show that $\boldsymbol{q} = (q_\ell)$ generates a set of dual variables $(\lambda_{\ell s})$ that is feasible for the dual problem (2.3.2.2) with $(m_{\ell s})$. Therefore, we have $d \leq \sum q_\ell = \sum q'_\ell = d'$, where we denote d and d' as the total surplus corresponding to $\boldsymbol{x}$ and $\boldsymbol{x}'$, respectively.

Finally, since the two assignments are symmetric in the problem, we can claim the other direction $d' \leq d$. Therefore, we conclude that $d' = d$. □

2.4 Model extension

In this section, we will explore three distinct models. In Subsection 2.3.1, we will extend DRAP by relaxing the constraint that governs the number of rebalancers, and present associated results. Subsequently, in Subsection 2.3.2 and Subsection 2.3.3, we will provide a brief overview of alternative pricing structures as substitutes for the prices determined by the origin-destination pair.

2.4.1 Model with extra agents

We consider an extended model of DRAP where the number of rebalancers exceeds the number of bikes in the system. In this scenario, instead of assuming that every rebalancer moves exactly one bike, we allow rebalancers to move at most one bike while still satisfying the bike supply and demand constraints. To preserve the decentralized nature of the problem, rebalancers who are not assigned a task are unable to generate any profit and have

a maximum profit of zero for all routes. This can be expressed as $\max_{(i,j)} p_{ij} - c_{\ell ij} = 0$, or equivalently, $p_{ij} - c_{\ell ij} \leq 0$ for all (i, j).

In the corresponding MIQP formulation, constraint (2.2.1.3d) is relaxed to $\sum_i \sum_j x_{\ell ij} \leq 1$, allowing for rebalancers to have at most one assignment. However, constraints (2.2.1.3e) and (2.2.1.3f) still remain intact as they apply only when $x_{\ell ij} = 1$. Therefore, the extended formulation can be expressed as follows:

$$\begin{aligned} \min_{\mathbf{p},\mathbf{x}} \quad & \sum_{\ell} \sum_{i} \sum_{j} p_{ij} x_{\ell ij} \\ \text{s.t} \quad & \sum_{i} \sum_{j} x_{\ell ij} \leq 1 \quad \forall \ell \\ & p_{ij} \leq c_{\ell ij} \text{ when } \sum_{i,j} x_{\ell ij} = 0 \\ & (2.2.1.3b) - (2.2.1.3c), (2.2.1.3e) - (2.2.1.3h) \end{aligned} \tag{2.4.1.1}$$

To gain further insights, we introduce a notation called the *fictitious* route, denoted as $f = (i_f, j_f)$. This notation represents a route that does not actually exist. Instead of considering the scenario where extra rebalancers are not assigned any task, we assume that they have chosen the fictitious route. Additionally, we make the assumption that on this fictitious route, all rebalancers have a zero cost, denoted as $c_{\ell i_f j_f} = 0$. Then we have the

following MIQP formulation:

$$
\begin{aligned}
\min_{\mathbf{p},\mathbf{x}} \quad & \sum_{\ell}\sum_{i}\sum_{j} p_{ij}x_{\ell ij} + \sum_{\ell} p_{i_f j_f} x_{\ell i_f j_f} \\
\text{s.t} \quad & \sum_{i}\sum_{j} x_{\ell ij} + x_{\ell i_f j_f} = 1 \quad \forall \ell \\
& p_{i_f j_f} \geq c_{\ell i_f j_f} x_{\ell i_f j_f} \quad \forall \ell, i, j \\
& p_{i_f j_f} - p_{i'j'} \geq (c_{\ell i_f j_f} - c_{\ell i'j'}) x_{\ell i_f j_f} - M(1 - x_{\ell i_f j_f}) \quad \forall \ell, i', j' \\
& p_{ij} - p_{i_f j_f} \geq (c_{\ell ij} - c_{\ell i_f j_f}) x_{\ell ij} - M(1 - x_{\ell ij}) \quad \forall \ell, i, j \\
& p_{i_f j_f} \geq 0 \\
& (2.2.1.3b) - (2.2.1.3c), (2.2.1.3e) - (2.2.1.3h)
\end{aligned}
\tag{2.4.1.2}
$$

Following the same process as in Section 2.2, we can demonstrate that (2.4.1.2) is equivalent to the following MILP:

$$
\begin{aligned}
\underset{\mathbf{q},\mathbf{x}}{\text{minimize}} \quad & \sum_{\ell}\sum_{i}\sum_{j} c_{\ell ij} x_{\ell ij} + \sum_{\ell} c_{\ell i_f j_f} x_{\ell i_f j_f} + \sum_{\ell} q_{\ell} \\
\text{s.t} \quad & \sum_{i}\sum_{j} x_{\ell ij} + x_{\ell i_f j_f} = 1 \quad \forall \ell \\
& q_{\ell} - q_s \geq \sum_{i,j} x_{sij}(c_{sij} - c_{\ell ij}) + x_{si_f j_f}(c_{si_f j_f)} - c_{\ell i_f j_f}) \quad \forall \ell, s \\
& (2.2.3.2b) - (2.2.3.2e), (2.2.3.2g)
\end{aligned}
\tag{2.4.1.3}
$$

Since $c_{\ell i_f j_f} = 0$ for all ℓ, the first constraint reduces to (2.2.3.2f), and the MILP can be

simplified as follows:

$$\begin{aligned} \underset{\mathbf{q},\mathbf{x}}{\text{minimize}} \quad & \sum_{\ell}\sum_{i}\sum_{j} c_{\ell ij}x_{\ell ij} + \sum_{\ell} q_{\ell} \\ \text{s.t} \quad & \sum_{i}\sum_{j} x_{\ell ij} \leq 1 \quad \forall \ell \\ & (2.2.3.2b) - (2.2.3.2d), (2.2.3.2f) - (2.2.3.2g) \end{aligned} \tag{2.4.1.4}$$

We will prove the fictitious route version (2.4.1.2) and (2.4.1.3) formulation is equivalent to (2.4.1.1).

Lemma 2.4.1. *In the optimal solution of (2.4.1.3), rebalancers who are assigned to the fictitious route gain zero surpluses.*

Proof. Assume that rebalancer ℓ_0 taking the fictitious route with surplus $q_{\ell_0} = q_0$. Then for any rebalancer ℓ, we have

$$q_{\ell} - q_{\ell_0} \geq c_{\ell_0 i_f j_f} - c_{\ell_0 i_f j_f} = 0 \implies q_{\ell} \geq q_{\ell_0} = p_0.$$

Then similarly, for ℓ_1 who assigned to a fictitious route as well, we have

$$q_{\ell_0} - q_{\ell_1} \geq 0 \implies q_{\ell_1} \leq q_{\ell_0} = p_0$$

Thus, we conclude that $q_{\ell_1} = q_{\ell_0} = p_0 \leq q_{\ell}$ for all ℓ, indicating that all rebalancers taking the fictitious route have the same surplus, which is the minimum among all existing surpluses. Since the minimization process ensures that $\min_{\ell} q_{\ell} = 0$, we obtain $p_0 = 0$. □

As a consequence of the lemma, we can deduce the existence of an optimal solution

such that $p_{i_f j_f} = c_{\ell_0 i_f j_f} + q_{\ell_0} = 0$. Based on this observation, we can re-evaluate (2.4.1.2). The first constraint implies the first constraint in (2.4.1.1). The second and fifth constraints become redundant. The third constraint reduces to $p_{i'j'} \leq c_{\ell i' j'}$ for $x_{\ell i_f j_f} = 1$, which is equivalent to the second constraint in (2.4.1.1). The fourth constraint remains unchanged and corresponds to (2.2.1.3e). Therefore, we can conclude that (2.4.1.2) is equivalent to (2.4.1.1). We refer to (2.4.1.4) (or equivalently, (2.4.1.3)) as the Decentralized Route Assignment Problem with Extra Agents (DRAP-EA).

Notice that the lemma states that the platform does not need to pay the rebalancers if they are not assigned any task. As a direct consequence, if we increase the number of rebalancers in the system while keeping the other factors constant, the total platform cost will decrease. This is because the increase in the number of rebalancers provides more combinations and options for the platform to choose from, potentially leading to lower costs.

We observe that the corresponding MACP-EA is obtained by replacing the last constraint with a $\leq$-inequality. If we determine the cost of rebalancers by distances between locations, i.e., $c_{\ell ij} = d_{ij} + \lambda(d_{\ell i} + d_{\ell j})$, and consider a scenario where there is a substantial number of rebalancers densely distributed across the region, we can reasonably infer that there are an ample number of individuals positioned at the most cost-effective locations along all potential routes. As a result, their cost only relies on the distances between the stations, leading to a total surplus of 0 in the optimal solution. Hence, the optimal assignment to DRAP-EA for this scenario coincides with that of MACP-EA.

The concept of fictitious routes allows us to view DRAP-EA as an extension of the standard DRAP by expanding the cost parameter $\boldsymbol{c} = (c_{\ell ij}, c_{\ell f})$, without making any

changes to the objective functions and constraints. This extension enables us to generalize the theoretical results in previous sections without requiring significant modifications to the existing framework.

Lastly, we observe that this idea can be further extended by relaxing the capacity assumption ($\sum_{i,j} x_{\ell ij} = 1$) to $\sum_{i,j} x_{\ell ij} \leq u_\ell$. This relaxation is equivalent to introducing $u_\ell - 1$ additional agents with the same personalized cost to the system.

2.4.2 Pricing over stations

In order to demonstrate the advantages of pricing over origin-destination pairs, we aim to compare the model that prices with other strategies. All the other assumptions of DRAP are kept.

We first consider a pricing structure where the platform sets prices separately for pick-up stations (p_i) and drop-off stations (q_j). The optimization problem can be formulated

as follows:

$$
\begin{aligned}
\min_{\mathbf{p},\mathbf{q}} \quad & \sum_i o_i p_i + \sum_j t_j q_j \\
\text{s.t} \quad & \sum_j \sum_\ell x_{\ell ij} = o_i \quad \forall i \\
& \sum_i \sum_\ell x_{\ell ij} = t_j \quad \forall j \\
& \sum_i \sum_j x_{\ell ij} = 1 \quad \forall \ell \\
& p_i + q_j \geq c_{\ell ij} x_{\ell ij} \quad \forall \ell, i, j \\
& p_i + q_j - (p_{i'} + q_{j'}) \geq (c_{\ell ij} - c_{\ell i' j'}) x_{\ell ij} - M(1 - x_{\ell ij}) \quad \forall \ell, i, j, i', j' \\
& x_{\ell ij} \in \{0, 1\}, \quad p_i \geq 0, \quad q_j \geq 0 \quad \forall \ell, i, j
\end{aligned}
\tag{2.4.2.1}
$$

It is evident that the solution to the problem is feasible for (2.2.1.3a) by defining $p_{ij} = p_i + q_j$ and but the opposite is not always true, thus resulting in the same or a higher objective value than DRAP.

2.4.3 Pricing over pick-up station

Next we consider the case where the company price the bike by pick-up location (p_i). If we maintain the decentralized pricing element, we can formulate the following program:

$$
\begin{aligned}
\min_{\mathbf{p},\mathbf{q}} \quad & \sum_i o_i p_i \\
\text{s.t} \quad & \sum_j \sum_\ell x_{\ell ij} = o_i \quad \forall i \\
& \sum_i \sum_\ell x_{\ell ij} = t_j \quad \forall j \\
& \sum_i \sum_j x_{\ell ij} = 1 \quad \forall \ell \\
& p_i \geq c_{\ell ij} x_{\ell ij} \quad \forall \ell, i, j \\
& p_i - p_{i'} \geq (c_{\ell ij} - c_{\ell i' j'}) x_{\ell ij} - M(1 - x_{\ell ij}) \quad \forall \ell, i, j, i', j' \\
& x_{\ell ij} \in \{0, 1\}, \quad p_i \geq 0 \quad \forall \ell, i, j
\end{aligned}
\tag{2.4.3.1}
$$

Unfortunately, unlike DRAP, there is no guarantee that this optimization problem will be feasible. A counter-example can be created from a star network where there is only one pick-up station but multiple drop-off stations.

Chapter 3: Computational methods

3.1 Overview

DRAP is a mixed-integer linear program (MILP), which is NP-hard in general [26]. To solve MIPs, branch-and-cut is a widely used approach that combines the branch-and-bound algorithm with the cutting plane method. In addition to these methods, this section also explores several search techniques that can be used to solve DRAP: hill-climbing as a local search method and simulated annealing as a global search method. We aim to evaluate these algorithms on DRAP and hopefully draw possible insights.

In the context of DRAP, a flow variable $\boldsymbol{y}$ defined by (2.3.1.1) determines a feasible solution and hence an objective value, as shown in the previous section. As a result, one possible approach to finding a solution to DRAP is to search through flows in the feasible set. This is indicated by solving the following optimization problem:

$$\begin{aligned} &\text{minimize} && f(y) \\ &\text{subject to} && (2.3.1.1) \end{aligned} \tag{3.1.0.1}$$

where the realization of $f(y)$ is defined by the subroutine we defined earlier in Algorithm 2. We observe that $f(y)$ is not expressible in closed form and therefore not differentiable.

3.2 Branch-and-cut

The branch-and-cut (B&C) method [27] is widely used to solve mixed-integer linear programming (MILP) problems by combining branch-and-bound and cutting plane techniques.

In the branch-and-bound framework, the problem is solved iteratively by traversing a tree-like structure. At each node, the linear programming (LP) relaxation of the MILP is solved. If the LP relaxation yields integer solutions, an upper bound is updated. Otherwise, the node is branched into two child nodes, creating subproblems. This branching process continues until infeasibility is detected or the optimal value of a subproblem exceeds the current upper bound. The algorithm terminates once all nodes have been explored.

The cutting plane method is employed to improve the lower bound and tighten the formulation of the MILP. Undesirable fractional solutions are removed by introducing cutting planes, namely valid inequalities. This method plays a crucial role in the branch-and-cut approach, reducing the search space and enhancing the efficiency of the branch-and-bound algorithm.

Branch-and-cut is a commonly employed technique in modern optimization solvers. For our research on DRAP, we will utilize Gurobi, a state-of-the-art MIP solver, to obtain computational results, which can serve as a benchmark for other proposed methods.

Algorithm 3 Branch-and-cut Framework

Initialize the tree: $T = \{o\}$ where o has no branching constraints; $z_u := +\infty$.
while T is nonempty **do**
  Select a node $o' \in T$,
  Initialize $z_l := -\infty$.
  if o' is feasible **then**
    Let z be the value by LP relaxation and update $z_l = \max\{z, z_l\}$,
    Apply cutting planes and update z_l accordingly,
  else
    $T := T \setminus \{o'\}$.
  end if
  if $z > z_u$ **then**
    $T := T \setminus \{o'\}$.
  else if $\boldsymbol{r}$ is integer **then**
    $T := T \setminus \{o'\}$,
    Update z_u by plugging solution $\boldsymbol{r}$, $\boldsymbol{q}$ into DRAP.
  else
    Branch, resulting in nodes o^* and o^{**}, $T := T \cup \{o^*, o^{**}\}$.
  end if
end while

3.3 Hill-Climbing algorithm

Hill-climbing, as a heuristic approach, is a classical local search algorithm for finding the optimal solution to a complex optimization problem. The algorithm works by starting with an initial solution and repeatedly searching for the direction of improvement in the search space. At each step, the algorithm adopts one of two main strategies: complete search where it selects the neighbor with the best objective function value and first-accept where it accepts the first move with an improvement. The algorithm terminates when a local optimum is reached. Although hill-climbing can be efficient for large-scale problems [28], it is easily stuck at a local optimum. A common way to tackle this is to randomly restart the program with different initial points multiple times [29].

In order to apply searching methods, we need to generate neighboring solutions for

a given flow variable. We can achieve this by considering the swapping of destinations between two bikes. For instance, if we have a flow y where $y_{i_1 j_1} > 0$ and $y_{i_2 j_2} > 0$, with $i_1 \neq i_2$ and $j_1 \neq j_2$, it implies that there is a bike moving from i_1 to j_1 and another bike moving from i_2 to j_2. By swapping the destinations of these two bikes, we create a new solution where the first bike travels from i_1 to j_2 and the second bike travels from i_2 to j_1. As a result, $y_{i_1 j_1}$ and $y_{i_2 j_2}$ decrease by 1, while $y_{i_1 j_2}$ and $y_{i_2 j_1}$ increase by 1. Importantly, this maintains the constraints (2.3.1.1). The algorithm to realize this process is outlined below:

Algorithm 4 Generate Neighbor

Initialize y, $y^{\text{new}} = y$
while True **do**
  random generate i_1, i_2, $i_1 \neq i_2$ from pick-up stations
  random generate j_1, j_2, $j_1 \neq j_2$ from drop-off stations
  if $y_{i_1 j_1} > 0$ and $y_{i_2 j_2} > 0$ **then**
    $y^{\text{new}}_{i_1 j_1} = y_{i_1 j_1} - 1$, $y^{\text{new}}_{i_2 j_2} = y_{i_2 j_2} - 1$
    $y^{\text{new}}_{i_1 j_2} = y_{i_1 j_2} + 1$, $y^{\text{new}}_{i_2 j_1} = y_{i_2 j_1} + 1$
  end if
end while
Return y^{new}

3.3.1 Hierarchical hill-climbing

We propose a new strategy to the classical hill-climbing method, which combines the complete search and first-accept strategies in hill-climbing algorithms. The search space is divided into ordered groups $G_1, \ldots, G_n$ with probability $p_1, \ldots, p_n$ where $p_1 > \cdots > p_n$ and $\sum p_i = 1$. The algorithm samples groups based on their probability, and evaluates all possible solutions within each group to select the one with the best objective value. This process is repeated for subsequent groups until the best solution within the group improves

the current solution. The strategy aims to enhance the efficiency of hill-climbing algorithms by narrowing down the search space while still obtaining high-quality solutions. However, it should be noted that this algorithm relies on having a predefined preference for neighbors, which may not be readily available in many problem scenarios. Additionally, we can see that the algorithm terminates at local optimum.

We provide a detailed description of the algorithm below where the solution to MACP is chosen to construct the initial point.

Algorithm 5 Hierarchical Hill-climbing

Initialize an ordered groups: $G_1, ..., G_n$ with probability $p_1, \ldots, p_n$, initial solution $\boldsymbol{y}$ obtained from MACP, $k = 0$.
while True **do**
  Sample a number k from $\{1, ..., n\}$ based probability $p_1, \ldots, p_n$.
  Initialize $f^k = 0$
  for $y' \in G_k$ **do**
    if $f(y') < f^k$ **then**
      Update: $f^k \leftarrow f(y')$, $y^k \leftarrow y'$
    end if
  end for
  if $f^k < f(y)$ **then**
    Break
  end if
  $k \leftarrow k + 1$
  $y \leftarrow y^k$
end while

To introduce preference in our problem (3.1.0.1), we assume that the rebalancer cost parameter $c_{\ell ij}$ is positively correlated with the distance d_{ij} between stations i and j. This allows us to deduce that rebalancers generally prefer pick-up and drop-off locations that are close to each other. When generating neighbors for a given flow, we consider swapping two origin-destination pairs, such as (i_1, j_1) and (i_2, j_2), to (i_1, j_2) and (i_2, j_1). To judge the preference of a neighbor, we compare the ratio between $d_{i_1 j_1} + d_{i_2 j_2}$ and $d_{i_1 j_2} + d_{i_2 j_1}$. A

larger ratio indicates a preferable swap. Notice that this preference is independent of flows and hence can be predefined.

Then we can group the neighbors in order from the most preferred to the least preferred. This creates a sorted list of groups for the algorithm, with each group containing similarly preferred neighbors. The probability of selecting the group can be adjusted through experimentation by algorithm tuning.

3.3.2 Extended local search

We propose an extended local search method, a modified version of the complete-search hill-climbing algorithm by combining tabu search [30] which uses a memory-based mechanism to guide the search process and escape local optima. In this approach, we maintain a tabu list that keeps track of all local optima reached by the algorithm, and we exclude these optima from the neighbor search process. This modification allows the algorithm to continue exploring and avoid getting trapped in local optima. The algorithm is designed to run indefinitely, adapting to any runtime duration. The steps of the algorithm are outlined below:

Algorithm 6 Extended Local Search

Initialize $y = y_0$, tabu list S.
for k=1,2,... **do**
  Search thoroughly excluding S and find the neighbor y' of y with minimal objective value
  if $f(y') \geq f(y)$ **then**
    add y to the S
  end if
  $y \leftarrow y'$
end for

3.4 Simulated annealing

Simulated annealing (SA) is an enhancement to the hill-climbing algorithm that introduces stochasticity. It draws inspiration from the annealing process in metallurgy, which involves heating and slowly cooling metal to enhance its properties by removing defects. The idea of simulated annealing was first introduced by Metropolis et al. [31], who developed a Monte Carlo simulation method at a fixed temperature level. Kirkpatrick et al. [32] later extended the Metropolis algorithm to incorporate varying temperatures, giving rise to the concept of simulated annealing.

In simulated annealing algorithm, an initial solution is iteratively modified, and these modifications are accepted or rejected based on a probabilistic criterion determined by the current temperature of the system. As the optimization process progresses, the temperature gradually decreases, enabling the algorithm to explore the solution space more effectively. This temperature annealing allows simulated annealing to escape local optima and potentially converge towards a global optimum, given certain conditions [33].

The initial temperature in simulated annealing is a critical parameter that influences the trade-off between exploration and exploitation in the search process. By controlling the likelihood of accepting worse solutions, the initial temperature guides the algorithm's behavior in the early stages. With a higher initial temperature, the algorithm is able to perform a random walk, allowing it to explore different regions of the search space and escape local optima. Conversely, a lower initial temperature prioritizes exploitation, causing the algorithm to focus on refining the current solution and potentially converge faster. The choice of the initial temperature depends on the specific problem and is typically determined through

experiments [34].

Cooling schedules in the simulated annealing algorithm control the temperature reduction throughout the search process. They determine the transition from exploration to exploitation by gradually decreasing the temperature. The convergence of simulated annealing relies on the cooling schedule's ability to escape local optima and converge towards the global optimum [35]. Popular cooling schedules include linear, logarithmic, exponential, and adaptive schedules [36]. The choice of a cooling schedule depends on the problem and desired search behavior, often requiring experimentation to find the most effective one.

Below is the pseudo-code for the simulated annealing algorithm with a geometric cooling schedule, where $T_{k+1} = \rho T_k$ and ρ is typically chosen from the range $[0.9, 1)$:

Algorithm 7 Simulated Annealing

Initialize the current solution y, the initial temperature T, and the number of steps K at each temperature
for $i = 1, 2, \ldots$ **do**
  for $k \leftarrow 1$ to K **do**
    Generate a new solution y' by swapping two origin-destination pairs defined in y
    Calculate the difference of objective value $\Delta f = f(y') - f(y)$
    if $\Delta f < 0$ **or** random$(0, 1) < e^{-\Delta f / T}$ **then**
      Accept the new solution: $y \leftarrow y'$
    end if
  end for
  Cool down the temperature: $T \leftarrow \rho T$
end for

3.4.1 Shaking procedure

In our simulated annealing algorithm, we utilize a shaking procedure to enhance exploration and avoid local optima. This procedure involves generating new candidate solutions by introducing random changes to the current solution. By incorporating this

shaking step, the algorithm gains the ability to explore a larger search space, allowing for the discovery of alternative solutions.

We propose a novel algorithm that integrates a shaking procedure to generate neighboring solutions. Our algorithm includes a mechanism to monitor the number of rejections, denoted as R, for a given solution. The shaking procedure is then employed with a probability that is determined by the inverse of the rejection count, i.e., $p = 1/R$. This adaptive shaking approach enhances the algorithm's exploration capability by encouraging it to explore a broader range of solutions, particularly when it is easier to transition away from the current solution. By dynamically adjusting the shaking probability based on the ease of transition, the algorithm increases its chances of finding improved solutions during the optimization process.

The neighbor generation algorithm and the corresponding modified simulated annealing algorithm are presented in Algorithm 8 and Algorithm 9, respectively.

Algorithm 8 Generate Neighbor with Probabilistic Shaking

Initialize y, p
Generate y' by swapping from y
if random$(0, 1) < p$ **then**
    Generate y'' by swapping from y'
    $y \leftarrow y''$
end if
Return y'.

Algorithm 9 Simulated Annealing with Probabilistic Shaking

Initialize the current solution y, the initial temperature T, the number of steps K at each temperature, number of rejection $r = 0$.
for $i = 1, 2, \ldots$ **do**
  for $k \leftarrow 1$ to K **do**
    Generate a new solution y' from y with shaking probability $p = 1/(R+1)$.
    Calculate the difference of objective value $\Delta f = f(y') - f(y)$
    if $\Delta f < 0$ **or** random$(0, 1) < e^{-\Delta f/T}$ **then**
      Accept the new solution: $y \leftarrow y'$
      Set $r = 0$.
    else
      $R \leftarrow R + 1$
    end if
  end for
  Cool down the temperature: $T \leftarrow \rho T$
end for

3.5 Conclusion

After conducting extensive testing on various instances of the DRAP model, it has been observed that all the search methods have shown promising performance. In the upcoming section, we will implement three specific algorithms, namely simulated annealing (SA), simulated annealing with probabilistic shaking (SA-PS), and extended local search (ELS), to obtain numerical results.

To ensure a comprehensive evaluation, we will employ branch-and-cut algorithm in Gurobi (GRB) as a benchmark. In order to enhance the performance of Gurobi, we will utilize a warm start solution obtained through hierarchical hill-climbing. This warm start solution is particularly crucial as it significantly increases the likelihood of generating feasible solutions using Gurobi.

Chapter 4: Numerical experiments

4.1 Overview

In this chapter, our objective is to perform a comprehensive numerical study of DRAP using the algorithms proposed in the previous chapter. To generate instances for our study, we collect real data from the DC area and present a detailed case study. We carefully design various scenarios and create 24 instances to analyze. To test the effectiveness of our approach, we propose four algorithms for experimentation. Subsequently, we analyze the numerical results obtained from these algorithms and provide visualizations of selected solutions to gain further insights into the problem.

4.2 Case study

Our research focuses on analyzing the operations of a bike-sharing company located in the Washington DC area. This company uses freelancers to help manage the re-distribution of their bikes, with each freelancer only allowed to move one bike. In our experiment, the payment for each freelancer is based on the bike's origin and destination, with each freelancer's cost calculated using their home location.

The company has developed an app for their freelancers to make the bike rebalancing

process easier. This app enables freelancers to register their availability three hours before the rebalancing process begins. Every evening at 10 pm, the company activates its app and presents a comprehensive list of rebalancing tasks, each accompanied by a designated payment amount. These tasks involve picking up bikes from specified locations and delivering them to assigned drop-off points. Freelancers who have registered with the company can access the app and review the available jobs, taking into account their own individual costs and potential profits. Once a freelancer selects a job, they are responsible for collecting the bike from the designated pick-up location and transporting it to the specified drop-off point. Upon successful completion of the task, the freelancer receives payment and returns where they started.

To simulate the bike distribution, we obtained real-time data from Capital Bikeshare, a station-based bike-sharing company in Washington DC. The data used for the simulation was collected from November 7th to November 8th, 2020. For the initial distribution, we utilized the data recorded at 11:00 PM on November 7th, which represents the end of operations for that day. Conversely, for the target distribution, we utilized the data recorded at 5:00 AM on the following morning, just prior to the commencement of operations for that day. This choice was motivated by the understanding that the desired morning bike distribution reflects the public demand that needs to be met, while the distribution at night represents the outcome after the system has been operational throughout the day. We specifically selected stations within DC for designing our instances, even though Capital Bikeshare stations extend beyond the city.

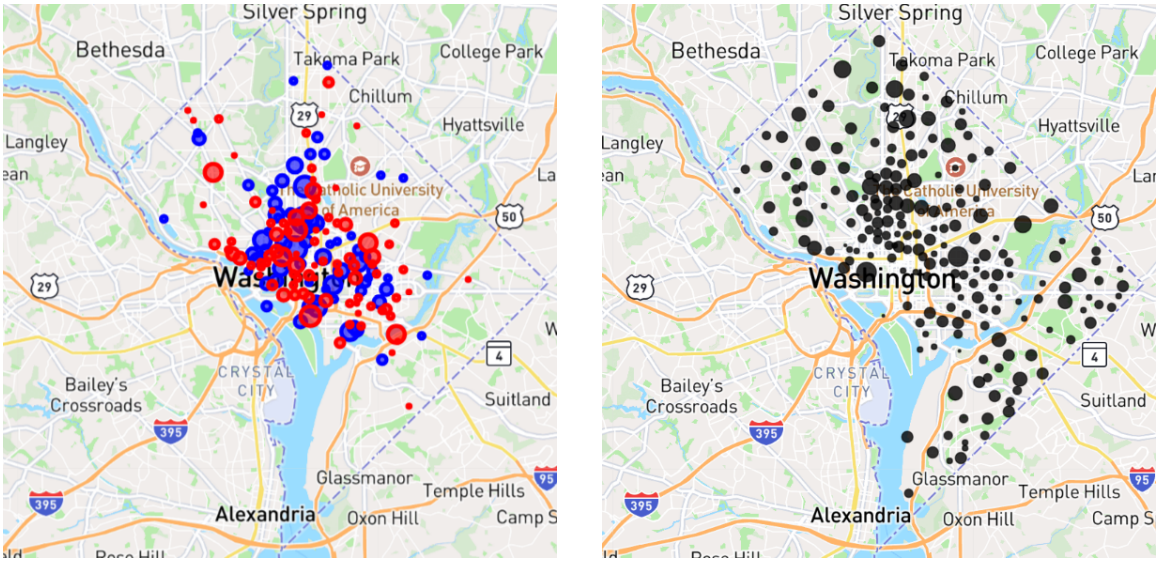
In our numerical experiment, we considered instance sizes ranging from $N = 20$, 30, 40, 50, where N represents the number of pick-up locations. For each instance size,

we randomly generated a set of pick-up and drop-off locations by sampling without replacement from the set of stations in the data, while ensuring that the total number of bikes available for pick-up and drop-off remained constant across all instances. The pick-up locations were determined based on stations with an increase in bike count at night, while the drop-off locations were chosen from stations with a decrease in bike count. Table 4.1 provides detailed information on the number of drop-off locations and bikes for each instance size N.

The rebalancers in our dataset were generated randomly from a spatial distribution proportional to the population density across the city of DC. This density information is publicly available. Each rebalancer in the dataset is assigned a home location, which signifies their starting and ending point for the rebalancing tasks. All locations in our dataset, including pick-up stations, drop-off stations, and rebalancer home locations, are represented by latitude and longitude coordinates.

Figure 4.1 provides a visual representation of the original regional distribution of stations and residency in Washington DC before being selected for our instances. The two figures highlight that the stations requiring rebalancing are primarily concentrated in the central area of the city, while the residents are more evenly distributed throughout the entire region. Larger circles in Figure 4.1 (a) indicate larger numbers of bikes. Red circles represent pickup stations, while blue circles represent drop-off stations. Similarly, in Figure 4.1 (b), larger circles indicate larger numbers of rebalancers in a particular area.

In our model, each rebalancer has an individual cost structure. We use a cost calculation based on the total travel distance associated with a particular job. This cost calculation consists of two components: the distance between stations and the distance be-



(a) Bike distribution

(b) Resident distribution

Figure 4.1: Bike and resident distribution from raw data

pick-up locations (N)	drop-off locations	bikes
20	19	100
30	30	155
40	35	210
50	45	271

Table 4.1: Number of drop-off stations and bikes for each instance size

tween the rebalancer home and the stations. The formula of the cost is represented as: $c_{\ell ij} = d_{ij} + \lambda(d_i^\ell + d_j^\ell)$, where d_{ij} denotes the distance between station i and station j, and d_i^ℓ represents the distance between station i and the home of rebalancer ℓ. The parameter λ chosen from $\{0.7, 1, 1.5\}$ plays as a tradeoff parameter reflecting the weight assigned to traveling empty versus moving the bike. In other words, it represents the ratio of the cost for a rebalancer to move between their home and stations compared to the cost of moving between stations. For instance, when $\lambda = 1$, the cost is equal. A smaller value of $\lambda = 0.7$ implies a higher cost for moving between stations, while a larger value of $\lambda = 1.5$ indicates a relatively higher cost for moving between the rebalancer home and stations. All distances are calculated from latitudes and longitudes using the Haversine formula. In addition, as discussed in Chapter 2, the scale of the cost parameter has no impact on the solution, except for a constant multiple. Therefore, we can use the distance as a reliable proxy for the utility of rebalancers.

In our experiments, we will compare two models: the original DRAP and its extended version, DRAP-EA, which were introduced in Chapter 2. In the DRAP model, the number of rebalancers is the same as the number of bikes. In contrast, the DRAP-EA model includes approximately 50% more rebalancers to serve as additional resources in the system. We introduce the parameter r to represent the ratio between the number of rebalancers and the number of bikes. In our subsequent discussions, we will refer to the DRAP model as the standard model with $r = 1$, and the DRAP-EA model as the EA (Enhanced Availability) model with $r = 1.5$.

In summary, our experimental setup focuses on four sets of regional distributions for bikes and rebalancers. This setup encompasses a total of 24 instances, which explore

parameter	values range
N	$20, 30, 40, 50$
λ	$0.7, 1, 1.5$
r	$1, 1.5$

Table 4.2: Values range for instances

different combinations of instance sizes (N), models (r), and cost structures (λ). The specific parameter settings for these instances are summarized in Table 4.2.

4.3 Numerical results

The primary objective of this section is to assess and compare the effectiveness of the proposed approaches, namely simulated annealing (SA), simulated annealing with probabilistic shaking (SA-PS), Gurobi (GRB), and extended local search (ELS). In our experiments with large instance sizes ($N = 40, 50$), we observed that Gurobi alone might not be able to find an incumbent solution, even with the assistance of its built-in heuristic function, within a reasonable time frame (approximately 3 days). To address this issue and further improve the performance, we employed the hierarchical hill-climbing algorithm as a warm-start strategy for Gurobi. All algorithms have been implemented in Python 3.9 and executed on a system equipped with an Intel Core i7 8700 CPU (3.70 GHz) and 16 GB RAM. Gurobi 9.5 Solver is employed for the optimization tasks.

Table 4.3 provides the hyperparameters that have been tuned for each instance size. The second column includes the initial temperature T_0 for both SA and SA-PS approaches. The third column shows the number of steps we iterated at each temperature level in SA and SA-PS. The fourth column indicates the number of groups utilized in the hierarchical hill-

climbing algorithm to determine Gurobi warm-start solutions. The final column displays the time limit extended as the problem instances grow in size. To strike a practical balance, we imposed a maximum time limit of three hours for all algorithms. During the search for Gurobi warm-start solutions, we imposed a time limit of 1500 seconds on the hierarchical hill-climbing algorithm. This restriction applies specifically to larger instance sizes. Both simulated annealing methods in this study utilize a geometric cooling schedule: $T_{k+1} = 0.95T_k$.

N	T_0	K	groups	time limit (s)
20	500	100	2	1800
30	1000	100	4	7200
40	1500	100	8	10800
50	2000	100	16	10800

Table 4.3: Algorithmic parameters for each instance size

Due to the inherently stochastic nature of the SA, SA-PS, and hierarchical hill-climbing algorithm, it is expected that the results obtained from these methods may vary across different runs. To ensure a robust evaluation, we conducted 10 instances for each of the three methods: SA, SA-PS, and Gurobi with warm-start. For each of the three methods, we recorded the average objective value (avg_obj) and standard deviation (std) across the 10 instances. Additionally, we compared the results of SA, SA-PS, and ELS with the warm-started Gurobi method by computing a column called gap. The gap is calculated using the formula:

$$\text{gap} = \frac{\text{obj} - \text{Gurobi\_obj}}{\text{Gurobi\_obj}} \times 100\%$$

with obj being the (average) objective value for the method and Gurobi_obj being the aver-

age objective value for the warm-started Gurobi method. A comprehensive summary of the numerical results can be found in Table 4.4. For better visualization and comparison of the four methods, we also present boxplots obtained from all runs for all instances in Figure 4.2 and Figure 4.3.

Based on the results in Table 4.4 and the boxplots shown in Figures 4.2 and 4.3, it is evident that the two simulated annealing methods consistently outperform the other two approaches in all tested scenarios. Both SA and SA-PS consistently achieve better solutions compared to the ELS and GRB methods, resulting in an average decrease in objective values of 5.8% from GRB. We also note that the ELS algorithm demonstrates comparable performance to GRB across all cases, with no significant superiority or inferiority observed. This is indicated by the even distribution of positive and negative values in the ELS gap column.

To visualize the solution evolution over time, we selected one instance for each of the stochastic methods (SA-PS, SA, GRB) and a single instance for the ELS algorithm, and plotted the results. Figure 4.4 illustrates the solution progression over time for instance $N = 20$ in the standard case. It is worth mentioning that the GRB method starts with a starting solution derived from hill-climbing, resulting in a later start in the recorded data. From the figure, we can observe that the ELS algorithm initially converges faster than the two SA methods. However, after approximately 600 seconds, the SA methods stabilize around their best solutions, while the ELS and GRB methods continue to make slow but steady improvements at much higher iterations. It is notable that Gurobi updates its best solution fewer than 7 times from its initial solution. ELS has the tendency to stagnate for a prolonged period before reaching a point where it can make multiple significant improve-

			SA-PS			SA			GRB		ELS	
N	r	λ	avg_obj	std	gap(%)	avg_obj	std	gap(%)	avg_obj	std	obj	gap(%)
20	1	0.7	612.625	1.392	6	613.431	1.641	5.9	651.836	9.958	672.976	-3.2
		1	822.429	2.779	5.6	821.797	4.611	5.7	871.584	15.435	844.24	3.1
		1.5	1166.909	8.077	5.5	1165.24	7.604	5.6	1234.757	15.059	1238.54	-0.3
	1.5	0.7	301.68	1.18	0.3	301.791	1.077	0.3	302.635	1.724	304.273	-0.5
		1	388.499	1.363	1.4	389.201	1.671	1.3	394.162	1.694	391.64	0.6
		1.5	530.69	2.891	2.5	530.521	2.933	2.5	544.174	3.74	546.86	-0.5
30	1	0.7	944.289	1.597	5	945.483	2.622	4.9	993.965	8.239	999.378	-0.5
		1	1271.879	3.161	4.9	1269.433	2.146	5.1	1338.035	10.368	1349.04	-0.8
		1.5	1814.55	4.409	4.9	1816.164	8.076	4.8	1907.238	16.748	1882.66	1.3
	1.5	0.7	441.774	3.224	2	441.193	2.275	2.1	450.583	0.431	452.401	-0.4
		1	565.976	3.23	3.7	569.648	3.561	3	587.564	0.652	590.16	-0.4
		1.5	769.578	6.895	5.5	769.235	5.59	5.5	814.432	0.639	816.47	-0.3
40	1	0.7	1378.792	4.225	2.1	1379.704	3.542	2.1	1408.978	5.013	1393.629	1.1
		1	1877.684	5.87	2.7	1878.439	4.171	2.7	1929.83	11.828	1907.89	1.1
		1.5	2703.257	5.135	3	2697.794	9.738	3.1	2785.529	5.345	2777.885	0.3
	1.5	0.7	598.617	3.733	17.8	599.037	3.956	17.8	728.677	104.472	655.234	10.1
		1	760.876	2.518	22.5	760.848	5.19	22.5	981.523	170.397	798.02	18.7
		1.5	1041.14	7.258	8.1	1034.648	10.783	8.7	1133.233	18.737	1122.845	0.9
50	1	0.7	1634.196	5.483	7.4	1635.53	6.696	7.3	1764.263	20.205	1763.054	0.1
		1	2227.026	9.811	7.7	2223.56	7.531	7.9	2413.179	26.608	2415.39	-0.1
		1.5	3188.783	8.42	8.7	3196.678	20.17	8.5	3491.768	34.245	3503.185	-0.3
	1.5	0.7	722.407	5.28	2.3	722.314	7.985	2.3	739.425	1.827	733.097	0.9
		1	941.844	10.541	3.6	942.3	9.486	3.6	977.488	2.88	975	0.3
		1.5	1283.24	10.866	6.3	1282.387	11.401	6.4	1369.846	4.247	1366.895	0.2

Table 4.4: Numerical results

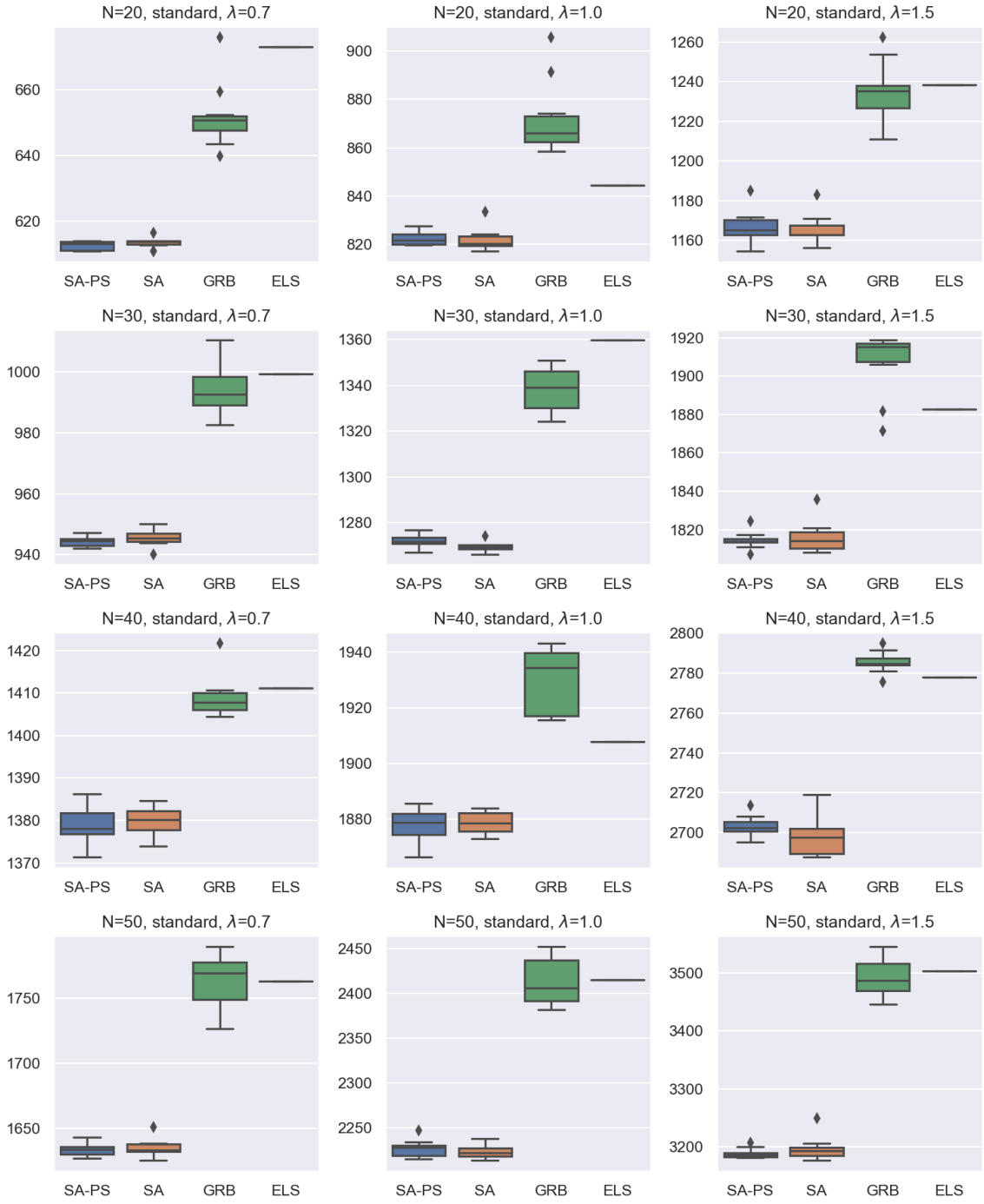


Figure 4.2: Boxplots of solutions for the standard model

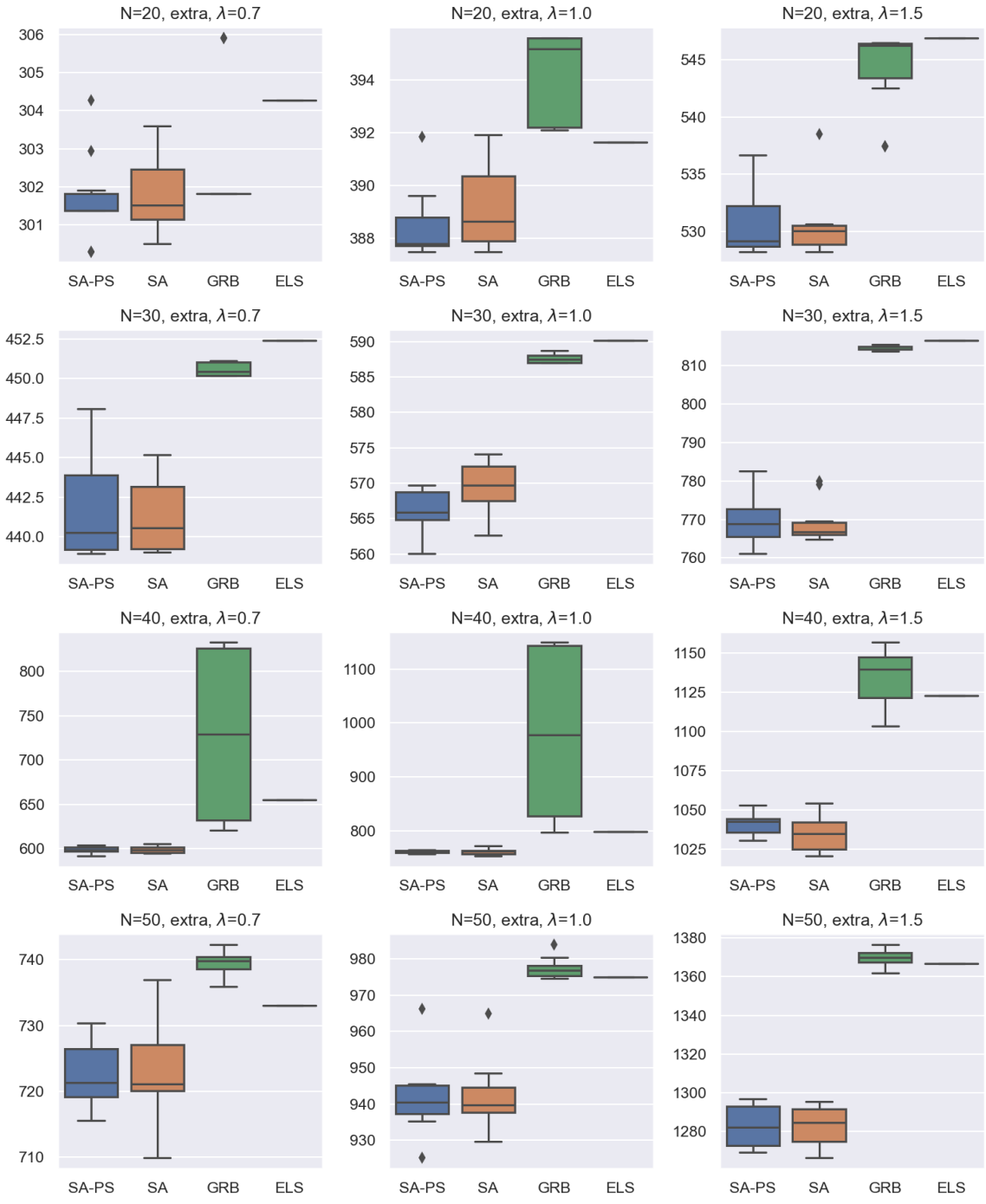


Figure 4.3: Boxplots of solutions for EA model

ments.

These findings provide insights into the performance dynamics of the four methods and their convergence behavior over time. The results suggest that for the instances we examined, the two SA methods are effective in finding high-quality solutions, while the ELS algorithm and GRB method exhibit a more gradual convergence.

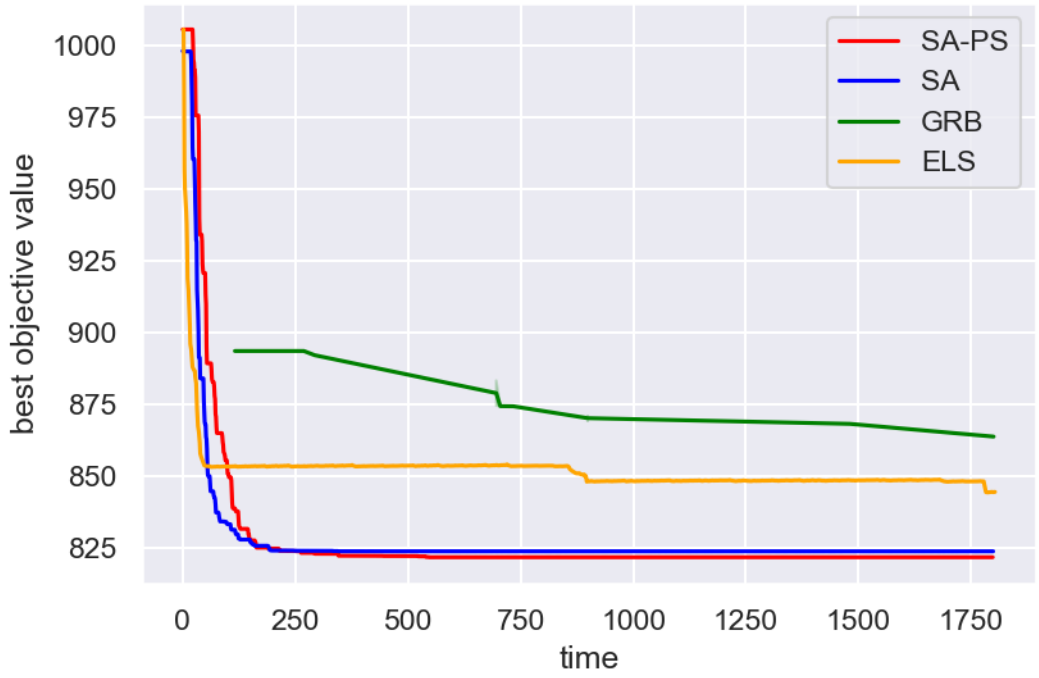


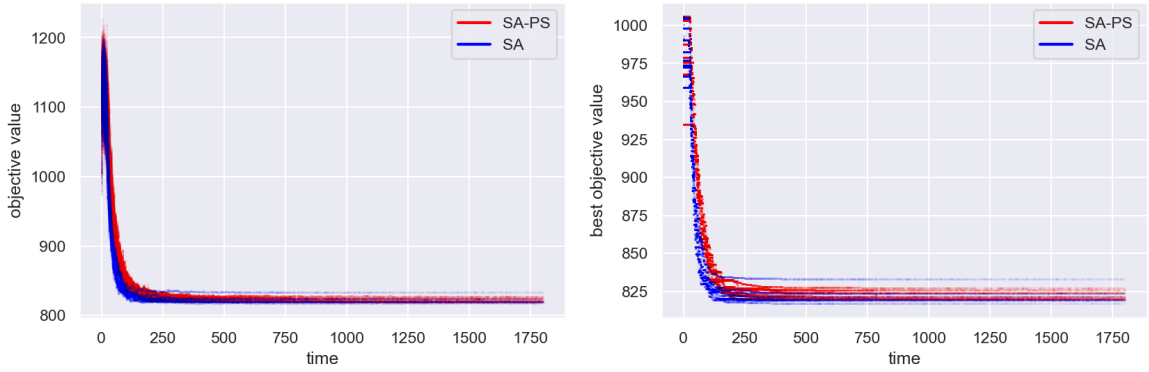
Figure 4.4: Evolution of four methods over time

4.3.1 SA methods comparison

In this subsection, we will compare the two simulated annealing methods, SA and SA-PS. We recall that SA-PS incorporates an adaptive neighborhood search technique by introducing a shaking procedure, which adjusts based on the number of rejections. The objective is to evaluate the influence of this added shaking procedure on the DRAP problem within this subsection of the study.

First, based on the numerical results presented in Table 4.4, we observe that the performance differences in terms of the obtained best objective values between the two methods, SA-PS and SA, are not substantial across all instances. On average, SA-PS achieves an objective value of 1166.1975, while SA achieves a slightly lower value of 1166.099. However, SA-PS consistently exhibits a smaller average standard deviation of 4.972 compared to SA's average standard deviation of 6.019, resulting in a decrease of 17.4%. This indicates that SA-PS demonstrates a higher level of robustness in its results.

Next, we analyze and compare the dynamic behaviors of the two methods using Figure 4.5. The graphs illustrate the results for all 10 instances of the standard model, where $N = 20$ and $\lambda = 1$. The numerical results indicate an average objective value of 822.429 for SA-PS and 821.797 for SA, specifically for this instance. It is worth noting that this particular instance is representative of the overall comparison between the two methods in terms of their behaviors across the remaining 23 instances. From the graphs, it can be observed that SA initially converges faster compared to SA-PS. However, once both methods reach a steady state, there is no significant difference in their convergence rates. Additionally, we note that SA exhibits a wider range of solution levels as the computation time increases, whereas SA-PS consistently produces more concentrated and consistent results across multiple runs. Furthermore, we observe that SA-PS displays a slightly higher peak in its solution search compared to SA, suggesting that SA-PS explores a broader range of solutions during its search process.



(a) objective value vs time

(b) best objective value vs time

Figure 4.5: Evolvement of SA methods over time

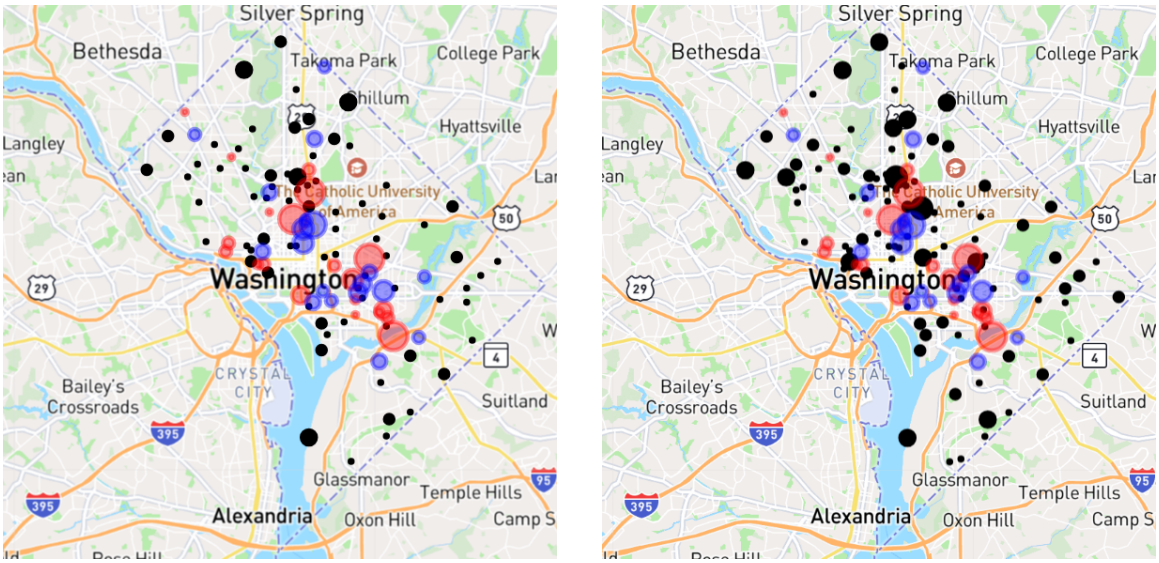
Overall, the comparison between the SA and SA-PS methods indicates similar convergence behavior and solution quality performance. SA demonstrates faster initial convergence, while SA-PS exhibits more robust results. This can be attributed to the fact that SA-PS tends to explore a broader region of the solution space before reaching convergence.

4.4 Solution visualization

In this section, we focus on visualizing the solutions for the instance with $N = 20$ and $\lambda = 1$ obtained by SA-PS. We consider two solutions in this case: one for the standard model and another for the EA model. These solutions correspond to objective values, representing the platform costs, of 819.73 and 388.52 respectively, from where we observe a decrease of 52.6% in the total cost for the extended model.

Figure 4.6 showcases the regional distribution of data for both the standard and EA models of the instance. Both models exhibit the same distribution of stations and bikes. In the standard model, there are a total of 100 rebalancers and bikes, while in the EA model, an additional 50 rebalancers are added, resulting in a total of 150 rebalancers with an excess

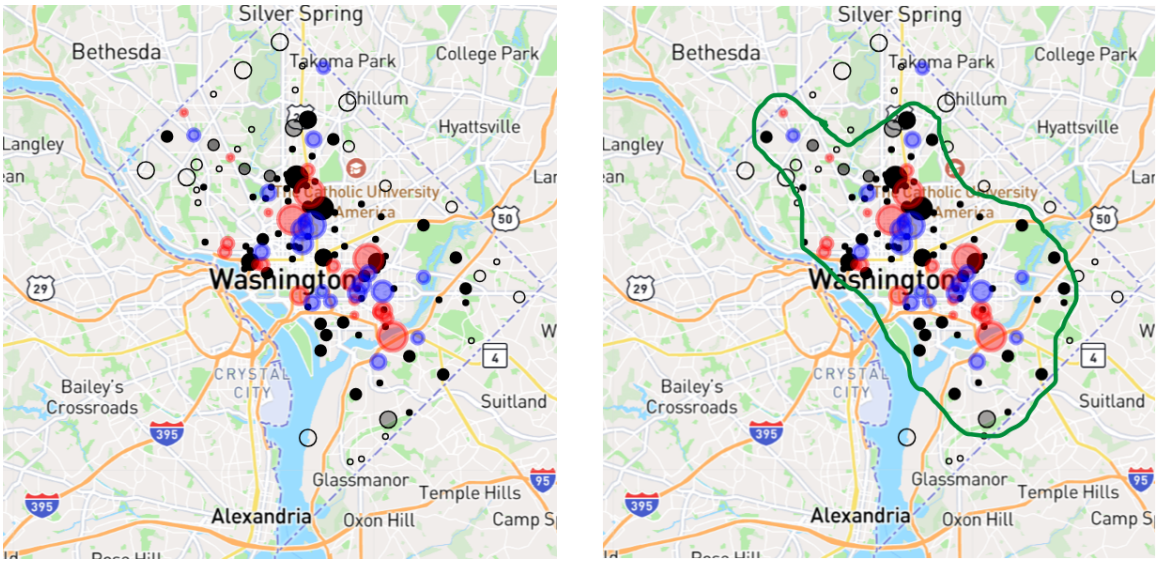
ratio of $r = 1.5$. The additional rebalancers in the EA model are randomly distributed, ensuring consistent overall distribution between the two models.



(a) The standard model (b) EA model

Figure 4.6: Bike and resident distribution for instance size $N = 20$

Based on our previous discussions in Chapter 2, we can infer that the platform cost for the EA model is lower compared to its standard counterpart. This is because the EA model provides more flexibility in selecting the 100 rebalancers to perform the actual job. As the cost is determined by the Euclidean distance between rebalancers and stations, we can expect the selected 100 rebalancers in the EA solutions to be more concentrated around the stations. This observation is clearly demonstrated by the solutions depicted in Figure 4.7. In the figure, each black circle represents the distribution of all rebalancers, while the opacity of the filled color represents the rebalancers that have been selected. A hollow black circle indicates that none of the rebalancers in that region have been picked, a black disk indicates that all rebalancers in that region have been assigned a job, and a gray disk indicates that some rebalancers have been selected. From the figure, it is evident that most



(a) (b)

Figure 4.7: Rebalancers picked for EA model

of the unpicked rebalancers are located far from the stations and tend to be distributed around the edges of the area. In other words, although the platform is not directly paying the cost of the empty travel time, it is still doing this indirectly because a low price is less likely to be accepted by a rebalancer who has to travel farther. In contrast, in regions where the stations are densely located, the black circles are completely filled in with black ink, indicating that all rebalancers in those regions have been selected. The right graph further illustrates this observation by drawing a rough circle within which the picked rebalancers reside.

Next, we examine the graphs depicting the flow distribution for the two solutions in Figure 4.8 and Figure 4.9. In these graphs, each line represents a connection between a pick-up and drop-off station, indicating the movement of bikes in a specific origin-destination direction. The presence of a line signifies that bikes are assigned to be relocated along that route, and the width of the line reflects the volume of bike traffic on that partic-

ular route. A wider line indicates a higher number of bikes being moved in that direction, while a thinner line represents a lower volume of bike movement.

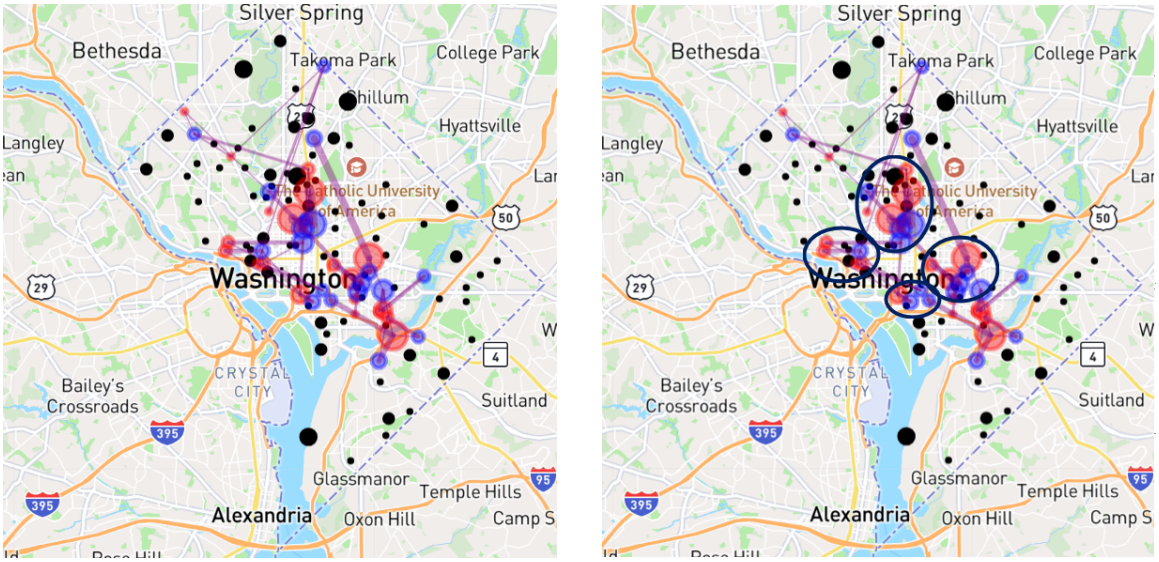
Upon observation, we can identify four distinct clusters of stations in the graph. To highlight these clusters, dark blue circles are placed on the right graphs of both figures, alongside the plain flow for comparison. In other words, most of the demand tends to be satisfied by matching stations that are very close together.

By comparing the two distributions, we can clearly observe significant variations in the flow after incorporating an additional 50 rebalancers into the system. Upon closer inspection, we notice that the flows become more concentrated within the clusters of stations, and fewer connections are established between different clusters. This observation aligns with our intuition, as it is more cost-effective to relocate bikes between stations that are close to each other rather than between stations that are farther apart.

So far, we can conclude from the visualization of the solutions that it effectively highlights the advantages of the EA model in terms of reducing platform costs and concentrating rebalancers around the stations for efficient operations. Additionally, the visualization demonstrates the aggregate distribution of bike flows, showcasing the increased concentration of flows within localized clusters. These insights provide a clear understanding of the improved efficiency and cost-effectiveness achieved by the EA model.

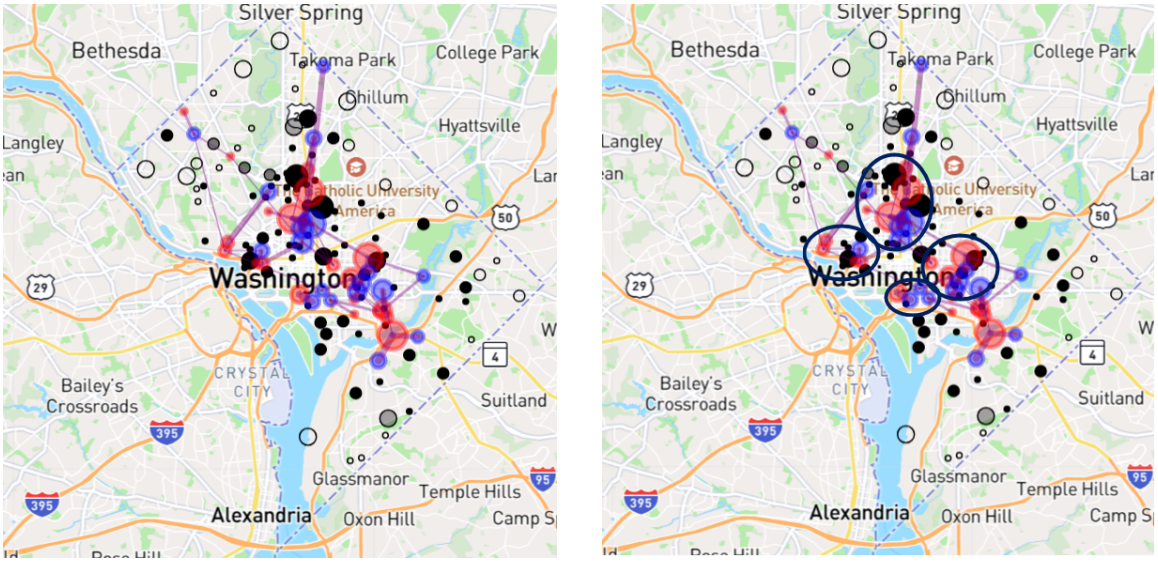
Lastly, we examine specific connections between rebalancers and their assigned origin-destination pairs in Figure 4.10. The colored labels highlight three locations of rebalancers, with the number beside indicating the count of rebalancers in each location.

The colored lines in the graph represent the origin-destination pairs assigned to the rebalancers, originating from the corresponding colored source points. The number on each



(a) (b)

Figure 4.8: Flow distribution from the standard model



(a) (b)

Figure 4.9: Flow distribution from EA model

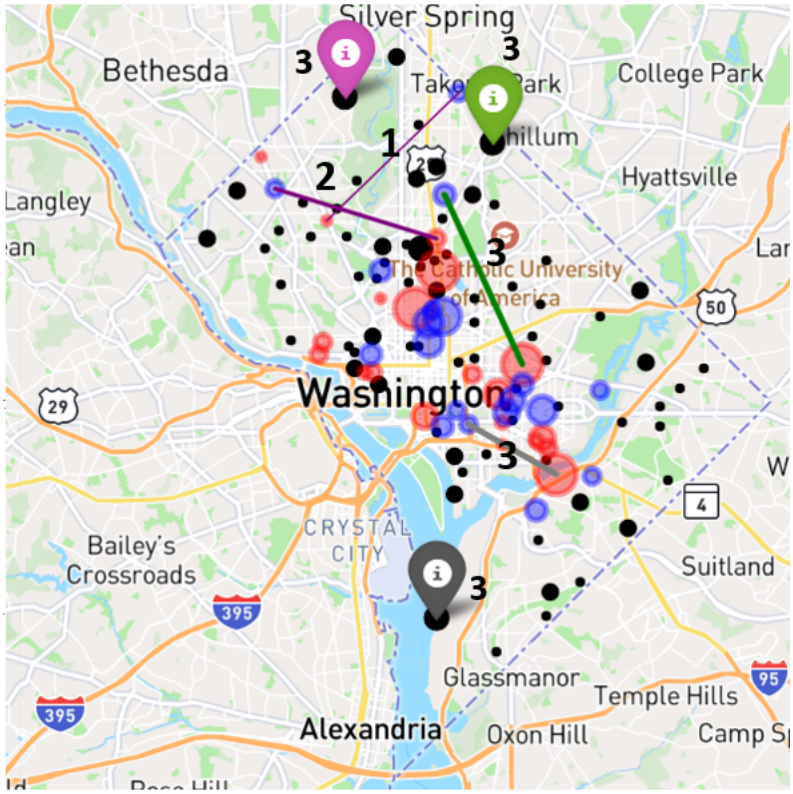


Figure 4.10: Specific connections of the standard model

line denotes the number of rebalancers from that location assigned to the specific route.

From the graph, we can observe the difference in assignments for rebalancers from different locations. For example, in the case of the purple rebalancers, one rebalancer is assigned to the thinner purple line, while the other two are assigned to the thicker purple line. These crossing lines depict distinct origin-destination directions and do not exhibit a similar distribution. This is a consequence of considering the needs of other rebalancers who are closer to the stations, as well as the size of the stations. In the cases of the green and grey points, both rebalancers are assigned to the same origin-destination pairs. Thus far, our observations indicate that the assigned origin-destination pairs are relatively close to the starting points for all three locations. However, we also note that the grey line does not represent the closest drop-off location in terms of optimizing personal cost. In fact, we can

identify a closer drop-off location located directly beneath the assigned pick-up location. This decision is made because the assignment process aims to optimize the overall cost, taking into account the needs of all rebalancers. Therefore, the assignment process involves a trade-off to minimize costs while satisfying the requirements of all rebalancers.

Chapter 5: Conclusion and future discussions

5.1 Conclusion

Our research primarily focused on addressing the rebalancing problem in bike-sharing systems. We formulated a rebalancing strategy by incorporating real-world observations of freelancers and developed a decentralized assignment optimization problem which we call DRAP. We then introduced theoretical results that provide a new perspective, emphasizing a lower-dimensional flow variable. We demonstrated the crucial role of this variable in determining the feasibility of DRAP and proposed an algorithm to address the remaining aspects of the problem. This subroutine significantly reduces the problem size, allowing us to employ search methods that mitigate the computational challenges.

To explore potential solution approaches, we considered various methods including branch-and-cut, simulated annealing, and the hill-climbing method. We made modifications to some of these approaches to enhance their performance in solving the problem. Specifically, we selected two simulated annealing methods: SA, which utilizes Gurobi with a warm-started solution obtained from a modified hill-climbing method called hierarchical hill-climbing, and SA-PS, an extended local search algorithm that does not terminate on its own.

In order to evaluate the performance of the different methods, we conducted numer-

ical experiments on 24 instances generated based on real data and a set of predefined parameters. Our experimental results consistently demonstrated that the simulated annealing methods, SA and SA-PS, outperformed the other approaches across different instances and scenarios. The extended local search algorithm also exhibited comparable performance to Gurobi in most cases, indicating its potential as a viable heuristic method. Additionally, the comparison between SA and SA-PS provided insights into the nature of the problem and the effectiveness of exploring slightly further from the initial starting point.

To gain a deeper understanding of the problem, we employed visualizations of the solutions obtained. By analyzing the regional and flow distributions, we were able to observe how the addition of rebalancers and variations in the cost matrix influenced the spatial patterns and traffic flows within the bike-sharing system. These visualizations help us better understand the trade-offs involved in rebalancer assignments and their impact on the overall performance of the system.

5.2 Future discussions

The model we examine represents a simplified version of the real-world rebalancing scenario we are interested in. In our study, we make the assumption that each rebalancer can only handle one bike at a time and that all the work must be completed within a single round. However, it is important to note that the actual real-world situation can be much more complex. Rebalancers may have the flexibility to handle multiple bikes simultaneously and perform consecutive tasks. As we continue our research, we plan to explore these aspects further and extend the model accordingly.

Our pricing structure has the potential for further extension. Currently, we have implemented a pricing mechanism based on the origin-destination pair. However, we aim to explore and evaluate the advantages and disadvantages of this pricing strategy by comparing it with alternative pricing approaches like the ones mentioned in Section 2.3.2 and 2.3.3 of Chapter 2. By conducting such comparisons, we can gain a deeper understanding of the strengths and weaknesses of our current pricing structure and potentially identify areas for improvement or alternative strategies that may offer additional benefits.

The algorithms we have primarily analyzed in our study rely on heuristic approaches, which offer upper bounds for the original optimization problem. However, obtaining lower bounds using Gurobi and branch-and-cut techniques based on LP relaxations has proven to be inefficient in our current implementation. Looking ahead, our future research efforts will focus on developing methods that can provide practical lower bounds for larger problem instances. This may involve devising specific algorithms or adopting heuristic approaches to enhance the estimation process of lower bounds. By doing so, we aim to improve the efficiency and effectiveness of obtaining lower bounds, enabling more accurate assessments of solution quality and facilitating decision-making in practical applications.

As an extension of traditional simulated annealing algorithms, the simulated annealing with probabilistic shaking technique deserves further attention and exploration. This research has demonstrated some of its benefits, particularly in terms of stability. However, there is still room for additional analysis and experimentation on various problem instances and parameter settings.

In Chapter 4, we analyzed the solutions obtained from the instances generated based on real data. Through this analysis, we gained insights that were visually represented

in the graphs. These insights have sparked our interest in developing heuristics specifically tailored to the characteristics of the instances, particularly focusing on refining the cost structure by incorporating real distance functions. A promising objective is to devise heuristics that can potentially improve the quality of solutions or expedite the convergence of algorithms. By leveraging these heuristics, we aim to enhance the overall efficiency and effectiveness of the bike rebalancing process.

Bibliography

[1] Meddin Bike-sharing. Meddin bike-sharing world map: Mid-2022 report. https://world.bikesharemap.com/, 2022. Accessed on March 24, 2023.

[2] Christine Fricker, Nicolas Gast, and Hanene Mohamed. Mean field analysis for inhomogeneous bike sharing systems. In *AofA*, volume DMTCS Proceedings vol. AQ, 23rd Intern. Meeting on Probabilistic, Combinatorial, and Asymptotic Methods for the Analysis of Algorithms (AofA'12), Montreal, Canada, July 2012. DMTCS.

[3] Financial Panther. How to become a bird charger and make money. https://financialpanther.com/bird-charger/, 2019. Accessed on March 24, 2023.

[4] Gilbert Laporte, Frédéric Meunier, and Roberto Wolfler Calvo. Shared mobility systems. *4OR*, 13(4):341–360, 2015.

[5] Gilbert Laporte, Frédéric Meunier, and Roberto Wolfler Calvo. Shared mobility systems: an updated survey. *Annals of Operations Research*, 271(1):105–126, 2018.

[6] Zhenpeng Zou, Hannah Younes, Sevgi Erdoğan, and Jiahui Wu. Exploratory analysis of real-time e-scooter trip data in washington, d.c. *Transportation Research Record*, 2674(8):285–299, 2020.

[7] Ashish Kabra, Elena Belavina, and Karan Girotra. Bike-share systems: Accessibility and availability. *Management Science*, 66(9):3803–3824, 2020.

[8] Chong Yang Goh, Chiwei Yan, and Patrick Jaillet. A locational demand model for bike-sharing. January 6 2019. Available at SSRN: https://ssrn.com/abstract=3311371 or http://dx.doi.org/10.2139/ssrn.3311371.

[9] Ezgi Eren and Volkan Emre Uz. A review on bike-sharing: The factors affecting bike-sharing demand. *Sustainable Cities and Society*, 54:101882, 2020.

[10] Jenn-Rong Lin and Ta-Hui Yang. Strategic design of public bicycle sharing systems with service level constraints. *Transportation Research Part E: Logistics and Transportation Review*, 47(2):284–294, 2011.

[11] Christine Fricker and Nicolas Gast. Incentives and redistribution in homogeneous bike-sharing systems with stations of finite capacity. *EURO Journal on Transportation and Logistics*, 5(3):261–291, 2016.

[12] Sharon Datner, Tal Raviv, Michal Tzur, and Daniel Chemla. Setting inventory levels in a bike sharing network. *Transportation Science*, 53(1):62–76, 2019.

[13] Ahmadreza Faghih-Imani, Robert Hampshire, Lavanya Marla, and Naveen Eluru. An empirical analysis of bike sharing usage and rebalancing: Evidence from barcelona and seville. *Transportation Research Part A: Policy and Practice*, 97:177–191, 2017.

[14] Hipólito Hernández-Pérez and Juan-José Salazar-González. The one-commodity pickup-and-delivery travelling salesman problem. In *Combinatorial Optimization — Eureka, You Shrink!*, pages 89–104. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[15] Jesus Osorio, Chao Lei, and Yanfeng Ouyang. Optimal rebalancing and on-board charging of shared electric scooters. *Transportation Research Part B: Methodological*, 147:197–219, 2021.

[16] Jinjia Huang, Mabel C. Chou, and Chung-Piaw Teo. Bike-repositioning using volunteers: Crowd sourcing with choice restriction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13):11844–11852, May 2021.

[17] Gérard P. Cachon, Kaitlin M. Daniels, and Ruben Lobel. The role of surge pricing on a service platform with self-scheduling capacity. *Manufacturing & Service Operations Management*, 19(3):368–384, 2017.

[18] Kostas Bimpikis, Ozan Candogan, and Daniela Saban. Spatial pricing in ride-sharing networks. *Operations Research*, 67(3):744–769, 2019.

[19] Siddhartha Banerjee, Daniel Freund, and Thodoris Lykouris. Pricing and optimization in shared vehicle systems: An approximation framework. *Operations Research*, 70(3):1783–1805, 2022.

[20] Henrik Sternberg and Magnus Andersson. Decentralized intelligence in freight transport—a critical review. *Computers in Industry*, 65(2):306–313, 2014.

[21] Ozgun Caliskan Demirag and Julie L. Swann. Capacity allocation to sales agents in a decentralized logistics network. *Naval Research Logistics (NRL)*, 54(7):796–810.

[22] Alexander Kleiner, Bernhard Nebel, and Vittorio Amos Ziparo. A mechanism for dynamic ride sharing based on parallel auctions. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume One*, IJCAI'11, page 266–272. AAAI Press, 2011.

[23] Mehdi Nourinejad and Matthew J. Roorda. Agent based model for dynamic ridesharing. *Transportation Research Part C: Emerging Technologies*, 64:117–132, 2016.

[24] Okan Arslan, Ola Jabali, and Gilbert Laporte. Exact solution of the evasive flow capturing problem. *Operations Research*, 66(6):1625–1640, 2018.

[25] Roy Jonker and Ton Volgenant. Improving the hungarian assignment algorithm. *Operations Research Letters*, 5(4):171–175, 1986.

[26] H. W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.

[27] Manfred Padberg and Giovanni Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33(1):60–100, 1991.

[28] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973.

[29] Molly A. O'Neil and Martin Burtscher. Rethinking the parallelization of random-restart hill climbing: A case study in optimizing a 2-opt tsp solver for gpu execution. In *Proceedings of the 8th Workshop on General Purpose Processing Using GPUs*, page 99–108. Association for Computing Machinery, 2015.

[30] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13(5):533–549, 1986. Applications of Integer Programming.

[31] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[32] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[33] V. Granville, M. Krivanek, and J.-P. Rasson. Simulated annealing: a proof of convergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):652–656, 1994.

[34] P.J.M. Van Laarhoven and E.H.L. Aarts. *Simulated annealing: Theory and applications*. Springer, 1987.

[35] Bruce Hajek. Cooling schedules for optimal annealing. *Mathematics of Operations Research*, 13(2):311–329, 1988.

[36] Yaghout Nourani and Bjarne Andresen. A comparison of simulated annealing cooling strategies. *Journal of Physics A: Mathematical and General*, 31(41):8373, oct 1998.