November 2021

Theory and Algorithms for Systems Optimization

Vahid Mahmoodian
University of South Florida

Theory and Algorithms for Systems Optimization

by

Vahid Mahmoodian

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

Co-Major Professor: Hadi Charkhgard, Ph.D.
Co-Major Professor: Yu Zhang, Ph.D.
Ankit Shah, Ph.D.
Andrei Barbose, Ph.D.
Changyun Kwon, Ph.D.
He Zhang, Ph.D.

Date of Approval:
October 15, 2021

Keywords: Free-floating bike sharing system, Query batching, Multiplicative programming,
Crowd-sourcing last-mile delivery

Dedication

To my lovely family.

Acknowledgments

First of all, I sincerely thank Dr. Hadi Charkhgard, for his incredible support and dedicated guidance throughout my Ph.D. since my admission to my last days at USF. It was an absolute pleasure to work under his supervision and I enjoyed every second of it. He encouraged, motivated, guided, helped, and taught right at the moment I needed and in the best possible way. I could never appreciate more what I learned during this four-year journey.

I would like to thank my second advisor Dr. Yu Zhang for her support, guidance and encouragement as well. I gained so much knowledge in transportation while working with her team at Smart Urban Mobility Laboratory whose value cannot be fit in words. I am also grateful of having the chance to work with so many knowledgeable professors, especially Dr. Dayarian, and colleagues both in Multi-Objective Optimization Laboratory and Smart Urban Mobility Laboratory that helped me grow not only in science but in personal life.

The last but not the least, I would like to appreciate all the faculty and staff at the department of Industrial and Management Systems Engineering for the extremely warm and friendly atmosphere they have provided for Ph.D. students to pursue their career and dreams. I also appreciate my committee for their valuable comments. I would like to acknowledge the financial support by the National Science Foundation under Grant No. 1849627.

Table of Contents

List of Tables

List of Figures

Abstract

This dissertation presents four sets of contributions in the field of theory and algorithms for system optimization. In the first set, we introduce a simulation optimization method for redistributing bikes in a free-floating bike sharing system. The second set of contributions is a framework for batching queries in large databases to optimize the data retrieval time. The third set includes two branch-and-bound algorithms to solve minimum multiplicative programming problems and one branch-and-bound algorithm to solve the maximum form of the mentioned problems. At last, the fourth set presents an approach to fairly assign delivery tasks in an outsourcing last-mile delivery system using Nash Social Welfare solution.

In this dissertations, we propose dynamic hubbing and hybrid rebalancing, i.e. combining user-based and operator-based strategies, along with a novel multi-objective simulation optimization approach to solve bike rebalancing problem in free-floating bike sharing systems as the first set of contributions. Thanks to advances in location and communication technologies, the need for docks and stations in bike sharing systems has been removed. But rebalancing bikes is still a challenging problem in these bike sharing systems. In the proposed model, the users are encouraged to return the bikes to predetermined hubs using an incentive program. Then, for the remaining unbalanced bikes, an operator-based rebalancing operation will be planned. This method determines the number of hubs, their locations, the start time for initiating the user incentive program, and the amount of incentive by considering two conflicting objectives, i.e., level of service and total rebalancing cost. Our results on a real case, Share-A-Bull running on the campus of the University of South Florida, show that a hybrid rebalancing and dynamic hubbing strategy significantly improves the level of

service and reduces the total rebalancing costs. We, also, show this method can entail some promising environmental impacts by decreasing greenhouse gas emission.

Our second set of contributions presents an approach to optimize query-batching in large database systems. Query batching in database systems is very similar to a well-studied problem called order batching in the warehousing context. However, unlike order batching, the literature of optimization techniques for query batching is outstandingly rare. In this dissertation, we develop a Mixed Binary Quadratic Program (MBQP) to optimize query-batching using predicted query retrieval time. We use a simple regression model for this prediction that can be quickly and dynamically trained. Also, two iterative heuristics are proposed to solve the mentioned problem fast enough. The results on two benchmark databases show that the returning data for batches formed by the proposed method can improve the performance by 19–61.8% comparing to returning the required data for all the queries together.

In the third set of contributions, this dissertation presents three branch-and-bound algorithms to solve multiplicative programming problems; two algorithms for minimization form and one for maximization form of the problem. We showed that the algorithms for the minimization form of the problem outperform SCIP, a genetic-purpose solver, by a factor of over 10 on many instances out of 960 test instances. We, also, numerically show that selecting the best algorithm between the proposed algorithms depends on the class of the instances. Specifically, the performance of the algorithms is highly dependant upon the dimension of decision and criterion space of the instances, since the search mechanism in one of the algorithms works over the decision space and the other one over the criterion space. The algorithm for maximization form is limited to only bi-linear maximum multiplicative programs that, yet, has many applications such as finding Nash bargaining solution (Nash social welfare optimization), capacity allocation market, reliability optimization, etc. We proposed several enhancements to tighten the bounds and improve the performance of the algorithm. The result on 400 randomly generated instances is compared with the one ob-

tained by the latest algorithm in the literature and the Second Order Cone Programming solver of CPLEX. We show that the proposed algorithm outperforms the winner of the two mentioned rival methods by the factor of 6.54 on average.

As the last set of contributions in this dissertation, we introduce an equitable crowdsourced last-mile delivery model using Nash social welfare solution for coalition points among different drivers. In this model, the efficiency of the delivery company is guaranteed by putting a cap on the deviation of its cost from the minimum value. Fairness considerations that capture non-monetary performance requirements including equitable service provision to external stakeholders have a long history in routing applications in the public/nonprofit sector. In the private logistics service sector, however, such considerations are new and growing due to public or governmental pressures to improve equity in workload allocation among internal stakeholders, i.e., the drivers or other personnel providing the service. This is more crucial when employing crowdsourced workforce considering their inherent heterogeneity in terms of skills, availability, and productivity levels. A column generation method is developed to solve and study the behavior of the proposed model. We studied the changes in company's cost, drivers' total profit, and the level of achieved equity among the drivers when the three parameters of the problem vary. We show the superiority of the proposed method to the well-known max-min approach in terms of balancing the workload among the driver and their efficiency. For example, we observe a 50.15% decrease in the average range of profit ratios among drivers only by 5% deviation of the company's cost from the minimum value.

Chapter 1: Introduction

In this Chapter, we discuss the research questions and motivation of the presented contributions in this dissertation. All of the research questions are related to the theories and applications of multi-objective optimization in equity and transportation. In the following, we break down these questions in more detail and describe their motivation in Section 1.1. Also, a list of our contributions, including both published and submitted papers, is provided in Section 1.2. Finally, the outline of the dissertation is presented in Section 1.3.

1.1 Motivation

The first research question that this dissertation tries to answer is about rebalancing bike distribution in a newly-emerged type of bike sharing systems called free-floating bike sharing systems. Spatial and temporal imbalance of demand and supply is an inevitable problem that roughly exists in all kinds of sharing businesses. Thanks to advances in real-time location and information technologies, such as GPS and the internet, the situation became ready to get rid of docks and traditional payment stations in bike sharing systems, which brought about the growth of free-floating bike sharing systems. Beside all the facilitation and flexibility, these new bike sharing systems offer no advantage comparing to their older generations in terms of rebalancing problem. There are two groups of rebalancing methods in the literature of bike sharing systems; first, static rebalancing that is done from time to time and commonly when the demand is very low, and second, dynamic rebalancing that is carried out continuously while the system runs. The first work in this dissertation takes advantage of two mentioned rebalancing methods and specifications of free floating bike sharing systems to answer the following question:

- Q1: what is the best way of rebalancing bikes in a free-floating bike sharing system considering both cost and level of service?

Traditional database management systems process queries individually, often in a first-come-first-serve order. However, by the growth that happened both in the volume of needed data and computer hardware, shared work systems that can process queries in batches were introduced. Typically, there is a high likelihood for the queries on a database to share computing resources such as IO (reading/writing data from/to disk), CPU (processing data by the processor), memory, etc. Therefore, executing queries as batches could introduce savings in terms of computing resource usage and processing time, which raises the research question of the second work in this dissertation:

- Q2: How should a given set of queries be "optimally" partitioned into one or multiple batches of possibly different sizes?

Most of the real world optimization problems have more than one objective that are often conflicting, where multi-objective optimization methods are used to provide the decision makers with a view of trade-offs among considered objectives. In recent years, however, a few studies have applied the multi-objective techniques to handle the complexity of some single objective problems such as minimum Multiplicative Programs and Maximum Multiplicative Programs. In the third work of this dissertation, we attempted to answer the following questions:

- Q3: Can we use the multi-objective optimization techniques to solve the general minimum multiplicative problem?

- Q4: Can we further accelerate the solution of the maximum bi-linear multiplicative program?

We developed two algorithms to address Q3 and proposed one algorithm that outperforms the fastest algorithm in the literature.

Finally, as the last work, we address optimizing the equity among volunteer drivers in a crowdsourcing last-mile delivery system. Workload assignment equity has been receiving more attention over the past decade and delivery systems are not an exception. Setting workload equity among a delivery system with a heterogeneous fleet of drivers is particularly difficult as the capacity and distance of origin and destination of drivers which directly influence their workload can be different. In this dissertation, we used the Nash Social Welfare as a measure and maximum multiplicative programming to formulate an equitable crowdsourcing last-mile delivery system. This research is answering the following research question:

- Q5: How to assign the delivery tasks among occasional drivers in a crowd-sourcing last-mile delivery system to distribute the available benefit as equitable as possible?

1.2 Contributions of the Dissertation

Our attempt to answer the mentioned questions led to three published, one accepted for publication, and one under review paper (by the time this dissertation was written). It should be mentioned all papers are published or accepted in peer-reviewed journals. In the following, we provide a list of our contributions.

- P1: Vahid Mahmoodian, Yu Zhang, Hadi Charkhgard (2021). Hybrid Rebalancing with Dynamic Hubbing for Free-Floating Bike Sharing Systems. *International Journal of Transportation Science and Technology* (To appear).

- P2: Eslami, M., Mahmoodian, V., Dayarian, I., Charkhgard, H., & Tu, Y. (2020). Query batching optimization in database systems. *Computers & Operations Research*. https://doi.org/10.1016/j.cor.2020.104983 (Available online).

- P3: Mahmoodian, V., Charkhgard, H., & Zhang, Y. (2021). Multi-objective optimization based algorithms for solving mixed integer linear minimum multiplicative pro-

grams. *Computers & Operations Research*. https://doi.org/10.1016/j.cor.2020.105178 (Available online).

- P4: Mahmoodian, V., Dayarian, I., Ghasemi, P., Zhang, Y. & Charkhgard, H. (2021). Criterion Space Branch-And-Cut Algorithm For Mixed Integer Bi-Linear Maximum Multiplicative Programs. *INFORMS Journal on Computing* (To appear).

- P5: Mahmoodian, V., Dayarian, & Charkhgard, H. (2021). Equitable Workload Allocation in VRP with Heterogeneous Drivers (Under review).

1.2.1 Related Publications and Preprints

We present our paper P1 in Chapter 2. In paper P1, we answer question Q1. We propose dynamic hubbing and hybrid rebalancing bikes in a free-floating bike sharing system and solve the problem with a novel multi-objective simulation optimization approach. Given historical usage data and real-time bike GPS location information, the basic concept is that dynamic hubs are determined to encourage users to return bikes to desired areas towards the end of the day through a user incentive program. Then, for the remaining unbalanced bikes, an operator-based rebalancing operation will be scheduled. The proposed modeling and optimization solution algorithm determines the number of hubs, their locations, the start time for initiating the user incentive program, and the amount of incentive by considering two conflicting objectives, i.e., level of service and rebalancing cost. We implemented the proposed method on the Share-A-Bull free-floating bike sharing system at the University of South Florida. Results show that a hybrid rebalancing and dynamic hubbing strategy can significantly reduce the total rebalancing cost and improve the level of service. Moreover, taking the advantage of crowd-sourcing (or job-sharing) reduces the negative impacts of the operation of rebalancing vehicles.

We present our paper P2 in Chapter 3. In paper P2, we answer question Q2. We develop a Mixed Binary Quadratic Program (MBQP) to optimize query-batching in a database

system. This model uses the coefficients of linear regression on the query retrieval time, trained by a large set of randomly generated query sets. We also propose two heuristics, the so-called restricted-cardinality search methods I and II, for solving the proposed MBQP. To demonstrate the effectiveness of our proposed techniques, we conduct a comprehensive computational study over randomly generated instances of two well-known database benchmarks.

We present our paper P3 and P4 in Chapter 4 and 5. In papers P3 and P4, we answer questions Q3 and Q4, respectively. We present two new algorithms for Mixed Integer Linear minimum Multiplicative Programs. A computational study on 960 instances demonstrates that the proposed algorithms outperform a generic-purpose solver, SCIP, by a factor of more than 10 on many instances. Also, we introduce a branch-and-bound algorithm to solve Mixed-Integer Bi-Linear Maximum Multiplicative Programs. The proposed algorithm applies multi-objective optimization principles to solve this class of problems exploiting a special characteristic that holds for the optimal solution. Moreover, several enhancements are applied and adjusted to tighten the bounds and improve the performance of the algorithm. The performance of the algorithm is investigated by 400 randomly generated sample instances. The obtained result is compared against the outputs of the mixed-integer Second Order Cone Programming solver in CPLEX and a state-of-art algorithm in the literature for this problem.

We present our paper P5 in Chapter 6. In paper P5, we answer question Q5. We introduce an equitable crowdsourced last-mile delivery model by adopting Nash social welfare solution as the coalition point among different drivers. While the ultimate goal is to maximize the equity and efficiency of drivers, the efficiency of the company is guaranteed by putting a cap on the deviation of the company's cost from the least-cost solution value. To solve the proposed new formulation, a column generation method is developed and used to study the behavior of the model. Through a comprehensive computational study, we investigate the

behavior of the system in terms of the company's cost, drivers' total profit, and the level of achieved equity among the drivers when the main parameters of the problem vary.

1.3 Outline of the Dissertation

The remaining content of this thesis is organized as follows:

- In Chapter 2, we propose a hybrid hubbing and rebalancing method for free-floating bike sharing systems. In this work, the literature on the redistribution problem in bike sharing systems is investigated, categorized, and discussed from different angles. Also, the difference and challenges of newly emerged free-floating bike sharing systems in terms of this problem are detailed. Then, a simulation optimization-based approach is introduced to daily determine the number and location of hubs and the specifications of an incentive program to redistribute bikes in a free-floating bike sharing system. The performance of the proposed method is examined and analyzed on a real case, called Share-A-Bull, that is running on the campus of the University of South Florida.

- In Chapter 3, we present a new query batching optimization method in database systems. In this work, we formulate the query batching problem in the form of a mixed binary quadratic program and introduce two heuristic algorithms to solve the problem in an iterative fashion. In this method, we use the linear regression to estimate the needed time for processing a certain batch of queries given the required tables and the number of queries. Also, the effectiveness of the proposed techniques is studied by a comprehensive computational experiment.

- In Chapter 4, we present two new algorithms to solve mixed-integer linear minimum multiplicative programs. One of these algorithms searches the decision space, and the other one uses a desirable feature in these problems to find the optimal solution in the criterion space. In specific, the optimal solution is one of the optimal non-dominated points when each of multiplicative terms is considered a separate objective.

Our experiment on 960 random instances shows that the proposed algorithms can outperform the available generic solver SCIP. We also show that the algorithms are faster than CPLEX for those instances that can be linearized.

- In Chapter 5, we extend our idea to solve mixed-integer bi-linear maximum multiplicative programs. This class of problems appears in many applications, especially in finding Nash bargaining solution. We featured the algorithm with several enhancements to tighten the bounds and improve the performance. The outcome is studied on 400 random instances and compared by those obtained from Second Order Cone Programming (SOCP) solver of CPLEX and the latest algorithm in the literature. The result shows that the proposed algorithm is over 6 times faster than the fastest rival solver.

- In Chapter 6, we formulate and solve the Nash bargaining problem among the drivers of a crowdsourcing last-mile delivery system. In this work, the ultimate goal is to maximize the efficiency and equity of drivers under a limited deviation of the company's cost from the optimal cost. We developed a column generation approach to solve and study the behavior of the model for different values of parameters. Our targeted characteristics of the model are the company's cost, drivers total efficiency, and achieved level of equity among the drivers.

- In Chapter 7, we summarise the results of the dissertation and provide some directions for future research.

Chapter 2: Hybrid Rebalancing with Dynamic Hubbing for Free-Floating Bike Sharing Systems

In this chapter, we present paper P1. In this work, we develop a simulation optimization-based dynamic redistribution approach for free-floating bike sharing systems. The objectives of this method are cost and level of service, and takes advantage of multi-objective optimization techniques to provide the decision makers with several non-dominated solutions.

2.1 Introduction

In recent years, bike sharing has received much attention [2, 3, 4, 5], as highlighted in a study conducted by Shaheen et al. [6] that indicates the existence of 150 bike sharing systems across more than 30 countries until 2010. This number was expected to increase to 1,608 programs by the end of 2018 [7], indicating a sharp rise in this business. The booming of bike sharing was driven by many factors, including that environmental issues in transportation have attracted more attention in recent years [8], and the public desires better first/last-mile solutions for seamless multi-modal transportation [9, 10].

It is known that for bike sharing systems, the flow of customers can change the temporal and spatial distribution of bike availability. Easing the imbalance of demand and supply is key to ensuring the efficiency of bike sharing systems [11]. As described in existing literature (see, for example, [12, 13]), rebalancing of bikes can be done either by users with an incentive program or by the operator with a fleet of rebalancing vehicles. In an operator-based rebalancing method, the operator collects and repositions bikes to balance a certain number of bikes to predetermined locations. The rebalancing can be static or dynamic or a combination of the static and dynamic [14]. Static rebalancing means that the bikes are

rebalanced without interference of user activities and is usually operated during the night when no customers borrow or return bikes. In contrast, dynamic rebalancing is operated periodically during the day when the borrowing and returning of bikes continuously occur.

Advances in real-time location and information technologies, such as GPS and internet, prepared the situation to remove the docks from bike sharing systems and allowed free-floating bike sharing systems (FFBSs) to flourish. This change reduced the investment costs, but at the same time, entailed some new operational challenges. For example, every single bike, in a FFBS, is a moving station holding one bike whose location can constantly alter during the service time window and within service area, which increases the complexity of rebalancing problem of FFBSs and makes the measures of level of service of FFBS different from those used commonly for SSBSs.

In this study, a dynamic hubbing and hybrid rebalancing method is introduced to redistribute the bikes to desired areas towards the end of the day considering two objectives, cost and level of service. The proposed approach, uses the prospect demand of the second day to locate the hubs in the current day and takes advantage of incentives to encourage the users to return the bikes back to the hubs in the favor of rebalancing. At the end of the day, an operator-based static relocation completes the rebalancing for the remaining unbalanced bikes. The number and location of the hubs, the time of starting user incentive program, and the amount of incentives for each discretized time slot are optimized using a multi-objective simulation optimization method.

The remainder of this paper is organized as follows. In Section 2.2, the literature of dynamic rebalancing, especially using incentives, is reviewed. Section 2.3, provides a detailed description of the problem on which this study focuses. Section 2.4 explains the proposed approach, Section 2.5 presents the numerical results and analyses of a real-world dataset, and Section 2.6 includes final remarks.

2.2 Literature Review

Static rebalancing has been studied widely in literature on shared mobility, especially bike sharing systems [15, 16, 17]. Some researchers started evolving the idea of dynamic rebalancing and developing methods to apply it to different types of vehicle sharing systems. Most of these studies focused on operator-based dynamic rebalancing methods; however, some deal with incentive programs to encourage users to contribute to dynamic redistribution. One of the first works in this category was by Chemla et al. [18], who built a theoretical framework for dynamic redistribution of bikes in a station-based bike sharing system (SBBS). The authors proved the hardness of the problem and proposed five heuristic algorithms to solve it by incorporating a pricing technique. Fricker and Gast [19] presented an Ordinary Differential Equation (ODE) model to a homogeneous bike sharing system and studied its steady-state behavior when there is no incentive or redistribution mechanism. They then investigated the influence of a two-choice incentive program in which it was suggested to users to drop off the bike at the least congested station. The result shows that this simple incentive mechanism improves the performance by an exponential factor. This research also included determining the speed by which one solo truck must re-distribute the bikes between empty or over-filled stations of a basic (without incentive) bike sharing system to satisfy a minimum quality of the service. Li and Shan [20] developed a Bidirectional Incentive Model (BIM) by dividing customers into two groups—commuters and leisure travelers. Bidirectional incentive refers to the idea that in this model, not only customers who are biking in a high traffic flow are charged (or penalized) but also inverse-traffic flow travelers are awarded. Accordingly, they categorized the travels of these customers into three groups based on the fact that their trip was in the direction of the peak flow or inverse flow or only one of their origin or destination was among high-traffic stations. The authors used the Stackelberg Game to model this problem in three scenarios—implementation of BIM by the operator

without government subsidy, implementation with government subsidy, and implementation by cooperation of both the operator and the government.

Some studies take advantage of mathematical modeling's benefits to deal with incentivized rebalancing problems. [21] presented an incentivized relocation model for station-based vehicle sharing systems in which the origin and destination of trips along with the willingness to enter into the incentive program are given by users in the reservation process. The model collects the information of received reservations for short time intervals and then optimizes the transfers in the destinations so that the profit of the vehicle sharing system is maximized. Their study examined two schemes in offering the incentives. In the first, the authors considered the profit of the system; in the second, the utility of users resulted from these destination changes and the received incentive was taken into account. The authors admitted that using the second scheme is impractical because of its need for inputting a large amount of information by the user through the mobile application's interface. Although the authors tested their model on a bike sharing system, the requirements of the model fit more with car sharing systems. In addition, the study did not consider any uncertainty in the customer behavior and assumed that a specific ratio of users would agree to enter the incentive program and accept an offered incentive. Haider et al. [22] developed a deterministic bi-level model to find the best price in a predetermined range in a bike sharing system. In their model, it is assumed that customers will always select the cheapest option from every suggested alternative origin-destination pair of stations. The added walking and biking of customers to pick up or drop off the bike at other stations rather than their actual desired ones is deterministically transformed to a cost in this model. They solved the model by changing it to a single level model, as the integrality constraints of the lower level model can be removed because of the unimodularity. Furthermore, their model requires the user destination in the reservation process, which is not applicable in most bike sharing systems. Pfrommer et al. [23] combined an operator-based dynamic rebalancing model with an incentive optimization model. The latter transforms the value of a user's wasted time as a result

of accepting the incentives to a random cost and assumes that he/she will select the most profitable offer. Such random behavior is simply formulated as a linear probability vector in the optimization model. In the model, the value of offered incentives is not optimized and is limited to only a maximum value.

Common features of the above-mentioned incentive-based rebalancing methods are that all are designed for station-based sharing systems, and system demand is measured and predicted based on pick-ups and drop-offs at stations. Accordingly, the objective of these models or performance measure is the number of rejected demands caused by empty stations (for pick-ups) or full stations (for drop-offs). However, a new type of bike sharing system, the dockless/free-floating bike sharing system, has emerged and is being implemented in many cities. Such systems do not need docking stations, which consume a large percentage of start-up investment in station-based bike sharing [9]. With the built-in GPS device, the free-floating bike sharing system allows users to leave a bike almost anywhere, which removes the docking capacity limitation in these systems. The locations of bikes are tracked by GPS and displayed on smart phones or web-based apps. Users can access the app to locate the bikes and reserve them if the program is designed to allow them to do so. This means that a demand can enter the system from any point inside or the margin of the system's service area. Given these advantages, free-floating bike sharing has been expanded dramatically around the world. However, the advantages of flexibility also raise operational challenges. Compared to station-based bike sharing, the rebalancing of a fully free-floating bike sharing system is more difficult to manage because of the higher uncertainty in demand. Thus, many bike sharing companies are running free-floating bike sharing systems as station-based or partially station-based, and some start-up bike sharing companies do not consider rebalancing at all.

Observing the needs, the research community has put in effort to develop methods to tackle the rebalancing problem of free-floating bike sharing [24, 25]. For example, Pal and Zhang [9] presented a mixed integer linear programming model for solving the static complete

rebalancing problem in a free-floating bike sharing system. They solve this problem with a hybrid nested large neighborhood search algorithm for both single and multiple vehicles. In another study, Pal et al. [26] extracted the mobility patterns and imbalance of a free-floating bike sharing system by considering the interaction of independent variables in a statistical model on historical trips and weather data. In two very recent studies, Caggiani et al. [27, 28] introduced methods for clustering the region under consideration based on the temporal patterns and optimizing the weighted objective of two measures through the clusters; one is the number of lost users and the other is the cumulative time in which the number of bikes in a cluster is below a threshold. Also, Du et al. [29] developed a mathematical model for complete static rebalancing problem in a free-floating bike sharing system. This model was generalized to consider collecting malfunctioning bikes as well as heterogeneous trucks, multiple depots, and multiple visits from a node.

In light of the above, we further the rebalancing research of free-floating bike sharing system. The main contributions of our research are (1) developing hybrid rebalancing by combining user-based incentive program with operator-based rebalancing to take the advantage of both; (2) introducing dynamic hubbing (i.e., geofencing areas varying from one day to another) for incentive program design; and (3) solving the problem with a novel multi-objective simulation optimization method. Specifically, on a daily basis, our approach makes three important decisions—the optimal number and location of *hubs*, the starting time for offering an incentive to customers, and the amount of the incentive. To make these three decisions, our approach considers two conflicting objectives—total rebalancing cost and system level of service.

2.3 Problem Description

In hybrid rebalancing with dynamic hubbing, dynamic rebalancing during the day is achieved by offering incentives towards the late hours of each day (or service time), and then the operator-based static rebalancing occurs during the night. It is assumed that the demand

patterns of the next day are known and that the goal of the rebalancing is to maximize the service level for tomorrow's first users with the lowest relocation cost. In light of these assumptions, this study attempted to answer the following key questions for a given day:

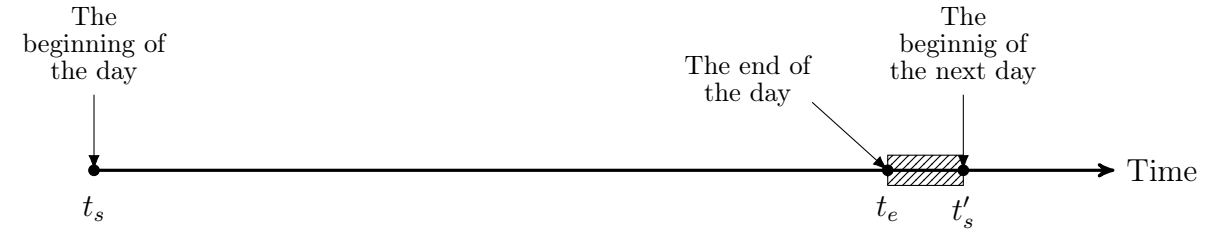
- What time of day should the user-based rebalancing or incentive program be started? The start time is denoted by t_0, and end of the service time is denoted by t_e in this study.

- What should be the incentive for the user-based rebalancing method? The incentive for time interval t is denoted by p_t in this study and shows the amount of the incentive. These incentives could be bonus points for a certain amount of minutes or times of free riding. The operator may put restrictions on use of the bonus points, e.g., allowing the points to be used only during non-peak hours.

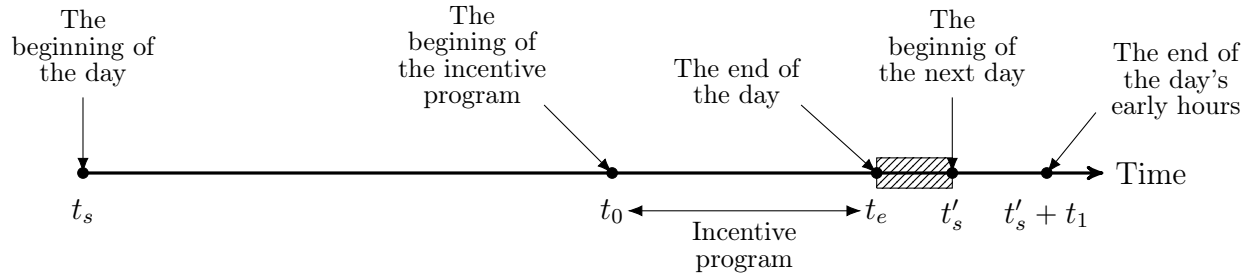
- What are the number and places of the hubs for rebalancing the bikes in both user-based and operator-based rebalancing methods in a daily fashion? These hubs are areas (for example, 50×50 m^2), but not stations at specific locations, as are those in station-based systems. In the redistribution process, we determine n^h of such hubs where bikes are relocated to respond to the demand in earlier hours of a second day.

The optimal values of t_0, $p_t, \forall t \in \{t_0, t_0 + \Delta, t_0 + 2\Delta, \ldots, t_e\}$ and n^h will be determined using a multi-objective simulation optimization approach for every day. The two objectives are system level of service (LOS) and cost. The system LOS is measured with two metrics: average amount of walking required for users to reach the bikes during early t_1 hours of the following day, W_1, and during the rest of day after the incentive program starts, W_2. The next day's walking in the first t_1 hours is controlled by the initial distribution of the bikes, which is the result of the hubbing decision and bike rebalancing. Also, when the incentive program starts, it will change the locations of usable bikes and affect the walking distance of users during the rest of the day. The incentive program does not affect user walking distance if their trips are completed before the program starts, i.e., t_0 (see Figure

2.1). For free-floating bike sharing systems, the lower value of this measure means that the bikes have better distribution to meet the demand, and, hence, provide a better level of service for users. The cost also has three components—operator-based rebalancing cost, cost of incentives, and cost of lost users. Compared to doing nothing during the day, the incentive program with hubbing encourages users to return bikes to the hubs. If this is done regardless of the demand distribution during the rest of the day, such incentive program may cause the loss of users because the dispersion of bikes can decrease and users cannot find a close bike (within a reasonable willingness-to-walk threshold). Considering conflicting objectives, we propose a multi-objective optimization approach to approximate the set of (the so-called) *Pareto-optimal* or non-dominated solutions of the problem, i.e., a solution in which it is impossible to improve the value of one objective without making the value of the other objective worse. Decision-maker(s) can then see the trade-offs and choose the most beneficial solution.



(a) Time-line of a day without incentive program



(b) Time-line of a day in proposed method with incentive program

Figure 2.1: Time-line of a day

2.4 Proposed Method

FFBSs are different from conventional station-based bike sharing. In SBBS literature, the LOS of the system is measured mostly as the number of rejected drop-off demands from full stations and pick-up demands from empty stations [22, 30, 31]. Since SBBSs have fixed stations and all customers in a certain radius from a station will refer to it, the arrival rate of customers to stations can be a good estimation of the demand in these systems. In FFBSs, however, there is no station that challenges both LOS evaluation and demand prediction using traditional methods. Instead, every bike can be considered a station that answers only one pick-up demand. Users decide if they will go to pick up a bike based on how far they have to walk to reach the bike; a closer distance makes it more likely they will use the system. Thus, in our study, we consider both the number of lost customers and the walking distance of customers who used the service, which can be two measures of LOS. We assume that the willingness to walk to reach available bikes is a piece-wise linear distribution. If a customer demand pops up in the system but the available bike is too far away for him/her to reach, the customer decides to not use the service and becomes a lost customer. Otherwise, the customer's walking distance to get the bike is counted into the walking distance measure.

A simulation model of a system is basically imitating the behavior of that system based on the formulated relationships between its components. If the internal and external variables of the system are deterministically given, a mathematical model can answer all questions about the system. However, the main difference between a mathematical model and a simulation model can be the capability of the simulation model to account for the uncertainty of input variables in high-complexity systems and capture the effect of this uncertainty in performance indicators or other outcomes of the system [32]. In other words, a simulation model helps to statistically study and identify reasons for the changes in the outputs of a system by defining "what-if" experiments, which may not be doable in a real system because of the high or even sometimes impossible cost, danger, or resource it requires.

In optimization, we look for the best values of some variables (usually called decision variables) in a system, which maximizes or minimizes a specific measure. Most of the time, mathematical modeling is used in optimization, and except in rare cases, it is done with many simplifications that results in not representing all the aspects of complex systems. Whereas simulation models can cover a significant part of this simplification, which mostly has roots in uncertainty. Needless to say, when the number of input variables and the complexity of the simulation model increases, the simulation may become computationally difficult as well [33]. Nevertheless, advances in computing technologies in recent years have alleviated this difficulty and turned coupled simulation and optimization into a strong tool to optimize complex systems [34]. Simply speaking, simulation optimization is used when the objective functions or constraints of a model can be evaluated only by simulation [35]. A variety of simulation optimization methods has been used to solve a broad group of optimization problems in the literature (see [36]), especially in the field of transportation (see [32]).

In this study, simulation optimization is used to handle the uncontrollable amount of uncertainty in the considered FFBS. The proposed simulation optimization method uses a simulation model of an FFBS to evaluate the objectives associated with the solutions of individuals in a population-based heuristic multi-objective optimization method (see Section 2.4.6). For each solution denoted by $(n^h, t_0, \boldsymbol{p})$, in the simulation process, we first determine the locations of the hubs based on the distribution of the demand for the early hours of the next day and the number of hubs (n^h). Then, the system is simulated throughout a day, mimicking the pop-up of customers, their decisions of approaching available bikes and using bikes to get to their destination, and the locations of bikes due to the dynamic usage of system. Once the incentive program starts at t_0, customers will respond to the incentive programs. A detailed simulation of the system during the incentive program period is explained in Section 2.4.4. The simulation will be run several times and, at the end, the average of both objective values over all runs will be computed. The average values will be considered as the fitness, i.e., objective values, of an individual solution $(n^h, t_0, \boldsymbol{p})$. Figure

2.2 schematically demonstrates the framework of the proposed method and relationships between its steps. Next, we explain these steps and needed details.

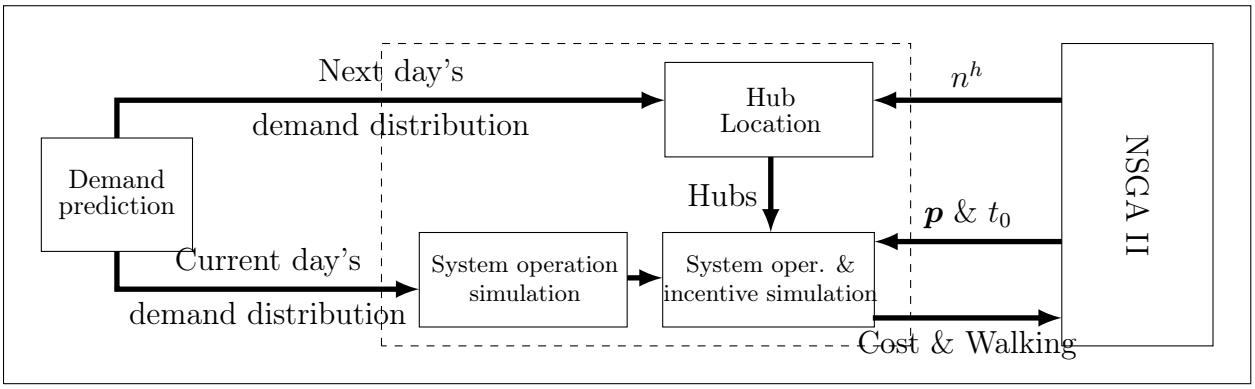


Figure 2.2: Framework of proposed simulation optimization method

2.4.1 Demand Prediction

Researchers have put much effort into predicting demand in bike sharing systems, both when there is no such service in the area yet [37, 38, 39, 40] and when future demand of an existing system is intended [41, 42, 43, 44]. In this study, we aim to predict and simulate demand for an existing FFBS system with historical pick-up and drop-off data. We first divide the service area of the system into small parts with equal square or hexagonal shapes called zones and service time windows into temporal intervals. We assume that demand arrivals to each zone follow a Poisson distribution and that the average of the aggregated demand in each time interval will depend on an arrival rate (λ_t), which will be predicted according to historical data. We also assume that the demand that pops up in each zone could be anywhere in the zone, i.e. spatially evenly distributed. Furthermore, for the destination of demands originating from a specific zone in time interval t, we simply elicit the percentage of the flow to other zones, denoted by OD_t. We use an Auto-Regressive Moving Average (ARMA) model to predict these parameters in this study.

The proposed method could be improved if the operators refine the data collection process and obtain the location where users start to use the app to search for available bikes. Such information could reveal better estimation of origins of users than the bike pick-up locations. This is because in FFBS, if a user starts searching for a bike in zone A, depending on the location of the nearest available bikes, he/she may end up actually picking a bike in zone B. Also, knowing the location when the user starts searching bikes, the nearest available bikes, and his/her final decision of picking-up the bike or not, we can analyze the demand better, understand the walking limit of the user, WL, and develop a better willingness-to-walk model for bike sharing users.

2.4.2 Finding Locations of Hubs

The locations of the hubs for organizing an incentive program and operator-based rebalancing on the current day are determined based on the distribution of the demand in the early hours of the following day, $\boldsymbol{D^2}$. n^h of such hubs are selected aiming at facilitating bike re-balancing on the current day to meet the demand of the early t_1 hours of the next day. The value of t_1 could be determined by operators who intend to implement the proposed method in this study. An intuitive way is aiming for the morning peak hours, usually 2-3 hours between 6:00 and 9:00 AM for most of cities. In this study, we used t_1 equal to 2 hours.

Finding the locations of the hubs given the predicted demand $D^2_t, \forall t \in [t'_s \;\; t'_s + t_1]$ is generally a continuous location-allocation problem, i.e., locating a set of facilities and allocating a set of given customers to different facilities while optimizing an objective function. For general transportation problems, the objective function is usually transportation cost, and the allocations are allowed to be partial [45]. However, in an FFBS system, each trip is responded by only one hub, so there is no partial allocation. As a result, this problem

reduces to the clustering problem, which can be formulated as follows:

$$\text{Model (1):} \qquad \min \sum_{k=1}^{n^h} \sum_{i \in \boldsymbol{D^2}} \xi_{ik} d(x_k, l_i),$$

$$\text{s.t.} \qquad \sum_{k=1}^{n^h} \xi_{ik} = 1 \qquad \forall i \in \boldsymbol{D^2},$$

$$\xi_{ik} \in \{0, 1\} \qquad \forall k = 1, ..., n^h, i \in \boldsymbol{D^2},$$

where $d(x_k, l_i)$ specifies the distance function between the location of the hub's center $x_k, \forall k = 1, ..., n^h$, and the location of the demand $l_i, \forall i \in \boldsymbol{D^2}$. Remember that $\boldsymbol{D^2}$ denotes the set of all predicted demands for the following day's early hours. Moreover, ξ_{ik} represents the binary decision variable that indicates the allocation of demand i to hub k. This problem has been proved to be NP-hard ([46]), and, because of its numerous applications, many heuristic algorithms have been introduced to deal with different types of it (see [47]).

There are three common distance functions, all of which make Model (1) non-linear—Manhattan, Euclidean, or general Minkowski distance [48]. In bike sharing systems, Manhattan distances are used in cities because of the blocky structures that buildings make, and Euclidean distances in open spaces such campuses can be used. However, both will have some errors because there is no city with perfectly blocky-shaped streets or completely open space places. The only way to take the accurate distance into account is diverting the problem into the discrete location-allocation problem. To this end, a set of candidate points are determined in the considered region, and the location of the facilities are selected among them. This method facilitates using the actual walking distance by creating a network between the candidate hub and demand points. Taking the center of all zones as candidate hub points, given the walking distance between zone j and customer i, d_{ij}, this problem is

formulated as follows:

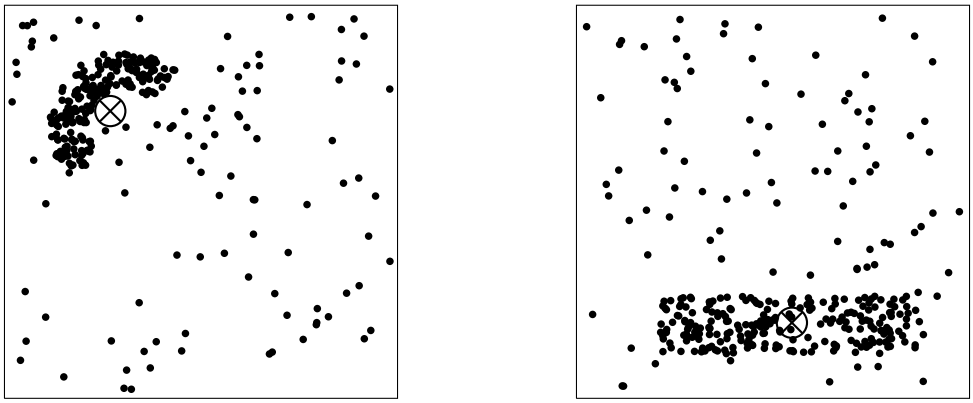
$$
\begin{aligned}
\text{Model (2):} \qquad & \min \sum_{j=1}^{n^z} \sum_{i \in \boldsymbol{D^2}} x_{ij} d_{ij}, & \\
\text{s.t.} \qquad & \sum_{j=1}^{n^z} x_{ij} = 1 & \forall i \in \boldsymbol{D^2}, \\
& x_{ij} \leqslant y_j & \forall i \in \boldsymbol{D^2}, \forall j = 1, \ldots, n^z, \\
& \sum_{j=1}^{n^z} y_j = n^h, & \\
& x_{ij}, y_j \in \{0, 1\} & \forall j = 1, ..., n^z, i \in \boldsymbol{D^2},
\end{aligned}
$$

where y_j indicates the selection of zone j as a hub and x_{ij} denotes the assignment of demand i to zone j.

In this study, we employ K-means [49] and DBSCAN [50] clustering algorithms using both Manhattan and Euclidean distance functions as a fast and approximate solution for the continuous location of hubs. While the number of the hubs is an input in K-means, the number of the hubs in DBSCAN is controlled by two distinct inputs—*neighborhood density* and *neighborhood radius*. As opposed to K-means, which finds only circular clusters, DBSCAN is able to find clusters with random shapes, which can be both helpful and adverse in our method. For example, Figure 2.3.a shows the case where locating a hub in the center of a cluster will reduce both walking and redistribution costs, but in Figure 2.3.b the center of the found cluster as a hub will not help the walking measure. The reason is that this method works based on the density of the objects and can join all demand points into one cluster if they are densely connected. Thus, while applying this clustering method we limit the size of the clusters by putting a cap on the distance that two demands can have in a cluster. Furthermore, we fix the two original parameters of the algorithm and attempt to change the number of the clusters, which determines the number of hubs, by changing the value of this limit. Another issue that arises using this algorithm is the existence of single demands that fall far apart from all clusters such that the algorithm cannot assign them to

any clusters. These points are called noise points in the algorithm's terminology. We solve this problem by assigning these demands to the closest cluster.

In addition to these heuristic methods for continuous hub location, we can also use Model (2) to select some of the zones as hubs. The results of using each of these methods for case study FFBS systems are analyzed in Section 2.5. It is worth mentioning that using DBSCAN and Model (2) allows us to use any type of distance, especially real distance. In contrast, K-means requires the coordinates of the points and is limited to the types of distance that can be calculated from the pair of points' coordinates. For some systems, depending on the geographic features of the service area, hub locations obtained from different methods may be very similar.



(a) A arc shape cluster for which using DBSCAN is helpful

(b) A long rectangular cluster for which using DBSCAN is adverse

Figure 2.3: Shape of clusters DBSCAN can find ($\otimes$ is center of the cluster or hub).

In sequel, we use the percentage of the total demand assigned to a cluster to determine the percentage of bikes required in a hub for the following day. Specifically, if $\gamma\%$ of all demands are assigned to a cluster, $\gamma\%$ of all bikes should be in the associated hub the next day. Consequently, the rebalancing methods including the incentive program and static relocation must ensure that by the end of the day, $\gamma\%$ of all bikes will move into that hub.

2.4.3 Incentive Program and Customer Behavior

Different incentives can affect customer behavior (*demand response*) in different ways. To evaluate any incentive program, it is necessary to have enough information about the customer's reaction to different values of the incentive and also the cost the operator bears for implementing the incentive program. The cost of the incentive becomes more important when there is no control over the accepted offers. For example, offering a discount on the price of a product in a store may lead to uncontrollable selling-out, whereas in online systems such as flight ticket reservations the discount can be changed and even stopped at any time according to the situation. In bike sharing systems, since the state of the system is monitored online and communication of the system with the customers is done through an online mobile application or web interface, the number of accepted offers can be controlled. However, we need to model the reaction of customers to the incentive program so it can be included in the overall multi-objective optimization problem.

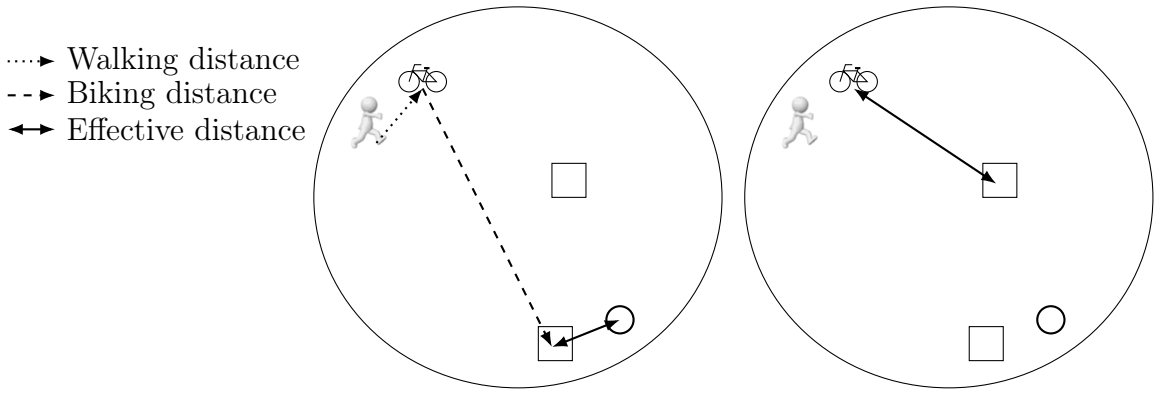
In this research, we explain the incentive program that is being planned to run on the case study bike sharing system to provide a better illustration of the proposed method. However, operators may come up with different designs of incentive programs. In the sample incentive program, any drop-off in the suggested hub is encouraged by $p_t, \forall t \in \{t_0, t_0 + \Delta, t_0 + 2\Delta, \ldots, t_e - \Delta\}$ points (or credit). Each unit of the incentive is worth the same amount of units of free ride outside the incentive time window in the system. In other words, if the unit of rides in the bike sharing system is one minute, two points means two minutes free ride some time outside the incentive window. Some studies model an incentive program in SSBSs by requesting customer destination information prior a trip [21, 22]. However, the incentive program designed in this study does not require customers to reveal their destination; instead, all incentivized hubs are suggested to all users, and at the drop-off point the system can know if the user accepts the offer or not.

Time value and price sensitivity are two major factors that influence a customer's decisions on accepting a specific offer in the incentive program. When simulating the customer's

reaction to the incentive program, we assume that there are two types of customers— *regular customers* and *opportunistic customers*. Regular customers accept the incentive program offer only if they need to ride a bike to a destination and the geofenced hub with incentive offer is relatively close to their destination. Compared to regular customers, opportunistic customers are those with relatively low perceived value of time, high price sensitivity, and schedule flexibility. Similar examples are leisure passengers who would take flights because of a significant discount offered by airlines. Opportunistic users will accept the incentive offer if riding a nearby bike to the closest hub-with-incentive-offer does not elongate their original trip (from where they are to their destination) and the earned credit is worth more than what they have to pay for riding the bike (e.g., paying one minute of riding the bike but earning five minutes of future free riding). Such customers were observed in bike sharing systems in China when Ofo and MoBike started "Red Envelop" incentive programs. Although details of those incentive programs vary from one service provider to another, the basic idea is the same, i.e., offering free ride credits as an incentive to encourage customers to help with bike rebalancing.

In the proposed model, the acceptance of offers is a probability function of the amount of the incentive p_t at time interval t and the specific distance that users have to walk or bike, called *effective distance*. The definition of the effective distance for both types of customers is illustrated in Figure 2.4. Regular customers are assumed to leave the bike at the hub closest to their actual destination in case they accept the offer. Therefore, the effective distance is how far they have to walk from dropping off the bike to their true destination (Figure 2.4a). However, opportunistic customers will reach the nearest bike and ride it to the closest hub for earning the credits. Therefore, the effective distance for these users is the distance between the pick-up point and the closest hub. Pfrommer et al. [23] modeled user reactions to an incentive by converting the customer's additionally-traveled distance to cost. However, we assume that the acceptance probability of an offer p is a decreasing function with respect to effective distance and can be learned from records of users. Overall,

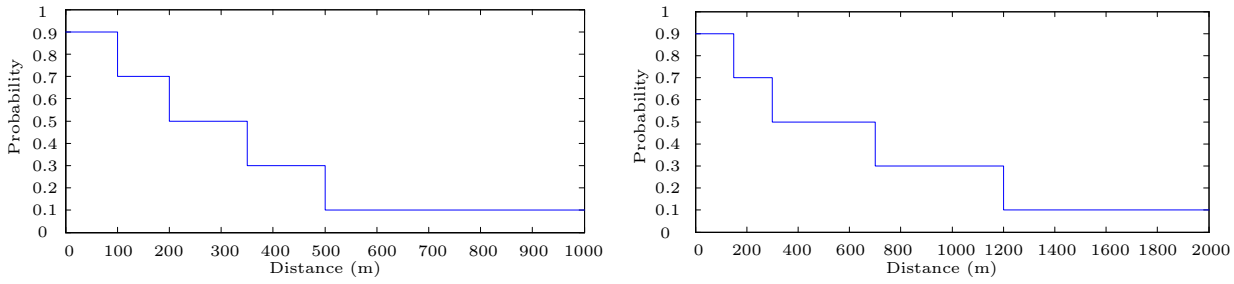
Figure 2.5 shows the simplest form of acceptance probability of an incentive offer versus the effective distance for $p = 1$ and $p = 1.5$. Here, the effect of p is assumed to be only on proportionally stretching or shrinking the curve along the horizontal axis. Apparently, higher values of p imply higher acceptance probability. It should be mentioned that these functions are different for two types of users, as the effective distance is walked by regular users but it is biked by opportunistic users. In this study, we used simple forms of acceptance probability function such as that shown in Figure 2.5.



(a) Effective distance for regular customers

(b) Effective distance for opportunistic customers

Figure 2.4: Effective distance based on customer decision to accept an offer



(a) Acceptance probability with $p = 5$

(b) Acceptance probability with $p = 10$

Figure 2.5: Acceptance probability vs. effective distance

2.4.4 Simulation

In this section, we explain the simulation of the system in detail. This proposed simulation model requires the initial distribution of bikes. Therefore, the real locations of

the bikes one or two hours before a potential moment of t_0, where we want to start optimizing the system, is obtained from the system and provided to the model. Needless to say, for a large-size FFBS, this makes the model computation costly, so we need to obtain the bike location inputs and start to run the simulation earlier. The normal simulation, with regular customers, is carried on until $t = t_0$. When the incentive program kicks in, out-of-hub bikes are incentivized and the users are encouraged to drop them off inside one of the predetermined hubs. In other words, hubs with fewer bikes than what they are planned will be offered to customers as incentivized destinations. Depending on the group they belong to, i.e., regular or opportunistic, customers will be simulated as following the behavior explained in Section 2.4.3.

At the end of the simulation ($t = t_e$), the static operator-based rebalancing will be conducted and the static relocation cost evaluated. To do so, the final location of bikes and hubs will be given to one of the fast and effective static rebalancing algorithms in the literature called NLNS + VND [9]. This algorithm is a recent output of our research group and takes advantage of both Large Neighborhood Search (LNS) and Variable Neighborhood Descent (VND). NLNS+VND is equipped with four granular neighborhoods and multiple large neighborhoods specifically designed for large-scale bike redistribution problems. Recall that after the static operator-based rebalancing, the number of bikes in each hub will be exactly equal to the number planned in the hub determination step, i.e., complete rebalancing. Next, we discuss the simulation process in more detail.

Figure 2.6 shows a detailed flowchart of the proposed simulation model, shown in Figure 2.2 by a dashed frame. The notation used in the flowchart can be found in Table 2.1. We explain the steps of this flowchart block by block following the direction of the arrows. After "Start", the first step addresses the demand generation for next day's first t_1 hours (see Section 2.4.1) and accordingly determining hub locations for a given n^h (explained in details in Section 2.4.2). Next, the current location of the bikes ($Bike.L$) and their availability ($Bike.Avail$), incentive program starting time t_0, and incentive level p_t are initialized. Then,

the user's origin (*Dem.Location*) and destination (*Dem.Dest*) for a unit of simulation time step are generated (through the function *Generate_ customers*). The simulation's time step, *step*, must be set to a small enough time interval (e.g., six minutes) to cover and record the most information about the behavior of the system. However, the smaller the *step* is, the more computational load the simulation requires. User generation follows the distribution of demand associated with each moment of the day. Therefore, the time counter, i.e., t, has been passed into the function *Generate_ customers* as an argument. After that, the number of generated users is computed by function *size*, and the part of the simulation related to the behavior of the customers starts.

For each regular user, first, the closest available bike will be found. Then, the model examines if the distance to the closest bike is acceptable to the user, i.e., it is not greater than walking limit WL. If the user chooses to not pick up the bike, meaning the closest bike is farther than WL, the system fails to serve the customer and, consequently, the number of lost customers, nLC will increase by 1. Otherwise, the user picks up the bike and, depending on the time and the value of t_0, if the incentive program has not started yet or it has started but the user decides to not accept the offer, he/she will ride the bicycle to his/her destination and drop off the bike. However, if the incentive program has started and the user decides to accept the offer, the user rides the bike to the incentivized hub that is the closest to his/her destination and the number of accepted offers (nAO) will increase by 1. For different scenarios, the status of the bike will be updated accordingly once it is dropped off. Note that the user's decision about an offered incentive is made based on the value of p_t and EDR as explained in Section 3.3.

Once the incentive program starts, i.e., $t > t_0$, in addition to regular users, the simulation will generate additional demand, which represents opportunistic users. The number of opportunistic users is considered proportional to the number of regular demands at the moment t, and it is generated around incentivized (or out-of-hub) bikes within walking limit WL. Each opportunistic user makes a decision on accepting the incentive offer based on ef-

fective distance (as explained in Section 3.3) and the comparison of the distance from his/her generated location to destination and the distance of where the bike will be dropped off to his/her destination. In other words, an opportunistic user accepts an offer if the effective distance to riding the bike is not too far and relocating to the incentivized hub does not make hos/her way to the destination any farther. If an opportunistic user decides to not accept an incentive offer, he/she will disappear from the system without changing any variables, i.e., not be counted as a lost customer. Under different scenarios, the status of bikes, *Bike.Avail* and *Bike.DT*, will be updated accordingly.

The simulation will continue until the time reaches the end of day, t_e, when the static operator-based rebalancing starts.

Table 2.1: Notations used in Figure 2.6

n^h	number of hubs
t_0	start time for incentive offering
p_t	incentive amount at time period t
α	ratio of opportunistic customers
t_e	time horizon of simulation
$step$	time step of simulation
WL	maximum distance that a customer walks to find a bike before giving up
D^2	tomorrow's predicted demand
$Hubs$	locations of the hubs
nAO	number of accepted offers
nLC	number of customers who have given up
EDR	effective distance for regular customers
EDO	effective distance for opportunistic customers
$Bike.L$	location of bikes
$Bike.Avail$	status of bike which is true if it is available and false, otherwise
$Bike.DT$	time by which rented bike will be dropped off
$Dem.L$	point in which a demand appears
$Dem.Dest$	destination of demand
$Cost$	value cost objective
$Walking$	value of walking objective

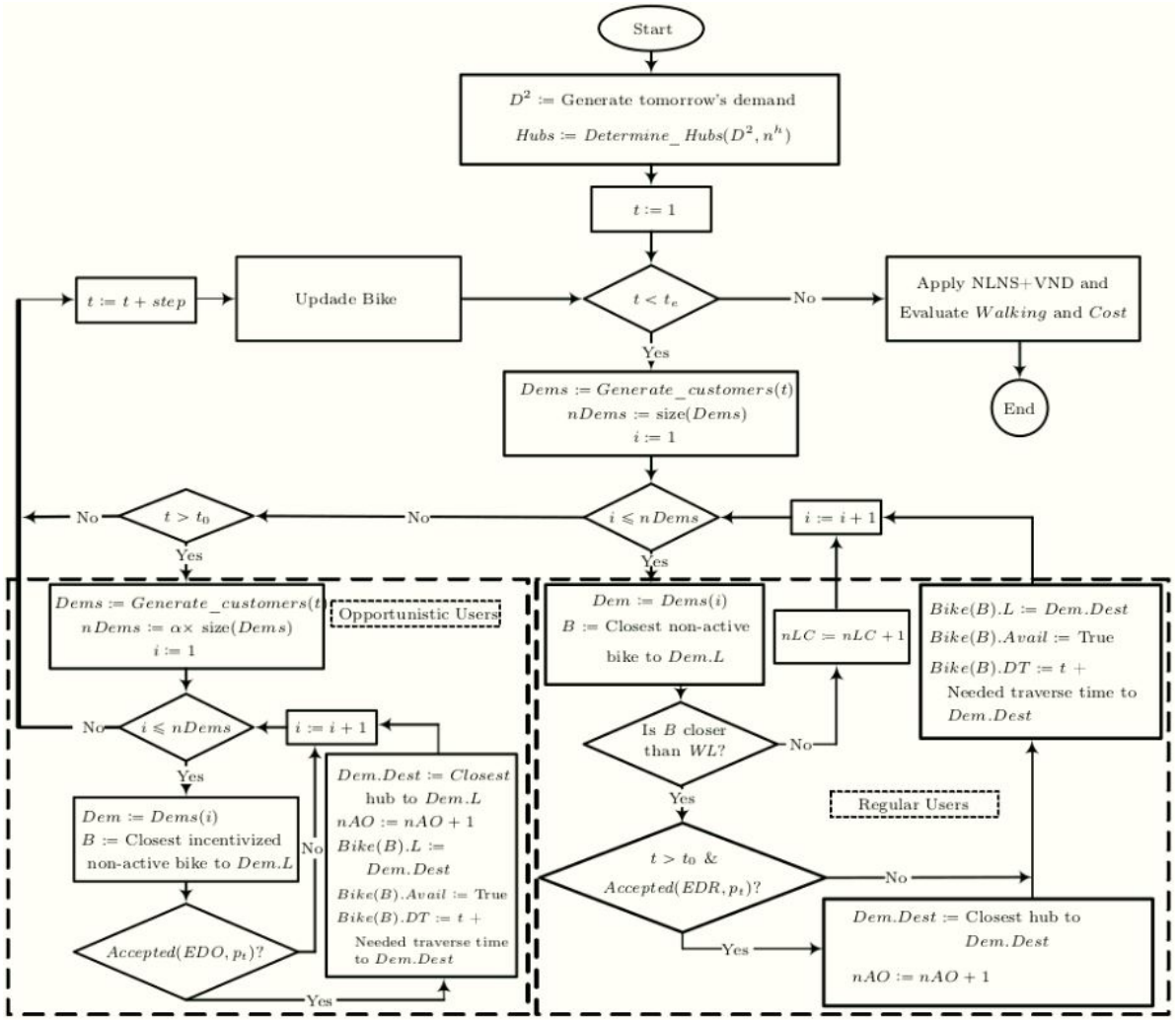


Figure 2.6: Flowchart of proposed simulation

2.4.5 Objective Evaluation

In this study, two objectives are considered—service level and cost. For free-floating bike sharing systems, service level is defined as the amount of customer walking to bikes and is denoted by W . In this study, we consider hourly average walking distance, which includes two parts. The first part, denoted by W_1, is the average walking distance of customers in the first t_1 hours of the next day, which is dependent on the determination of the hubs. We use simulation to calculate W_1 by recording the walking done to pick up a bike from hubs and dividing it into t_1. The second part, W_2, is the average required walking of users in the current day after the incentive offering starts, which is computed through the simulation

as well. Specifically, W_2 is the summation of distances that regular users need to walk to reach a picked-up bike (after starting the incentive program) divided by the length of the incentive offering period. The reason for considering average hourly walking instead of total distance is because the start time of the incentive program, t_0, is also a decision variable. The value of W_2 is dependent on t_0 and will naturally be smaller for a bigger t_0 (or shorter incentivized period). Note that, to compute W_2, we ignore lost users because they are taken into account as a cost. Moreover, since the goal of the system is improving the service level for regular customers, the walking of the opportunistic users is also not considered in this objective function.

The second objective, i.e., cost, includes several parts. The first and major part is the cost of static operator-based rebalancing, calculated by applying the algorithm NLNS + VND. The second part is the cost of lost customers, i.e., summation of the income of the rides the system misses because of losing users. The third part is the cost of accepted offers, which sums the number of accepted offers during t multiplied by the cost of p_t for the period of incentive program. In other words, we consider the worst case and assume that all gained points will be used by the customers in their future rides.

2.4.6 Multi-objective Optimization

For this hybrid rebalancing with dynamic hubbing, we solve the multi-objective optimization and generate a Pareto-optimal frontier to help decision-makers understand the trade-off between objectives. Non-dominated Sorting Genetic Algorithm (NSGA II) is a widely used heuristic algorithm in solving multi-objective problems [51, 52, 53]. This algorithm uses the framework of a genetic algorithm in the context of multi-objective optimization to optimize the non-dominated solutions. Moreover, it is equipped with a fast subroutine that can draw the layers of non-dominated points in a given population. In general, this algorithm improves the quality of approximate Pareto-optimal frontier in each generation,

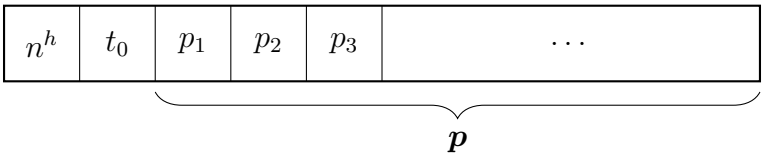


Figure 2.7: The solution representation in the proposed NSGA II

imitating the idea of evolution on the information obtained from the Pareto-optimal solutions in the previous generations.

In light of the above, we use NSGA II as the optimizer in our proposed simulation optimization approach. Specifically, the simulation model discussed in Section 2.4.4 plays the role of fitness evaluator in NSGA II, i.e., it calculates the value of objectives associated with each solution $(n^h, t_0, \boldsymbol{p})$. Figure 2.7 shows the solution representation in the proposed algorithm. Depending on the length of the intervals for which we want to determine the amount of the incentive, i.e., the number of entries of $\boldsymbol{p}$, the length of the solution representation differs.

It should be mentioned that the results of a simulation will be statistically reliable only if their average after enough duplications tends to be a stable value. The number by which the simulation is duplicated to evaluate each solution is denoted by *Duplication* and was set to 20 in this study.

Algorithm 1 summarizes the steps of the proposed simulation optimization approach. In this algorithm, first, the parameters of the algorithm are initialized. These parameters include the number of population (*nPop*), *Duplication*, and the termination condition of the algorithm (*Termination_Condition*). The number of the population shows the number of solutions with the form $(n^h, t_0, \boldsymbol{p})$ that we want to keep in the current generation's pool. The larger this parameter is, the better the algorithm will perform, which, in turn, requires more computational time. Termination condition determines when the algorithm will finish and can be set based on different measures in the algorithm. In this study, we use the most common termination condition that is an upper bound for the number of generations or iterations (*Termination_Condition*).

In the next step, *nPop* number of solutions $(n^h, t_0, \boldsymbol{p})$ are generated randomly within a given interval (*Pop*), and their objective values are evaluated by simulation. This interval for variables can be obtained by a sensitivity analysis or expert estimation. The function *Simulation*, moreover, receives the value of the variables and runs the process, as explained in Sections 2.4.4 and 2.4.5. Note that this step is repeated *Duplication* times for i'th solution and the average values will be considered as the final objectives' values, *Pop(i).Obj*. Then, the population will be sorted by function *Sort_Frontiers* based on two measures in the context of multi-objective optimization; ranking and distribution in the space of objectives. This sorting algorithm is the main part of NSGA II; interested readers can refer to [52] to learn more about it.

The algorithm's main loop runs until the termination condition is met. The first step of this loop is the selection of *Parents*, which is a set of solutions selected from the population. In this study, the parents are chosen randomly from the population by the function *Select_Parents*. Then, some new solutions (which are 80% of *nPop* in this study) are generated by applying reproduction operators on parent solutions. Reproduction is mainly done by operations called crossover and mutation in a Genetic algorithm, and there are several methods to apply them [54, 55]. In the proposed NSGA II algorithm, we use the random linear combination of two parents to generate the offspring. The only integer part of the solution representation is the number of the hubs, which is replaced by the closest rounded number if the linear combination results in a fractional value. For mutation, we randomly add or reduce a random value from each part of the solution representation. These random additions or subtractions are limited to 10 percent of the provided range used to generate the initial population. Moreover, if any value exceeds the bounds of the ranges, it will be replaced by the value of the violated bound. These reproduction steps are summarized within function *Recombine* in Algorithm 1. Then, the new solutions are evaluated and added to the population. Finally, each iteration is completed by sorting the population and then

removing those solutions that are not eligible to survive for the next generation. This is done by keeping *nPop* number of the best solutions.

Algorithm 1: Proposed simulation optimization method

```
Initialize nPop, Duplication and Termination_Condition
for i = 1 : nPop do
    Pop(i).[n^h, t_0, p] ← Generate solution [n^h, t_0, p] randomly within the given ranges
    Cost ← 0; Walking ← 0; Obj ← [0, 0]
    for j = 1 : Duplication do
        [Cost, Walking] ← Simulation(Pop(i).[n^h, t_0, p])
        Obj ← Obj + [Cost, Walking]
    Pop(i).Obj ← Obj/Duplication
Pop ← Sort_frontiers(Pop)
while Termination_Condition is not met do
    Parents ← Select_parents(Pop)
    Children ← Recombine(Parents)
    for i = 1 :size(Children) do
        Cost ← 0; Walking ← 0; Obj ← [0, 0]
        for j = 1 : Duplication do
            [Cost, Walking] ← Simulation(Children(i).[n^h, t_0, p])
        Obj ← Obj + [Cost, Walking]
    Children(i).Obj ← Obj/Duplication
    Pop ← [Pop, Children]
    Pop ← Sort_frontiers(Pop)
    Pop ← Pop(1:nPop)
```

2.5 Numerical Results

The proposed method was implemented in Julia 0.6.4 [56] and applied on a real dataset obtained from a free-floating bike sharing system (Share-A-Bull) that operates at the University of South Florida (USF) in Tampa. The university serves more than 38,000 students with 1,700 faculty and staff on a 1500×3000-meter campus. SoBi has provided a free-floating bike sharing system called Share-A-Bull at USF since 2015. Currently, after four years, this system includes 300 bikes and covers the main USF campus and student housing areas in the vicinity. Trip data of this system were analyzed and used to extract the demand distribution in different zones through each hour of 12 randomly-selected days from each month. The results show that the demand was significantly low in some months because of breaks (in Summer and Winter). Therefore, we ignored 4 days and considered 8 days in the experiments. The service area was split into 50×50-meter zones and, as the data show only pick-up points, we uniformly spread each zone's demand in neighbor zones

to get the actual points from which the demands had arisen. It should be mentioned that in this system, the demand approaches zero from 00:00–06:00 AM; hence, the operator starts static rebalancing at 12:00 mid-night. In the experiments, we assumed that this bike sharing system uses a rebalancing crew with 10 trailers, each with the capacity of 10 bikes that can load or unload a bike in 60 seconds. In the simulation, the time was discretized in 6-minute intervals and the start of the day t_s was set to 06:00 AM. Also, the walking limit and the ratio of opportunistic users were respectively set to 200 meters and 20 percent. The results for all the experiments are based on the average of 20 replications in simulation part.

2.5.1 Comparison of Hub Determination Methods

First, we investigated the performance of the clustering methods introduced in Section 2.4.2. In this experiment, we selected representative days and generated the demand in the first two hours following the simulation procedure described earlier. We set the number of hubs to 15 and applied different methods to find the locations of the 15 hubs and compare the corresponding average walking distance of users obtained from simulation. As shown in Table 2.2, for some days, the methods gave out fewer than 15 hubs. Possible reasons include the following: 1) for continuous location methods, hubs falling into a same zone are considered as one hub; 2) the number of clusters in DBSCAN was not as controllable as K-means or mathematical modeling and can be different given a fixed set of parameters; 3) for some days or during some duplicated simulations, the generated demand was low, so fewer hubs would be needed. We repeated the experiment several times and reported the averages. As shown in Table 2, Model (2) almost always had the minimum average walking distance among all methods. However, the performance of the clustering methods was not so different from that of Model (2), which means we can save on computational time using clustering methods for this part of the model. Furthermore, there was no clear dominance between the two clustering methods for a specific type of distance, although K-means was expected to perform better on Euclidean distance. Thus, in our case study, we used K-means along

Table 2.2: Results for different hub determination methods

Distance Type	Day	DBSCAN		K-means		Model (2)	
		Walking (m)	n^{h*}	Walking (m)	n^{h*}	Walking (m)	n^{h*}
Euclidean	1	1005.6	9	975.3	10	**972.2**	15
	2	1370.0	11	1383.0	13	**1282.9**	15
	3	375.8	6	410.1	5	**360.0**	14
	4	565.9	3	475.1	3	**443.7**	13
	8	1840.2	10	**1636.2**	13	1788.6	15
	9	2146.4	13	2097.3	15	**2041.8**	15
	10	2889.9	15	2844.9	15	**2748.3**	15
	11	2744.9	14	2784.0	15	**2599.3**	15
Manhattan	1	473.2	9	474.5	10	**434.3**	15
	2	833.3	12	864.9	14	**828.9**	15
	3	202.9	7	200.8	5	**187.6**	15
	4	190.4	4	188.9	4	**164.5**	13
	8	881.6	11	912.2	12	**855.9**	15
	9	1094.7	15	1148.3	14	**1081.5**	15
	10	1670.4	17	1693.2	15	**1554.1**	15
	11	1665.7	16	1585.9	15	**1580.0**	15

*Number of hubs that are the average of several runs are rounded to the closest integer

with Euclidean distance, which is a better fit for a campus area, to determine the locations of hubs.

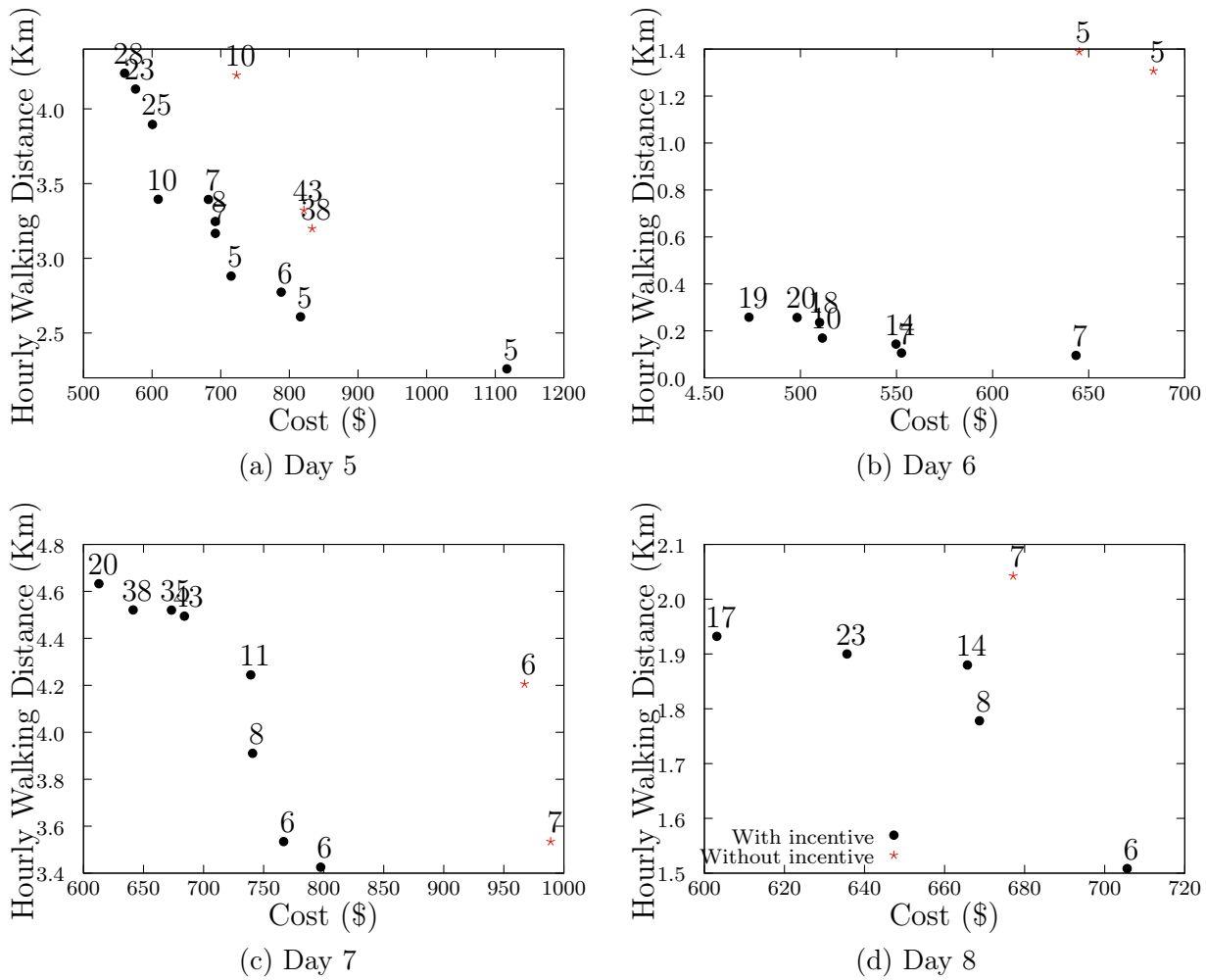
2.5.2 Multi-objective Result

In this section, we present the results of the multi-objective simulation optimization model. For this purpose, the allowed ranges that the variables $(n^h, t_0, \boldsymbol{p}$ can take were set to $([5\ 50], [16\text{PM}\ \ 24\text{PM}], [0\ 15])$, respectively. It is worth mentioning that in this experiment, variable p_t, according to the original model, can take different values for each hour during the incentive program (from 16:00 PM to 24:00 PM). We also set the population to 15 and the termination condition to 10 generations. We used Julia's parallel processing features to accelerate the evaluation process in the algorithm. To combine the different parts of the cost objective, we set the labor and equipment cost, incentive cost, and lost customer cost to 300\$/hour, 0.3\$/point, and 3\$/user, respectively, based on the estimation of system experts.

Figure 2.8 shows the trade-offs obtained by the (approximate) Pareto-optimal solutions in the space of the objective values for two scenarios; when the incentive program is considered (black circles), and when incentive program is canceled from the optimization (red stars) by fixing the variable, $\boldsymbol{p}$), to 0. This setting will remove the effect of the incentive program from relocation. In the second scenario, the optimization is basically focused on only one variable, n^h, and since there is a smaller feasible region, the result is expected to be more robust. The number printed on top of each point indicates the corresponding number of hubs as the most sensitive variable. In these Pareto solutions for scenario one, the increase in the number of hubs likely leads to lower cost and higher walking. Another experiment (see Figure 2.10) shows that the required time (and accordingly its cost) to relocate bikes is higher when there are more hubs. This reveals the dominance of two other parts of the cost objective function. We observe that there are not so many Pareto solutions for scenario two because the optimization is done based on only one variable and the feasible region is limited, which helps the algorithm have even a better performance compared to the first scenario. Nevertheless, comparing the Pareto frontier of these two scenarios shows the difference that an optimized incentive program makes in the system. We observe that the difference (or improvement) varies for different days.

In multi-objective decision making, the best possible values of objective functions obtained from the exact Pareto-optimal frontier are called ideal points. Sometimes, proximity to the ideal point is a simple way to select one solution among the Pareto-optimal frontier. Although it is the decision-makers' job to select a final solution, if we assume that they will use proximity to the approximate ideal point, Figure 2.8 shows that the system will have different values for the number of hubs and other variables every day, which points out the dynamic function of the model.

Green House Gas (GHG) emission is an environmental effect that is recently considered in all the industries and businesses due to global warming. According to U.S. Environmental Protection Agency (EPA) report in 2017, transportation has the largest share

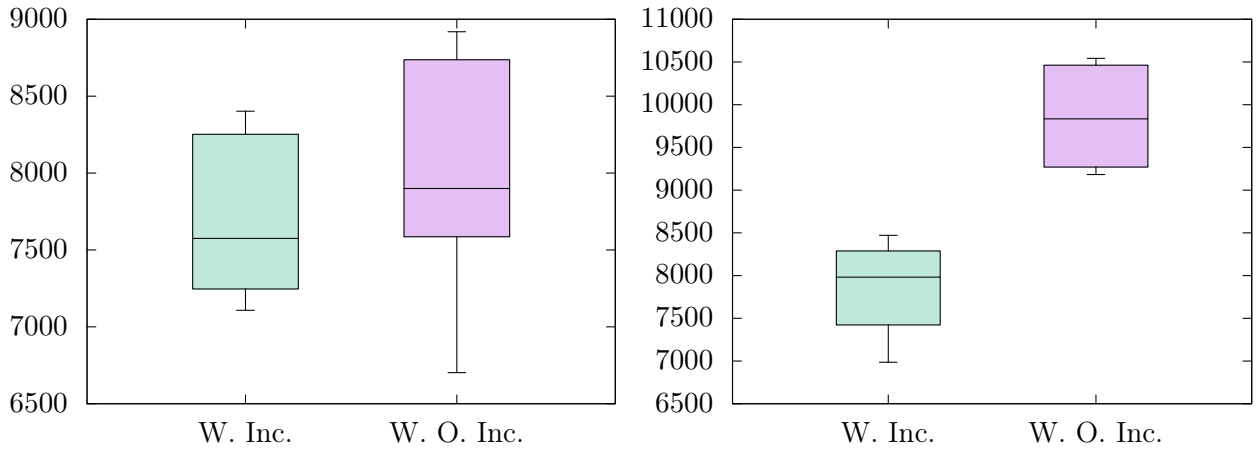


(a) Day 5 (b) Day 6

(c) Day 7 (d) Day 8

Figure 2.8: Trade-offs obtained by (approximate) Pareto-optimal solutions for days 5–8

of GHG emission, as CO_2, the product of combustion in engines, is known as the major GHG. Therefore, environmental effects are another aspect from which using incentive to re-distribute bikes in a BSS are important. To show how much a simple incentive program can save on the operation of trucks, we calculated the average operation of each truck in non-dominated points under two scenarios—with and without an incentive program—for days 4 and 7. We used EPA's last report on automotive CO_2 emissions to find the average emission rate (350 g/mile) and converted the operation to the amount of carbon dioxide. Although a decision maker or the policy of a company determines which non-dominated point will be chosen, we draw these amounts in the form of a box-plot (Figure 2.9). This graph helps to



(a) Day 4

(b) Day 7

Figure 2.9: Box plot of average CO2 emissions of rebalancing trucks with and without incentive program (g)

compare the interval, quarters, and median of CO_2 for the scenarios in general. The figure shows that the proposed method (or offering of an incentive in general) can make a difference in terms of GHG emission.

2.5.3 Sensitivity Analysis on Variables

We examined the behavior of the system for a range of valid values for each variable, keeping the other variables and parameters fixed. Figure 2.10 shows how the number of hubs, i.e., n^h, changes the static relocation cost when $p_t = 10, \forall t \in [18\ 24]$ and $t_0 = 18:00$ PM For all days, although there were some fluctuations (because of the uncertain nature of the process), the required time to relocate the bikes is increasing. This is not surprising because by increasing the number of hubs, the system needs more dispersion of bikes, which requires more labor-hour resource. It should be mentioned that this experiment was done with $n^h \in [5\ 50]$. However, for days that have fewer demands in the next day's morning, the number of hubs is different from what was given since the model combines the hubs that are closer than 50 meters. Figure 2.11 shows the next day's walking per hour for the same experiment. As expected, the walking distance of users in the first two hours of next day

decreases (service level increases) because the visibility of the bikes becomes higher in the demand-prone regions.

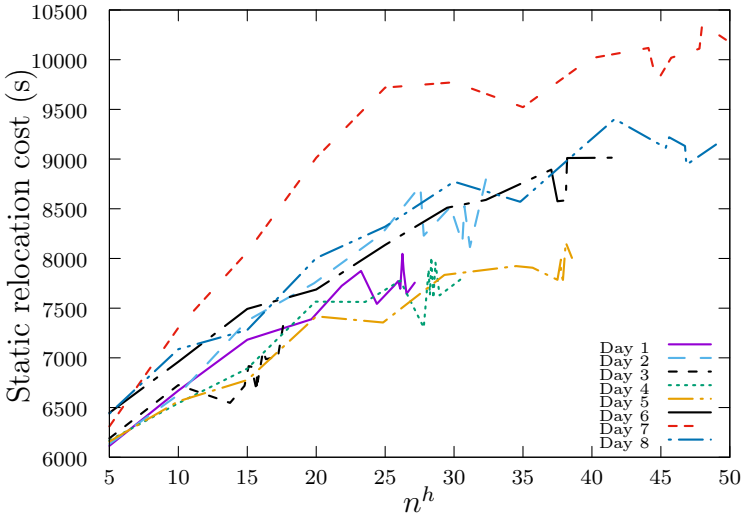


Figure 2.10: Static relocation cost for different number of hubs

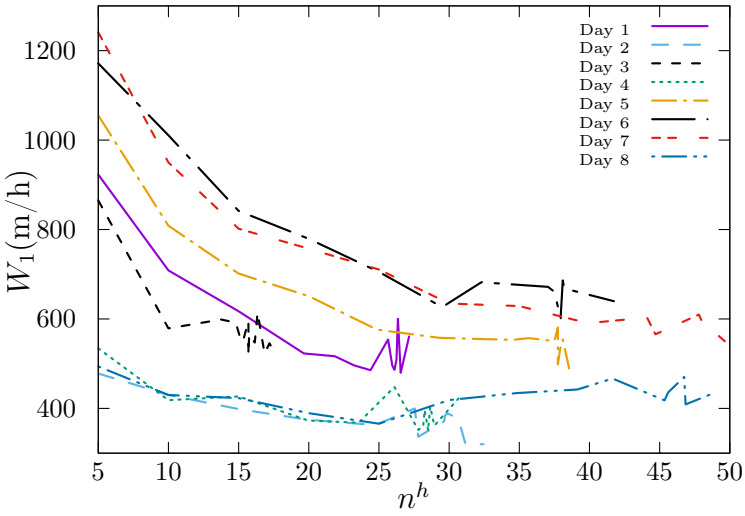


Figure 2.11: Tomorrow's walking for different number of hubs

We also examined the changes in the average walking of users during the incentivized period, W_2. Figure 2.12 shows an increasing trend of this measure up to around 25 hubs. However, there is some decrease in the track of W_2 for the higher number of hubs, especially for days with higher demand (see Day 7 at 30 hubs and Day 8 at 35 hubs). The reason is

that when the number of hubs is low and the incentive program starts soon, the collection of bikes to the hubs decreases the bike dispersion in the service area and users need to walk more to pick up a bike. This negative influence disappears when the number of hubs increases, and the collection of bikes to the hubs becomes a redistribution by itself. The inverse behavior, but with the same pattern, is seen in curves that show the number of lost (rejected) users in the system (Figure 2.14). Basically, these two graphs together reveal that part of the increase in the walking is because the system serves more customers rather than rejecting them when the number of hubs goes up. The walking limit plays a critical role in this trade-off, as discussed in Section 2.5.4. Also, the number of accepted offers increases as the number of hubs increases (Figure 2.13). A higher number of hubs leads to shorter effective distances for both types of users, which makes the acceptance of offers more likely. Hence, dynamic relocation is intensified when there are more hubs in the system, whereas a higher number of hubs adversely affects the static relocation time (see Figure 2.10).

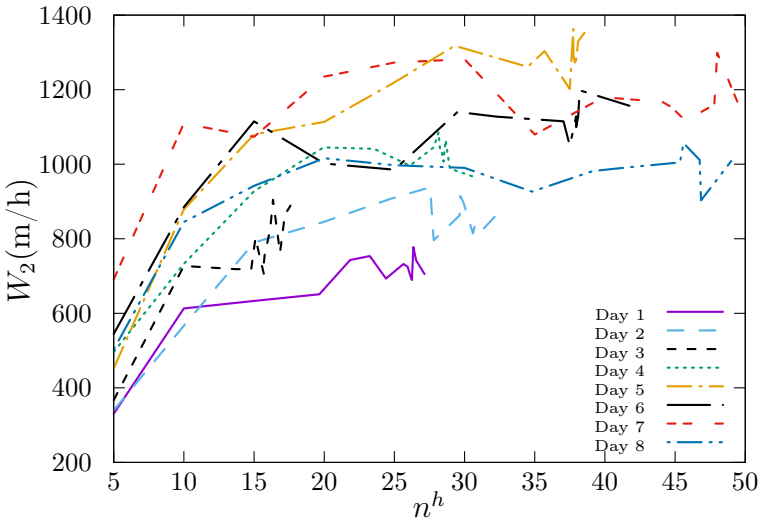


Figure 2.12: Today's walking for different number of hubs

A similar analysis was done on the start time of the incentive program for $t_0 \in [16\ 24]$ when there were 35 hubs and $p_t = 10, \forall t \in [16\ 24]$. We increased the number of hubs to 35 to have enough accepted offers in the system and to be able to study their influence on the outcomes. We selected days 5–8 for this experiment because the following day of days

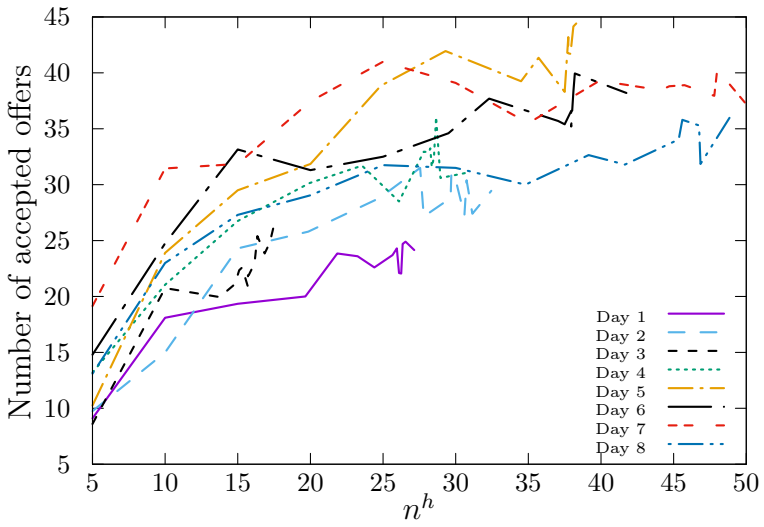


Figure 2.13: Number of accepted offers for different number of hubs

5–8 had enough demand to keep the number of hubs around 35. Figure 2.15 shows the changes in the static relocation cost. The graphs exhibit different and sometimes multiple local minimals for different days. This variety in behaviors relates to the demand distribution of the days because sometimes a high demand in the middle of the incentive time window can disperse all collected bikes by the incentive and basically cancel the effect of accepted offers on bike redistribution. However, regardless of fluctuations, an increasing track can be seen for the last hours of the days, which shows that incentive program helps the static relocation costs because it is less likely to have a high enough demand in the last hours of the day to anticipate enough accepted offers or relocation. Hourly walking and accepted offers demonstrate a decreasing trend in three of the days (6, 7, and 8), and day 5 had a peak at 22:00 PM (see Figures 2.16 and 2.17). The hourly number of lost users for days 6, 7, and 8 fluctuates around a consistent value until 20:30 PM. and takes either an increasing or decreasing trend after that (see Figure 2.18). However, this measure for day 5 keeps fluctuating, and it shows a peak around 22:00 PM. This reveals that there must be an instant increase in the demand of day 5 at this time. This also justifies the lower effect of incentive program on the static relocation cost for day 5 (Figure 2.15). The last variable whose influence in the system was investigated is the amount of the incentive. Since analysis

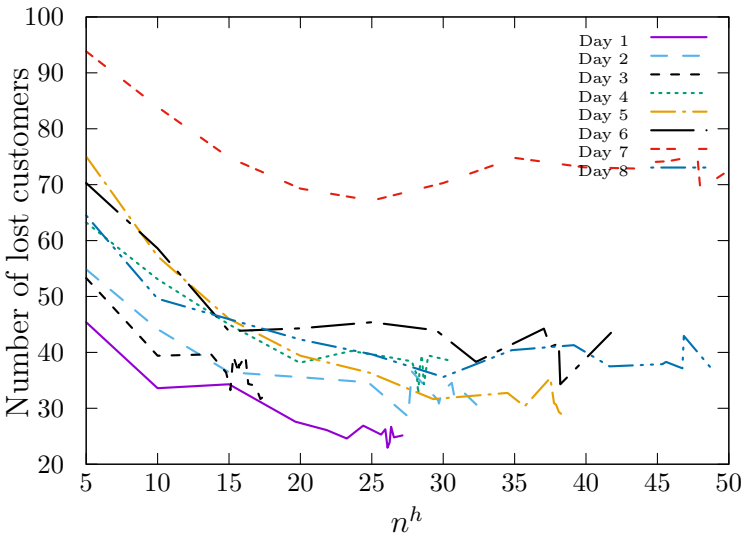


Figure 2.14: Number of lost users for different number of hubs

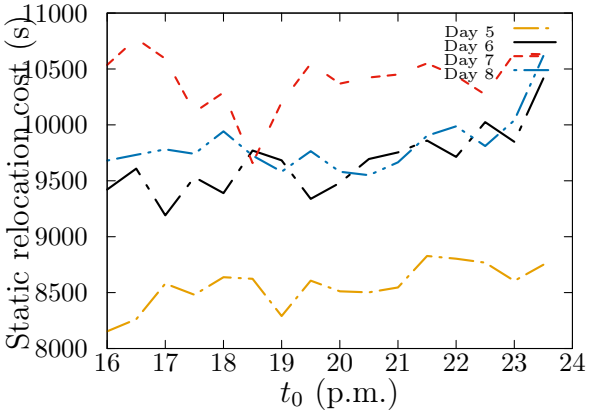


Figure 2.15: Static relocation cost for different start times of incentive program

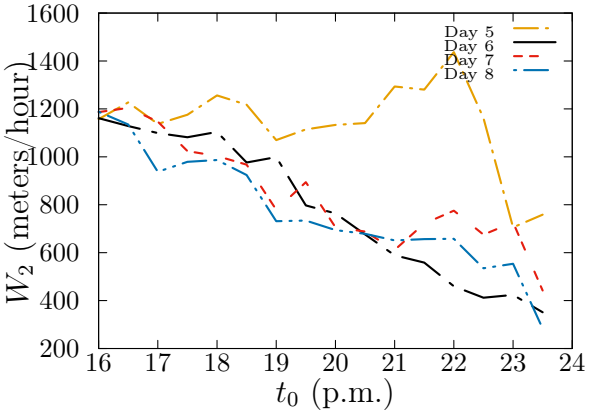


Figure 2.16: Todays' walking for different start times of incentive program

on the start time showed that in the late hours we can cancel a major part of the demand distribution's impacts and see a trend in the system's behavior, in this experiment, we fixed the start time to 9:00 PM. The number of hubs stays at 35, and all other parameters have the same values as the previous experiments. Figure 2.19 shows how the amount of the incentive affects the static relocation cost in the system. For all days, the cost had a decreasing track until around 10 points of incentive and remained consistent for higher values. The reason is that in the late hours of the days in the considered system, the demand is not high and, for more than 10 points, the number of accepted offers does not change (see Figure 2.20).

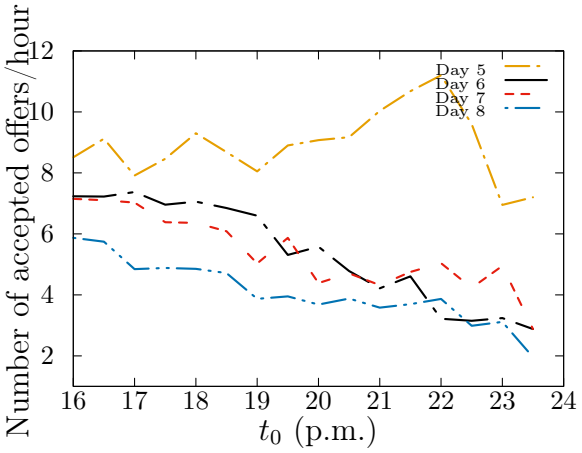


Figure 2.17: Number of accepted offers for different start times of incentive program

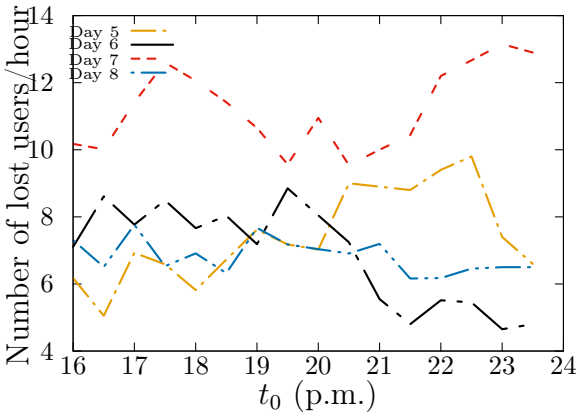


Figure 2.18: Number of lost users for different start times of incentive program

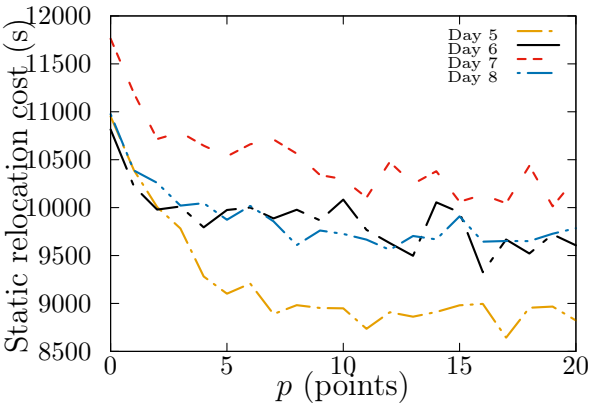


Figure 2.19: Static relocation cost for different amounts of incentive

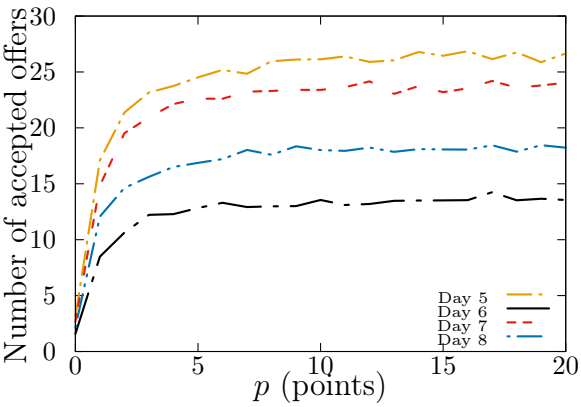


Figure 2.20: Number of accepted offers for different amounts of incentive

2.5.4 Sensitivity Analysis on Parameters

In the proposed model, there are parameters whose values are not determined based on any historical data since they are not recorded in the current system, such as walking limit and ratio of opportunistic users. In this section, we discuss more experiments that were conducted to explore how the variation of these two parameters would change the outcomes of rebalancing.

First, we fixed the variables as $n^h = 35, t_0 = 18$p.m., and$p_t = 10,\ \forall t \in [18\ 24]$, and all parameters were the same as in previous analyses except walking limit, which changed in the

range [100 1500]. Generally, by increasing the walking limit in the system, the number of lost users decreases and, as a result, the system serves more customers. Thus, this analysis can provide information about the system's behavior to demand increase as well. Figures 2.21 and 2.22 show the changes of the lost users and accepted offers in the system for different values of walking limits. A quick look at these graphs reveals that the number of lost users drops by increasing the walking limit, and these saved users proportionally lead to a higher accepted offer. From another point of view, this means that the number of served customers increases. Figure 2.23 shows that when the number of served customers(or the demand in general) is high during late hours, an earlier start of the incentive program will not help to save on static relocation costs. This outcome stems from the fact that high demand disperses the collected bikes in the hubs again and, consequently, cancels the effect of accepted offers on static relocation cost. Also, the result demonstrates a consistent increase in walking as the walking limit grows.

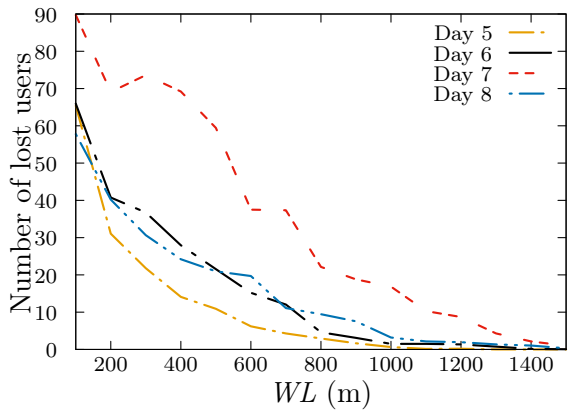


Figure 2.21: Number of lost users for different walking limits

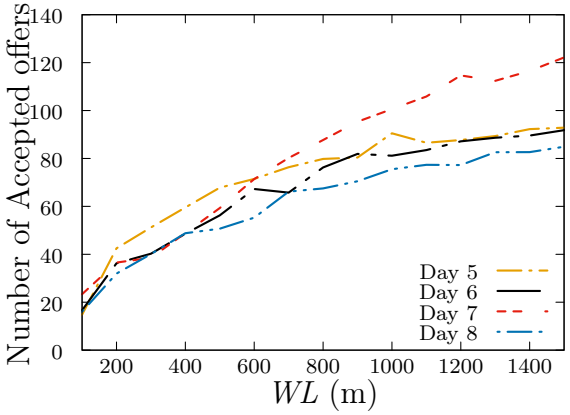


Figure 2.22: Number of accepted offers for different walking limits

Second, a similar analysis was performed on the ratio of opportunistic users from 0 to 50 percent of regular users. It shows no remarkable changes in the values of the considered measures. For example, Figure 2.24 shows that the number of accepted offers increases by a maximum 15, which has no significant effect on static relocation cost (see Figure 2.25).

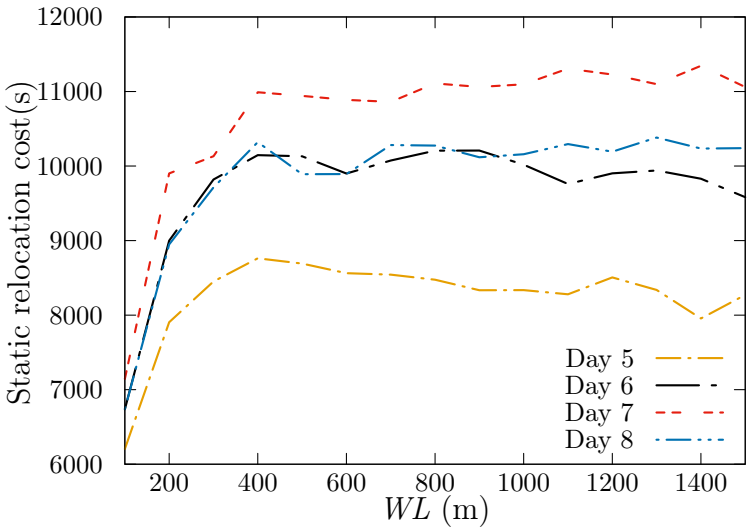


Figure 2.23: Static relocation cost for different walking limits

Although this result can be different for other values of the controlled variables and parameters, it implies that the behavior of the system is not sensitive to this parameter unless the incentive program causes a bigger than 50 percent increase in demand.

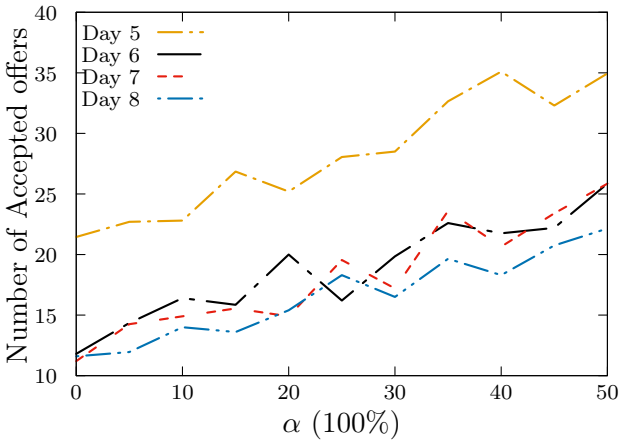


Figure 2.24: Number of accepted offers for different number of opportunistic users

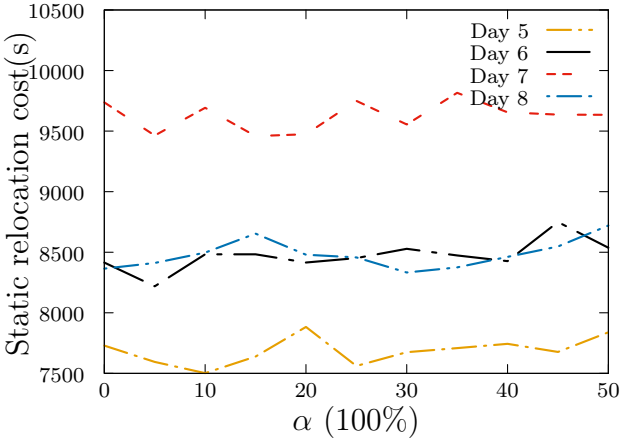


Figure 2.25: Static relocation cost for different number of opportunistic users

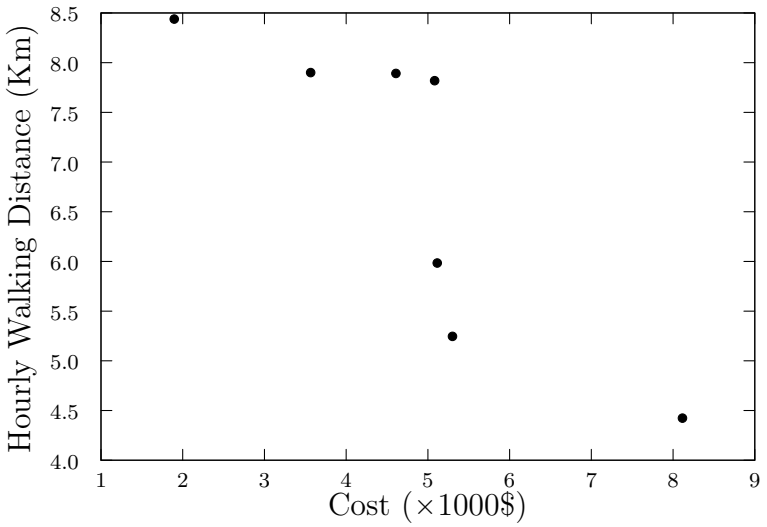


Figure 2.26: Pareto frontier for MoBike case

2.5.5 Large-scale Case

To show the capability of the algorithm on larger scales of the problem, we used the trip data of MoBike in Shanghai. MoBike is a free-floating bike sharing system recently acquired by Meituan Dianping, a Chinese group with a website for local food delivery services, consumer products, and retail services. A part of the MoBike Shanghai sharing system with a 10×10 km^2 area was selected and split into 250×250 m^2 zones. The walking limit was set to 300 meters and the K-means and Manhattan distance were used. As we did not have access to detailed information of this bike sharing system, we assumed that the operation costs were the same as the previous numerical experiment and that there were 100 trucks available for static relocation. The ranges for variables $(n^h, t_0, \boldsymbol{p})$ were set to $([20\ 150], [0\ 15], [16\text{PM}\ 24\text{PM}])$, respectively, and the termination condition was set to 15 generations. Figure 2.26 shows the Pareto-optimal frontier for this case. It clearly shows the trade-off between average hourly walking distance and the total cost. By increasing about 1 Km (from 4.5-5.5) hourly walking distance, the operator can save more than 30% of total cost.

2.6 Conclusion

In this study, we proposed a hybrid rebalancing strategy, i.e., combining an incentive program and an operator-based rebalancing and dynamic hubbing concept, i.e., determining daily hubs (number and locations) to assist hybrid rebalancing. We presented a multi-objective simulation optimization method for solving the proposed strategy for free-floating bike sharing systems by considering two objectives—cost and service level. The optimized outputs are the number and location of hubs, start time of incentive program, and incentive level. Using the solution algorithm proposed in our previous work, we also obtained the routing of rebalancing trucks and the number of bikes being picked up or dropped off at each stop along the routes.

We showed the effectiveness of the proposed approach on a real data set at the University of South Florida and applied this model on a large-scale free-floating bike sharing system in Shanghai to demonstrate its applicability on systems with different characteristics. The numerical example shows the trade-off between the hourly walking distance of users and the total cost of the operator. Given the Pareto frontier, the operator may make decisions on which solution to implement based on its business interests. Free-floating bike-sharing has been promoted as an environmentally-friendly micromobility solution. However, the air pollution emissions caused by operator-based rebalancing offset the environmental benefits of the mode. Due to the intrinsic mismatch of supply and demand (spatial and temporal), rebalancing is necessary to improve the efficiency of the system. Hence, this study presents a way to make free-floating bike sharing more sustainable.

In future research, we will apply user-based redistribution for rebalancing bikes during the whole day according to demand distribution of the near future (for example, two hours ahead) instead of the following day. Additionally, obtaining and using offer-acceptance behavior of users by artificial intelligence based on his/her record is another direction of future research.

Chapter 3: Query Batching Optimization in Database Systems

In this chapter [1], we present paper P2. In this work, we take advantage of machine learning techniques to learn about the processing time of a batch of queries in database systems. This estimated time is used to develop a query batching model that clusters a given set of queries into some batches minimizing the total processing time. We propose two fast heuristic algorithms to solve the model. The copyright permissions for reusing previously published material in this chapter can be found in Appendix C1.

3.1 Introduction

Despite being barely 50 years old, database research has had a profound impact on the economy and society, creating an industry sector valued between US$37-US$50 billion annually [57]. The history of database research over the past 40 years has demonstrated a tremendous productivity that has led to the database system becoming arguably the most important development in the field of software engineering. The database system software is the foundation of today's Information Technology (IT) infrastructure and has fundamentally changed the way many organizations operate. In particular, the developments in this technology over the last few years have produced systems that are more powerful and more intuitive to use [58, 59]. This development has resulted in increasing availability of database systems for a wider variety of users.

Traditional database management systems process user-issued tasks (called *queries*) one by one, often by the order they arrive in the system. However, as a result of ongoing

[1] This chapter was published at C&OR. Eslami, M., Mahmoodian, V., Dayarian, I., Charkhgard, H., & Tu, Y. (2020). Query batching optimization in database systems. *Computers & Operations Research*. Permission is included in Appendix C.1

increases in both the number of users/requests and data size, as well as improvement in computer hardware, the last fifteen years have seen a shift from *one-query-at-a-time* approaches towards shared work systems where queries are executed in batches. A batch of query includes one or more queries on the same database that are executed together. Typically, there is a high likelihood for the queries on a database to share computing resources such as IO (reading/writing data from/to disk), CPU (processing data by processor), memory, etc. Therefore, executing queries as batches could introduce savings in terms of computing resource usage and processing time. To process a batch of queries, two main questions must be answered:

1. How should a given set of queries be "optimally" partitioned into one or multiple batches of possibly different sizes? and,

2. How should a given single batch of queries be "optimally" processed?

Most previous studies have focused on the second question and have presented different methods to process a batch of queries by efficiently retrieving required data by the queries of a batch from a database. The proposed methods are different ways of sharing the computation resources or taking advantage of query overlaps [60]. As a result of that stream of research, many single-batch processing databases including PsiDB [61], SharedDB [62], Datapath [63], MQO [64], Qpipe [65] have been developed. As for the query batching optimization (the first question above), however, even though it has been stated in many of these articles as a means to improve the performance and a line of potential future study, to the best of our knowledge, no prior article has thoroughly investigated that field.

The main goal of the query batching problem is to partition a set of queries into different batches in order to minimize the total query processing time. In that sense, the query batching problem has a close resemblance to the order batching problem [66, 67]. The order batching problem is an optimization problem, which involves the operation of retrieval of goods in a warehouse [68, 69]. Specifically, the order batching consists of grouping a

set of orders together, forming a batch, which is then assigned to a picker whose job is to retrieve all the orders within the same batch on a single tour through the warehouse. A common objective of order batching problem is to reduce picker travel time. In order batching problem, the retrieving time of a given batch of orders can be pre-calculated by solving a traveling salesman problem [70]. In the query batching problem, however, the processing time of a given batch is unknown until it is actually processed/executed. This is because of several reasons including (1) the size of the data that will be retrieved to process a query batch is unknown in advance; (2) defining the similarities between queries is not a trivial task; and (3) the processing time depends highly on the methods used for processing the batch.

As mentioned above, the processing time of queries depends on the database that is used. In this study, we mainly focus on the so-called relational database management systems (RDBMS). Such a commonly used database system software stores data in the form of tables, using columns and rows and follows the database normalized structure introduced by Codd [71]. Most of the well-known industrial database systems such as Microsoft SQL Server, Oracle, MySql, Postgres are for relational databases. The single-batch processing databases that have been mentioned before, are a type of RDBMS. Although the approach that we develop for generating the batches is independent from the methods used for processing each batch, our focus is mainly on PsiDB for processing each batch in this study (because its implementation/code was available to us). PsiDB is a single-batch processing database system, which is newly developed and has shown to perform better than retrieving queries one by one, by a factor of almost 30x on workloads with large number of queries [61].

Our main contributions in this paper can be summarized as follows.

1. An effective query batching requires an understanding of the processing time of each potential batch of queries, before that batch is formed and executed. We therefore, develop a quadratic function based on the resulting coefficients of a linear regression model to predict the processing time of a given batch using PsiDB.

2. Building upon our batch processing time predictor, we develop a Mixed Binary Quadratic Program (MBQP) to efficiently partition any set of queries into batches (of possibly different sizes). We also propose an effective symmetry breaking technique to strengthen the proposed MBQP.

3. Since the main goal of the proposed algorithm is to minimize the query processing time, the solution time of the MBQP becomes critical. We first prove that the proposed MBQP is in fact an NP-hard problem. Consequently, one cannot afford to solve the MBQP for large-scale problems using exact approaches, as it could be too time demanding. Therefore, we develop two heuristics, the so-called Restricted-Cardinality Search Methods I and II (RCSA-I and RCSA-II), to generate high-quality solutions for the proposed MBQP in a relatively short amount of time.

4. We conduct a comprehensive computational study on three well-known benchmark database systems: TPC-H [72], TPC-DS [73], and Join-Order Benchmark (JOB) [74]. We consider instances of TPC-H with 5 tables and up to 10 GB of data, instances of TPC-DS with 10 tables and up to 5 GB of data, and instances of JOB with 23 tables and up to 4 GB of data. We generate a total of 600 random instances based on these database systems, each containing between 32 to 4096 queries. The computational results show that the proposed heuristics can generate optimal solutions to the MBQP for the instances for which checking the optimality was possible (instances with up to 512 queries). More importantly, we show that the total processing time of the queries by batching generated by the MBQP is up to 61.8% smaller than using PsiDB directly, i.e., processing all queries as a single batch. Moreover, we show that the proposed quadratic time prediction function has a high accuracy, i.e., R-squared value between 0.86 and 0.98.

The rest of this paper is organized as follows. In Section 5.2, we provide preliminaries on database systems. In Section 3.3, we describe the problem in hand in details. In Sec-

tion 3.4, a detailed description of the proposed solution approach is given. In Section 3.5, we conduct a comprehensive computational study. Finally, in Section 3.6 some concluding remarks are given.

3.2 Preliminaries

In this section, we provide a high-level description of a well-known type of database systems, Relational Database Management System (RDBMS), and its main components [71, 75, 76]. Due to the significant number of applications of this class of database systems, it is the main focus of this research. One of the critical components of a RDBMS is the so-called *query optimizer*, which has raised a lot of research questions in the last four decades. In this section, we briefly review the existing body of literature on query optimizer and explain a research gap, which is the main motivation of this study. In addition to the traditional RDBMS, we also cover a few new database systems based on RDBMS, developed for specific needs such as single-batch processing database systems.

3.2.1 A Relational Database Management System

Since in this paper we mainly focus on relational databases, it is essential to have fundamental knowledge about databases. Hence, we survey a few key components of relational databases in this section. The system software that handles the database functionality is called a Database Management System (DBMS). An RDBMS is a type of DBMS which follows certain rules designed by [71] using the relational model. In the following, we explain the structure of data in relational databases and the methods of retrieving data efficiently from such a database. Figure 3.1 shows the main components of an RDBMS. We briefly explain each component in this section.

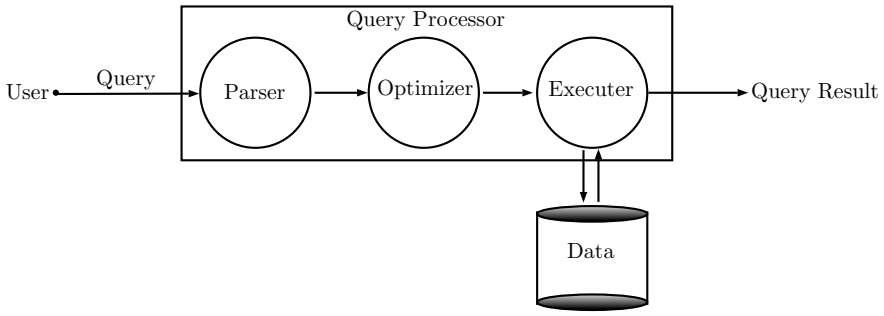


Figure 3.1: A relational database management system for one-query-at-a-time processing

3.2.1.1 Data Structure

Data values in an RDBMS are stored in the form of *tables*. A table is a fixed data structure that consists of *columns* (also called *attributes*) and *rows* (Tuples). A table has a specified number of columns, but can have any number of rows. Each row is a record stored in the database and each column is a type of data whose domain (e.g., data type, length, range of value) is pre-described. In designing a database table, certain attribute(s) of a table are set to be unique identifiers of the entire row, and such attribute(s) is called a *key*. There can be multiple keys in a table, e.g., Social Security Number (SSN) and Student ID are both keys in a University Student table as they both can be viewed as IDs of students (rows). Typically there are two kinds of key: *Primary key*, which is chosen among all keys in a table by the database designer, and *foreign keys*, referring to another table's primary key. The computations in such systems are described by an algebraic system named *Relational Algebra* (RA). Among the RA operators, concatenating rows from different tables is called *join*.

3.2.1.2 Queries

A query is a language expression that describes data to be retrieved from a database based on specific criteria. All the queries in an RDBMS are in Structured Query Language (SQL) form. The SQL language has changed through the time and deviated in several ways from its theoretical foundation. Our focus in this paper is on standard query expression of

SQL. Such expressions are done in a *declarative* manner, which focuses on descriptions of the query results. A simple SQL statement (i.e. `SELECT` $a_1, ..., a_w$ `FROM` $t_1, ..., t_m$ `WHERE` $c_1, ..., c_f$) has the following parts:

- `SELECT`: Specifies the columns in the result
- `FROM`: The table(s) which data is stored in
- `WHERE`: A set of conditions which specifies criteria for the retrieved rows

The scope of this study is on the queries of the form `SELECT..FROM..WHERE` (SFW). Such queries are known to be one-to-one mappings to core operators in relational algebra, which is the query language used to described actual computations required to process the queries.

3.2.1.3 SQL Parser

SQL parser handles two tasks: Syntax validations and query transformation. Any query that is being executed, needs to be translated to the relational algebraic statement first. We have the same main components in the relational algebra:

- Projection: The set of columns/attributes written as $\Pi_{a_1,...,a_w}$;
- Join: The tables needed to execute the queries;
- Selection: The specific conditions on the attributes $\sigma_{c_1,...,c_f}$.

A SFW query in SQL language, translates to $\Pi_{a_1,...,a_w}\sigma_{c_1,...,c_f}(t_1 \times ... \times t_m)$ in algebraic form. After this transformation, the query is ready to be optimized and then executed.

3.2.1.4 Query Optimizer

The query optimizer is a main component in terms of database performance enhancement [58]. The query optimizer attempts to determine the most efficient way to execute a query by creating multiple execution plans and choose the optimal one. This happens right

after parsing the query and before accessing the data. A query plan determines the query execution plan and how to access the data. Two different plans for the same query would have the same results but the execution time and resource consumption for each of them can be different.

3.2.1.5 SQL Executor

Given a query execution plan, the query processor fetches the data in the tables and generates the results. There are many factors that play major roles in a query execution such as buffering or sequential reading, which are outside the scope of this paper.

3.2.2 A Research Gap

As mentioned above, query optimizer plays a key role in any database system. Query optimizer techniques have been an active research topic in the last four decades. Up to this day, there are three main lines/streams of research:

1. How to efficiently process a given single query and retrieve the results?

   In traditional database systems, each query is expected to be executed independently [77, 78, 79, 80]. Figure 3.1 shows all the components of an RDBMS for executing a single query. A request from the user translates into a query, and the query will go through the parser, optimizer and executor to get the data from the storage and send the results to the user [81, 76]. The underlying idea of this line of research is to compute optimal execution plans for a single query. As a result of this line of research, several query optimizer techniques such as indexing and sequential scanning are developed [82, 83, 84, 85].

2. How to efficiently process a given set of queries as a single batch?

   The landscape of data management systems has changed dramatically over the last decade, and traditional RDBMS are frequently criticized for not meeting requirements

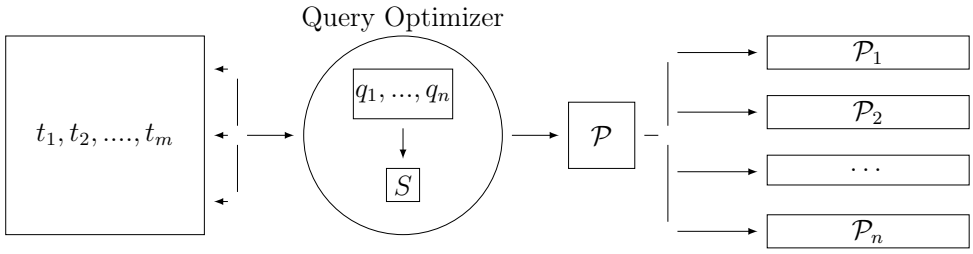


Figure 3.2: A single batch-processing, query optimizer model. Here $t_1, t_2, \ldots, t_m$ are Tables, $q_1, \ldots, q_n$ are the queries and $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n$ are output data for each query

of modern data-intensive applications. Today's database systems often need to serve a multitude of user queries in a short period of time. For example, Alibaba, a leading Chinese online retailer, needs to process a workload at the rate of 10,000 query a second. Under such high demand, the one-query-at-a-time model falls short in meeting performance requirements because it can easily lead to resource contentions. On the other hand, the database community observed that queries in a workload may have a significant overlap in the data access paths and intermediate query results. Plausible efforts to harness computation and data access sharing among processing multiple queries have been reported [63, 61, 62, 65, 64].

In light of the above, the database community developed several database systems that combine a set of queries into a single batch and attempts to efficiently process the batch. Examples include: relational database management systems for batch processing i.e., PsiDB and SharedDB designed for workloads with large number of users. These databases are capable of handling a set of queries instead of one single query. The idea of this line of research is demonstrated in Figure 3.2. The query optimizer in Figure 3.2 gets a set of queries $q_1, \ldots, q_n$ and combines them into one query S so it can be processed by the database. After executing the combined query, $\mathcal{P}$ is the table containing all information required by the set of queries, which in the next step is distributed among them.

3. How to efficiently partition a given set of queries into batches and process each batch individually?

   While there are many studies for the first and second lines of research, there are no studies for this line of research (to the best of our knowledge). Therefore, the goal of our research is to study whether partitioning the queries into batches can improve the batch-processing database systems performance.

   The idea of such a database system is demonstrated in Figure 3.3. In this database system, a set of queries $q_1, \ldots, q_n$ are partitioned into multiple batches $S_1, \ldots, S_v$. Associated with each batch S_i, there is a result $\mathcal{P}_i$, leading into a set of results: $\mathcal{P}_1, \ldots, \mathcal{P}_v$. Each $\mathcal{P}_i$ is then distributed among the queries in batch S_i.

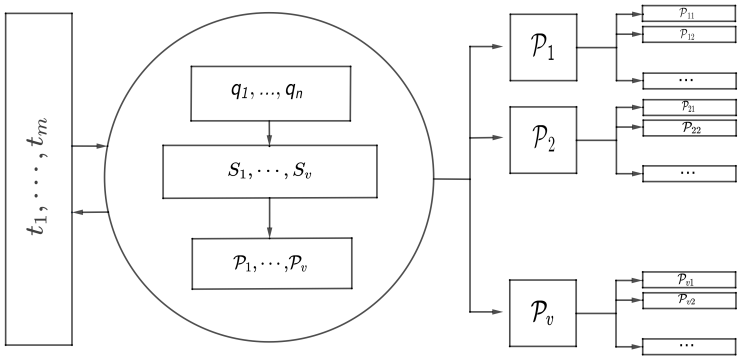


Figure 3.3: A batch-processing model with partitioned query. Here $t_1, t_2, \ldots, t_m$ are Tables, $q_1, \ldots, q_n$ are the queries, $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_v$ are batches of queries and $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_v$ are output data for each batch

3.3 Problem Description

Batching is one of the main functions of modern databases due to how directly it affects the performance of data retrieval. On a PsiDB, the performance is impacted by the way we batch a set of queries because it impacts the required run time to retrieve the data. In this research, the main goal of the query batching problem (QBP) is to find the best way

of partitioning a given set of queries into batches in order to minimize the total processing time. For example, let $Q = \{q_1, q_2, q_3\}$ be a set of queries that needs to be processed. In this case, Q can be processed in 5 different ways by partitioning it into 1, 2, or 3 batches as follows: $\{(q_1), (q_2), (q_3)\}$, $\{(q_1, q_2), (q_3)\}$, $\{(q_1, q_3), (q_2)\}$, $\{(q_1), (q_2, q_3)\}$, and $\{(q_1, q_2, q_3)\}$. The total processing time of each solution of the query batching problem may vary due to the correlation between the queries. The correlation between queries can be established by comparing the data that each query is attempting to retrieve. For example, if q_1, q_2, and q_3 request similar data from similar tables, it will probably be faster to process them as a single batch.

Overall, in order to solve the query batching problem, it is important to know how long the processing time of any potential batch would be. However, computing such a processing time is not a trivial task. This is because to process a batch, a database system typically takes two steps and each one requires some computational efforts [61, 86]. In the first step, it combines/converts all the queries in the batch into a single global query containing all the attributes and records of each query in the batch. The database system will then process the global query and retrieve the entire data requested by the global query. It is also worth mentioning that processing the global query in an efficient manner is not a trivial task by itself because it requires operations such as joining tables and filtering. More importantly, the amount of data retrieved as a result of processing the global query is unknown in advance. In the second step, some additional computational efforts should be made for searching the retrieved data (that its size is unknown in advance) and distributing it among queries in the batch. Due to the existence of unknown parameters in these two steps, computing the exact total processing time of a given batch (in advance) is a challenging task that depends highly on the single-batch-processing method used.

In light of the above, if one can generate a function for predicting the processing time of any possible batch, then a mathematical program for obtaining an optimal solution of the QBP can be developed. Specifically, let $Q := \{q_1, \ldots, q_n\}$ be the set of queries. Observe

that since the total number of queries is n, the maximum number of possible batches would be n, i.e., each batch will contain exactly one query. We denote batch $j \in \{1, \ldots, n\}$ by $S_j \subseteq Q$ and its predicted processing time by $P(S_j)$. By using these notations, the QBP can be stated as follows:

$$\min_{S_1,\ldots,S_n \subseteq Q} \Big\{ \sum_{j=1}^{n} P(S_j) : \cup_{i=1}^{n} S_j = Q \text{ and } S_i \cap S_j = \varnothing \quad \forall i, j \in \{1, \ldots, n\}, i \neq j \Big\}.$$

Note that due to the significant variability of processing times in practice, it would be hard to predict the processing times accurately. So, instead of predicting processing times, we predict the value of an indicator of processing times. In other words, in the remaining of this paper, $P(S_j)$ basically represents the predicted value for the processing-time indicator of batch S_j. So, for the convenience, whenever we say the 'predicted processing time', we mean the 'predicted value for the processing-time indicator' in the remaining of this paper.

In the next section, we will explain that the processing-time indicator of batch S_j is defined as the square of $\log_2$-transformation of the processing time of the batch S_j in this paper (see Eq. (3.3) for details). Hence, $P(S_j)$ attempts to predict the value of the indicator for the batch S_j. We observe that the indicator consists of a 'square' function and a 'log-transformation' function. We use the square function because of two main reasons including: (1) linear functions cannot capture the impact of the number of queries allocated to batches when being used in our developed optimization model for the QBP, see Section 3.4.2 for details; and (2) square function can be handled effectively by (commercial) mixed integer programming solvers such as Gurobi or CPLEX since it is convex. Also, we use the log-transformation because that is a common heuristic way to reduce the variability of data. Through this article, we use base 2 for the log-transformation function because it is a natural choice given that computers use binary encoded systems. However, during the course of this study, we observed that our proposed method is not sensitive to the base value.

Finally, we assume that queries are not empty. In other words, we assume that each query in Q should involve (i.e., search) at least one of the tables in the database system under consideration.This assumption is needed only for the purpose of defining valid instances when proving that the QBP is NP-hard (see the Appendix).

3.4 The Proposed Approach

In this section, we explain our proposed approach for solving the query batching optimization problem. Our approach contains three phases as follows.

1. Developing the batch processing time prediction function
2. Developing an optimization model to effectively partition the queries into batches
3. Developing effective solution methods to solve the optimization model

In the remaining of this section, we will provide further details about each phase.

3.4.1 The Batch Processing Time Prediction Function

In this section, we develop a function $P(S')$ that serves to predict batch processing time for any arbitrary query batch S' of a given database. An ideal function $P(S')$ has the two following desired characteristics: (1) The function has high accuracy; and (2) The function is simple as the complexity of the optimization model for the QBP and its solution approaches highly depend on it. The use of the QBP is justified only if the sum of the solution time of the QBP and the processing time of the batches prescribed by the QBP is dominated by the processing time of the same set of queries when the QBP is not employed. Accordingly, if one cannot solve the QBP quickly then the benefit of batching may not be significant.

As mentioned in Section 3.3, for a given query batch S', its predicted (indicator of) processing time, $P(S')$, depends on the size of the data that must be retrieved for processing

batch S'. However, such information is not known in advance and becomes available only while executing the batch. Therefore, in our approach we benefit from some other attributes (or features) of S' that are either available in advance or easy to compute to predict the processing time of batch S'.

Let $T := \{1, \ldots, m\}$ denote the index set of all tables in the database (where m is the total number of tables). To predict the processing time of a batch S', we introduce $|T^{S'}| + 1$ attributes associated with batch S', where $T^{S'} \subseteq T$ denotes the index set of all tables required by the queries in batch S'. The first $|T^{S'}|$ attributes are indicators of the size of tables required for batch S'. Specifically, we introduce the attribute $\log_2(s_t)$ for each $t \in T^{S'}$ with s_t being the number of records in table t. Note that the size of tables in a database can be significantly different and hence the proposed $\log_2$-transformation can be helpful in normalizing the size of the tables in such cases. We also introduce one additional attribute, indicating the number of queries in batch S', denoted by n'. Using the proposed attributes, we construct a function that can predict the processing time of any given batch. This function takes the following form.

$$P(S') := \left[\beta_0 + \beta_1 n' + \sum_{t \in T} \beta_{t+1} \log_2(s_t) y'_t\right]^2, \tag{3.1}$$

where $y'_t \in \{0, 1\}$ is a parameter indicating whether table t is required to process queries in batch S'. Also, $\beta_0, \beta_1, \ldots, \beta_{|T|+1} \in \mathbb{R}$ are the coefficients that should be estimated. In Section 3.5, we will numerically show that the proposed function has high prediction accuracy on the three different database benchmarks that we consider. This implies that the proposed function addresses the first required characteristic, mentioned at the beginning of this section. Additionally, we observe that the proposed $P(S')$ is a quadratic function of n' and $y'_1, \ldots, y'_m$. Despite being a quadratic function, $P(S')$ can still be handled easily by commercial solvers such as CPLEX and Gurobi when used in the optimization model that we develop for the QBP, which satisfies the second desired characteristic mentioned above. Note that we do

not use a linear function of the form,

$$\beta_0 + \beta_1 n' + \sum_{t \in T} \beta_{t+1} \log_2(s_t) y'_t \tag{3.2}$$

instead of Eq. (3.1), since Eq. (3.2) has a significant weakness when being employed in the QBP. In Section 3.4.2, we will explain in detail why such a linear function is not suitable.

We estimate coefficients $\beta_0, \beta_1, \ldots, \beta_{|T|}$ through a training procedure over a large set of randomly generated batches of queries. In Section 3.5, we discuss in details how a random batch is generated. For each batch S', one can compute the value of each attribute and then actually process the batch to compute its real processing time, denoted by RealTime. The goal is to estimate the coefficients of the proposed processing time prediction function $P(S')$ such that,

$$P(S') \approx [\log_2(\text{RealTime})]^2. \tag{3.3}$$

This implies that,

$$\sqrt{P(S')} = \beta_0 + \beta_1 n' + \sum_{t \in T} \beta_{t+1} \log_2(s_t) y'_t \approx \log_2(\text{RealTime}).$$

This itself implies that in order to estimate the coefficients of the proposed function, one can simply fit a linear regression function. Specifically, $\log_2(\text{RealTime})$ can serve as the dependent variable and the proposed attributes can serve as independent variables. In this study, we employed the package *linear_model* from *sklearn* in Python to compute the coefficients based on Ridge linear regression [87].

3.4.2 The Optimization Model

In this section, we develop a MBQP to solve the QBP using the proposed processing time prediction function. In the remaining of this section, it is assumed that coefficients

$\beta_0, \beta_1, \ldots, \beta_{m+1}$ are already estimated and are available as parameters. However, the assignment of the queries to batches, and consequently the number of queries in each batch and the set of tables required for processing each batch are unknown and are to be determined by the optimization model. Let $Q_t \subseteq \{1, \ldots, n\}$ be the set of queries that requires table $t \in T$. Note that, by assumptions, each query requires at least one table. Let k be a user-defined parameter denoting the maximum number of possible batches that are allowed to be formed. Note that in the worst case the number of batches equals the number of queries; one batch per query. Hence, one can set k to n if no information about the value of k is available. Let x_{ij} be a binary decision variable that equals 1 if query $i \in \{1, \ldots, n\}$ is assigned to batch $j \in \{1, \ldots, k\}$, and 0 otherwise. Also, let y_{jt} be a binary decision variable indicating whether table $t \in T$ is used by at least one query in batch $j \in \{1, \ldots, k\}$. For each batch $j \in \{1, \ldots, k\}$, we introduce the continuous variable n_j for indicating the number of queries in batch j. As an aside, although n_j is defined as continuous decision variable, it will naturally take integer values in the proposed formulation for each batch $j \in \{1, \ldots, k\}$. So, one can define n_j as an integer variable in practice for each batch $j \in \{1, \ldots, k\}$. Finally, for each batch $j \in \{1, \ldots, n\}$, we introduce the binary decision variable z_j that takes the value of 1 if at least one query is assigned to batch j, i.e., batch j is not empty. Using these notations, the query batching problem can be formulated as the following MBQP,

$$\min \sum_{j=1}^{k} \left[\beta_0 z_j + \beta_1 n_j + \sum_{t \in T} \beta_{t+1} \log_2(s_t) y_{jt}\right]^2 \tag{3.4}$$

$$\text{s.t.} \sum_{j=1}^{k} x_{ij} = 1 \qquad \forall i \in \{1, 2, \ldots, n\} \tag{3.5}$$

$$n_j = \sum_{i=1}^{n} x_{ij} \qquad \forall j \in \{1, 2, \ldots, k\} \tag{3.6}$$

$$x_{ij} \leqslant y_{jt} \qquad \forall t \in T,\ \forall i \in Q_t,\ \forall j \in \{1, 2, \ldots, k\} \tag{3.7}$$

$$y_{jt} \leqslant z_j \qquad \forall t \in T,\ \forall j \in \{1, 2, \ldots, k\} \tag{3.8}$$

$$y_{jt} \leqslant \sum_{i \in Q_t} x_{ij} \qquad \forall j \in \{1, 2, \ldots, k\},\ \forall t \in T \tag{3.9}$$

$$z_j \leqslant n_j \qquad \forall j \in \{1, 2, \ldots, k\} \tag{3.10}$$

$$n_j \geqslant 0 \qquad \forall j \in \{1, 2, \ldots, k\} \tag{3.11}$$

$$x_{ij}, y_{jt}, z_j \in \{0, 1\} \qquad \forall j \in \{1, 2, \ldots, k\},\ \forall i \in \{1, \ldots, n\},\ \forall t \in T, \tag{3.12}$$

where the objective function (3.4) measures the total predicted processing time of non-empty batches. Constraint (3.5) ensures that each query is assigned to exactly one batch. Constraint (3.6) computes the value of n_j for any batch $j \in \{1, \ldots, k\}$. Constraint (3.7) ensures that if table $t \in T$ is required by at least one query in batch j, then $y_{jt} = 1$. Constraint (3.8) guarantees that if batch j needs some tables to be processed, i.e., $y_{jt} = 1$ for some $t \in T$, then $z_j = 1$ because the batch j must be non-empty in that case. Constraint (3.9) ensures that if for any given batch $j \in \{1, \ldots, k\}$ table $t \in T$ is not needed, i.e., $\sum_{i \in Q_t} x_{ij} = 0$, then we must have that $y_{jt} = 0$. Finally, Constraint (3.10) guarantees that if no query is assigned to a given batch $j \in \{1, \ldots, k\}$, i.e., $n_j = 0$, then we must have that $z_j = 0$. We note that Constraints (3.9) and (3.10) are not necessary for solving the QBP if $\beta_0, \beta_1, \ldots, \beta_{m+1} \geqslant 0$. However, for the purpose of studying the computational complexity of the QBP (see Section 3.4.2.1) we include them in the formulation. These two additional inequalities combined with the assumption that each query should involve at least one table guarantee that the (predicted) processing time of each batch is computed properly even if there exists $l \in \{0, 1, \ldots, m+1\}$ such that $\beta_l < 0$. Moreover, these two constraints ensure that if a batch is empty and/or do not need to explore any table then its associated predicted processing time is zero. Also, if a batch does not need some of the tables then those will not impact the processing time of that batch. In the remaining of this paper, we denote the formulation (3.4)-(3.12) by MBQP(k) where k is the input parameter set by users.

Observe that if $k = n$ then the objective function of the proposed formulation simply captures $\sum_{j=1}^{n} P(S_j)$ where $P(S_j)$ is the quadratic function obtained in Section 3.4.1. So, it is natural to ask why cannot $P(S_j)$ be a linear function, i.e., Eq. (3.2)? Note that the

objective of the QBP is to minimize the total predicted processing time of a set of queries by regrouping them into batches. A linear function may have high accuracy for predicting the processing time of each batch individually but when used in the optimization framework for minimizing the total predicted processing time, it will perform poorly. Specifically, if one uses a linear processing time prediction function, the objective function of the proposed formulation (when $k = n$) will change to

$$\sum_{j=1}^{n} \beta_0 z_j + \beta_1 n_j + \sum_{t \in T} \beta_{t+1} \log_2(s_t) y_{jt} = \beta_1 n + \sum_{j=1}^{n} \beta_0 z_j + \sum_{t \in T} \beta_{t+1} \log_2(s_t) y_{jt}.$$

This implies that the number of queries in the batches will not play any role in the optimization process because $\beta_1 n$ is a constant. In that case, by assuming that $\beta_0, \beta_1, \ldots, \beta_{m+1} \geqslant 0$ (which is likely to be the case in practice), the optimal solution for the QBP is to create only one batch containing all queries. Therefore, such linear functions are not suitable for showing the effectiveness of batching. Hence, the main purpose of using the proposed quadratic function is that it allows to capture the importance of the number of queries assigned to different batches. We next present a technique to strengthen the proposed formulation.

The proposed model can be strengthen by reducing the level of existing symmetry in the formulation. As a result of the existing symmetry, one can expect that commercial solvers such as CPLEX and Gurobi struggle to solve the proposed MBQP without applying symmetry breaking techniques [88]. To break the symmetry, we propose to add the following constraints to the model,

$$x_{ij} = 0 \qquad \forall i \in \{1, \ldots, n\} \text{ and } \forall j \in \{i+1, \ldots, k\}. \tag{3.13}$$

Observe that without using the proposed symmetry breaking constraints, a query $i \in \{1, \ldots, n\}$ can be assigned to any batch $j \in \{1, \ldots, k\}$ in the optimization model. This is in fact one of the main reasons that symmetry exists in our proposed formulation. Hence, our proposed symmetry breaking technique limits this flexibility by forcing the optimization

model to assign query $i \in \{1, \ldots, n\}$ to only a batch $j \in \{1, \ldots, k\}$ with $j \leqslant i$. This implies that query 1 is only allowed to be assigned to batch 1, query 2 is only allowed to be assigned to either batch 1 or 2, query 3 is only allowed to be assigned to either batch 1 or 2 or 3, and so on. This process obviously does not remove any feasible solution of the QBP and will (partially) break the symmetry in the proposed formulation. Finally, in terms of implementation, instead of explicitly adding the proposed symmetry breaking constraints, it would be computationally better not to generate x_{ij} for all $i \in \{1, \ldots, n\}$ and $j \in \{i+1, \ldots, k\}$ when creating the model. This efficient implementation is used in this study.

3.4.2.1 Computational Complexity

We now explore the computational complexity of the QBP when its objective function is the proposed quadratic function. We again note that although Constraints (3.9) and (3.10) are not necessary for computing optimal solutions for the QBP when $\beta_0, \beta_1, \ldots, \beta_{m+1} \geqslant 0$, we assume that they are included in the formulation for the purpose of studying the computational complexity of the QBP. This is because for identifying the computational complexity of the QBP, the decision problem of the QBP should be explored. The decision problem of the QBP (when employing the proposed quadratic function), denoted by `QBP`, can be stated as follows: does there exist a solution that can satisfy the following set of constraints,

$$\sum_{j=1}^{k} \left[\beta_0 z_j + \beta_1 n_j + \sum_{t \in T} \beta_{t+1} \log_2(s_t) y_{jt}\right]^2 \leqslant U$$

$$(3.5) - (3.12)$$

where U is a given parameter. The `QBP` is basically a feasibility problem in which the objective function of the proposed MBQP has changed to a constraint. Therefore, in order to make an accurate analysis on the computational complexity of the `QBP`, the objective value

of each feasible solution must be captured accurately in the QBP. That is the main reason that Constraints (3.9) and (3.10) are included in the QBP.

Theorem 3.1. *The query batching problem when employing the proposed processing time prediction function* (3.1) *is NP-hard.*

Proof. The proof is rather lengthy and is provided in the appendix. □

3.4.3 Heuristic Solution Methods

Observe that if $k = n$ then the size of the proposed formulation, i.e., MBQP(n), is $O(mn^4)$ because the number of constraints is $O(mn^2)$ and the number of variables is $O(n^2)$. This combined with Theorem 3.1 implies that there is little hope that the proposed MBQP can be solved quickly enough for practical-sized instances with possibly thousands of queries and multiple tables. In Section 3.5, we will show that instances with up to 500 queries on a database benchmark with 5 tables and around 10 GB of data can be solved to optimality using commercial solvers within a few hours (on a typical computing node). Obviously, for larger instances, the solution time for the MBQP is expected to increase significantly. As a result, even if optimal or near optimal batching can offer some savings in terms of processing time of a set of queries, these savings can be dominated by long solution time of the MBQP. As a consequence, in this section, we propose two simple heuristic algorithms permitting us to quickly generate high-quality feasible solutions to the proposed MBQP.

We refer to the first algorithm as the Restricted-Cardinality Search Method I (RCSA-I). The underlying idea of RCSA-I is to set the value of k to values smaller than n. Observe that for such values the size of MBQP(k) is significantly smaller than the MBQP(n). Hence, one can expect to solve the MBQP(k) faster than MBQP(n) using commercial solvers when k is smaller than n. To improve the solution time even further, RCSA-I restricts the formu-

lation even more by adding the following set of constraints to the MBQP(k),

$$\sum_{j=1}^{k} y_{jt} \leqslant 1 \qquad \forall t \in T. \tag{3.14}$$

Constraint (3.14) forces all queries requiring a given table t to be assigned to the same batch. The addition of Constraint (3.14) results in a restricted variant of MBQP(k) that we refer to as the MBQP-I(k) and it plays a key role in RSCA-I. Starting from $k = 1$, RCSA-I iteratively increases k with a stepsize one and solves the corresponding MBQP-I(k) in each iteration considering a time limit. As an aside, we note that the solution for $k = 1$ is trivial and hence MBQP-I(k) does not need to be solved for $k = 1$. If in a given iteration there is an improvement in the obtained objective value of MBQP-I(k) compared to the previous iteration, the algorithm increases k by one and starts a new iteration. Otherwise, the algorithm terminates and returns the best solution found and the corresponding number of batches, k^*. It is worth mentioning that for implementing RCSA-I, three points should be considered. First, the solution of each iteration can be used as a warm-start for the next iteration. Second, the proposed symmetry breaking technique developed for strengthening MBQP(k) can be used for MBQP-I(k). Third, because of Constraint (3.14), the maximum number of iterations of RCSA-I is m.

We refer to the second algorithm as the Restricted-Cardinality Search Method II (RCSA-II). RCSA-II is a two-phase method and its underlying idea is to improve the solution obtained by RCSA-I further (if possible). Hence, RCSA-II first calls RCSA-I in its first phase. Let k^* be the value of k in the last iteration of RCSA-I. For the second phase, the algorithm initializes k by k^* and then iteratively solves the MBQP(k), instead of MBQP-I(k). Similar to the first phase, the algorithm increases the value of k by one as long as some improvements in the obtained objective value of MBQP(k) is observed (compared to its last iteration). Otherwise, the algorithm terminates and returns the best solution found.

It is worth mentioning that the implementation points discussed for RCSA-I should also be considered for the RCSA-II.

3.5 A Computational Study

In this section, we investigate the performance of our proposed approach on three database benchmarks. For conducing the experiments, we employ SQL Server 2017 and Julia programming Lagrange 0.6.4. Also, to solve optimization models, we use CPLEX 12.9 and we employ PsiDB for processing each batch of queries in database. All the computational experiments are conducted on a workstation with Intel quad-core 3.6GHz i7-7700 processor, 32GB of DDR4-2400 memory, a 256GB SSD system disk, and a 2TB 7200 RPM hard drive for database storage. Also, in this study we impose a time limit on each iteration of RCSA-I and RCSA-II. The proposed time limit is 300 seconds for each iteration. All the data involved are stored on the same hard drive to ensure consistent Input/Output (I/O) rate across different experimental runs. The workstation runs on Windows 10 Enterprise Version 1709.

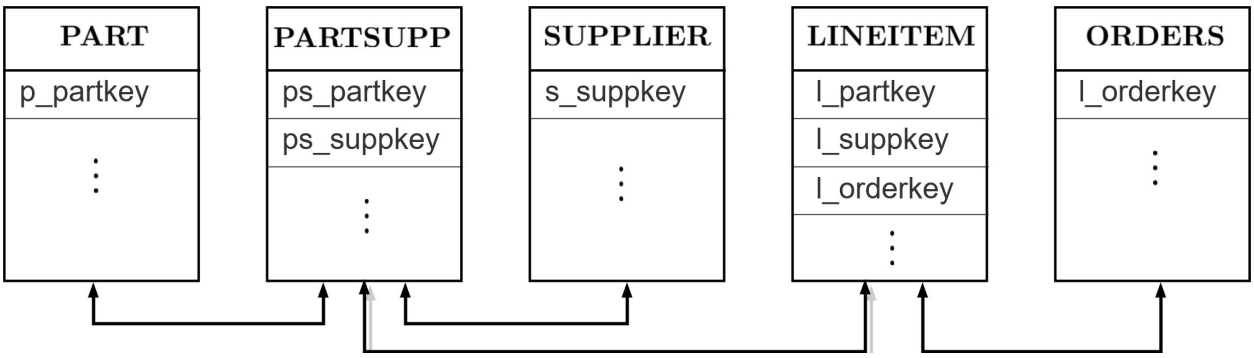


Figure 3.4: The structure of a TPC-H database

3.5.1 Database Benchmark TPC-H

In this section, we employ the well-known TPC-H database benchmark for showing the effectiveness of the QBP and our proposed heuristics [89, 72]. TPC-H database represents

a customer-supplier system. It contains information of the provided parts by the suppliers and the orders by the customers.

This database benchmark takes a parameter, the so-called Scale Factor (SF), as an input. This parameter indicates the approximate size (in gigabytes) of the random TPC-H database that will be generated. In this section, we consider three different values for the SF, i.e., $\{1, 5, 10\}$. That is, we generate three random TPC-H database replicas with the size of approximately 1, 5, and 10 GB, respectively. The generated databases have 5 tables with established relationships between some of them as shown in Figure 3.4. The relationships are defined by the so-called *foreign keys* and shown by arrows in Figure 3.4. These relationships are important because PsiDB uses them when processing a batch of queries. For example, if there is a batch of queries on only tables PART and SUPPLIER, PsiDB has to also use table PARTSUPP (when joining tables) as it contains both foreign keys of tables PART and SUPPLIER. As an aside, we note that since there are only 5 tables, in theory, there should be $2^5 - 1 = 31$ scenarios for a query to involve the tables. However, because of the relationships between tables, only 17 scenarios are feasible. The number of records in each table for different values of SF are reported in Table 3.1.

Table 3.1: The number of records in each table of TPC-H for three values of SF

Table Name	**SF=1**	**SF=5**	**SF=10**
LINEITEM	6,001,215	30,006,075	60,012,150
ORDERS	1,500,000	7,500,000	15,000,000
PARTSUPP	800,000	4,000,000	8,000,000
PART	200,000	1,000,000	2,000,000
SUPPLIER	10,000	50,000	100,000

We first attempt to estimate the coefficients of the batch processing time prediction function, i.e., $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_{m+1})$. Since each SF basically defines a new database, the coefficients should be estimated for each one independently. Hence, for each SF, we randomly generate 2754 data points, i.e., batches of queries. These data points are divided over 81 classes, each containing 34 batches, based on the number of queries that they involve. Specifically, the number of queries in each class of batches is taken from the set

$\{50, 75, 100, 125, \ldots, 2050\}$. To create the batches of each class, we first generate two random scenarios for showing the percentage of the data that should be retrieved as the result of processing a batch. Each scenario can take a value from the interval $(0\%, 5\%]$. For each scenario, we create 17 batches. For example, suppose that the first scenario for the class 50 is 1% and the second scenario is 5%. So, we create 17 (random) batches that each has 50 queries and requires to return 1% of the total size of the database for processing its queries. Similarly, we create 17 (random) batches that each has 50 queries and requires to return 5% of the total size of the database for processing its queries. The reason that we create 17 random batches for each scenario is that, as mentioned earlier, there are 17 possible ways of utilizing tables in (a batch of) queries for TPC-H databases. So, for each one, we randomly generate one batch of queries such that the amount of data that should be retrieved for it matches its associated scenario. For example, tables PART, PARTSUPP, and SUPPLIER construct one valid combination. So, for the class 50 and the scenario 1% that we mentioned earlier, one random batch will only utilize tables PART, PARTSUPP, and SUPPLIER. In other words, to process (all 50 queries of) that batch, 1% of the total size of the database will be returned from only tables PART, PARTSUPP, and SUPPLIER.

After creating random batches for each SF, we execute them individually and record their associated values for the dependent and independent variables of linear regression (see Section 3.4.1). We randomly pick 70% of data as the training set and use the remaining 30% as the test set. Overall, the results show that

$$\boldsymbol{\beta} = (1.6032, 0.0023, 0.1165, 1.7646, 1.0794, 2.4355, 0.9743),$$

$$\boldsymbol{\beta} = (10.8716, 0.0028, 0.1977, 3.0068, 1.9187, 4.0763, 1.6231),$$

and

$$\boldsymbol{\beta} = (17.1205, 0.0028, 0.2480, 3.7526, 2.4400, 5.1369, 2.0199)$$

for SF=1, SF=5, and SF=10, respectively. Table 3.2 shows the accuracy of the regression model on this database for different values of SF where 'MSE' refers to Mean Squared Error and 'MAPE' refers to Mean Absolute Percentage Error.

Table 3.2: The statistics of the regression model for different values of SF on TPC-H databases

SF	**Training set**			**Test set**		
	R-squared	**MSE**	**MAPE(%)**	**R-squared**	**MSE**	**MAPE(%)**
1	0.94	0.04	3.44	0.94	0.05	3.46
5	0.95	0.04	2.15	0.95	0.04	2.13
10	0.95	0.09	1.83	0.96	0.04	1.82

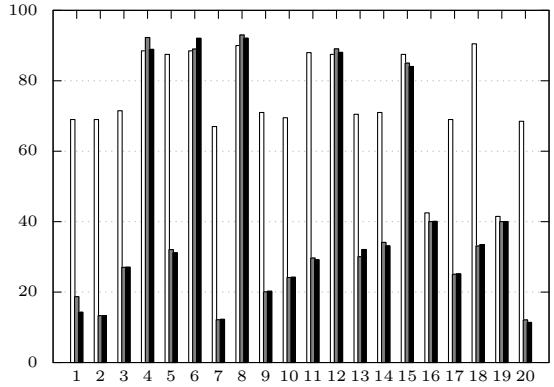
In order to show the effectiveness of solving the query batching problem, a total of 120 instances are generated for each SF. An instance includes a specific number of queries on SF 1, 5, or 10 of the TPC-H database. With SF taking three values 1, 5, and 10, a total of 360 instances are generated in this experiment. The instances associated with each SF are regrouped into three classes of small, medium, and large instances. For the small-sized instances (denoted by S), the number of queries is taken form the set $\{32, 64\}$. For the medium-sized instances (denoted by M), the number of queries is taken form the set $\{256, 512\}$. Finally, for the large-sized instances (denoted by L), the number of queries is taken form the set $\{2048, 4096\}$. Therefore, each class of instances contains two subclasses. For example, for Class S, the first subclass has instances with 32 queries each and the second subclass has instances with 64 queries each. Within each instance subclass, 20 random instances are generated. We note that since TPC-H simulates a customer-supplier system database in the real world, it comes with some suggested scenarios for generating a set of queries to make sense in practice (see the details in [89, 72]). Hence, when creating an instance, we follow the suggested scenarios. Specifically, to generate each instance, we first randomly select a subset of all 17 valid combinations of employing tables and then generate all randomly queries based on the selected combinations. Table 3.3 summarizes the performance of RCSA-I and RCSA-II on all 360 instances of this experiment. Numbers reported in this table for each instance subclass are averages over 20 instances.

Table 3.3: Performance of RCSA-I and RCSA-II on TPC-H databases

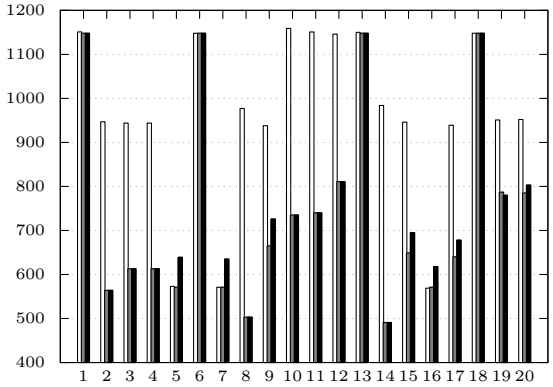
SF	n	Default	RCSA-I			RCSA-II			Imp (%)	
		Total Time (s)	# of batches	Opt Time (s)	Total Time (s)	# of batches	Opt Time (s)	Total Time (s)	RCSA-I	RCSA-II
1	32	75.0	2.4	0.6	41.4	2.4	0.3	41.3	44.8	45.0
	64	130.9	2.3	0.2	83.1	2.3	0.7	83.1	36.5	36.5
	256	306.5	2.6	0.2	182.6	2.8	1.0	182.1	40.4	40.6
	512	414.6	1.9	0.2	319.9	2.3	2.1	317.8	22.9	23.4
	2048	756.7	2.2	4.3	550.2	2.7	40.1	550.6	27.3	27.2
	4096	964.4	2.1	13.1	745.1	2.6	353.3	761.8	22.7	21.0
5	32	453.5	2.1	0.2	268.1	2.1	0.2	272.3	40.9	40.0
	64	710.2	2.1	0.1	453.7	2.1	0.2	453.7	36.1	36.1
	256	1548.1	2.4	0.4	964.3	2.4	1.1	954.5	37.7	38.3
	512	2335.7	2.0	0.4	1822.2	2.0	2.9	1822.2	22.0	22.0
	2048	3772.3	2.1	1.9	2835.8	2.4	32.3	2873.4	24.8	23.8
	4096	5176.5	2.3	9.4	3751.3	2.6	217.1	3787.3	27.5	26.8
10	32	1120.8	2.1	0.1	702.6	2.1	0.1	703.2	37.3	37.3
	64	1638.4	2.3	0.1	969.5	2.3	0.2	969.5	40.8	40.8
	256	3259.4	2.4	0.2	2057.7	2.4	1.0	2054.2	36.9	37.0
	512	4479.9	2.2	0.4	3192.3	2.2	2.8	3192.3	28.7	28.7
	2048	8165.0	2.3	2.6	5484.4	2.5	51.0	5460.7	32.8	33.1
	4096	10502.2	2.3	10.1	7505.9	2.3	186.4	7507.4	28.5	28.5

In Table 3.3, the columns labeled '# of batches' report the number of batches obtained by using a heuristic; the columns labeled 'Opt Time (s)' show the solution time of a heuristic in seconds; the columns labeled 'Total Time (s)' show the total processing time of an instance (in seconds) which include the solution time of a heuristic (if used any) and the processing time of the prescribed batches by the model; the column labeled 'Default' contains the results for the default setting of PsiDB, which is a single batch containing all queries; Finally, the columns labeled 'Imp (%)' report the percentage improvement in the total time obtained by using a heuristic to solve the QBP. As an aside, we note that PsiDB is shown to be faster than processing queries one by one by a factor of around 30 [61]. Consequently, we do not report any results for processing queries individually.

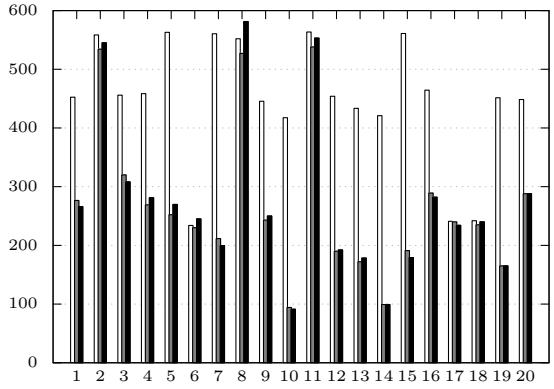
The last two columns of Table 3.3 reveal an improvement ranging from 21.0% to 45.0% in the query processing time, based on the proposed heuristic methods when compared to the default setting. Another interesting outcome from comparing the results of the two heuristics for large instances is that their total times are close. This implies that the time that RCSA-II's solution saves by a better batching is compromised by the additional time it spends in the optimization phase. For example, the last row of Table 3.3 shows that the processing time reduction achieved by potentially improved batching of RCSA-II can barely



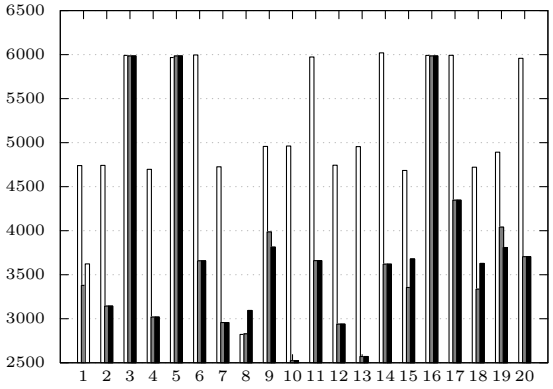
(a) SF = 1, $n = 32$



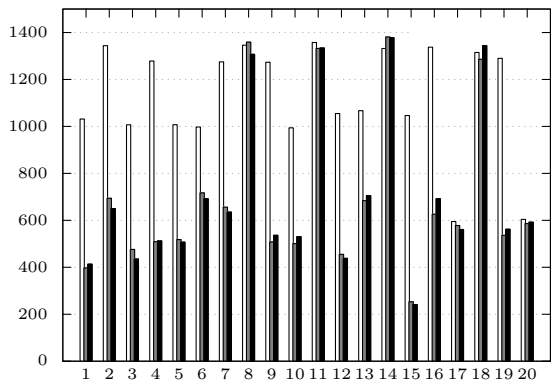
(b) SF = 1, $n = 4096$



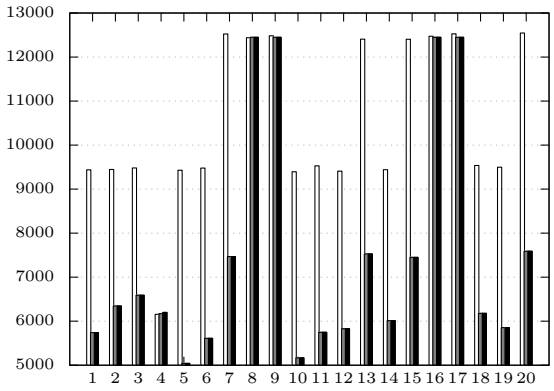
(c) SF = 5, $n = 32$



(d) SF = 5, $n = 4096$



(e) SF = 10, $n = 32$



(f) SF = 10, $n = 4096$

Figure 3.5: Total time (in seconds) of all 20 instances of the smallest and largest subclasses on TPC-H databases (white bars: default setting, gray bars: RCSA-I, black bars: RCSA-II)

compensate the 176.3 seconds of extra computational time that it needs for optimization compared to RCSA-I.

Figure 3.5 shows the total time of all instances of the subclasses $n = 31$ and $n = 4096$ for different SF values. In the Figure, white bars, gray bars, and black bars represent the total time (in seconds) of the default setting, RCSA-I, and RCSA-II, respectively. Observe that batching the queries using the proposed heuristics reduces the total time except for the cases in which final recommendation of the optimization model coincides the default setting (assigning all queries to one single batch).

Table 3.4 shows the (relative) optimality gap between the objective values reported by the heuristic methods and the global dual bounds reported by CPLEX when solving the exact MBQP formulation, i.e., MBQP(n), on instances with $n \leqslant 512$ over a two-hour time limit. In this table, the column labelled 'Exact Opt Time (s)' shows the solution time of CPLEX for solving the exact MBQP. Again, numbers reported in this table for each instance subclass are averages over 20 instances. CPLEX was able to solve all instances with $n \leqslant 256$ and most instances with $n = 512$ to optimality within the imposed time limit. Among the instances with $n = 512$, only two, three, and two of them were not solved to optimality for SF= 1, SF = 5, and SF = 10, respectively. From the table, we observe that RCSA-II was able to generate an optimal solution for all instances that their optimal solutions were known, i.e., solved within 2 hours time limit by CPLEX. Even for those few large instances that CPLEX was not able to prove optimality within 2 hours, the RCSA-II relative optimality gap from the global dual bound is below 5%. This provide evidences that RCSA-II can at least provide near optimal solutions for such instances. Note that the optimality gaps of RCSA-I are similar to the optimality gaps of RCSA-II but slightly worse for instances with $n = 512$. This explains the observation that we made earlier about the similarity of the total time of RCSA-I and RCSA-II for large instances in Table 3.3.

Table 3.4: Comparing the quality of the solutions obtained by the proposed heuristics with optimal solutions on the TPC-H

SF	n	Exact Opt Time (s)	RCSA-I gap (%)	RCSA-II gap (%)
1	32	0.4	0.00	0.00
	64	1.2	0.00	0.00
	256	324.1	0.02	0.00
	512	2463.0	2.54	2.45
5	32	0.6	0.00	0.00
	64	0.9	0.00	0.00
	256	309.3	0.00	0.00
	512	2887.8	3.37	3.37
10	32	0.5	0.00	0.00
	64	1.0	0.00	0.00
	256	392.9	0.00	0.00
	512	2143.2	2.14	2.14

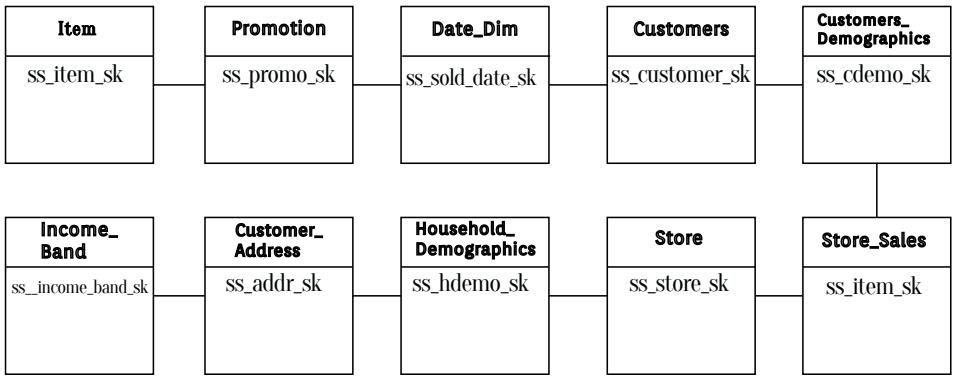


Figure 3.6: The structure of the TPC-DS database

3.5.2 Database Benchmark TPC-DS

In this section, we employ another database benchmark, i.e., TPC-DS [73, 72], that consists of 10 tables and 5GB of data for showing the effectiveness of the QBP and our proposed heuristics. The database describes a retail product supplier. The supporting schema contains vital business information, such as customer, order, and product data. All the tables in this database and their relationships are shown in Figure 3.6. Note that while there were only 17 valid combinations for involving tables in the TPC-H benchmark, the TPC-DS benchmark involves a significantly larger number of valid combinations. Specifically, the total number of potential combinations is $2^{10}-1$ but because of the relationships between the tables not all combinations are valid. In this section, we only select 55 valid combinations and employ them for generating instances. These 55 valid combinations are obtained based on the suggested scenarios that come with the TPC-DS benchmark to ensure the generated queries make sense in practice (see the details in [73]).

We again start by estimating the coefficients of the batch processing time prediction function, i.e., $\boldsymbol{\beta}$. We randomly generate 4,455 data points, i.e., batches of queries. We generate the data points for the TPC-DS benchmark following a similar procedure that we used for the TPC-H benchmark. Specifically, the data points are divided over 81 classes, each containing 55 batches, based on the number of queries that they involve. The number of queries in each class of batches is again taken from the set $\{50, 75, 100, \ldots, 2050\}$. For each class, we generate a single scenario for showing the percentage of the data that should be retrieved as the result of processing a batch. Each scenario can take a value from the interval $(0\%, 5\%]$. For each scenario, we created 55 batches. Specifically, there are 55 selected valid ways of including tables in (a batch of) queries for the TPC-DS databases. So, for each one, we randomly generate one batch of queries such that the amount of data that should be retrieved for it matches its associated scenario.

After creating random batches, we execute them individually and record their associated values for the dependent and independent variables of linear regression (see Sec-

tion 3.4.1). We randomly pick 70% of data as the training set and use the remaining 30% as the test set. Overall, the results show an R-squared of around 0.86 (see Table 3.5) can be obtained by setting,

$$\boldsymbol{\beta} = (0.326, 0.003, 0.002, 0.004, 0.001, 0.002, 0.002, 0.004, 0.002, 0.004, -0.000, 0.003).$$

Observe that β_{11} is negative and close to zero, i.e., the coefficient corresponding to the ninth table is negative. This is mainly because of the size of this table (which is small) and the structure of the database.

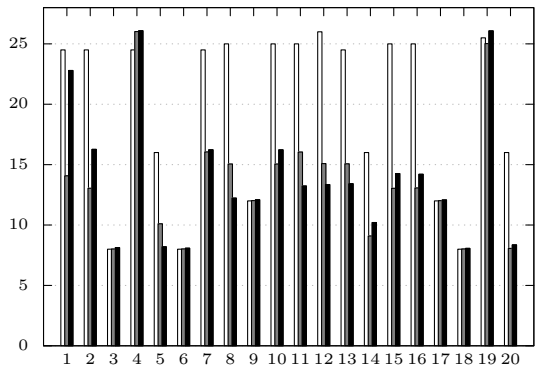
Table 3.5: The statistics of the regression model on the TPC-DS

Training set			**Test set**		
R-squared	**MSE**	**MAPE(%)**	**R-squared**	**MSE**	**MAPE(%)**
0.87	0.48	10.46	0.86	0.55	10.43

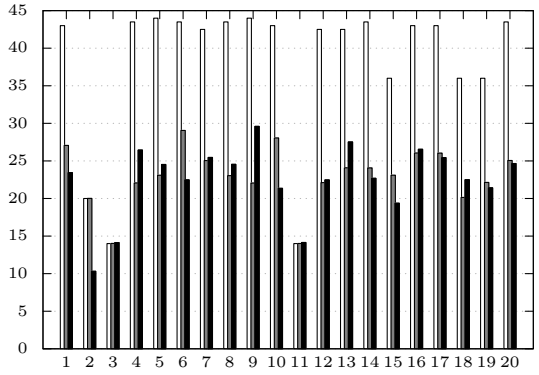
Table 3.6: Performance of RCSA-I and RCSA-II on the TPS-DS

n	**Default**	**RCSA-I**			**RCSA-II**			**Imp (%)**	
	Total Time (s)	**# of batches**	**Opt Time (s)**	**Total Time (s)**	**# of batches**	**Opt Time (s)**	**Total Time (s)**	**RCSA-I**	**RCSA-II**
32	19.8	1.7	0.1	13.6	1.7	0.5	14.0	31.2	29.2
64	38.1	1.9	0.1	23.0	1.9	0.4	22.5	39.5	41.0
256	137.4	3.0	0.3	58.8	3.4	61.8	112.3	57.2	18.3
512	399.0	3.2	0.6	181.7	4.5	188.4	306.8	54.4	23.1
2048	4524.8	3.3	4.5	2068.4	5.0	668.3	2019.8	54.3	55.4
4096	18049.2	3.8	27.3	7566.9	4.1	532.1	6889.1	58.1	61.8

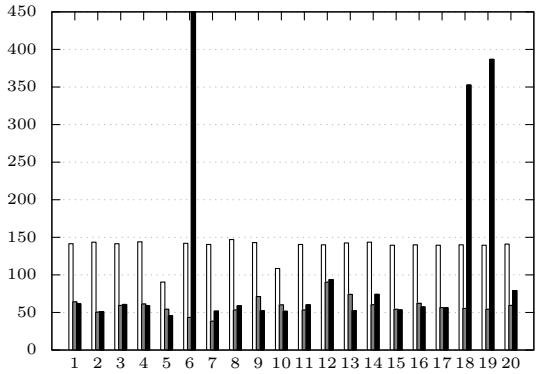
In order to show the effectiveness of solving the query batching problem on the TPC-DS database, a total of 120 instances are generated. Similar to what we did for the TPC-H database, the instances are divided over three classes of S, M, and L (based on their number of queries). Each class contains two subclasses, each with 20 instances. Table 3.6 shows the results of the proposed heuristics on the TPC-DS database. Numbers reported in this table for each instance subclass are averages over 20 instances. From columns 'Opt Time(s)', we observe that the solution time of our proposed heuristics has increased compared to the previous database, i.e., TPC-H. This can be because more batches are constructed (on average) using our heuristics on the TPC-DS database as indicated in columns '# of batches'. We also



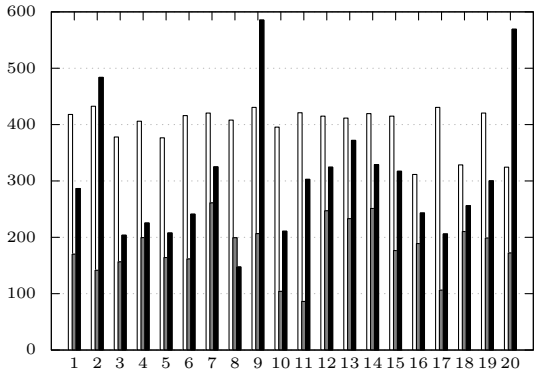
(a) $n = 32$



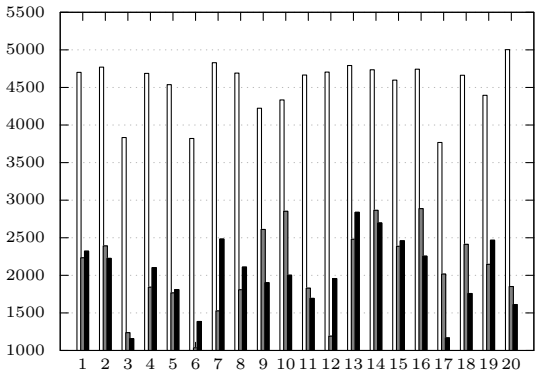
(b) $n = 64$



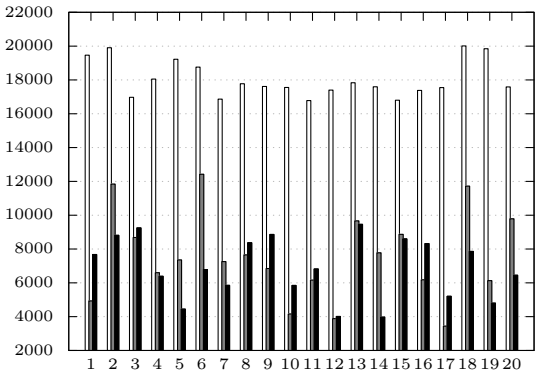
(c) $n = 256$



(d) $n = 512$



(e) $n = 2048$



(f) $n = 4096$

Figure 3.7: Total time (in seconds) of all 20 instances of different subclasses on the TPC-DS (white bars: default setting, gray bars: RCSA-I, black bars: RCSA-II)

observe from the table that our proposed algorithms can result in processing time reduction of between 18.3% to 61.8% on average. This is much larger than the improvement that we observed for the TPC-H and can be because of the structure of the TPC-DS database. Finally, we observe that RCSA-I and RCSA-II are competitive. In some subclasses of instances RCSA-I is better and in other subclasses of instances RCSA-II is better. The main reason that RCSA-I is better than RCSA-II in some cases is that the second phase of RCSA-II has been very time consuming, as shown in columns 'Opt Time(s)'. Figure 3.7 highlights this observation further because it shows the total time of all instances of all subclasses under default setting, RCSA-I, and RCSA-II. From the figure, we observe that RCSA-II has performed poorly for the Class M, i.e., instances with $n = 256$ and $n = 512$, which is again because the second phase of RCSA-II has been time consuming. Note that the imposed time limit for each iteration of RCSA-I/RCSA-II is 300 seconds in our study. Hence, one can make the results of RCSA-II close to RCAS-I by significantly decreasing the time limit imposed on each iteration of the second phase of RCSA-II.

Table 3.7: Comparing the quality of the solutions obtained by the proposed heuristics with optimal solutions on the TPC-DS

n	**Exact Opt Time (s)**	**RCSA-I gap (%)**	**RCSA-II gap (%)**
32	9.8	0.24	0
64	1144.3	4.53	3.84

Table 3.7 shows the (relative) optimality gap between the objective values reported by the heuristic methods and the global dual bounds reported by CPLEX when solving the exact MBQP formulation, i.e., MBQP(n), on instances with $n \leqslant 64$ over a two-hour time limit. Note that we did not report any result for larger instances, e.g., instances with $n = 256$ or $n = 512$, because CPLEX was unable to handle them as their optimality gap was close to 100% in 2 hours (and even hours beyond that). This observation is not surprising because as we mentioned earlier, even the second phase of RCSA-II was very time consuming for instances with $n = 256$ or $n = 512$. We note that in Table 3.7, 39 out of 40 instances were optimally solved within two hours using CPLEX and the optimality gap of only one

instance was 46.53% after two hours. With this mind, we observe that both RCSA-I and RCSA-II have performed well on generating optimal solutions. Specifically, the optimality gap of RCSA-II is zero for all 39 instances (that we know an optimal solution) and only for one instance is non-zero.

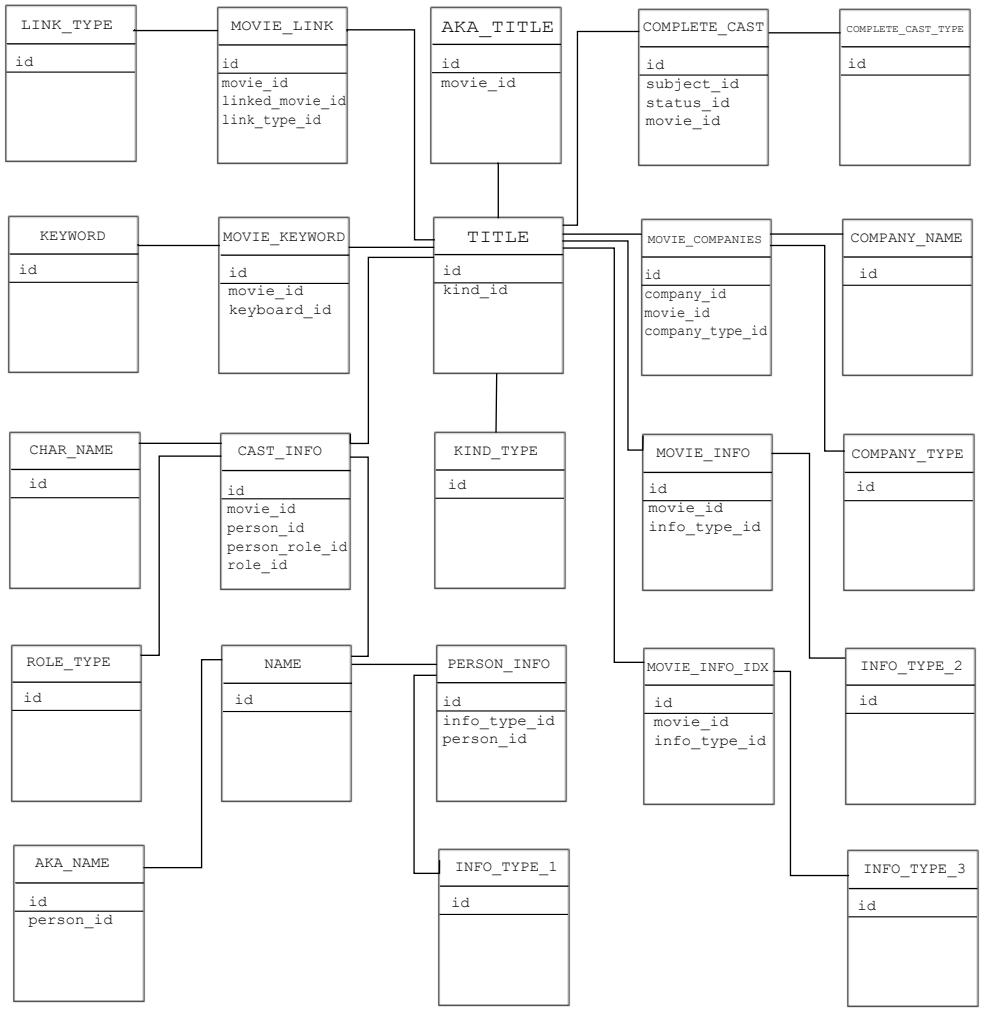


Figure 3.8: The structure of the JOB database

3.5.3 Database Benchmark JOB

The last benchmark we use to test the effectiveness of the QBP and our proposed heuristics is the Join-Order Benchmark (JOB) [74]. JOB is a fixed-sized database with almost 4GB of data which is created based on the well-known Internet Movie Database (IMDB).

It contains detailed information about movies and related facts about actors, directors, production companies, etc. This database contains 23 tables, and relations among them. All the tables in this database and their relationships are shown in Figure 3.8. Again note that while there were only 17 valid combinations for involving tables in the TPC-H benchmark, the JOB benchmark involves a significantly larger number of valid combinations. Specifically, the total number of potential combinations is $2^{23}-1$ but because of the relationships between the tables not all combinations are valid. In this section, we only select 165 valid combinations and employ them for generating instances. These 165 valid combinations are obtained based on the suggested scenarios that come with the JOB benchmark to ensure the generated queries make sense in practice (see the details in [74]).

Table 3.8: Performance of RCSA-I and RCSA-II on the JOB

n	Default	RCSA-I			RCSA-II			Imp (%)	
	Total Time (s)	# of batches	Opt Time (s)	Total Time (s)	# of batches	Opt Time (s)	Total Time (s)	RCSA-I	RCSA-II
32	145.1	2.4	0.2	115.3	1.9	0.3	111.2	20.6	23.3
64	325.4	2.2	0.1	263.9	1.9	0.3	257.4	18.9	20.9
256	1031.0	2.4	0.3	836.5	2.2	1.0	805.7	18.9	21.8
512	1757.6	1.9	0.4	1408.3	1.7	2.8	1377.0	19.9	21.7
2048	3694.3	1.8	1.6	3094.9	1.8	74.5	3083.2	16.2	16.5
4096	4502.8	2.3	14.6	3394.0	2.1	293.8	3765.2	24.6	16.4

Again, we first start by estimating the coefficients of the batch processing time prediction function, i.e., $\boldsymbol{\beta}$. We randomly generate 13,365 data points, i.e., batches of queries. We generate the data points for the JOB benchmark following a similar procedure that we used for the TPC-H benchmark. Specifically, the data points are divided over 81 classes, each containing 165 batches, based on the number of queries that they involve. The number of queries in each class of batches is again taken from the set $\{50, 75, 100, \ldots, 2050\}$. For each class, we generate a single scenario for showing the percentage of the data that should be retrieved as the result of processing a batch. Each scenario can take a value from the interval $(0\%, 5\%]$. For each scenario, we created 165 batches. Specifically, there are 165 selected valid ways of including tables in (a batch of) queries for the JOB databases. So,

for each one, we randomly generate one batch of queries such that the amount of data that should be retrieved for it matches its associated scenario.

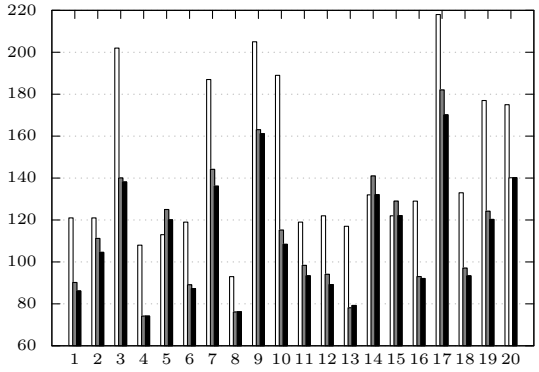
After creating random batches, we execute them individually and record their associated values for the dependent and independent variables of linear regression (see Section 3.4.1). We randomly pick 70% of data as the training set and use the remaining 30% as the test set. Overall, the results show an R-squared of around 0.98 (see Table 3.9) can be obtained by setting,

$$\begin{aligned}\boldsymbol{\beta} = (&2.3028, 0.0004, 0.0278, 0.3033, 0.2566, 0.2659, 0.2682, 0.2899, 0.2718, 0.2552,\\ &0.2516, 0.2580, 0.2620, 0.2466, 0.2462, 0.2602, 0.2625, 0.2661, 0.2879, 0.3121,\\ &0.3717, 0.3307, 0.3476, 0.2571, 0.2639).\end{aligned}$$

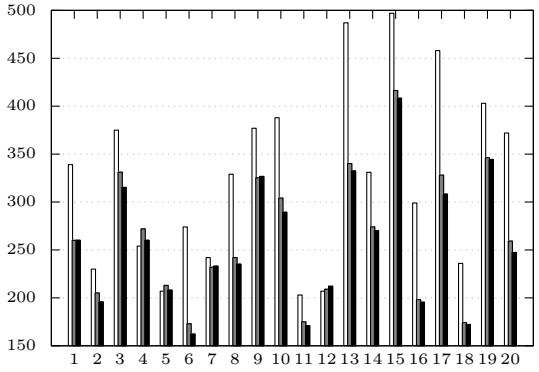
Table 3.9: The statistics of the regression model on the JOB

Training set			**Test set**		
R-squared	**MSE**	**MAPE(%)**	**R-squared**	**MSE**	**MAPE(%)**
0.98	0.04	2.18	0.98	0.04	2.17

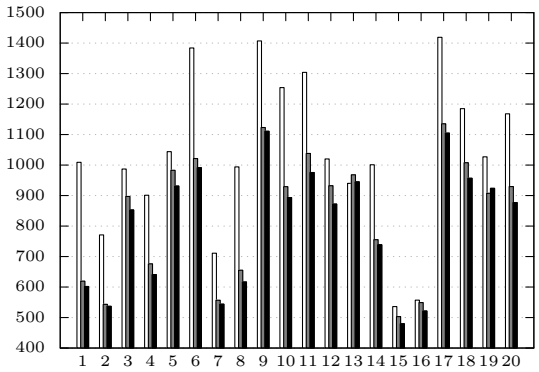
In order to show the effectiveness of solving the query batching problem on the JOB database, a total of 120 instances are generated. Similar to what we did for TPC-H database, the instances are divided over three classes of S, M, and L (based on their number of queries). Each class contains two subclasses, each with 20 instances. Table 3.8 shows the results of the proposed heuristics on the JOB database. Numbers reported in this table for each instance subclass are averages over 20 instances. The last two columns demonstrate that the proposed heuristics reduce the total time up to 24.6%. Observe that since the JOB database contains more tables compared to the TPC-H and TPC-DS databases, the difference between the performances of RCSA-I and RCSA-II is more noticeable. Although, RCSA-II performs better in small instances (in terms of the total time), RCSA-I outperforms



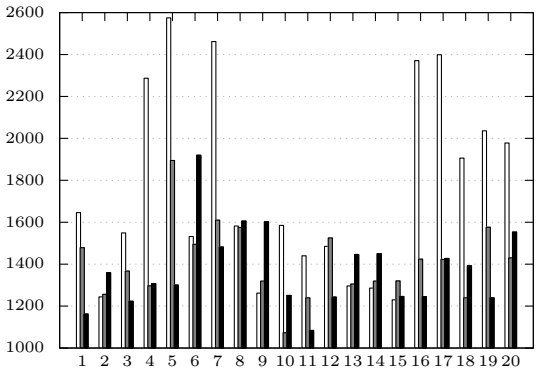
(a) $n = 32$



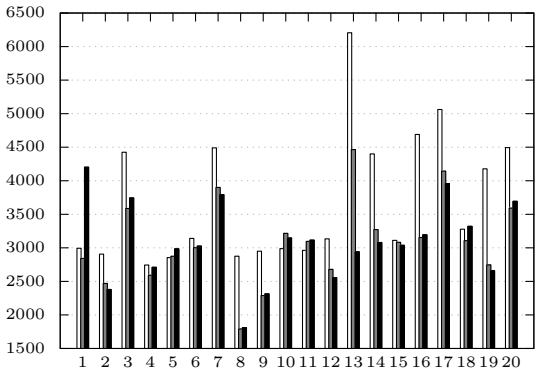
(b) $n = 64$



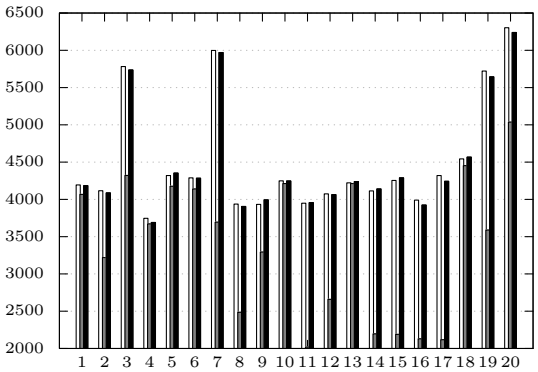
(c) $n = 256$



(d) $n = 512$



(e) $n = 2048$



(f) $n = 4096$

Figure 3.9: Total time (in seconds) of all 20 instances of different subclasses on the JOB (white bars: default setting, gray bars: RCSA-I, black bars: RCSA-II)

RCSA-II by around 8.2% for the largest subclass of instances. This can be mainly because of the significant increase in the optimization time of RCSA-II. Figure 3.9 highlights this observation further because (similar to Figure 3.5) it shows the total time of all instances of all subclasses under default setting, RCSA-I, and RCSA-II. Another interesting point from Table 3.8 is that, unlike the TPC-H databases, the number of batches for RCSA-II is smaller than the number of batches for RCSA-I. That is, RCSA-II sometimes assigns queries requiring the same table(s) to different batches, which can potentially result in a smaller number of batches.

Table 3.10 shows the (relative) optimality gap between the objective values reported by the heuristic methods and the global dual bounds reported by CPLEX when solving the exact MBQP formulation on instances with $n \leqslant 256$ over a two-hour time limit. CPLEX was able to solve all instances with $n = 32$ and most instances with $n = 64$ and $n = 256$ to optimality within the imposed time limit. Among the instances with $n = 64$, only 2 instances and among the instances with $n = 256$, 8 instances were not solved to optimality. From the table, we observe that both RCSA-I and RCSA-II performed similarly. They were both able to generate an optimal solution for all instances that their optimal solutions were known, i.e., solved within 2 hours time limit by CPLEX. Even for those few large instances that CPLEX was not able to prove the optimality within 2 hours, both heuristics relative optimality gap from the global dual bound is below 11%, while their required computational time is a small fraction of the time required by CPLEX.

Table 3.10: Comparing the quality of the solutions obtained by the proposed heuristics with optimal solutions on the JOB

n	**Exact Opt Time (s)**	**RCSA-I gap (%)**	**RCSA-II gap (%)**
32	5.9	0.00	0.00
64	726.9	0.93	0.93
256	3938.2	10.10	10.10

3.6 Final Remarks

In this paper, for the first time (to the best of our knowledge), we studied the query batching problem. This problem aims at partitioning a given set of queries into some batches before retrieving them from a database system in order to minimize the total retrieving/processing time. This optimization problem is challenging because predicting the time required for processing a given batch of queries is not a trivial task. We developed a generic three-phase approach for solving the problem for any given database system. In the first phase, using a quadratic function, our approach attempts to predict the processing time of any batch of queries for the given database. In the second phase, our approach uses the obtained quadratic function and develops a mixed binary quadratic programming formulation for the query batching problem accordingly. Finally, in the last phase, our proposed approach uses two custom-built heuristic approaches, i.e., RCSA-I and RCSA-II, to quickly solve the obtained formulation in practice. We tested our proposed approach on three well-known database benchmarks by conducing a comprehensive computational study. The results showed that a reduction of up to 61.8% is achievable for the total processing times in the database benchmarks when employing our proposed approach.

We hope that the simplicity, versatility, and performance of our proposed approach encourage practitioners/researchers to consider employing/developing effective query batching optimizers. There are several future research directions that can be considered for this study. One direction can be developing better exact or heuristic solution approaches for solving instances with larger number of queries. Alternatively, developing effective machine learning methods for predicting the outcome, i.e., an optimal solution, of the proposed mixed binary quadratic program can be an interesting research direction too. Another research direction can be developing some theories and/or methodologies for identifying an optimal amount of time that one should wait to accumulate queries before starting to solve the query batching problem.

Chapter 4: Multi-objective Optimization Based Algorithms for Solving Mixed Integer Linear Minimum Multiplicative Programs

In this chapter [2], we present paper P3. In this work, we develop two algorithms to solve mixed-integer linear minimum multiplicative programs. The copyright permissions for reusing previously published material in this chapter can be found in Appendix C2.

4.1 Introduction

Multi-objective optimization provides decision-makers with a complete view of the trade-offs between the objective function values that are attainable by feasible solutions. It is thus a critical tool in engineering, where competing goals must often be considered and balanced when making decisions. Combined with the fact that many problems can be formulated as mixed integer linear programs, the development of fast and reliable multi-objective mixed integer programming solvers has significant benefits for problem solving in industry and government. Consequently, it is not surprising that in the last decade, many researchers have focused on developing effective algorithms for solving multi-objective mixed integer linear programs, such as the ones conducted by Ehrgott and Gandibleux [90], Özlen and Azizoğlu [91], Özpeynirci and Köksalan [92], Dächert et al. [93], Lokman and Köksalan [94], Eusébio et al. [95], Kirlik and Sayın [96], Soylu and Yıldız [97], Boland et al. [98], and Przybylski and Gandibleux [99].

While promising, a non-trivial (but interesting) question that has received less attention in the literature is whether multi-objective optimization methods can be useful for

[2]This chapter was published at C&OR. Mahmoodian, V., Charkhgard, H., & Zhang, Y. (2021). Multi-objective optimization based algorithms for solving mixed integer linear minimum multiplicative programs. *Computers & Operations Research*. Permission is included in Appendix C.2

solving single-objective optimization problems? In recent years, a few studies have explored this question and have shown the superiority of multi-objective optimization based techniques (compared to the existing methods) on solving two classes of single-objective optimization problems including minimum Multiplicative Programs (mMPs; see for instance [100, 1]) and Maximum Multiplicative Programs (MMPs; see for instance [101] and [102]). A mMP is the problem of the form

$$\min \left\{ \prod_{i=1}^{p} y_i(\boldsymbol{x}) : \ \boldsymbol{x} \in \mathcal{X}, \ \boldsymbol{y}(\boldsymbol{x}) \geqslant \mathbf{0} \right\}, \tag{4.1}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$ represents the set of feasible solutions and it is assumed to be bounded. Also, $\boldsymbol{y}(\boldsymbol{x}) := \big(y_1(\boldsymbol{x}), \ldots, y_p(\boldsymbol{x})\big)$ is a vector of affine/linear functions. As an aside, throughout this article, vectors are always column-vectors and are denoted in bold fonts. It is assumed that the optimal objective value of a mMP is strictly positive. We note that if $\mathcal{X}$ is defined by a set of linear constraints, the problem is referred to as Linear mMP (L-mMP). If in a L-mMP, all decision variables are integers, the problem is referred to as Integer Linear mMP (IL-mMP). Finally, if in a L-mMP, some but not all decision variables are integers, the problem is referred to as Mixed Integer Linear mMP (MIL-mMP).

It is worth mentioning that if in a mMP, one changes 'min' to 'max', a MMP will be constructed. Although MMPs and mMPs look almost the same, they are very different in terms of their computational complexity. Specifically, L-MMPs (meaning linear MMPs) are polynomially solvable but L-mMPs are NP-hard [101, 1]. This negative result may have contributed to the fact that (to the best of our knowledge), all (multi-objective based) methodological studies on mMPs have focused only on solving mMPs involving no integer decision variables. For example, Konno and Kuno [103] showed how L-mMPs when $p = 2$ can be solved by combining the parametric simplex method and any standard convex minimization procedure. Benson and Boger [104] proposed a heuristic method for solving L-mMPs. In another study, Kuno [105] proposed a branch-and-bound algorithm for solving L-mMPs. Kim et al. [106] presented an outcome-space outer approximation algorithm for L-mMPs. Recently, Shao and Ehrgott [1] proposed two objective-space algorithms for solving L-mMPs.

It is worth mentioning that there are also some other studies such as those conducted by Van Thoai [107], Kuno et al. [108], Benson [109], Gao et al. [110], Oliveira and Ferreira [111], Gao et al. [110], and Shao and Ehrgott [100] on developing algorithms for mMPs involving no integer decision variables (but not necessarily L-mMPs) that interested readers may refer to.

In light of the above, there is a clear gap in the literature about developing effective multi-objective optimization based techniques for solving mMPs involving integer decision variables. Consequently, the goal of this study is to develop new exact multi-objective optimization-based algorithms that can solve any MIL-mMP by just solving a number of single-objective (mixed integer) linear programs. In other words, we attempt to develop new algorithms that can exploit the power of commercial single-objective mixed integer linear programming solvers such as IBM ILOG CPLEX for solving a MIL-mMP (which is a non-linear optimization problem).

We note that developing more effective techniques for MIL-mMPs can have significant impacts on problem solving in practice as MIL-mMPs (and in general mMPs) have several applications in different fields of study including (but not limited to) bond portfolio management, economic analysis, and VLSI chip design [112] as well as conservation planning or natural resource management [113]. For example, in conservation planning, a typical goal is to preserve biodiversity. For doing so, solution $\boldsymbol{x} \in \mathcal{X}$ typically represents a subset of parcels (or sites) that should be selected for protection within a geographical region. Also, $y_i(\boldsymbol{x})$ typically represents the probability of species $i \in \{1, \ldots, p\}$ being extinct if solution $\boldsymbol{x}$ is implemented. In other words, the objective function of a mMP in conservation planning represents the probability of all species being extinct and the goal is to minimize it (to preserve biodiversity).

The proposed exact algorithms in this study are developed based on a critical observation that an optimal solution of Problem (4.1) is an *efficient* or *Pareto-optimal* solution (i.e., a solution in which it is impossible to improve the value of one objective without making

the value of at least one other objective worse) of the following multi-objective optimization problem,

$$\min \Big\{ y_1(\boldsymbol{x}), \ldots, y_p(\boldsymbol{x}) : \ \boldsymbol{x} \in \mathcal{X}, \ \boldsymbol{y}(\boldsymbol{x}) \geqslant \mathbf{0} \Big\}. \tag{4.2}$$

Specifically, by minimizing the function $\prod_{i=1}^{p} y_i(\boldsymbol{x})$ over the set of efficient solutions of Problem (4.2), an optimal solution of Problem (4.1) can be obtained. We prove this observation in Section 5.2 but similar proofs can be found in Benson and Boger [104], Charkhgard et al. [101], Saghand et al. [102] and Shao and Ehrgott [100, 1]. Overall, this critical observation indicates that Problem (4.1) can be viewed and solved as a special case of the problem of *optimization over the efficient set* in multi-objective optimization. It is worth noting that, in optimization over the efficient set, the goal is to compute an optimal solution directly, i.e., without enumerating all efficient solutions if possible [114, 115, 116, 117, 118].

The main contribution of our research is that we employ the above observation and develop two new algorithms for solving MIL-mMPs. The first proposed algorithm is a decision space search algorithm, i.e., an algorithm that works in the space of decision variables of a multi-objective optimization problem. The algorithm performs similar to the classical branch-and-bound algorithm for solving single-objective mixed integer linear programs. Specifically, the algorithm solves a number of L-mMPs in order to solve a MIL-mMP. To solve each L-mMP, existing multi-objective optimization algorithms can be employed and they solve only a number of linear programs to compute an optimal solution. In other words, our first algorithm solves only (single-objective) linear programs for solving each MIL-mMP and we use the power of commercial linear programming solvers to solve each one. For improving the performance of our first algorithm, we also propose some enhancement techniques and show their effectiveness in a computational study.

Our second proposed algorithm is a criterion space search algorithm, i.e., an algorithm that works in the space of objective values of a multi-objective optimization problem. The proposed algorithm solves a number of (single-objective) mixed integer linear programs (rather than linear programs) to solve each MIL-mMP. So, the main advantage of the second

algorithm is that it uses the power of commercial mixed integer linear programming solvers. To show the effectiveness of the proposed algorithms, we conduct a comprehensive computational study on 960 instances and compare the performance of our proposed algorithms with a generic-purpose solver, SCIP. Note that SCIP is one of very few solvers freely available to academics capable of solving MIL-mMPs (at the time of conducting this research), and the solver has been widely used in research articles for solving L-mMPs, e.g., [1]. Our results show that our proposed algorithms can outperform SCIP by a factor of more than 10 on many instances in terms of solution time. By a detailed comparison, we show that for many instances with $p = 2$, our proposed criterion space search algorithm outperforms the proposed decision space search algorithm by a factor of more than 8 in terms of solution time. However, as p increases, our criterion space algorithm starts to struggle. Hence, we show that for instances with $p = 4$, the proposed decision space search algorithm significantly outperforms our proposed criterion space search algorithm in terms of optimality gap obtained within the imposed time limit. We also numerically show that as the number of integer decision variables increases, the proposed decision space search algorithm starts to struggle. Hence, for instances with $p = 3$, if the number of integer decision variables is large, the proposed criterion space search is the best choice. Otherwise, the proposed decision space search algorithm is the best approach. Furthermore, we show that linearizing the objective function of IL-mMPs, i.e., when no continuous variable exists, is possible but it will result in large single-objective integer linear programs. So, in the computational study, we show that a commercial solver, CPLEX, struggles to solve such linearized instances. Hence, using the proposed algorithms is a better choice.

The rest of the paper is organized as follows. In Section 5.2, some preliminaries about MIL-mMPs are introduced. In Section 4.3, the details of the proposed decision space search algorithm are provided. In Section 4.4, the proposed criterion space search algorithm is explained. In Section 5.5, a comprehensive computational study is presented. Finally, in Section 4.6, some concluding remarks are given.

4.2 Preliminaries

A Mixed Integer Linear minimum Multiplicative Program (MIL-mMP) can be stated as follows,

$$\begin{aligned} \min \ & \prod_{i=1}^{p} y_i \\ \text{s.t. } & \boldsymbol{y} = C\boldsymbol{x} + \boldsymbol{d} \\ & A\boldsymbol{x} \geqslant \boldsymbol{b} \\ & \boldsymbol{x}, \boldsymbol{y} \geqslant \boldsymbol{0}, \quad \boldsymbol{x} \in \mathbb{R}^{n_c} \times \mathbb{Z}^{n_i}, \quad \boldsymbol{y} \in \mathbb{R}^p, \end{aligned} \tag{4.3}$$

where n_c and n_i denote the number of continuous and integer decision variables, respectively. Also, C is a $p \times n$ matrix where $n := n_c + n_i$ and p shows the number of variables in the objective function. Finally, $\boldsymbol{d}$ is a vector of length p, A is an $m \times n$ matrix, and $\boldsymbol{b}$ is a vector of length m.

We refer to the set $\mathcal{X} := \{\boldsymbol{x} \in \mathbb{R}^{n_c} \times \mathbb{Z}^{n_i} : A\boldsymbol{x} \geqslant \boldsymbol{b},\ \boldsymbol{x} \geqslant \boldsymbol{0}\}$ as *the feasible set in the decision space* and to the set $\mathcal{Y} := \{\boldsymbol{y} \in \mathbb{R}^p : \boldsymbol{x} \in \mathcal{X},\ \boldsymbol{y} = C\boldsymbol{x} + \boldsymbol{d},\ \boldsymbol{y} \geqslant \boldsymbol{0}\}$ as *the feasible set in the criterion space*. We assume that $\mathcal{X}$ is bounded (which implies that $\mathcal{Y}$ is compact) and the optimal objective value of the problem is strictly positive, i.e., $\boldsymbol{0} \notin \mathcal{Y}$. Throughout this article, we refer to $\boldsymbol{x} \in \mathcal{X}$ as a *feasible solution* and to $\boldsymbol{y} \in \mathcal{Y}$ as a *feasible point* ($\boldsymbol{y}$ is the image of $\boldsymbol{x}$ in the criterion space).

Definition 4.1. A feasible solution $\boldsymbol{x} \in \mathcal{X}$ is called *efficient*, if there is no other $\boldsymbol{x}' \in \mathcal{X}$ such that $\boldsymbol{y}' \leqslant \boldsymbol{y}$ and $\boldsymbol{y}' \neq \boldsymbol{y}$ where $\boldsymbol{y} := C\boldsymbol{x} + \boldsymbol{d}$ and $\boldsymbol{y}' := C\boldsymbol{x}' + \boldsymbol{d}$. If $\boldsymbol{x}$ is efficient, then $\boldsymbol{y}$ is called a *nondominated point*. The set of all efficient solutions is denoted by $\mathcal{X}_E$. The set of all nondominated points is denoted by $\mathcal{Y}_N$ and referred to as the *nondominated frontier*.

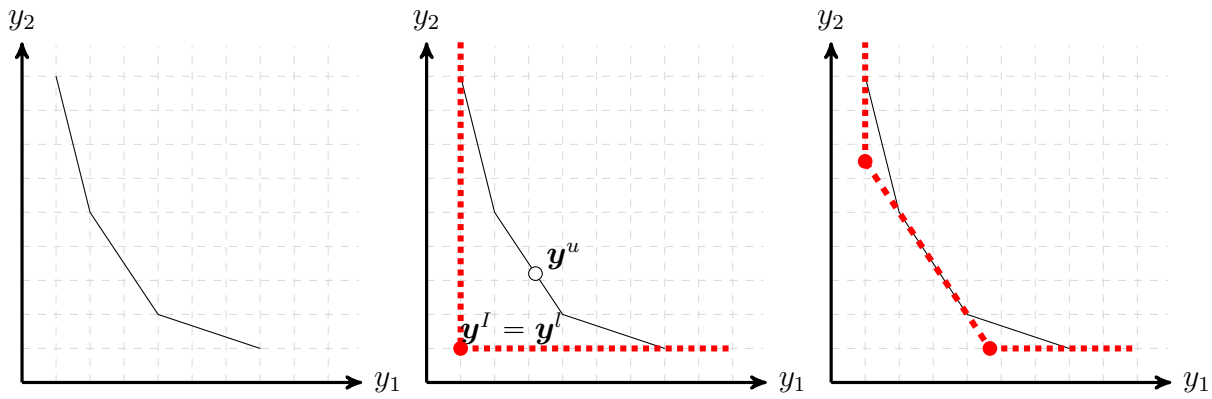
Proposition 4.1. *An optimal solution of Problem* (4.3)*, denoted by* $\boldsymbol{x}^*$*, is an efficient solution and therefore its corresponding image in the criterion space, denoted by* $\boldsymbol{y}^*$ *where* $\boldsymbol{y}^* := C\boldsymbol{x}^* + \boldsymbol{d}$*, is a nondominated point.*

Proof. Let $\boldsymbol{x}^*$ be an optimal solution of Problem (4.3) but not an efficient solution. By definition, this implies that there must exist a feasible solution $\boldsymbol{x} \in \mathcal{X}$ such that it dominates $\boldsymbol{x}^*$. In other words, we must have that $\boldsymbol{y} \leqslant \boldsymbol{y}^*$ and $\boldsymbol{y} \neq \boldsymbol{y}^*$ where $\boldsymbol{y} := C\boldsymbol{x} + \boldsymbol{d}$ and $\boldsymbol{y}^* = C\boldsymbol{x}^* + \boldsymbol{d}$. Also, by assumptions of Problem (4.3), we know that $\boldsymbol{y}, \boldsymbol{y}^* > \boldsymbol{0}$. Therefore, the inequalities $0 < \prod_{i=1}^{p} y_i < \prod_{i=1}^{p} y_i^*$ must hold. However, this immediately implies that $\boldsymbol{x}^*$ is not an optimal solution (a contradiction). □

Proposition 5.1 implies that Problem (4.3) is equivalent to $\min_{\boldsymbol{y} \in \mathcal{Y}_N} \prod_{i=1}^{p} y_i$ and this is precisely optimization over the efficient set. This observation indicates that developing multi-objective optimization techniques for solving MIL-mMPs is possible and this idea will be explored in this paper.

4.3 A Decision Space Search Algorithm

In this section, we propose a new decision space search algorithm for solving MIL-mMPs. We often refer to the proposed algorithm in this section as DSSA throughout this paper. As mentioned in the Introduction, in the field of multi-objective optimization, decision space search algorithms refer to solution approaches that work in the space of decision variables. To the best of our knowledge, developing decision space search algorithms has not yet been explored for MIL-mMPs. However, it is recently studied (see [102]) in the literature of MIL-MMPs for instances with $p = 2$. Note that as mentioned in the Introduction, MIL-MMPs are quite different from MIL-mMPs. Overall, DSSA is similar to the classical branch-and-bound algorithm for solving single-objective mixed integer linear programs with the main difference being the process of computing dual bounds in each node of the search tree. Specifically, the underlying idea of DSSA is to effectively employ a technique that can solve any L-mMP within a classical branch-and-bound framework to compute dual bounds. Note that the combination of the classical branch-and-bound algorithm with the method that we use for computing dual bounds (as a whole) is one of the contributions of our study, but none of them individually is our contribution. Our initial idea for developing DSSA



(a) The nondominated frontier of a L-mMP

(b) Critical points in the first iteration

(c) The relaxed set at the end of the first iteration

Figure 4.1: An illustration of the workings of the method proposed by Shao and Ehrgott [1] on a L-mMP with $p = 2$

was to use it as a benchmark for comparison purposes in the computational study. However, surprisingly as it will be discussed in the computational study, such a simple algorithm works well for some classes of instances after including some enhancement techniques. In the rest of this section, we first review one of the fastest algorithms for solving L-mMPs. After that, the proposed branch-and-bound framework is explained. Finally, a few enhancement techniques are introduced.

4.3.1 A Method for Solving L-mMPs

Although our proposed algorithm can employ any method for solving L-mMPs, in this section we briefly review the one that we found to perform the best within DSSA during the course of our research. The method is developed by Shao and Ehrgott [1] and is one of the fastest and (relatively) easy-to-implement algorithms for solving L-mMPs. This method acts as the default solution approach for solving L-mMPs at the heart of our proposed branch-and-bound framework.

The method of Shao and Ehrgott [1] maintains a global lower bound and a global upper bound and it terminates whenever these two values converge. The underlying idea of

the method is to use Proposition 5.1 for solving a L-mMP. Hence, the method attempts to find an optimal point of the L-mMP by searching over its nondominated frontier. It is worth mentioning that the nondominated frontier of a feasible L-mMP is known to be a convex curve and consists of only some plane/line segments. An illustration of the nondominated frontier of a L-mMP when $p = 2$ can be found in Figure 4.1a. Overall, the method of Shao and Ehrgott [1] is iterative but in order to start, it requires to be initialized by the so-called *ideal* point of the L-mMP, i.e., the imaginary point in the criterion space that has the minimum possible value for each objective. Note that to compute the ideal point, denoted by $\boldsymbol{y}^I$, p linear programs should be solved since $y_i^I := \min_{\boldsymbol{y} \in \mathcal{Y}^R} y_i$ for $i = 1, \ldots, p$ where $\mathcal{Y}^R := \{\boldsymbol{y} \in \mathbb{R}^p : \boldsymbol{x} \in \mathcal{X}^R,\ \boldsymbol{y} = C\boldsymbol{x} + \boldsymbol{d},\ \boldsymbol{y} \geqslant \boldsymbol{0}\}$ and $\mathcal{X}^R := \{\boldsymbol{x} \in \mathbb{R}^n : A\boldsymbol{x} \geqslant \boldsymbol{b},\ \boldsymbol{x} \geqslant \boldsymbol{0}\}$. An illustration of the ideal point, denoted by $\boldsymbol{y}^I$, when $p = 2$ can be found in Figure 4.1b. Observe that the set of nondominated points of the L-mMP is a subset of $S := \{\boldsymbol{y} \in \mathbb{R}^p : \boldsymbol{y} \geqslant \boldsymbol{y}^I\}$. We call S as the relaxed set and it plays a key role in the method of [1].

In each iteration, the method computes a global dual/lower bound for the optimal objective value of the L-mMP using the relaxed set. Specifically, it can be shown that an extreme point of the relaxed set, denoted by $\boldsymbol{y}^l$, that has the minimum value for the multiplicative function, i.e., the objective function of the L-mMP, is a global dual bound. For example, in the first iteration, the only extreme point in the relaxed set is the ideal point, i.e., $\boldsymbol{y}^l = \boldsymbol{y}^I$, and $\prod_{i=1}^p y_i^I$ is obviously a global lower/dual bound (see Figure 4.1b). As an aside, to compute the vertices of the relaxed set, the algorithm proposed by Chen et al. [119] can be employed. In each iteration, the method also uses $\boldsymbol{y}^l$ to compute a new feasible point, denoted by $\boldsymbol{y}^u$ with $\boldsymbol{y}^l \leqslant \boldsymbol{y}^u$. In order to do so, the following linear program needs to be solved that attempts to find a feasible point with the minimum Chebyshev distance from $\boldsymbol{y}^l$,

$$\boldsymbol{y}^u \in \arg\min_{\boldsymbol{y} \in \mathcal{Y}^R} \big\{ z \in \mathbb{R} : z \geqslant 0 \text{ and } y_i - y_i^l \leqslant z\ \forall i \in \{1, \ldots, p\} \text{ and } y_i^l \leqslant y_i\ \forall i \in \{1, \ldots, p\} \big\}, \tag{4.4}$$

where z is a continuous decision variable that captures the Chebyshev distance. An illustration of $\boldsymbol{y}^u$ can be found in Figure 4.1b. Note that whenever a new nondominated point is found, it can provide an upper bound for the optimal objective value of the L-mMP because it is a feasible point. So, the global upper/primal bound needs to be updated after computing a new nondominated point in each iteration. Moreover, it can be shown (see [1]) that the inequality $\boldsymbol{\lambda}^\intercal \boldsymbol{y} \geqslant \boldsymbol{\lambda}^\intercal \boldsymbol{y}^u$ will not remove any nondominated point of the L-mMP where $\boldsymbol{\lambda}$ is a vector of length p. For each $i \in \{1, \ldots, p\}$, λ_i captures the dual value associated with constraint $y_i - y_i^l \leqslant z$ in Problem (4.4). So, the inequality $\boldsymbol{\lambda}^\intercal \boldsymbol{y} \geqslant \boldsymbol{\lambda}^\intercal \boldsymbol{y}^u$ will be added at the end of each iteration for the purpose of restricting/updating the relaxed set. An illustration of the relaxed set after adding the cut can be found in Figure 4.1c.

4.3.2 The Proposed Branch-and-bound Framework

The key idea of DSSA is similar to the classical branch-and-bound framework for (single-objective) integer linear programming. The search in DSSA starts by relaxing the integrality condition on binary/integer decision variables of the input problem, which is a MIL-mMP. By doing so, a L-mMP will be generated that if being solved, it provides a dual/lower bound for the MIL-mMP. Note that to solve any L-mMP during the course of DSSA, we employ the method of Shao and Ehrgott [1]. If the solution obtained by solving the L-mMP turns out to be naturally integer, i.e., the integrality condition on all binary/integer decision variables hold, then the obtained solution has to be optimal for the MIM-mMP. Otherwise, the algorithm will pick one of the binary/integer variables that has taken a fractional value and branch on it exactly similar to the classical branch-and-bound algorithm. For example, suppose that the integer decision variable x_j has taken the fractional value of v is chosen by the algorithm for branching. In that case two child nodes will be created by branching: one with additional constraint $x_j \leqslant \lfloor v \rfloor$ and the other with additional constraint $x_j \geqslant \lceil v \rceil$. We note that there are multiple strategies for selecting a variable for branching. During the course of this research, we implemented four well-

known variable selection strategies (see [102] for details) including *random branching*, *most infeasible branching*, *pseudo-cost branching*, and *reliability branching*. The most promising variable selection strategy for our proposed framework is pseudo-cost branching and hence it is used as the default setting of our algorithm in the rest of this paper.

The algorithm then starts to explore the unexplored nodes in the search tree. Similar to the classical branch-and-bound algorithm, DSSA maintains a global upper/primal bound, denoted by G_{UB}, and a global dual/lower bound, denoted by G_{LB}. The algorithm updates the global primal bound as soon as it finds a better feasible solution than the one it already has. Also, at any point in time, the global dual bound represents the best/minimum dual bound among all unexplored nodes. The algorithm explores the nodes as long as $G_{UB} - G_{LB} \geqslant \varepsilon_1$ and $\frac{G_{UB}-G_{LB}}{G_{UB}} \geqslant \varepsilon_2$, where $\varepsilon_1, \varepsilon_2 \in (0, 1)$ are the user-defined absolute and relative optimality gap tolerances, respectively. Specifically, in each iteration, one node will be picked by the algorithm and its corresponding L-mMP will be solved for computing a dual bound for that particular node. If the node becomes infeasible and/or its associated dual bound is worse than the global primal bound, it will not be explored further. We note that there are multiple strategies for selecting a node for exploring. During the course of this research, we implemented five well-known node selection strategies (see [102] for details) including *depth-first search*, *best-bound search*, *two-phase method*, *best-expected bound*, and *best estimate*. The most promising node selection strategy for our proposed framework is the best estimate and hence it is used as the default setting of our algorithm in the rest of this paper.

4.3.3 Enhancements

In this section, we introduce two enhancement techniques that can be used to possibly improve the performance of DSSA.

4.3.3.1 Enhancement-I (Preprocessing)

According to Proposition 5.1, optimal points of a MIL-mMP must be nondominated points. Therefore, any nondominated point can potentially result in computing a good primal/upper bound for a MIL-mMP. So, one trivial enhancement technique is to compute a few nondominated points before starting DSSA and then feeding the best primal bound obtained from them as the initial $\mathrm{G_{UB}}$ to DSSA. In order to generate nondominated points, we use the so-called weighted sum operation. This operation attempts to compute a nondominated point $\bar{\boldsymbol{y}}$ by solving the following optimization problem,

$$\bar{\boldsymbol{y}} \in \arg\min \{\sum_{i=1}^{p} \lambda_i y_i : y \in \mathcal{Y}\},$$

where $\lambda_1, \ldots, \lambda_p > 0$ are user-defined weights. We denote the weighted sum operation by $\mathrm{WSO}(\boldsymbol{\lambda})$. As an aside, it is known that the weighted sum operation always returns a nondominated point but not all nondominated points can be necessarily found using the weighted sum operation (even by considering all possible values of $\lambda_1, \ldots, \lambda_p$) for non-convex optimization problems [120, 121].

In the literature of multi-objective optimization, there exist several exact algorithms for effectively choosing values for the vector $\boldsymbol{\lambda}$ (see for instance [92]). However, such approaches can be very time consuming for MIL-mMPs and we do not want to spend the valuable computational time on generating the nondominated points. So, instead, we propose a (preprocessing) heuristic approach which is motivated from the study of [122] for computing some nondominated points. Specifically, we impose a time limit for generating nondominated points in our heuristic approach. It is worth mentioning that during the course of this research, we found that setting the time limit to $0.01n$ seconds, where n is the number of decision variables, results in the best performance for our algorithm (in our test instances). So, the default value for the time limit of our heuristic approach is $0.01n$ in the remainder of this paper.

The proposed heuristic is an iterative approach that terminates whenever the time limit is reached. In each iteration, the components of the weight vector $\boldsymbol{\lambda}$ will be generated randomly from the standard normal distribution, denoted by RandNormal$(0, 1)$. However, we use the absolute value of the generated random numbers. Also, for convenience, we normalize the components of each weight vector to assure that the summation of its components will be equal to one. After generating $\boldsymbol{\lambda}$ in each iteration, the proposed heuristic calls WSO($\boldsymbol{\lambda}$) to compute a nondominated point $\bar{\boldsymbol{y}}$. Afterwards, the proposed heuristic updates the global upper bound value, i.e., $\mathrm{G_{UB}}$, if $\prod_{i=1}^{p} \bar{y}_i$ is better than its current value.

Algorithm 2: A preprocessing approach

```
Input: A feasible instance of Problem (4.3)
G_UB ← +∞
while Time ⩽ TimeLimit do
    foreach i ∈ {1, ..., p} do
        λ_i ← |RandNormal(0, 1)|
    foreach i ∈ {1, ..., p} do
        λ_i ← λ_i / (Σ_{j=1}^{p} λ_j)
    ȳ ← WSO(λ)
    if Π_{i=1}^{p} ȳ_i < G_UB then
        G_UB ← Π_{i=1}^{p} ȳ_i
    Y ← Y ∩ {y ∈ R^p : Σ_{i=1}^{p} λ_i y_i ⩾ λ_i ȳ_i}
return Y, G_UB
```

It is worth mentioning that an automatic advantage of solving the weighted sum operation (in each iteration) is that a valid inequality/cut can be added to the set of constraints defining $\mathcal{Y}$ which can potentially remove many fractional solutions and improving dual bounds. Specifically, after computing $\bar{\boldsymbol{y}}$ by calling WSO($\boldsymbol{\lambda}$), the following cut is valid (due to the optimality of $\bar{\boldsymbol{y}}$ for the weighted sum operation),

$$\sum_{i=1}^{p} \lambda_i y_i \geqslant \sum_{i=1}^{p} \lambda_i \bar{y}_i.$$

A detailed description of our proposed preprocessing heuristic for computing a good initial global primal bound and adding cuts using the weighted sum operation can be found in Algorithm 2.

4.3.3.2 Enhancement-II (Exchanging Bounds)

In our proposed decision space search algorithm, one can view DSSA as a solution approach for solving a master problem, i.e., a MIL-mMP, and the method proposed by [1] as a solution approach for solving its corresponding subproblems, i.e., L-mMPs. Both DSSA and the method of [1] have internal procedures for computing (primal and also dual) bounds when solving their corresponding problems. So, the underlying idea of our second enhancement technique is to effectively employ/exchange the information about these bounds between the master problem and the subproblems. Specifically, during the course of solving a subproblem, it is possible that the method of [1] finds some solutions that are feasible for not only the L-mMP (that it is trying to solve) but also its corresponding MIL-mMP, i.e., the solutions are luckily not fractional. So, such solutions can be immediately used for updating the global primal/upper bound, i.e., G_{UB}, of DSSA (if they are better). Moreover, we know that at any time during the course of solving a subproblem, the method of [1] maintains a global dual bound for the L-mMP. So, one can immediately terminate solving the L-mMP, if its dual bound is not strictly better/smaller than the global upper bound that DSSA has already obtained, i.e., G_{UB}.

4.4 A Criterion Space Search Algorithm

In the previous section, we proposed a decision space search algorithm that solves a number of L-mMPs in order to solve a MIL-mMP. To solve each L-mMP, DSSA solves a number of single-objective linear programs. So, in some sense, DSSA relies on the power of commercial linear programming solvers for solving a MIL-mMP. In this section, we propose a completely different approach. Specifically, we introduce a new criterion space search

algorithm for solving MIL-mMPs. We often refer to the proposed algorithm in this section as CSSA throughout this paper. As mentioned in the Introduction, in the field of multi-objective optimization, criterion space search algorithms refer to solution approaches that work in the space of objective functions. The underlying idea of CSSA is to solve a MIL-mMP by solving a number of single-objective mixed integer programs. So, in some sense, CSSA relies on the power of commercial mixed integer linear programming solvers for solving a MIL-mMP.

In the remainder of this section, we first provide a high-level description of CSSA, then we give its detailed description, and finally we explain some implementation issues. However, before doing so, we introduce a key operation that is frequently used in CSSA and also a critical proposition that guarantees the correctness of the algorithm. The operation is used for solving a min-max optimization problem and takes a reference point, denoted by $\boldsymbol{y}^l \in \mathbb{R}^p$, as input. The reference point basically defines a *search region*, i.e., $\{\boldsymbol{y} \in \mathbb{R}^p : \boldsymbol{y} \geqslant \boldsymbol{y}^l\}$, in the criterion space for the min-max operation. For a given reference point $\boldsymbol{y}^l \in \mathbb{R}^p$, the operation attempts to compute a feasible point, denoted by $\boldsymbol{y}^u$, where $\boldsymbol{y}^l \leqslant \boldsymbol{y}^u$ that has the minimum Chebyshev distance from the reference point by solving the following single-objective mixed integer linear program,

$$\boldsymbol{y}^u \in \arg\min_{\boldsymbol{y}\in\mathcal{Y}} \big\{ z \in \mathbb{R} : z \geqslant 0 \text{ and } y_i - y_i^l \leqslant z \ \forall i \in \{1,\dots,p\} \text{ and } y_i^l \leqslant y_i \ \forall i \in \{1,\dots,p\} \big\}. \tag{4.5}$$

We denote this operation by MIN-MAX($\boldsymbol{y}^l$). Note that if $\boldsymbol{y}^u$ =null then Problem (4.5) is infeasible. Also, in this paper, we sometimes refer to the point $\boldsymbol{y}^l$ as the lower bound point and to the point $\boldsymbol{y}^u$ as the upper bound point. Next we provide a proposition that builds the foundations of our algorithm. The importance of this proposition will be explained in Section 4.4.3.

Informally, the proposition claims that because of using the min-max operation, we can always create a *hypercube* (in the p-dimensional criterion space) containing $\boldsymbol{y}^l$ and $\boldsymbol{y}^u$

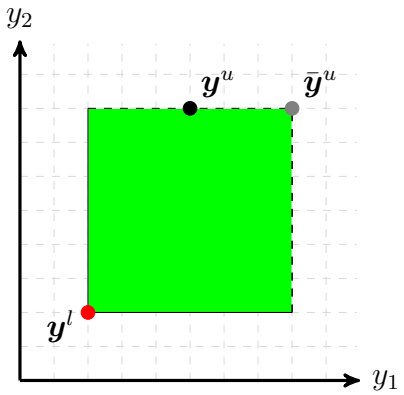
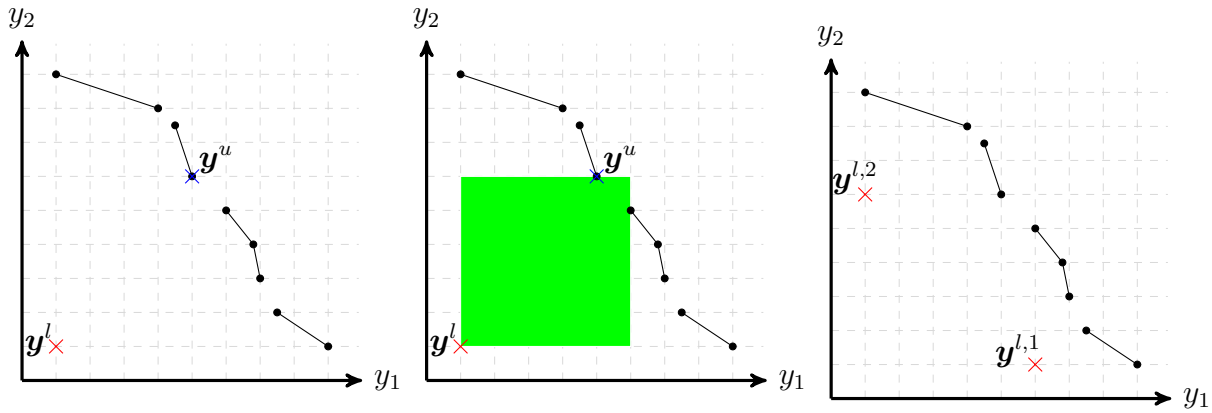


Figure 4.2: An illustration of Proposition 4.2 when $p = 2$

with no feasible point in its interior. For example, in Figure 4.2, an illustration of the points $\boldsymbol{y}^l$ and $\boldsymbol{y}^u$ when $p = 2$ can be found. Observe that in the figure, $y_2^u - y_2^l > y_1^u - y_1^l$. So, let $\bar{y}_1^u := y_1^l + y_2^u - y_2^l$ and $\bar{y}_2^u := y_2^l + y_2^u - y_2^l$. It is evident that the area between $\boldsymbol{y}^l$ and $\bar{\boldsymbol{y}}^u$ is a square, i.e., a hypercube of dimension 2. The proposition claims that there cannot exist any feasible point $\boldsymbol{y}^f$ with $\boldsymbol{y}^f \geqslant \boldsymbol{y}^l$, i.e., $y_i^f \geqslant y_i^l$ for $i = 1, \ldots, p$, and $\boldsymbol{y}^f < \bar{\boldsymbol{y}}^u$, i.e., $y_i^f < \bar{y}_i^u$ for $i = 1, \ldots, p$, because of solving the min-max operation to optimality. A formal description of the proposition and its proof are given next.

Proposition 4.2. *Let $\boldsymbol{y}^u$ be an optimal point obtained by* MIN-MAX($\boldsymbol{y}^l$) *and $z^* := \max\{y_1^u - y_1^l, \ldots, y_p^u - y_p^l\}$ be the Chebyshev distance of $\boldsymbol{y}^u$ from $\boldsymbol{y}^l$. Also, let $\bar{\boldsymbol{y}}^u \in \mathbb{R}^p$ be a point in the criterion space with $\bar{y}_i^u := y_i^l + z^*$ for $i = 1, \ldots, p$. There cannot exist any feasible point $\boldsymbol{y}^f \in \mathcal{Y}$ such that $\boldsymbol{y}^f \geqslant \boldsymbol{y}^l$ and $\boldsymbol{y}^f < \bar{\boldsymbol{y}}^u$.*

Proof. Suppose not. So, there exists a feasible point $\boldsymbol{y}^f \in \mathcal{Y}$ with $\boldsymbol{y}^f \geqslant \boldsymbol{y}^l$ and $\boldsymbol{y}^f < \bar{\boldsymbol{y}}^u$. We know that z^* is the Chebyshev distance of $\boldsymbol{y}^u$ from $\boldsymbol{y}^l$ and by construction it should be minimum because of calling MIN-MAX($\boldsymbol{y}^l$). However, since $y_i^f < \bar{y}_i^u = y_i^l + z^*$ for $i = 1, \ldots, p$, we have that $\max\{y_1^f - y_1^l, \ldots, y_1^f - y_1^l\} < z^*$. This immediately implies that $\boldsymbol{y}^u$ cannot be an optimal point of the min-max operation since $\boldsymbol{y}^f$ is feasible and has a strictly smaller Chebyshev distance from $\boldsymbol{y}^l$ (a contradiction). □



(a) Result of the min-max opeartion in the first iteration

(b) Result of Proposition 4.2 in the first iteration

(c) Branching and creating new lower bound points

Figure 4.3: An illustration of the key steps of the proposed criterion space search algorithm when $p = 2$

4.4.1 A High-level Description

CSSA starts by computing the initial lower bound point. The initial lower bound point is basically the (imaginary) ideal point of the MIL-mMP that we are trying to solve. So, computing the initial lower bound point, denoted by $\boldsymbol{y}^l$, can be done by solving p mixed integer linear programs. Specifically, $y_i^l := \min_{\boldsymbol{y} \in \mathcal{Y}} y_i$ for $i = 1, \ldots, p$. Observe that $\prod_{i=1}^{p} y_i^l$ is a global dual/lower bound for the optimal objective value of the MIL-mMP. Based on the initial lower bound point, we can now apply the min-max operation to compute an upper bound point $\boldsymbol{y}^u$. Observe that $\prod_{i=1}^{p} y_i^u$ is a global primal/upper bound for the optimal objective value of the MIL-mMP. An illustration of the nondominated frontier of a MIL-mMP as well as $\boldsymbol{y}^l$ and $\boldsymbol{y}^u$ when $p = 2$ can be found in Figure 4.3a. Note that the nondominated frontier of a MIL-mMP can be very complicated since it may not be a convex and/or continuous curve as opposed to the nondominated frontier of a L-mMP. Interested readers may refer to [123] to learn more about the nondominated frontier of multi-objective mixed integer linear programs.

Next we can apply Proposition 4.2 to create an empty-interior hypercube. An illustration of such a hypercube when $p = 2$ is shown in Figure 4.3b. Afterwards, we can remove the discovered empty region from the search by branching and creating p new lower bound points, denoted by $\boldsymbol{y}^{l,1}, \ldots, \boldsymbol{y}^{l,p}$. In order to do so, we first set $z^* = \max\{y_1^u - y_1^l, \ldots, y_p^u - y_p^l\}$. Now, for each $j \in \{1, \ldots, p\}$, we can define $\boldsymbol{y}^{l,j}$ by setting $y_j^{l,j} = y_j^l + z^*$ and $y_i^{l,j} = y_i^l$ for all $i \in \{1, \ldots, p\} \backslash \{j\}$. An illustration of the new lower bound points when $p = 2$ can be found in Figure 4.3b.

Observe that each new lower bound point defines a new (but smaller) search region for which the product of the components of its lower bound point is a dual bound. In other words, for each $j \in \{1, \ldots, p\}$, the search region corresponding to $\boldsymbol{y}^{l,j}$ is $\{\boldsymbol{y} \in \mathbb{R}^p : \boldsymbol{y} \geqslant \boldsymbol{y}^{l,j}\}$ and we know that $\prod_{i=1}^p y_i^{l,j}$ is a dual bound for that search region. So, in each iteration, one can easily update the global dual bound of the MIL-mMP. Specifically, in each iteration, we can first compute the dual bound of each existing (meaning not yet explored) search region and then set the global lower bound to the minimum one. Similarly, the global primal bound can be updated in each iteration. Specifically, in each iteration, whenever the algorithm finds an upper bound point, we can first compute the product of its components and then set the global primal bound to it if the current value of global primal bound is (strictly) larger. It is evident that the algorithm can terminate as soon as the global primal bound and global dual bound converge.

4.4.2 A Detailed Description

CSSA maintains a queue of lower bound points, denoted by TREE. The algorithm also maintains a global upper bound, denoted by $\mathrm{G_{UB}}$, and a global lower bound, denoted by $\mathrm{G_{LB}}$. At the beginning, CSSA sets $\mathrm{G_{UB}} = +\infty$ and $\mathrm{G_{LB}} = -\infty$. The algorithm also computes the ideal point, i.e., $(\min_{\boldsymbol{y} \in \mathcal{Y}} y_1, \ldots, \min_{\boldsymbol{y} \in \mathcal{Y}} y_p)$, and then initializes $\boldsymbol{y}^l$ by setting it equal to the ideal point. After that, the algorithm updates $\mathrm{G_{LB}}$ by setting it equal to $\prod_{i=1}^p y_i^l$ and initializes the queue by $\boldsymbol{y}^l$. The algorithm then explores the queue as long as it is

nonempty and $\mathrm{G_{UB}} - \mathrm{G_{LB}} \geqslant \varepsilon_1$ and $\frac{\mathrm{G_{UB}} - \mathrm{G_{LB}}}{\mathrm{G_{UB}}} \geqslant \varepsilon_2$, where $\varepsilon_1, \varepsilon_2 \in (0,1)$ are the user-defined absolute and relative optimality gap tolerances, respectively. Next, we explain how each element of the queue is explored.

In each iteration, the algorithm pops out an element of the queue and denotes it by $\boldsymbol{y}^l$. During the course of this research, the best bound strategy, i.e., selecting the node with the minimum dual bound, was found to perform the best for popping out an element. So, in the remainder of this paper, the best bound strategy is set as the default setting of our proposed algorithm. After popping out an element, the algorithm then calls MIN-MAX($\boldsymbol{y}^l$) to compute an upper bound point $\boldsymbol{y}^u$.

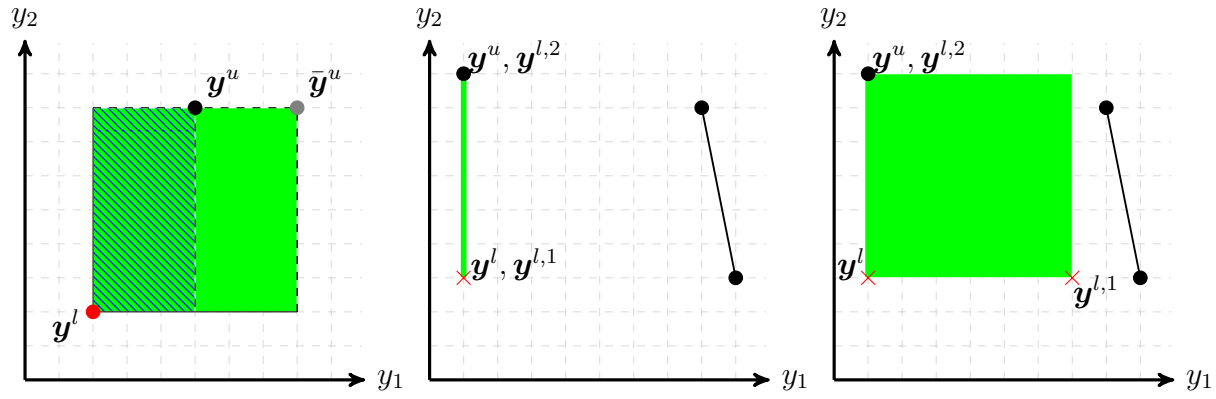
If $\boldsymbol{y}^u$=null then the search region corresponding to $\boldsymbol{y}^l$ is infeasible. So, the algorithm simply updates G_{LB} (as discussed in Section 4.4.1) and starts a new iteration. Otherwise, because $\boldsymbol{y}^u$ is a feasible point, the algorithm checks whether it would be possible to update G_{UB}. Specifically, if $\prod_{i=1}^{p} y_i^u < \mathrm{G_{UB}}$ then the algorithm sets $\mathrm{G_{UB}}$ to $\prod_{i=1}^{p} y_i^u$ and also the best feasible point found, i.e., $\boldsymbol{y}^*$, to $\boldsymbol{y}^u$. Next, the algorithm computes the Chebyshev distance z^* of $\boldsymbol{y}^u$ from $\boldsymbol{y}^l$, i.e., it sets z^* to $\max\{y_1^u - y_1^l, \ldots, y_p^u - y_p^l\}$. Afterwards, the algorithm attempts to create at most p new nodes. To create node $j \in \{1, \ldots, p\}$, the algorithm first generates $\boldsymbol{y}^{l,j}$ based on $\boldsymbol{y}^l$ (as discussed in Section 4.4.1). The algorithm then adds $\boldsymbol{y}^{l,j}$ to the queue if there is hope to find better feasible points there, i.e., $\mathrm{G_{UB}} - \prod_{i=1}^{p} y_i^{l,j} \geqslant \varepsilon_1$ and $\frac{\mathrm{G_{UB}} - \prod_{i=1}^{p} y_i^{l,j}}{\mathrm{G_{UB}}} \geqslant \varepsilon_2$. Finally, the algorithm updates G_{LB} before starting a new iteration.

For further details about our proposed criterion space search algorithm, readers can refer to Algorithm 3. We complete this subsection by making one final comment. In multi-objective optimization, one generic issue about criterion space search algorithms (especially for large values of p) is that some nodes in the priority queue could be redundant (see for instance [98, 124, 125]). Hence, in our algorithm, whenever we want to add a node to the priority queue with the lower bound point $\overline{\boldsymbol{y}}^l$, we check whether there is already a node in the tree with the lower bound point $\underline{\boldsymbol{y}}^l$ such that $\underline{y}_i^l \leqslant \overline{y}_i^l$ for all $i = 1, \ldots, p$. If $\underline{\boldsymbol{y}}^l$ exists then

Algorithm 3: The Proposed Criterion Space Search Algorithm

Input: A feasible instance of Problem (4.3)
Queue.create(TREE)
$\mathrm{G_{LB}} \leftarrow -\infty$; $\mathrm{G_{UB}} \leftarrow +\infty$
$\boldsymbol{y}^l \leftarrow (\min_{\boldsymbol{y}\in\mathcal{Y}} y_1, \ldots, \min_{\boldsymbol{y}\in\mathcal{Y}} y_p)$
$\mathrm{G_{LB}} \leftarrow \prod y_i^l$
TREE.*add*$(\boldsymbol{y}^l)$
while *not Queue.empty*(TREE) *&* $\mathrm{G_{UB}} - \mathrm{G_{LB}} \geqslant \varepsilon_1$ *&* $\frac{\mathrm{G_{UB}}-\mathrm{G_{LB}}}{\mathrm{G_{UB}}} \geqslant \varepsilon_2$ **do**
    TREE.*PopOut*$(\boldsymbol{y}^l)$
    $\boldsymbol{y}^u \leftarrow$ MIN-MAX$(\boldsymbol{y}^l)$
    if $\boldsymbol{y}^u \neq null$ **then**
        if $\prod_{i=1}^p y_i^u < \mathrm{G_{UB}}$ **then**
            $\mathrm{G_{UB}} \leftarrow \prod_{i=1}^p y_i^u$
            $\boldsymbol{y}^* \leftarrow \boldsymbol{y}^u$
        $z^* \leftarrow \max\{y_1^u - y_1^l, \ldots, y_p^u - y_p^l\}$
        foreach $j \in \{1, \ldots, p\}$ **do**
            $\boldsymbol{y}^{l,j} \leftarrow \boldsymbol{y}^l$; $y_j^{l,j} \leftarrow y_j^l + z^*$
            if $\mathrm{G_{UB}} - \prod_{i=1}^p y_i^{l,j} \geqslant \varepsilon_1$ *&* $\frac{\mathrm{G_{UB}}-\prod_{i=1}^p y_i^{l,j}}{\mathrm{G_{UB}}} \geqslant \varepsilon_2$ **then**
                TREE.*add*$(\boldsymbol{y}^{l,j})$
    Update $\mathrm{G_{LB}}$
return $\boldsymbol{y}^*$

$\overline{\boldsymbol{y}}^l$ is redundant and should not be added to the priority queue. This is because in that case the search region corresponding to $\overline{\boldsymbol{y}}^l$ is a subset of the search region corresponding to $\underline{\boldsymbol{y}}^l$.



(a) An illustration of a box vs a hypercube

(b) Branching based on the area of a box

(c) Branching based on the area of a hypercube

Figure 4.4: An illustration of the importance of Proposition 4.2 when $p = 2$

4.4.3 Implementation Issues

The main purpose of this section is to explain the importance of Proposition 4.2 for the correctness of our algorithm. Before doing so, it is worth mentioning that Enhancement-I which is developed for DSSA (see Section 4.3.3) can be directly used for CSSA. However, during the course of this research, we found out that the performance of CSSA does not improve by employing Enhancement-I in practice because (unlike DSSA) the proposed criterion space search algorithm naturally finds feasible solutions/points for a MIL-mMP in each iteration. Also, in theory, one should be able to employ Enhancement-II for possibly improving the performance of CSSA. However, since each min-max operation in CSSA will be solved using a commercial solver (and not an open-source solver) in this paper, it would be technically impossible (to the best of our knowledge) to implement Enhancement-II in practice.

We now explain the importance of Proposition 4.2. Observe that the branching procedure in CSSA relies on Proposition 4.2. However, in the literature of criterion space

search algorithms for multi-objective optimization problems, this type of branching is not typical. This is because in the literature of multi-objective optimization, there are multiple ways for generating a (nondominated) point and employing a Chebyshev-based operation (similar to the proposed min-max operation) is just one of them [126]. By using a min-max operation, it is always possible to identify an empty-interior hypercube in the criterion space according to Proposition 4.2. However, this observation may not be true for other operations. Specifically, instead of identifying an empty-interior hypercube, other operations can possibly identify empty-interior boxes. However, branching based on boxes rather than hypercubes can be problematic and hence requires further considerations/research.

To see how a box can be created rather than a hypercube, consider Figure 4.4a in which $p = 2$. Suppose that by calling MIN-MAX($\boldsymbol{y}^l$), we were able to compute $\boldsymbol{y}^u$. By Proposition 4.2, the area defined by $\boldsymbol{y}^l$ and $\bar{\boldsymbol{y}}^u$ is an empty-interior hypercube. However, one can claim that the box between $\boldsymbol{y}^l$ and $\boldsymbol{y}^u$ is also empty-interior. This implies that if we apply a different operation (than the proposed min-max operation) and find $\boldsymbol{y}^u$ then we may not be able to claim that $\boldsymbol{y}^l$ and $\bar{\boldsymbol{y}}^u$ is an empty-interior hypercube but at least we can claim that the area between $\boldsymbol{y}^l$ and $\boldsymbol{y}^u$ is an empty-interior box. This is because even if we do not use Proposition 4.2, we know that there cannot exist any feasible point that strictly dominates $\boldsymbol{y}^u$. So, the question is now why not branching based on the obtained box rather than a hypercube? Of course one can argue that the area that the hypercube will remove is larger. So, it is probably better to use a hypercube. However, we now illustrate a more significant problem that can be created as a result of branching based on a box using Figures 4.4b and 4.4c.

Suppose that the nondominated frontier of a MIL-mMP with $p = 2$ consists of a single point and a line segment as shown in Figures 4.4b. Observe that the result of MIN-MAX($\boldsymbol{y}^l$) is the top endpoint of the nondominated frontier. In this case, the box between MIN-MAX($\boldsymbol{y}^l$) and $\boldsymbol{y}^u$ is basically a line as shown in Figure 4.4b. So, if we apply branching based on this line, the new lower bound points will be $\boldsymbol{y}^{l,1} = \boldsymbol{y}^l$ and $\boldsymbol{y}^{l,2} = \boldsymbol{y}^u$. Observe that it is not

necessary to add $\boldsymbol{y}^{l,2}$ because $\boldsymbol{y}^{l,2}$ is a feasible point (and Line 17 of Algorithm 3 will take this observation into account). However, $\boldsymbol{y}^{l,1}$ should be added to the queue and this implies that the algorithm falls into an infinite loop since $\boldsymbol{y}^{l,1}$ is exactly $\boldsymbol{y}^{l}$ (and we have already explored that). However, this issue will never happen if we apply branching based on hypercubes as shown in Figure 4.4b because $\boldsymbol{y}^{l,1} \neq \boldsymbol{y}^{l}$.

4.5 A Computational Study

In this section, we conduct an extensive computational study and compare the performance of DSSA and CSSA with a generic-purpose solver which is capable of solving MILmMPs, i.e., SCIP 6.0.0. We use Julia 0.6.4 [3] in this section and employ CPLEX 12.8 for solving (single-objective) linear programs and mixed integer linear programs arising during the course of DSSA and CSSA. In addition, all the computational experiments are conducted on a Dell PowerEdge R640 system with two Intel Xeon (Silver 4116) 2.1GHz 12-core processors, 128GB of memory, and RedHat Enterprise Linux 7.4 operating system, and using only a single thread. A time limit of 3600 seconds is imposed for solving each instance and this includes the time of employing any enhancement technique. Our instance generator and codes of this study can be found at https://github.com/vahidmhn/MMP_Instance_Generator and https://github.com/vahidmhn/MIMMP.jl, respectively.

In our computational study, a total of 960 instances are generated and solved. Specifically, for each $p \in \{2, 3, 4\}$, we generate 320 instances. For each $p \in \{2, 3, 4\}$, its corresponding instances are divided into two equal-sized classes denoted by Class-I and Class-II. For the first class, we set $n_c = n_i = n/2$. However, for the second class, we set $n_c = 0$ and $n_i = n$. For convenience, we assume that all integer decision variables are binary. Note that because by our assumptions $\mathcal{X}$ is bounded, in theory, any instance with generic integer decision variables can be transformed to an instance with only binary decision variables with the cost of introducing an additional set of binary variables and constraints. Furthermore, by assuming

[3] Since the start time of this study, several newer versions of Julia are introduced. However, users can still use our implementation if they employ the configurations described in the links.

that all integer decision variables are binary, Class-II contains only pure binary instances that can be easily linearized and solved directly by CPLEX. So, this assumption enables us to directly compare the performance of our methods with CPLEX for linearized instances. For example, consider a MIL-mMP with an objective function of the following form,

$$\min\ (2x_1 + 3x_2)(x_2 + 4x_3) = 2x_1x_2 + 8x_1x_3 + 3x_2^2 + 12x_2x_3,$$

where $x_1,\ x_2$, and x_3 are binary variables. It is evident that $x_i^2 = x_i$ and also any bi-linear term x_ix_j where both x_i and x_j are binary variables can be linearized by adding the following three constraints and introducing a new binary variable q,

$$x_ix_j := \Big\{q \in \{0,1\} :\ q \leqslant x_i,\ q \leqslant x_j,\ x_i + x_j - 1 \leqslant q\Big\}.$$

Since the example is in the form of minimization and its coefficients are non-negative, the first two constraints, i.e. $q \leqslant x_i$ and $q \leqslant x_j$, are redundant and can be eliminated. In light of this observation, the linearized objective function is as follows,

$$\begin{aligned}
&\min\ 2q_1 + 8q_2 + 3x_2 + 12q_3 \\
&\text{s.t. } x_1 + x_2 - 1 \leqslant q_1, && (x_1x_2) \\
&\qquad x_1 + x_3 - 1 \leqslant q_2, && (x_1x_3) \\
&\qquad x_2 + x_3 - 1 \leqslant q_3, && (x_2x_3) \\
&\qquad q_i \in \{1,0\} && \forall i \in \{1,2,3\}.
\end{aligned}$$

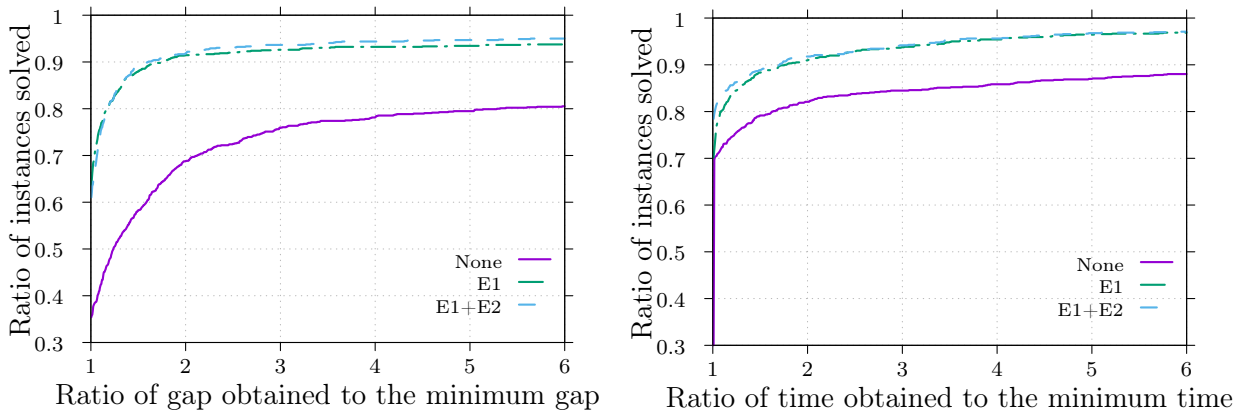
Given the possibility of linearizing instances of Class-II, it is natural to ask whether it is better to use our proposed algorithms to solve them or first linearize such instances and then solve them directly as (single-objective) mixed integer linear programs using CPLEX? We refer to the second approach as *L-CPLEX* in the remainder of this paper and we will ex-

plore this question in this computational study. Next, we explain how each class of instances are generated.

Each class of instances in this study contains 16 subclasses with different number of variables n and constraints m. Specifically, each subclass is denoted by $m \times n$ for which we have that $n \in \{400, 800, 1200, 1600\}$ and $m = \alpha \times n$ where $\alpha \in \{1, 2, 3, 4\}$. Each subclass contains 10 random instances and to generate each, the value of the parameters in A, C, and $\boldsymbol{d}$ are randomly generated from the discrete uniform distribution in the interval $[1, 10]$. In addition, the sparsities of the matrices A and C are set to 50%, i.e. we set 50% of their elements to zeros. To ensure that $\mathcal{X}$ is bounded (which is an assumption of this study), we impose an upper bound of equal to one for all n decision variables when generating an instance, i.e., we set $x_i \leqslant 1$ for all $i = 1, \ldots, n$. Finally, to ensure the feasibility of an instance, the value of b_i is randomly generated from the discrete uniform distribution in the interval $[0, \sum_{j=1}^{n} a_{ij}]$ for each $i \in \{1, \ldots, m\}$.

In this computational study, we frequently use *performance profiles* [127]. A performance profile presents cumulative distribution functions for a set of algorithms being compared with respect to a specific performance metric, i.e., the run time or optimality gap in this study. The run time performance profile for a set of algorithms is constructed by computing for each algorithm and for each instance the ratio of the run time of the algorithm on the instance and the minimum of the run times of all algorithms on the instance. The run time performance profile then shows the ratios on the horizontal axis and, on the vertical axis, for each algorithm, shows the percentage of instances with a ratio that is smaller than or equal to the ratio on the horizontal axis. This implies that values in the upper left-hand corner of the graph indicate the best performance. The optimality gap performance profile can be constructed and interpreted similarly. As an aside, in this study, the optimality gaps and/or the run times are sometimes close to zero and this can create problems when computing ratios in the performance profiles. To resolve this issue, when creating performance

profiles, we first add 0.01% to all reported optimality gaps and 0.01 second to all reported run times.

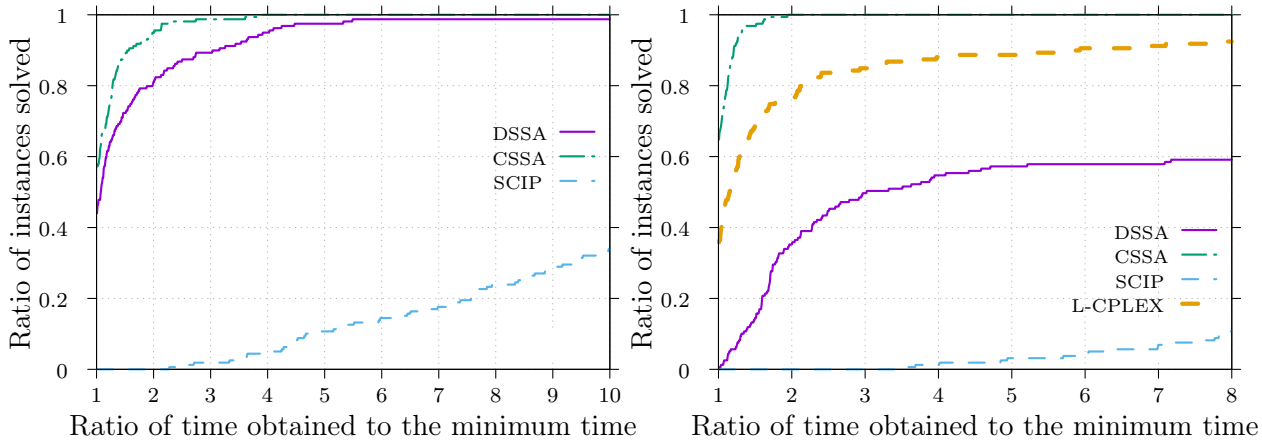


(a) Optimality gap

(b) Run time

Figure 4.5: Performance profiles of the proposed enhancement techniques on DSSA over all instances

Finally, as mentioned in Section 4.3, DSSA can be potentially improved by employing Enhancement-I (E1) and Enhancement-II (E2). So, before we compare the performance of different algorithms, we check whether activating the proposed enhancement techniques can be helpful. Figure 4.5 illustrates the run time and optimality gap performance profiles of DSSA under three different scenarios: no enhancement is active (Scenario 1), only E1 is active (Scenario 2), both E1 and E2 are active (Scenario 3), on all our 960 instances. We observe that both the run time and optimality gap performance profiles under Scenario 3 are slightly better than Scenario 2 but they are significantly better than Scenario 1. Specifically, by comparing Scenarios 1 and 3, we observe that around 10% of instances are solved 6 times faster. Also, the optimality gap of around 15% of instances are at least 6 times smaller. Hence, in the remainder of this section, whenever we refer to DSSA, we assume that both enhancements are active.

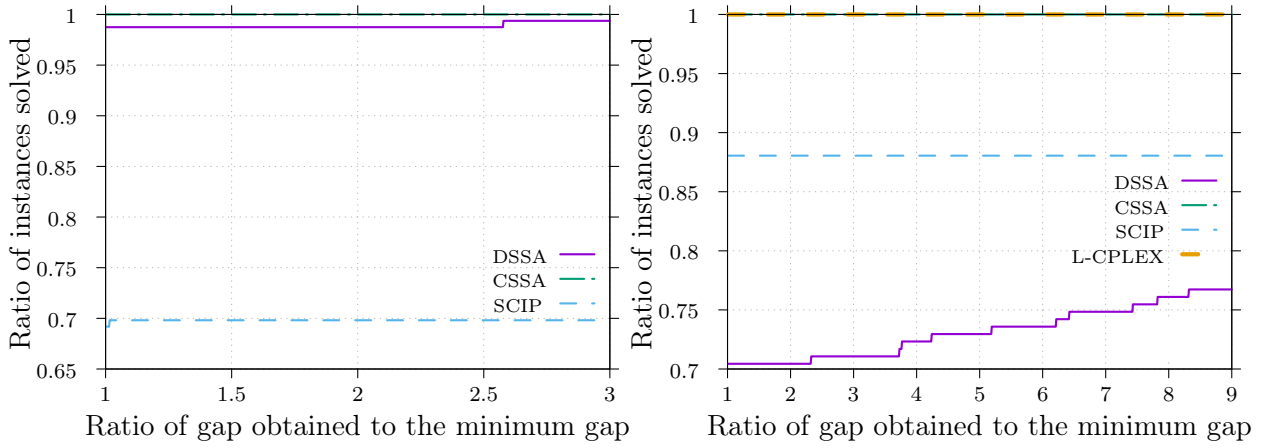


(a) Class-I (b) Class-II

Figure 4.6: Run time performance profiles when $p = 2$

4.5.1 Two Objectives ($p = 2$)

In this section, we compare the performance of different methods including DSSA, CSSA, SCIP, and L-CPLEX, on solving instances with $p = 2$. The run time performance profiles of the different methods on instances of Class-I and Class-II can be found in Figure 5.5. Also, the optimality gap performance profiles of different methods on instances of Class-I and Class-II can be found in Figure 4.7.



(a) Class-I (b) Class-II

Figure 4.7: Optimality gap performance profiles when $p = 2$

Note that L-CPLEX is only tested on instances of Class-II since they do not have any continuous decision variables. Observe that for both Class-I and Class-II, CSSA has the best performance and SCIP has the worst performance. For Class-I, CSSA has been able to solve around 10% and 98% of instances at least 3 times faster than DSSA and SCIP, respectively. For Class-II, CSSA has been able to solve around 16%, 50%, and 99% of instances at least 3 times faster than L-CPLEX, DSSA, and SCIP, respectively. So, the second best method for solving instances of Class-II has been L-CPLEX. However, the difference between Algorithm 3 and L-CPLEX is significant in Class-II and this is highlighted by this observation that around 8% of instances are solved at least 8 times faster using CSSA. In terms of the optimality gap, a similar pattern can be observed. Specifically, Algorithm 3 has been able to solve all instances to optimality within the imposed time limit. However, DSSA has performed poorly on Class-II but has been able to solve almost all instances of Class-I within the imposed time limit. This is mainly because, by construction, Class-I contains significantly fewer integer decision variables compared to Class-II. Therefore DSSA, which is a decision space search algorithm, struggles to solve such instances because the size of its search tree depends highly on the number of integer decision variables. Finally, we also observe that, for around 30% of instances, CSSA has reached an optimality gap which is at least 3 times smaller than SCIP on Class-I and DSSA on Class-II.

Table 4.1 provides further details about the performance of DSSA and CSSA on instances with $p = 2$. Columns labeled 'Time (s)' show the solution time in seconds, columns labeled 'Gap (%)' show the optimality gap, and columns labeled 'Solved (#)' show the number of instances solved to optimality. Numbers in this table, except those in Columns labeled 'Solved (#)', are averages over 10 instances. The table clearly shows that CSSA outperforms DSSA significantly. Observe that the largest subclass of instances are solved to optimality in around 50.72 seconds and 259.16 seconds using CSSA under Class-I and Class-II, respectively. However, DSSA was not able to solve 1 and 6 instances of the largest

subclass of instances to optimality within the imposed time limit under Class-I and Class-II, respectively.

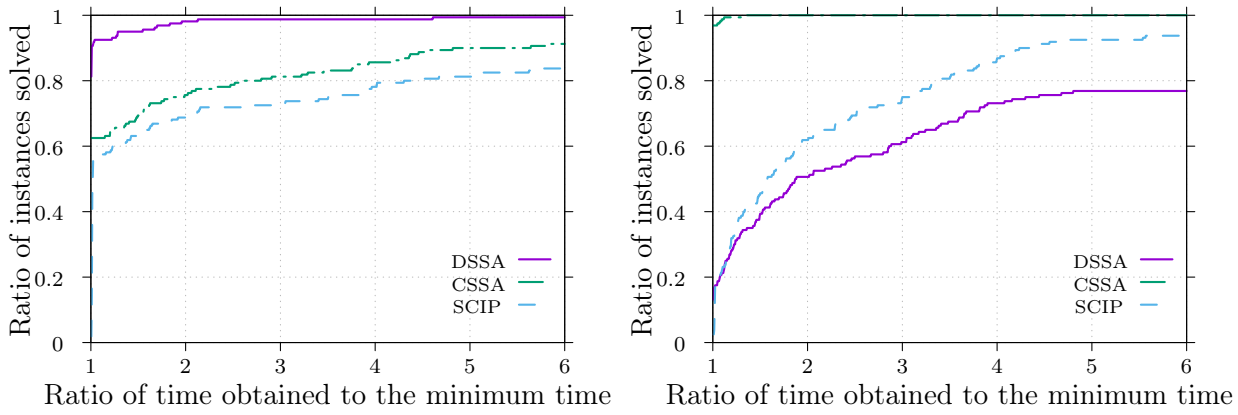
Table 4.1: Performance comparison of DSSA and CSSA on instances with $p = 2$

Subclass	**Class-I**						**Class-II**					
	DSSA			**CSSA**			**DSSA**			**CSSA**		
	Time (s)	**Gap** (%)	**Solved** (#)	**Time** (s)	**Gap** (%)	**Solved** (#)	**Time** (s)	**Gap** (%)	**Solved** (#)	**Time** (s)	**Gap** (%)	**Solved** (#)
400 × 400	5.43	0.00	10	3.07	0	10	5.53	0.00	10	1.06	0	10
800 × 400	20.17	0.00	10	7.00	0	10	501.63	0.03	9	2.26	0	10
1200 × 400	47.15	0.00	10	14.46	0	10	1674.59	0.08	6	3.56	0	10
1600 × 400	20.25	0.00	10	14.90	0	10	2199.49	1.09	4	7.08	0	10
800 × 800	10.96	0.00	10	6.57	0	10	10.14	0.00	10	1.15	0	10
1600 × 800	20.72	0.00	10	32.72	0	10	796.25	1.45	8	4.40	0	10
2400 × 800	147.65	0.00	10	40.35	0	10	733.05	0.54	8	7.05	0	10
3200 × 800	334.57	0.00	10	98.65	0	10	2224.25	3.37	4	16.63	0	10
1200 × 1200	15.17	0.00	10	15.83	0	10	15.95	0.00	10	3.30	0	10
2400 × 1200	105.38	0.00	10	50.35	0	10	880.78	0.47	8	8.38	0	10
3600 × 1200	506.98	0.18	9	68.54	0	10	1464.86	1.41	7	19.94	0	10
4800 × 1200	132.27	0.00	10	145.50	0	10	2888.05	1.43	2	46.29	0	10
1600 × 1600	20.16	0.00	10	17.16	0	10	24.60	0.00	10	5.69	0	10
3200 × 1600	74.18	0.00	10	102.61	0	10	860.45	0.22	9	19.83	0	10
4800 × 1600	80.60	0.00	10	129.56	0	10	3077.95	5.25	4	38.06	0	10
6400 × 1600	695.01	0.00	9	259.16	0	10	2621.50	10.00	4	50.72	0	10

4.5.2 Three Objectives ($p = 3$)

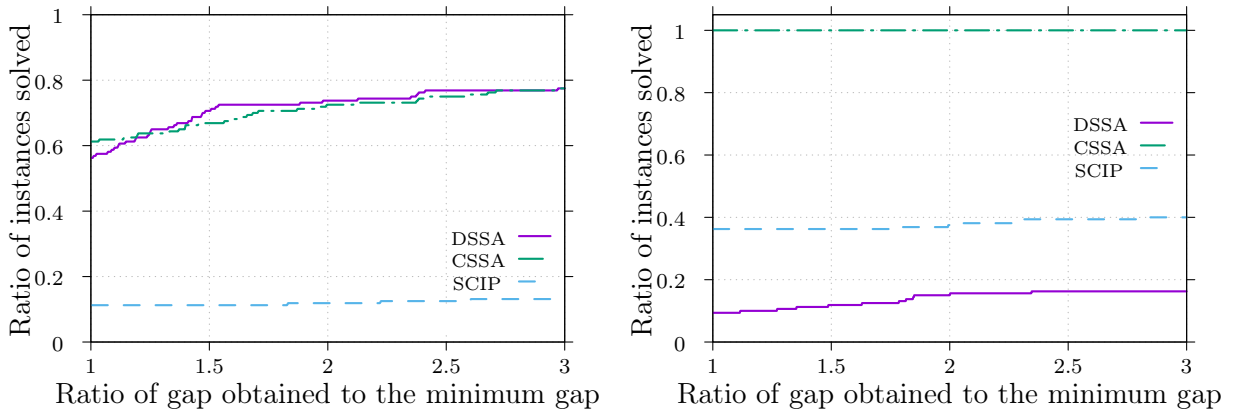
In this section, we compare the performance of different methods including DSSA, CSSA, and SCIP, on solving instances with $p = 3$. Note that, in the remainder of this paper, we do not provide any further result for L-CPLEX because for $p > 2$ many constraints and decision variables need to be added for linearizion and CPLEX will struggle to solve such linearized instances. With this in mind, the run time performance profiles of the different methods on instances of Class-I and Class-II can be found in Figure 4.8. Also, the optimality gap performance profiles of different methods on instances of Class-I and Class-II can be found in Figure 4.9.

One interesting observation is that, unlike instances with $p = 2$, DSSA outperforms CSSA for Class-I. Specifically, DSSA has been able to solve around 20% and 30% of instances at least 3 times faster than CSSA and SCIP, respectively. This is mainly because of two reasons. (1) Class-I involves less integer decision variables and hence a decision space search



(a) Class-I (b) Class-II

Figure 4.8: Run time performance profiles when $p = 3$



(a) Class-I (b) Class-II

Figure 4.9: Optimality gap performance profiles when $p = 3$

algorithm can be expected to perform well for such instances. (2) CSSA is a criterion space search algorithm and hence as p increases, more nodes need to be added after exploring each node of the search tree. So, the global dual bound is expected to improve at slower rate during the search. For Class II, however, CSSA has the best performance and DSSA has the worst performance. Specifically, CSSA has been able to solve around 40% and 30% of instances at least 3 times faster than DSSA and SCIP, respectively. In terms of the optimality gap, for Class-I, DSSA and CSSA have similar performance. However, for around 70% of instances of Class-I, CSSA has reached to an optimality gap which is at least 3 times smaller than SCIP. For Class II, DSSA has performed poorly in terms of the optimality gap. Specifically, we observe that, for around 80% of instances in Class-II, CSSA has reached to an optimality gap which is at least 3 times smaller than DSSA.

Table 4.2: Performance comparison of DSSA and CSSA on instances with $p = 3$

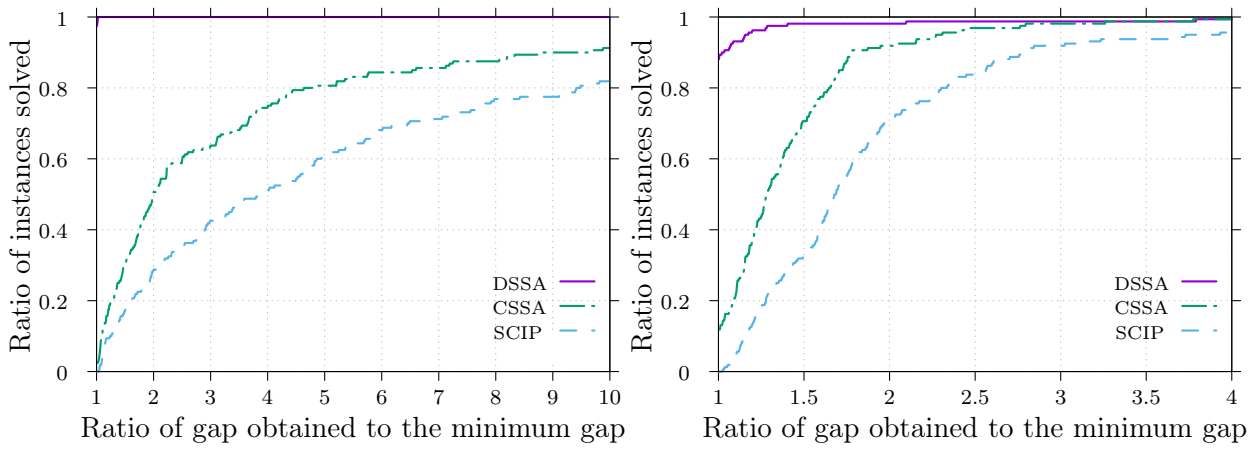
Subclass	**Class-I**						**Class-II**					
	DSSA			**CSSA**			**DSSA**			**CSSA**		
	Time (s)	**Gap** (%)	**Solved** (#)	**Time** (s)	**Gap** (%)	**Solved** (#)	**Time** (s)	**Gap** (%)	**Solved** (#)	**Time** (s)	**Gap** (%)	**Solved** (#)
400 × 400	1762.16	3.01	6	2184.25	0.18	5	3135.94	9.73	2	98.45	0.00	10
800 × 400	1925.29	2.99	5	3334.84	0.34	2	3600.00	18.25	0	377.61	0.00	10
1200 × 400	1956.08	10.02	5	3405.11	0.19	2	3600.00	14.38	0	682.83	0.00	10
1600 × 400	3000.39	4.99	2	3357.12	0.23	1	3600.00	16.55	0	1403.17	0.30	9
800 × 800	1234.85	1.52	7	1991.53	0.04	6	2882.26	5.65	2	103.00	0.00	10
1600 × 800	2229.45	3.27	4	3295.56	2.80	2	3600.00	16.24	0	680.56	0.00	10
2400 × 800	3258.70	5.65	1	3600.00	2.26	0	3600.00	24.81	0	1098.76	0.00	10
3200 × 800	3204.88	5.24	2	3600.00	3.46	0	3600.00	23.62	0	2226.01	0.34	8
1200 × 1200	1115.15	0.75	7	1612.77	0.02	8	2618.78	5.86	4	231.66	0.00	10
2400 × 1200	2125.83	4.38	6	3385.22	10.12	1	3600.00	15.21	0	1130.86	0.00	10
3600 × 1200	3198.68	4.89	2	3600.00	5.37	0	3600.00	29.85	0	2907.52	3.05	5
4800 × 1200	3331.31	5.90	1	3600.00	12.24	0	3600.00	24.61	0	3145.63	5.61	4
1600 × 1600	504.71	0.35	9	1768.84	0.76	7	2634.85	3.41	4	241.15	0.00	10
3200 × 1600	2954.51	8.21	3	3250.68	8.63	1	3600.00	14.94	0	1354.44	0.00	10
4800 × 1600	2799.84	6.64	4	3156.85	9.78	2	3600.00	18.46	0	2839.52	2.92	5
6400 × 1600	3600.00	9.54	0	3600.00	21.70	0	3600.00	26.24	0	3320.17	7.10	4

Table 4.2 provides further details about the performance of DSSA and CSSA on instances with $p = 3$. Observe that the solution times have increased dramatically for instances with $p = 3$ compared to instances with $p = 2$. It is evident that DSSA has been able to solve significantly more instances to optimality for Class-I. However, for Class-II, CSSA has solved significantly more instances to optimality. The average optimality gap

reported for each subclass indicates a similar pattern. Specifically, for the largest subclass of instances, we observe that the average optimality of CSSA is more than 2.2 times larger than DSSA under Class-I but it is more than 3.6 times smaller under Class-II.

4.5.3 Four Objectives ($p = 4$)

In this section, we compare the performance of different methods including DSSA, CSSA, and SCIP, on solving instances with $p = 4$. Unlike the previous cases, i.e., $p = 2$ and $p = 3$, we do not provide any run time performance profiles because almost all instances are not solved to optimality by any of the solution methods within the imposed time limit. The optimality gap performance profiles of different methods on instances of Class-I and Class-II can be found in Figure 4.10.



(a) Class-I (b) Class-II

Figure 4.10: Optimality gap performance profiles when $p = 4$

From the figure, we observe that DSSA has completely outperformed CSSA in terms of the optimality gap. Specifically, for around 20% and 40% of instances of Class-I, DSSA has reached to an optimality gap which is at least 5 times better than those obtained by CSSA and SCIP, respectively. Similarly, for around 30% and 70% of instances of Class-II, DSSA has reached to an optimality gap which is at least 1.5 times better than those obtained by CSSA and SCIP, respectively. Table 4.3 provides further details about the performance

of DSSA and CSSA on instances with $p = 4$. Observe that very few instances are solved to optimality by the methods within the imposed time limit. The average optimality gap reported for each subclass indicates that DSSA has completely outperformed CSSA. This is another evidence that for large values of p, the proposed decision space search algorithm is a better choice.

Table 4.3: Performance comparison of DSSA and CSSA on instances with $p = 4$

Subclass	**Class-I**						**Class-II**					
	DSSA			**CSSA**			**DSSA**			**CSSA**		
	Time (s)	**Gap** (%)	**Solved** (#)	**Time** (s)	**Gap** (%)	**Solved** (#)	**Time** (s)	**Gap** (%)	**Solved** (#)	**Time** (s)	**Gap** (%)	**Solved** (#)
400 × 400	3282.43	15.88	1	3600.00	59.25	0	3600.00	24.73	0	3600.00	36.57	0
800 × 400	3600.00	12.43	0	3600.00	30.31	0	3600.00	31.26	0	3600.00	56.83	0
1200 × 400	3600.00	12.47	0	3600.00	38.53	0	3600.00	27.60	0	3600.00	63.95	0
1600 × 400	3600.00	10.43	0	3600.00	36.79	0	3600.00	36.72	0	3600.00	67.02	0
800 × 800	3273.97	9.69	1	3600.00	55.89	0	3600.00	23.57	0	3504.52	40.83	1
1600 × 800	3087.01	18.51	2	3600.00	65.06	0	3600.00	36.15	0	3600.00	72.58	0
2400 × 800	3600.00	22.51	0	3600.00	63.75	0	3600.00	36.73	0	3600.00	73.35	0
3200 × 800	3600.00	11.90	0	3600.00	68.32	0	3600.00	30.67	0	3600.00	76.33	0
1200 × 1200	3600.00	12.18	0	3600.00	47.08	0	3600.00	32.21	0	3600.00	54.36	0
2400 × 1200	3600.00	18.90	0	3600.00	69.25	0	3600.00	45.16	0	3600.00	69.40	0
3600 × 1200	3600.00	18.00	0	3600.00	81.08	0	3600.00	50.78	0	3600.00	78.44	0
4800 × 1200	3600.00	40.70	0	3600.00	70.59	0	3600.00	24.53	0	3600.00	85.33	0
1600 × 1600	3323.96	6.67	1	3600.00	50.39	0	3600.00	35.52	0	3600.00	47.60	0
3200 × 1600	3567.38	12.69	1	3600.00	74.71	0	3600.00	49.32	0	3600.00	68.65	0
4800 × 1600	3600.00	55.16	0	3600.00	84.29	0	3600.00	47.78	0	3600.00	76.75	0
6400 × 1600	3600.00	49.08	0	3600.00	82.58	0	3600.00	61.08	0	3600.00	84.45	0

4.6 Conclusions

We studied a class of single-objective non-convex non-linear (mixed) integer optimization problems, i.e., MIL-mMPs, with applications in different fields of study, e.g., natural resource management. We showed that because the objective function of a MIL-mMP is a multiplication of p non-negative linear functions, a MIL-mMP can be viewed as the problem of optimization over the efficient set of a multi-objective mixed integer linear program with p objective functions. Consequently, we developed two novel multi-objective optimization based algorithms for solving MIL-mMPs. The first algorithm is a decision space search algorithm that relies on the power of (single-objective) linear programming solvers for solving a MIL-mMP. The second algorithm, however, is a criterion space search algorithm that relies

on the power of (single-objective) mixed integer linear programming solvers for solving a MIL-mMP.

We developed several enhancement techniques for the proposed algorithms and tested the performance of our algorithms on 960 random instances against the non-linear optimization solver of SCIP. Our numerical results showed that SCIP is significantly outperformed by our proposed algorithms. We also showed that commercial mixed integer programming solvers struggle to directly solve instances of MIL-mMP when their objective functions are linearized in advance. The results showed that when $p = 2$ the proposed criterion space search algorithm is the best solution method. However, as p increases, the proposed criterion space search algorithm starts to struggle because the dimension of the criterion space increases. Consequently, for $p = 4$, we observed that the proposed decision space search algorithm outperforms the proposed criterion space search algorithm. Similarly, we also observed that as the number of integer decision variables increases, the proposed decision space search algorithm starts to struggle because the dimension of the decision space increases. Consequently, for $p = 3$, we observed that the decision space search algorithm was the best choice for our mixed integer instances because they involved fewer integer decision variables (compared to our pure integer instances). However, for pure integer instances with $p = 3$, the criterion space search algorithm was the best choice because they involved more integer decision variables (compared to our mixed integer instances).

We hope that the simplicity, versatility, and performance of our proposed algorithms encourage practitioners/researchers to consider developing/employing multi-objective optimization based algorithms for solving certain single-objective optimization problems in particular multiplicative programs. There are several future research directions that can be considered for this study. One direction could be exploring whether the proposed algorithms can be combined together through an effective parallelization framework to integrate their advantageous. Another research direction is to explore how the proposed methods can be

customized effectively for solving the so-called 'generalized' MIL-mMPs in which the only difference is that each term in the objective function may have a positive power.

Chapter 5: A Criterion Space Branch-And-Cut Algorithm For Mixed Integer Bi-Linear Maximum Multiplicative Programs

In this chapter, we present paper P4. In this work, we introduce a branch-and-bound algorithm to solve Mixed-Integer Bi-Linear Maximum Multiplicative Programs.

5.1 Introduction

A maximum multiplicative program (MMP) can be formulated as

$$\max \left\{ \prod_{i=1}^{p} y_i(\boldsymbol{x}) : \ \boldsymbol{x} \in \mathcal{X}, \ \boldsymbol{y}(\boldsymbol{x}) \geqslant \mathbf{0} \right\}, \tag{5.1}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$ is the set of feasible solutions, and $\boldsymbol{y}(\boldsymbol{x})$ is a vector of p non-negative linear functions of $\boldsymbol{x} \in \mathcal{X}$. If $\mathcal{X}$ consists of a set of linear constraints, the problem is called linear maximum multiplicative program (L-MMP). If the values of all variables are expected to be integers, it is referred to as an integer linear maximum multiplicative program (IL-MMP). Finally, if the variable set incorporates both integer and continuous variables, the problem becomes a mixed integer linear maximum multiplicative (MIL-MMP). Further details about extensions of MMPs can be found in the work done by [128].

The MMP, in its general form, has numerous applications. The MMP essentially finds the Nash solution in a bargaining problem [129]. Due to the importance of the MMP in solving real-world bargaining problems, the MMP is sometimes referred to as the Nash social welfare program in the literature [130]. For example, Caragiannis et al. [131] employed IL-MMPs to solve the allocation of indivisible goods among heirs, which is a real-world bargaining problem. Other application domains of bargaining problems that require the

MMP include but not limited to energy [132, 133], conservation planning [134], and healthcare [135, 136]. Supply chain management is another important domain that gives rise to bargaining problems [137, 138]. Forming collaboration in supply chains through solving bargaining problems involving two players, i.e., $p = 2$, has been the topic of several studies in the literature [see 139, 140, and the citations therein]. In addition to the bargaining problems, the MMP can be used for computing an equilibrium for Fisher's linear or Kelly's capacity allocation market [141, 142, 143, 144]. Systems reliability is another application of such optimization problems [145, 146, 147].

Although the MMP has only one objective function, it has strong connections to the problem of *optimization over the efficient set* in multi-objective optimization [116, 118, 115, 117, 114]. In multi-objective optimization, the efficient set is the set of solutions for which it is impossible to improve the value of one objective without making the value of at least one other objective worse. While a large body of literature on multi-objective optimization has sought to compute the efficient set of multi-objective (mixed integer) optimization problems both in the linear form [94, 95, 96, 97, 98] and nonlinear form [148, 149, 150], the goal of the problem of optimization over the efficient set is to directly find the most desirable efficient solution without necessarily exploring the entire efficient set. This will be done by optimizing a super-criterion function over the efficient set of a multi-objective optimization problem.

The connection between the MMP and optimization over the efficient set in multi-objective optimization comes from an idea that in Problem (5.1), each element of $\boldsymbol{y}$ can be viewed as a dummy objective function. Without loss of generality, the multi-objective optimization problem counterpart to Problem (5.1) can be stated as

$$\max \Big\{ y_1(\boldsymbol{x}), \ldots, y_p(\boldsymbol{x}) : \ \boldsymbol{x} \in \mathcal{X}, \ \boldsymbol{y}(\boldsymbol{x}) \geqslant \mathbf{0} \Big\}. \tag{5.2}$$

with p, $\mathcal{X}$ and $\boldsymbol{y}(\boldsymbol{x})$ bearing the same definitions as in Problem (5.1). It is known that an optimal solution of Problem (5.1) lies on the efficient set of its multi-objective problem coun-

terpart. Charkhgard et al. [101] proved this property for L-MMPs, however the same proof applies to MMPs in general as used by Saghand et al. [102] and Saghand and Charkhgard [128] for MIL-MMPs (with $p = 2$) and IL-MMPs, respectively. Hence, the optimal solution of Problem (5.1) can be found by identifying the element in the efficient set of Problem (5.2) that maximizes the objective function of Problem (5.1); namely $\prod_{i=1}^{p} y_i(\boldsymbol{x})$. This implies that instead of directly solving Problem (5.1) using single-objective optimization solvers, the super-criterion $\prod_{i=1}^{p} y_i(\boldsymbol{x})$ can be maximized over the efficient set of Problem (5.2).

The above-mentioned idea has been the basis of developing novel and highly-effective multi-objective optimization based solution approaches for solving the MMP in recent years [101, 102, 128, 151]. However, such approaches are mainly designed for solving IL-MMPs and not MIL-MMPs. This is not surprising as the multi-objective optimization counterpart of an IL-MMP is a multi-objective pure integer linear program (MOPILP). However, the multi-objective optimization counterpart of an MIL-MMP is a multi-objective mixed integer linear program (MOMILP). In multi-objective optimization, it is known [123] that MOMILPs are significantly more challenging than MOPILPs because the image of the efficient set of a MOMILP in the criterion space, i.e., the space of objective values, is not necessarily a collection of discrete points and may contain disjoint continuous segments. In fact, this is the main reason that the literature of solution approaches for MOMILPs has been mainly limited to cases involving only two objectives, e.g., [152], and there are very few studies on cases with more than two objectives, e.g., [153].

In light of the above, the focus of this study is on developing effective multi-objective optimization based solution approaches for solving any MIL-MMP with $p = 2$, referred to as mixed-integer bi-linear maximum multiplicative program (MIBL-MMP) in the remaining of this paper. We introduce a new approach that, from a computational time perspective, outperforms not only the existing approaches in the literature, but also CPLEX 12.10—the latest version of the commercial solver, while guaranteeing the optimality of the solution. Our contributions can be summarized as follows.

- We introduce a novel criterion space branch-and-cut (CSBnC) algorithm to solve any MIBL-MMP by solving a finite number of single-objective mixed integer linear programs. The proposed CSBnC framework explores the efficient set of the multi-objective problem counterpart of the MIBL-MMP through a criterion space-based branch-and-cut paradigm. Starting with an initial set of primal and dual bounds, the proposed branch-and-bound iteratively improves the bounds until the desired optimality gap is achieved. The bounds will be obtained using novel custom-built operations developed based on Chebyshev distance and piecewise McCormick envelopes. CSBnC employs a criterion-space cut-generating mechanism adopted from [101] and [128] that generate a cut for any feasible solution with strictly positive objectives.

- We conduct an extensive computational study by first evaluating numerically the contribution of the main algorithmic component of the proposed framework. We also obtain the optimal number of pieces for the piecewise McCormick envelopes. Next, we compare the performance of our proposed CSBnC algorithm with a state-of-the-art multi-objective optimization solution approach [151] and the mixed integer Second Order Cone Programming (SOCP) solver of CPLEX 12.10. Our tests demonstrate that our approach outperforms the SOCP solver by a factor of 6.60 on average while the SOCP solver itself outperforms the state-of-the-art approach by a factor of 1.25 on average.

The remainder of this paper is organized as follows. In Section 5.2, we introduce some preliminaries about MIBL-MMPs, introduce some existing methods, and explain some important concepts and notations. In Section 5.3, we introduce the key operations/components of CSBnC. In Section 5.4, we provide the details of CSBnC. Section 5.5 includes a comprehensive computational study, and finally, in Section 5.6, some concluding remarks are provided.

5.2 Preliminaries

A Mixed Integer Bi-Linear Maximum Multiplicative Program (MIBL-MMP) can be stated as

$$\begin{aligned} \max\ & y_1 y_2 \\ \text{s.t. } & \boldsymbol{y} = C\boldsymbol{x} + \boldsymbol{d} \\ & A\boldsymbol{x} \leqslant \boldsymbol{b} \\ & \boldsymbol{x}, \boldsymbol{y} \geqslant \boldsymbol{0}, \quad \boldsymbol{x} \in \mathbb{R}^{n_c} \times \mathbb{Z}^{n_i}, \quad \boldsymbol{y} \in \mathbb{R}^2, \end{aligned} \tag{5.3}$$

where n_c and n_i denote the number of continuous and integer decision variables, respectively. Given $n = n_c + n_i$, C is a 2×n matrix and $\boldsymbol{d}$ is a vector of size two, which include the coefficients and constants of the two multiplicative terms, respectively. Also, A is an $m \times n$ matrix that denotes the technological coefficients, with m being the number of linear constraints. Accordingly, $\boldsymbol{b}$ represents the m-sized vector of right-hand side values. We refer to the set $\mathcal{X} := \{\boldsymbol{x} \in \mathbb{R}^{n_c} \times \mathbb{Z}^{n_i} : A\boldsymbol{x} \geqslant \boldsymbol{b},\ \boldsymbol{x} \geqslant \boldsymbol{0}\}$ as *the feasible set in the decision space* and to the set $\mathcal{Y} := \{\boldsymbol{y} \in \mathbb{R}^2 : \boldsymbol{x} \in \mathcal{X},\ \boldsymbol{y} = C\boldsymbol{x} + \boldsymbol{d},\ \boldsymbol{y} \geqslant \boldsymbol{0}\}$ as *the feasible set in the criterion space*. We assume that $\mathcal{X}$ is bounded (which implies that $\mathcal{Y}$ is compact) and the optimal objective value of the problem is strictly positive. Throughout this article, we refer to $\boldsymbol{x} \in \mathcal{X}$ as a *feasible solution* and to $\boldsymbol{y} \in \mathcal{Y}$ as a *feasible point* ($\boldsymbol{y}$ is the image of $\boldsymbol{x}$ in the criterion space).

5.2.1 Existing Solution Approaches

Note that Problem (5.3) can be stated as $\max_{\boldsymbol{y} \in \mathcal{Y}} y_1 y_2$. Considering this observation, in order to solve an MIBL-MMP, there are two standard single-objective optimization based solution approaches. The first one applies the log-transformation, i.e., $\max_{\boldsymbol{y} \in \mathcal{Y}} \log y_1 + \log y_2$, and then solves the transformed problem by standard mixed integer convex programming solvers. The second approach, which is a faster approach, transforms an MIBL-MMP into a

mixed integer SOCP [101], i.e.,

$$\begin{aligned} &\max_{\boldsymbol{y}\in\mathcal{Y}} \gamma \\ &\text{s.t. } \gamma^2 \leqslant y_1 y_2 \text{ and } \gamma \geqslant 0 \end{aligned}$$

and then solves the transformed problem by powerful commercial solvers such as CPLEX. In addition to the standard single-objective optimization approaches, it is also possible to develop multi-objective optimization-based solution approaches for solving any MIBL-MMP based on the following definition and proposition.

Definition 5.1. A feasible solution $\boldsymbol{x} \in \mathcal{X}$ is called *efficient*, if there is no other $\boldsymbol{x}' \in \mathcal{X}$ such that $\boldsymbol{y}' \geqslant \boldsymbol{y}$ and $\boldsymbol{y}' \neq \boldsymbol{y}$ where $\boldsymbol{y} := C\boldsymbol{x} + \boldsymbol{d}$ and $\boldsymbol{y}' := C\boldsymbol{x}' + \boldsymbol{d}$. If $\boldsymbol{x}$ is efficient, then $\boldsymbol{y}$ is called a *nondominated point*. The set of all efficient solutions is denoted by $\mathcal{X}_E$. The set of all nondominated points is denoted by $\mathcal{Y}_\mathcal{N}$ and referred to as the *nondominated frontier*.

Proposition 5.1. *An optimal solution of Problem* (5.3)*, denoted by* $\boldsymbol{x}^*$*, is an efficient solution and therefore its corresponding image in the criterion space, denoted by* $\boldsymbol{y}^*$ *where* $\boldsymbol{y}^* := C\boldsymbol{x}^* + \boldsymbol{d}$*, is a nondominated point.*

Proof.

It is straightforward and can be found in [102].

Proposition 5.1 implies that Problem (5.3) is a spacial case of the problem of optimization over the efficient set because it is equivalent to $\max_{\boldsymbol{y}\in\mathcal{Y}_\mathcal{N}} y_1 y_2$. A straightforward way to solve any optimization over the efficient set problem is to first compute its nondominated frontier and then search the nondominated frontier for a point that optimizes the super-criterion function (which is $y_1 y_2$ in this paper). Such a two-phase multi-objective optimization-based approach can be computationally expensive as the first phase, i.e., computing the entire nondominated frontier, can be time consuming in practice. Therefore, in the literature of optimization over the efficient set, the focus has been mainly on develop-

ing multi-objective optimization-based solution methods that can directly find the optimal point without having to generate the entire nondominated frontier by employing bounding mechanisms and/or cut-generating techniques [118, 116].

In the context of MIBL-MMPs, [102] are the first to develop such an algorithm. They embed the multi-objective linear programming-based algorithm proposed by [101] for solving L-MMPs with $p = 2$ into a branch-and-bound framework to solve any MIBL-MMP. Thus, their approach relies on the power of (commercial) single-objective linear programming solvers to solve linear programs arising during the course of solving MIBL-MMP. [151] propose a multi-objective optimization-based approach that outperforms the approach proposed previously by the same authors as it relies on the power of (commercial) single-objective mixed integer programming solvers to solve any MIBL-MMP. Specifically, in each iteration, their algorithm solves a mixed integer linear program which maximizes $y_1 + y_2$ over the remaining part of the feasible set in the criterion space for finding a new efficient solution. After finding a new solution, their algorithm takes the two following steps. In the first step, it seeks to compute the best possible feasible solution that can be obtained by changing only the values of continuous variables while fixing the values of integer variables to the integer support vector of the solution found in that iteration. This will be done by simply fixing the integer support vector of the solution obtained (in each iteration) in the mixed integer SOCP counterpart of the MIBL-MMP. By doing so, the mixed integer SOCP reduces to a continuous SOCP that is then solved by a commercial solver such as CPLEX. In the second step, a no-good constraint [154] is added to the model to remove the integer support vector explored in that iteration permanently from future iterations.

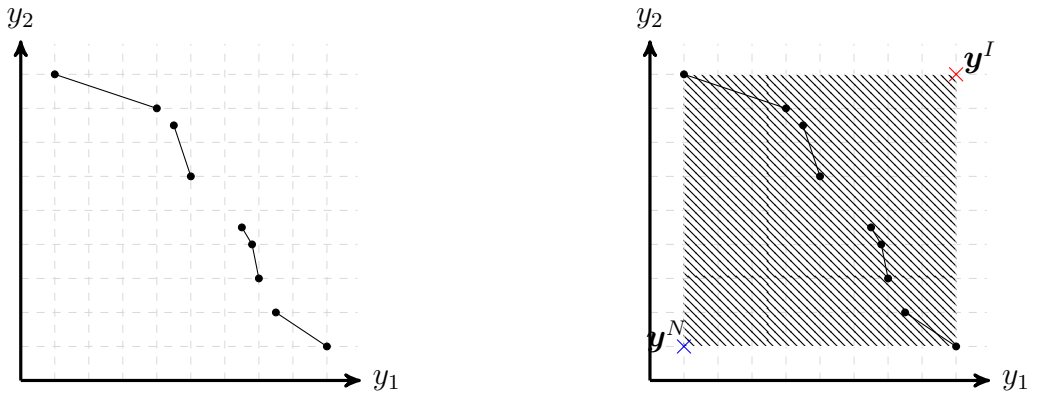
The two steps that the algorithm of [102] relies on are computationally expensive in practice; the first step is expensive due to its non-linear nature and the second step is expensive as it is a cut in the decision space that increases the size of the model in each iteration but has no strong impacts on the criterion space. Therefore, the main motivation of this study is to present an effective multi-objective optimization-based approach capable

of solving mixed integer instances without requiring such expensive steps. We do this by employing new search mechanisms, bounding procedures, and cut-generating technique. As an aside, we note that the proposed approach in this study cannot be directly used for solving $\min_{\boldsymbol{y}\in\mathcal{Y}} y_1 y_2$. At first glance, this problem may look similar to an MIBL-MMP. However, they are substantially different in theory. This can be observed from the fact that for the case of maximization, if there is no integer decision variable, the problem is polynomially solvable. However, for the case of minimization, the problem is NP-hard even if there is no integer decision variable [100, 1, 155].

5.2.2 Concepts and Notations

In this section, we introduce a few concepts and notations that are important for understanding the motivation behind the main operations in our proposed approach for solving an MIBL-MMP. The first concept is the *ideal* point, denoted by $\boldsymbol{y}^I \in \mathbb{R}^2$, which is a point in the criterion space reflecting the best (maximum) value of each objective function (when considered individually) over the nondominated frontier. The second concept is the *nadir* point, denoted by $\boldsymbol{y}^N \in \mathbb{R}^2$, which is a point in the criterion space reflecting the worst (minimum) value of each objective function (once again when considered individually) over the nondominated frontier. Both ideal and nadir points are often imaginary points in the criterion space as they may not be feasible.

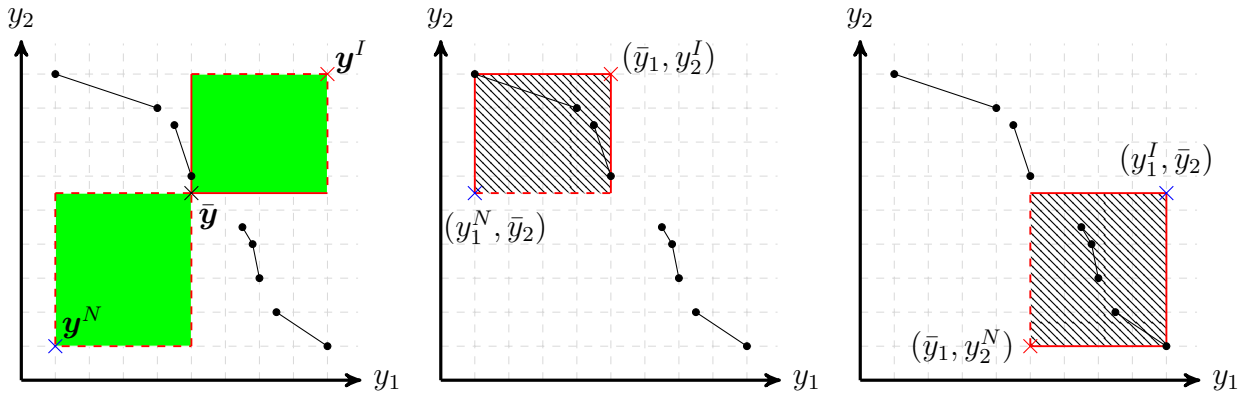
Fortunately for $p = 2$, computing the ideal and nadir points is straightforward and can be done by simply computing the endpoints of the nondominated frontier. We note that this observation does not hold for $p > 2$ [96]. An illustration of the entire nondominated frontier of the multi-objective counterpart of an MIBL-MMP and its corresponding ideal and nadir points can be found in Figure 5.1. In order to compute each endpoint, a lexicographic operation can be applied. The lexicographic operation is basically two sequential optimization problems that should be solved. In the first one, a primary objective function should be optimized over the entire feasible set and in the second one a secondary objective should



(a) The nondominated frontier (b) Ideal and nadir points

Figure 5.1: An illustration of the entire nondominated frontier of the multi-objective counterpart of an MIBL-MMP

be optimized over all feasible solutions that are optimal for the primary objective function. To compute the bottom endpoint of the nondominated frontier, i.e., (y_1^I, y_2^N), the primary objective function is y_1 and the secondary objective function is y_2. Hence, $y_1^I = \max_{\boldsymbol{y}\in\mathcal{Y}} y_1$ and $y_2^N = \max_{\substack{\boldsymbol{y}\in\mathcal{Y} \\ y_1=y_1^I}} y_2$. To compute the top endpoint of the nondominated frontier, i.e., (y_1^N, y_2^I), the primary objective function is y_2 and the secondary objective function is y_1. Hence, $y_2^I = \max_{\boldsymbol{y}\in\mathcal{Y}} y_2$ and $y_1^N = \max_{\substack{\boldsymbol{y}\in\mathcal{Y} \\ y_2=y_2^I}} y_1$.



(a) Boxes to remove (b) Top box (c) Bottom box

Figure 5.2: An illustration of new boxes after finding a nondominated point

It is evident that after computing the ideal and nadir points, one can claim that the entire nondominated frontier of the multi-objective optimization counterpart of an MIBL-MMP must be in the box, denoted by $\text{Box}(\boldsymbol{y}^N, \boldsymbol{y}^I)$, between the nadir and ideal points

(as shown in Figure 5.1b). Now, suppose that by searching $\text{Box}(\boldsymbol{y}^N, \boldsymbol{y}^I)$ we find a *weakly* nondominated point denoted by $\bar{\boldsymbol{y}}$. We call $\bar{\boldsymbol{y}}$ a weakly nondominated point if there exists no feasible point $\bar{\boldsymbol{y}}' \in \mathcal{Y}$ that can strictly dominate $\bar{\boldsymbol{y}}$, i.e., $\bar{y}_1 < \bar{y}'_1$ and $\bar{y}_2 < \bar{y}'_2$. An illustration of a weakly nondominated point can be found in Figure 5.2a. From the figure it is clear that $\bar{\boldsymbol{y}}$ is not a nondominated point as there is a point slightly above it. Observe too that after finding a weakly nondominated point two boxes can be removed from $\text{Box}(\boldsymbol{y}^N, \boldsymbol{y}^I)$ as shown in Figure 5.2a. Specifically, $\text{Box}(\boldsymbol{y}^N, \bar{\boldsymbol{y}})$ can be fully removed because all the points in that box are dominated by $\bar{\boldsymbol{y}}$ and dominated points cannot be optimal for an MIBL-MMP based on Proposition 5.1. Also, $\text{Box}(\bar{\boldsymbol{y}}, \boldsymbol{y}^I)$ except its two sides that intersect in $\bar{\boldsymbol{y}}$ can be removed because there cannot exist any feasible point within the box due to the fact that $\bar{\boldsymbol{y}}$ is a weakly nondominated point.

Another important property of working with $p = 2$ is that after removing $\text{Box}(\boldsymbol{y}^N, \bar{\boldsymbol{y}})$ and $\text{Box}(\bar{\boldsymbol{y}}, \boldsymbol{y}^I)$, the remaining region which actually contains the entire nondominated frontier is basically two boxes (referred to as the top box and bottom box) that are shown in Figures 5.2b and 5.2c. We note that this observation does not hold for $p > 2$, i.e., the remaining region is not two boxes anymore and instead would be a complex volume in the p-dimensional space. We also observe from the figures that the ideal and nadir points of the portion of the nondominated frontier trapped in each box can be bounded by using $\boldsymbol{y}^I$, $\boldsymbol{y}^N$, and $\bar{\boldsymbol{y}}$. This means that a point can be constructed based on $\boldsymbol{y}^I$, $\boldsymbol{y}^N$, and $\bar{\boldsymbol{y}}$ to bound its nadir point from below. Similarly, a point can be constructed based on $\boldsymbol{y}^I$, $\boldsymbol{y}^N$, and $\bar{\boldsymbol{y}}$ to bound its ideal point from the above. For example, in Figure 5.2c, $(\bar{y}_1, y_2^N)$ is a lower bound point for the nadir point of the portion of the nondominated frontier trapped in the box and $(y_1^I, \bar{y}_2)$ is an upper bound point for its ideal point. This observation plays a key role in the branching mechanism embedded in the search tree of our proposed algorithm.

In specific, the search tree of our proposed algorithm contains only boxes. Each box has a lower bound point, denoted by $\boldsymbol{y}^l$, and an upper bound point, denoted by $\boldsymbol{y}^u$. The lower and upper bound points simply bound the ideal and the nadir points of the portion of

the nondominated frontier trapped in $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$. The main property of $\boldsymbol{y}^l$ and $\boldsymbol{y}^u$ is that by construction $\boldsymbol{y}^u \geqslant \boldsymbol{y}^I$ and $\boldsymbol{y}^l \leqslant \boldsymbol{y}^N$ where $\boldsymbol{y}^I$ and $\boldsymbol{y}^u$ denote the ideal and nadir points of the portion of the nondominated frontier trapped in $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$. When exploring a box $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$, our algorithm attempts to find a weakly nondominated point within the box. It then creates two new boxes (if needed) using a process similar to the above-mentioned process with some slight changes that will be explained later in Section 5.3.3.

The last concept that plays a key role in our proposed method is McCormick envelopes/relaxation of $y_1 y_2$ [156]. Let w be a continuous variable representing the value of $y_1 y_2$. The value of w in $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$ is bounded by the following four linear inequalities (referred to as McCormick envelopes/relaxation)

$$w \leqslant y_1 y_2^u + y_1^l y_2 - y_1^l y_2^u \tag{5.4}$$

$$w \leqslant y_1^u y_2 + y_1 y_2^l - y_1^u y_2^l \tag{5.5}$$

$$w \geqslant y_1^u y_2 + y_1 y_2^u - y_1^u y_2^u \tag{5.6}$$

$$w \geqslant y_1^l y_2 + y_1 y_2^l - y_1^l y_2^l \tag{5.7}$$

$$w \geqslant 0. \tag{5.8}$$

Constraints (5.6) and (5.7) are not needed in this study since in an MIBL-MMP, we seek to maximize $y_1 y_2$. Thus, in the remaining of this section, we drop these two constraints. McCormick relaxation can be improved at the cost of introducing some additional binary variables and constraints. The improved version is referred to as piecewise McCormick relaxation [157, 158]. In the piecewise McCormick relaxation, the box $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$ should be partitioned either horizontally or vertically into k boxes where $k \in \{0, 1, 2, \dots\}$ is a user-defined parameter. For example, $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$, can be horizontally partitioned into $\text{Box}(\boldsymbol{y}^{l,1}, \boldsymbol{y}^{u,1})$, $\dots$, $\text{Box}(\boldsymbol{y}^{l,k}, \boldsymbol{y}^{u,k})$ where for each $j \in \{1, \dots, k\}$ we define

$$y_1^{l,j} := y_1^l, \quad y_2^{l,j} := y_2^l + \big[(j-1) \times \frac{y_2^u - y_2^l}{k}\big],$$

and

$$y_1^{u,j} := y_1^u, \quad y_2^{u,j} := y_2^l + \big[j \times \frac{y_2^u - y_2^l}{k}\big].$$

Note that in this study, if $k = 0$ then the piecewise McCormick relaxation automatically becomes inactive and will not be used. After creating the boxes, the McCormick envelopes of each (smaller) box will be added. For each box $j \in \{1, \ldots, k\}$, one binary variable, denoted by q_j, will be defined and some constraints and non-negative variables will be added to ensure that exactly one box and its corresponding McCormick envelopes can be active at any time. With this in mind, the piecewise McCormick relaxation of $y_1 y_2$ can be stated as

$$w \leqslant \sum_{j=1}^{k} y_{1,j} y_2^{u,j} + y_1^{l,j} y_{2,j} - y_1^{l,j} y_2^{u,j} q_j \tag{5.9}$$

$$w \leqslant \sum_{j=1}^{k} y_1^{u,j} y_{2,j} + y_{1,j} y_2^{l,j} - y_1^{u,j} y_2^{l,j} q_j \tag{5.10}$$

$$\sum_{j=1}^{k} q_j = 1 \tag{5.11}$$

$$y_1^{l,j} q_j \leqslant y_{1,j} \leqslant y_1^{u,j} q_j \qquad \forall j = 1, 2, \ldots, k \tag{5.12}$$

$$y_2^{l,j} q_j \leqslant y_{2,j} \leqslant y_2^{u,j} q_j \qquad \forall j = 1, 2, \ldots, k \tag{5.13}$$

$$y_1 = \sum_{j=1}^{k} y_{1,j} \tag{5.14}$$

$$y_2 = \sum_{j=1}^{k} y_{2,j} \tag{5.15}$$

$$q_j \in \{0, 1\} \qquad \forall j = 1, 2, \ldots, k \tag{5.16}$$

$$y_{1,j}, y_{1,j} \geqslant 0 \qquad \forall j = 1, 2, \ldots, k \tag{5.17}$$

$$w \geqslant 0 \tag{5.18}$$

where $y_{1,1},\dots y_{1,k}$ are additional non-negative variables that exactly one of them takes the value of y_1 and the remaining ones take the value of zero based on Constraints (5.11)-(5.15). Similarly, $y_{2,1},\dots y_{2,k}$ are additional non-negative variables that exactly one of them takes the value of y_2 and the remaining ones take the value of zero based on Constraints (5.11)-(5.15). In the remaining of this paper, for the sake of convenience, we use the following notation to refer to the piecewise McCormick relaxation,

$$\textsc{McCormick}(\boldsymbol{y^l}, \boldsymbol{y^u}, k) := \{w \in \mathbb{R} : (5.9) - (5.18)\}.$$

5.3 Algorithmic Components of CSBnC

In this section, we introduce the key components/operations of our proposed CSBnC algorithm to solve an MIBL-MMP. The algorithm builds upon a search tree structure following a branch-and-cut scheme, by exploiting the structure of the nondominated frontier of the multi-objective optimization counterpart of the MIBL-MMP. The latter is constructed by taking each multiplicative term as an independent objective function. Similar to any branch-and-cut algorithm, our proposed method contains four key components: dual bound finder, primal bound finder, branching scheme, and cut generator. The novelty of our proposed approach comes directly from the novelty of the first three components while the last one is adopted from the work of [101]. Next, we provide the details of each component.

5.3.1 Dual Bound Finder

The search tree of our proposed method contains boxes that need to be explored. Our proposed method employs two approaches for computing dual (upper) bound for a $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$. The first approach is trivial and comes from this observation that by construction $\boldsymbol{y}^u \geqslant \boldsymbol{y}^I$ where $\boldsymbol{y}^I$ is the ideal point of the portion of the nondominated frontier trapped in $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$. This immediately implies that $y_1^u y_1^u$ must be a dual bound for the best objective value of the MIBL-MMP that can be found within the $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$. The second or

the main mechanism for computing a dual bound is built based on the piecewise McCormick relaxation of $y_1 y_2$. Specifically, in order to find a dual bound, one can solve the following optimization problem,

$$(\hat{w}, \hat{\boldsymbol{y}}) \in \arg\max \big\{ w : w \in \textsc{McCormick}(\boldsymbol{y^l}, \boldsymbol{y^u}, k^D) \ \wedge \ \boldsymbol{y} \in \mathcal{Y} \big\},$$

where k^D is a user defined parameter showing the number of pieces in the piecewise McCormick relaxation. Observe that by construction $\hat{w}$ provides a dual bound for the best objective value of the MIBL-MMP that can be found within the $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$. For convenience, we denote the second/main operation (i.e., the above-mentioned optimization problem) for finding a dual bound by $\textsc{Dual}(\boldsymbol{y}^l, \boldsymbol{y}^u, k^D)$ in this study. Next, we will make two remarks. The first one is that setting $k^D = 0$ would make the above optimization problem irrelevant. Thus, in that case, $\textsc{Dual}(\boldsymbol{y}^l, \boldsymbol{y}^u, k^D)$ simply reports $\hat{w} = y_1^u y_2^u$ and $\hat{\boldsymbol{y}} = \mathbf{0}$ to show that the operation has been irrelevant. The second remark is that if $\textsc{Dual}(\boldsymbol{y}^l, \boldsymbol{y}^u, k^D)$ is infeasible then it reports $\hat{w} = \text{null}$ and $\hat{\boldsymbol{y}} = \textbf{null}$.

5.3.2 Primal Bound Finder

Similar to computing dual bounds, our proposed algorithm employs two approaches for computing primal (lower) bound for a $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$. The first approach is computationally free as it simply uses the outcome of $\textsc{Dual}(\boldsymbol{y}^l, \boldsymbol{y}^u, k^D)$. Specifically, we know that $\hat{\boldsymbol{y}} \in \mathcal{Y}$ by construction. Therefore, $\hat{y}_1 \hat{y}_2$ must provide a primal bound. The second or the main mechanism for computing a primal bound simply seeks to find a feasible point that has the minimum Chebychev distance from $\boldsymbol{y}^u$ within the $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$. Specifically, in order to find a primal bound, one can solve the following optimization problem,

$$(\bar{z}, \bar{\boldsymbol{y}}) \in \arg\min \big\{ z \in \mathbb{R} : \ z \geqslant 0 \ \wedge \ y_1^u - y_1 \leqslant z \ \wedge \ y_2^u - y_2 \leqslant z \ \wedge$$
$$\boldsymbol{y} \in \mathcal{Y} \ \wedge \ \boldsymbol{y} \in \text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u) \ \wedge$$

$$w \in \text{McCormick}(\boldsymbol{y^l}, \boldsymbol{y^u}, k^P) \ \wedge \ \text{Glb} \leqslant w\big\},$$

where Glb denotes the best global primal (lower) bound known by the proposed algorithm for the optimal objective value of MIBL-MMP at the time of exploring $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$. Also, k^P is a user defined parameter showing the number of pieces in the piecewise McCormick relaxation. Finally, continuous variable z is defined to capture the Chebychev distance of any feasible point from $\boldsymbol{y}^u$. The first three constraints, i.e., $z \geqslant 0$, $y_1^u - y_1 \leqslant z$, and $y_2^u - y_2 \leqslant z$, ensure that z will take the value of Chebychev distance when being minimized. Constraints $\boldsymbol{y} \in \mathcal{Y}$ and $\boldsymbol{y} \in \text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$ ensure that the point that will be found is feasible and located in $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$. The latter means that $y_1 \geqslant y_1^l$, $y_2 \geqslant y_2^l$, $y_1 \leqslant y_1^u$, and $y_2 \leqslant y_2^u$. Finally, constraints $w \in \text{McCormick}(\boldsymbol{y}^l, \boldsymbol{y}^u, k^P)$ and $\text{Glb} \leqslant w$ ensure that the relaxed value of $y_1 y_2$, i.e., w, should be at least as good as the best global primal bound known.

For convenience, we denote the second/main operation for finding a primal bound by $\text{Primal}(\boldsymbol{y}^l, \boldsymbol{y}^u, k^P, \text{Glb})$ in this study. Next, we will make three remarks. The first one is that if users set $k^P = 0$ then $w \in \text{McCormick}(\boldsymbol{y}^l, \boldsymbol{y}^u, k^P)$ and $G_{LB} \leqslant w$ will become irrelevant, in which case, those constraints will be removed when calling $\text{Primal}(\boldsymbol{y}^l, \boldsymbol{y}^u, k^P, \text{Glb})$. The second remark is that if $k^P > 0$ then the constraint $\boldsymbol{y} \in \text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$ is redundant because it naturally exists in $w \in \text{McCormick}(\boldsymbol{y}^l, \boldsymbol{y}^u, k^P)$ – see Constraints (5.11)-(5.15). Therefore, in that case, $\boldsymbol{y} \in \text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$ will be removed. Finally, the third remark is that if $\text{Primal}(\boldsymbol{y}^l, \boldsymbol{y}^u, k^P, \text{Glb})$ is infeasible then it reports $\bar{z} = \text{null}$ and $\bar{\boldsymbol{y}} = \textbf{null}$.

5.3.3 Branching Scheme

The branching scheme developed in this study is similar to the process previously explained in Section 5.2.2 about Figure 5.2 but with some slight changes. For now, we ignore the slight changes and assume that the branching scheme is precisely the same as the one presented in Section 5.2.2. Specifically, based on our discussions in Section 5.2.2, by finding a weakly nondominated point, denoted by $\bar{\boldsymbol{y}}$ in $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$, we replace $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$

with two new child nodes/boxes including the top box denoted by $\text{Box}(\boldsymbol{y}^{l,t}, \boldsymbol{y}^{u,t})$ and the bottom box denoted by $\text{Box}(\boldsymbol{y}^{l,b}, \boldsymbol{y}^{u,b})$ where $\boldsymbol{y}^{l,t} := (y_1^l, \bar{y}_2)$, $\boldsymbol{y}^{u,t} := (\bar{y}_1, y_2^u)$, $\boldsymbol{y}^{l,b} := (\bar{y}_1, y_2^l)$, and $\boldsymbol{y}^{u,b} := (y_1^u, \bar{y}_2)$. Suppose that all the boxes that currently exist in the search tree are created based on our discussions in Section 5.2.2. The following proposition is helpful.

Proposition 5.2. *Let $(\bar{z}, \bar{\boldsymbol{y}})$ be the (optimal) outcome of calling* PRIMAL$(\boldsymbol{y}^l, \boldsymbol{y}^u, k^P,$ GLB$)$. *If $\bar{z} \neq$ null, i.e., the operation is feasible, then $\bar{\boldsymbol{y}}$ is a weakly nondominated point.*

Proof.

Suppose $\bar{\boldsymbol{y}}$ is not a weakly nondominated point. In that case, by definition, there must exist a feasible point $\boldsymbol{y}' \in \mathcal{Y}$ with the property that $\bar{y}_1 < y_1'$ and $\bar{y}_2 < y_2'$. We now show that $\boldsymbol{y}'$ must be feasible for PRIMAL$(\boldsymbol{y}^l, \boldsymbol{y}^u, k^P,$ GLB$)$. This can be obtained from following three observations. (1) By assumptions, we have that $\boldsymbol{y}' \in \mathcal{Y}$. (2) We must have that $\boldsymbol{y}' \in \text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$. This is because PRIMAL$(\boldsymbol{y}^l, \boldsymbol{y}^u, k^P,$ GLB$)$ guarantees that $\bar{\boldsymbol{y}} \in \text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$. Based on the branching scheme proposed in Section 5.2.2, for any (unexplored) box other than $\bar{\boldsymbol{y}} \in \text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$ in the search tree denoted by $\text{Box}(\widetilde{\boldsymbol{y}}^l, \widetilde{\boldsymbol{y}}^u)$ we will have either $\widetilde{y}_1^u \leqslant y_1^l$ or $\widetilde{y}_2^u \leqslant y_2^l$. This combined with the fact that (by definition) $\bar{y}_1 < y_1'$ and $\bar{y}_2 < y_2'$ imply that $\boldsymbol{y}' \in \text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$. (3) The piecewise McCormick relaxation value of $y_1' y_2'$ is not smaller than GLB. This is because PRIMAL$(\boldsymbol{y}^l, \boldsymbol{y}^u, k^P,$ GLB$)$ guarantees that the McCormick relaxation value of $\bar{y}_1 \bar{y}_2$ not to be smaller than GLB. Therefore, since by definition $\bar{y}_1 < y_1'$ and $\bar{y}_2 < y_2'$, the piecewise McCormick relaxation value of $y_1' y_2'$ cannot be smaller than GLB too.

From discussions given above, we know that $\boldsymbol{y}'$ is feasible for PRIMAL$(\boldsymbol{y}^l, \boldsymbol{y}^u, k^P,$ GLB$)$. Also, by definition, we know that $\bar{y}_1 < y_1'$ and $\bar{y}_2 < y_2'$. Therefore, the Chebychev distance of $\boldsymbol{y}'$ from $\bar{\boldsymbol{y}}^u$ should be strictly smaller than $\bar{z}$. So, $\bar{\boldsymbol{y}}$ cannot be optimal (a contradiction). □

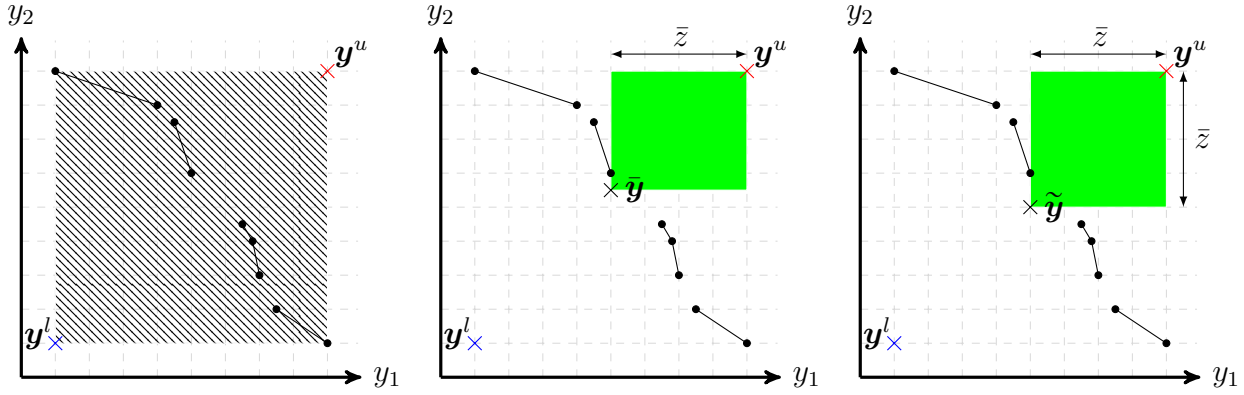
Based on Proposition 5.2, by calling PRIMAL$(\boldsymbol{y}^l, \boldsymbol{y}^u, k^P,$ GLB$)$, a weakly nondominated point (if any) will be found in $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$. So, the branching scheme proposed in Section 5.2.2 can be used when exploring each box based on the outcome of PRIMAL$(\boldsymbol{y}^l, \boldsymbol{y}^u, k^P,$ GLB$)$. Although the branching scheme is valid, it suffers from one major weakness. That is when

exploring $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$, the same box will be created again as a child node/box if the weakly nondominated point $\bar{\boldsymbol{y}}$ obtained is equal to either (y_1^l, y_2^u) or (y_1^u, y_2^l) which are two other corner points of $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$. So, the algorithm may get trapped in an infinite loop. As mentioned in Section 5.2.1, the existing algorithms such as the one proposed by [151] uses no-good constraints to break the infinite loop (which is costly). However, in this study, we use a property of $\text{PRIMAL}(\boldsymbol{y}^l, \boldsymbol{y}^u, k^P, \text{GLB})$, described in the following proposition, to modify our branching scheme.

Proposition 5.3. *Let $(\bar{z}, \bar{\boldsymbol{y}})$ be the (optimal) outcome of calling $\text{PRIMAL}(\boldsymbol{y}^l, \boldsymbol{y}^u, k^P, \text{GLB})$. If $\bar{z} \neq$ null, i.e., the operation is feasible, then there cannot exist a feasible point $\boldsymbol{y}'$ for operation $\text{PRIMAL}(\boldsymbol{y}^l, \boldsymbol{y}^u, k^P, \text{GLB})$ with $y'_1 > y_1^u - \bar{z}$ and $y'_2 > y_2^u - \bar{z}$.*

Proof.

Suppose not, i.e., there exists a feasible point $\boldsymbol{y}'$ for operation $\text{PRIMAL}(\boldsymbol{y}^l, \boldsymbol{y}^u, k^P, \text{GLB})$ with $y'_1 > y_1^u - \bar{z}$ and $y'_2 > y_2^u - \bar{z}$. This implies that $\max\{y_1^u - y'_1, y_2^u - y'_2\} < \bar{z}$, i.e., the Chebychev distance of $\boldsymbol{y}'$ from $\bar{\boldsymbol{y}}^u$ is strictly smaller than $\bar{z}$. Therefore, $\bar{\boldsymbol{y}}$ cannot be optimal (a contradiction). □



(a) $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$ (b) Point $\bar{\boldsymbol{y}}$ (c) Point $\tilde{\boldsymbol{y}}$

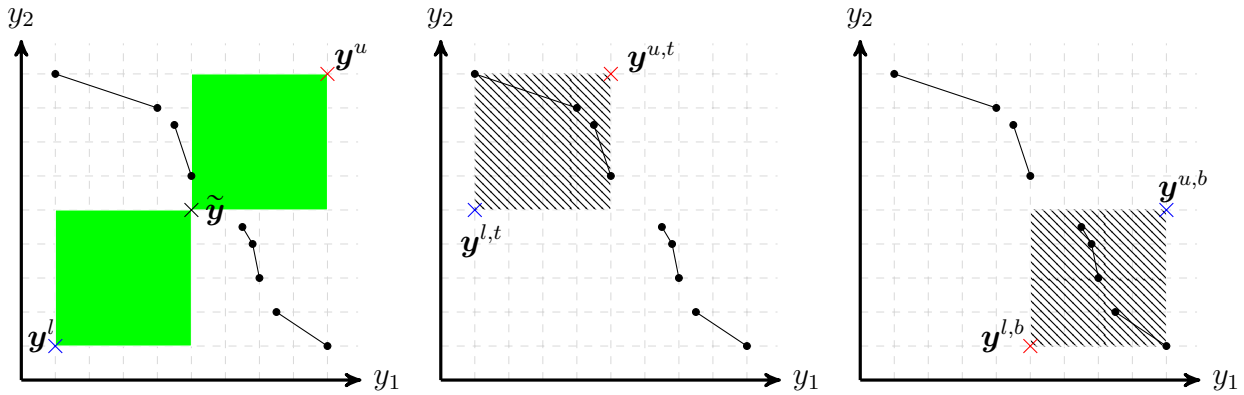
Figure 5.3: An illustration of the impact of Proposition 5.3

An illustration of Proposition 5.3 can be found in Figure 5.3. Suppose that we are exploring $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$ as shown in Figure 5.3a. Suppose that when exploring $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$,

we find the weakly nondominated point $\bar{\boldsymbol{y}}$ with the Chebychev distance of $\bar{z}$ from $\boldsymbol{y}^u$ as shown in Figure 5.3b. From discussions given in Section 5.2.2, we know that since $\bar{\boldsymbol{y}}$ is weakly nondominated, there cannot exist any feasible point in $\text{Box}(\bar{\boldsymbol{y}}, \boldsymbol{y}^u)$ (except its two sides intersecting at $\bar{\boldsymbol{y}}$) as shown in Figure 5.3b. Proposition 5.3 shows that because we have used $\textsc{Primal}(\boldsymbol{y}^l, \boldsymbol{y}^u, k^P, \textsc{Glb})$ to compute $(\bar{z}, \bar{\boldsymbol{y}})$ we can change the green box shown in Figure 5.3b into a perfect square. Specifically, let $\widetilde{\boldsymbol{y}}$ be a point in the criterion space such that

$$\widetilde{y}_1 := y_1^u - \bar{z} \text{ and } \widetilde{y}_2 := y_2^u - \bar{z}.$$

Proposition 5.3 shows that optimal solutions of an MIBL-MMPs cannot be in $\text{Box}(\widetilde{\boldsymbol{y}}, \boldsymbol{y}^u)$ (except possibly on its two sides intersecting at $\widetilde{\boldsymbol{y}}$) as shown in Figure 5.3c. Note that, by construction, we always have $\text{Box}(\bar{\boldsymbol{y}}, \boldsymbol{y}^u) \subseteq \text{Box}(\widetilde{\boldsymbol{y}}, \boldsymbol{y}^u)$ and consequently $\text{Box}(\boldsymbol{y}^l, \widetilde{\boldsymbol{y}}) \subseteq \text{Box}(\boldsymbol{y}^l, \bar{\boldsymbol{y}})$.



(a) Boxes to remove (b) Top box (c) Bottom box

Figure 5.4: An illustration of the modified branching scheme

Overall, Proposition 5.3 is helpful because it suggests that instead of branching based on $\bar{\boldsymbol{y}}$, we can branch based on $\widetilde{\boldsymbol{y}}$ as shown in Figure 5.4. Specifically, based on the outcome of $\textsc{Primal}(\boldsymbol{y}^l, \boldsymbol{y}^u, k^P, \textsc{Glb})$, the algorithm will generate $\widetilde{\boldsymbol{y}}$. We then remove two boxes $\text{Box}(\boldsymbol{y}^l, \widetilde{\boldsymbol{y}})$ and $\text{Box}(\widetilde{\boldsymbol{y}}, \boldsymbol{y}^u)$ from $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$ as shown in Figure 5.4a. Note that $\text{Box}(\widetilde{\boldsymbol{y}}, \boldsymbol{y}^u)$ can be removed based on Proposition 5.3. Also, $\text{Box}(\boldsymbol{y}^l, \widetilde{\boldsymbol{y}})$ can be removed since we know that $\text{Box}(\boldsymbol{y}^l, \widetilde{\boldsymbol{y}}) \subseteq \text{Box}(\boldsymbol{y}^l, \bar{\boldsymbol{y}})$ and $\text{Box}(\boldsymbol{y}^l, \bar{\boldsymbol{y}})$ can be removed based on our discussions in

Section 5.2.2. We can then replace $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$ with two new child nodes/boxes: the top box denoted by $\text{Box}(\boldsymbol{y}^{l,t}, \boldsymbol{y}^{u,t})$ and the bottom box denoted by $\text{Box}(\boldsymbol{y}^{l,b}, \boldsymbol{y}^{u,b})$ where $\boldsymbol{y}^{l,t} := (y_1^l, \widetilde{y}_2)$, $\boldsymbol{y}^{u,t} := (\widetilde{y}_1, y_2^u)$, $\boldsymbol{y}^{l,b} := (\widetilde{y}_1, y_2^l)$, and $\boldsymbol{y}^{u,b} := (y_1^u, \widetilde{y}_2)$. As final remark, we note that branching based on $\widetilde{\boldsymbol{y}}$ would never result in a child node represented by $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$. This is because $\text{Box}(\widetilde{\boldsymbol{y}}, \boldsymbol{y}^u)$ is always a square while $\text{Box}(\bar{\boldsymbol{y}}, \boldsymbol{y}^u)$ would be a line segment if $\bar{\boldsymbol{y}}$ is equal to either (y_1^l, y_2^u) or (y_1^u, y_2^l).

5.3.4 Cut Generator

During the course of our algorithm, whenever we find a feasible point $\boldsymbol{y}' \in \mathcal{Y}$ with $y_1', y_2' > 0$, we add a cut known as the *hypotenuse cut* [128, 101] to restrict $\mathcal{Y}$. The hypotenuse cut can be stated as

$$\frac{y_1}{y_1'} + \frac{y_2}{y_2'} \geqslant 2, \tag{5.19}$$

and for convenience we use the notation $H(\boldsymbol{y}')$ to represent the hypotenuse cut. To understand why this cut is valid, consider the following set

$$T(\boldsymbol{y}') := \{\boldsymbol{y} \in \mathbb{R}^2 : y_1, y_2 \geqslant 0 \text{ and } \frac{y_1}{y_1'} + \frac{y_2}{y_2'} \leqslant 2\}.$$

The set $T(\boldsymbol{y}')$ basically represents a right-angle triangle in the criterion space where $\frac{y_1}{y_1'} + \frac{y_2}{y_2'} = 2$ is its hypotenuse. [101] proved that the (unique) optimal point of $\max_{\boldsymbol{y} \in T(\boldsymbol{y}')} y_1 y_2$ is $\boldsymbol{y}'$. Therefore, there is no need to search the triangle.

5.4 Criterion Space Branch-and-cut Algorithm

In this section, we explain the details of our proposed algorithm, CSBnC, which is shown in Algorithm 4. The algorithm maintains a queue (or search tree) of boxes and their corresponding *initial* upper (dual) bounds. The initial upper bound of each node is basically the best (minimum) of the upper bound obtained for its parent and its trivial upper bound, i.e., the first upper bound approach explained in Section 5.3.1. For the root node, since there

Algorithm 4: The CSBnC Algorithm

Inputs: A feasible instance of Problem (5.3), ε_1, ε_2, k^D, k^P
Queue.create(TREE) //creating an empty search tree//
$\boldsymbol{y}^l \leftarrow (-\infty, -\infty)$ and $\boldsymbol{y}^u \leftarrow (+\infty, +\infty)$ //computing the initial lower bound and upper bound points//
$y_1^u \leftarrow \max_{\boldsymbol{y}\in\mathcal{Y}} y_1$ and $y_2^l \leftarrow \max_{\substack{\boldsymbol{y}\in\mathcal{Y} \\ y_1=y_1^u}} y_2$
$y_2^u \leftarrow \max_{\boldsymbol{y}\in\mathcal{Y}} y_2$ and $y_1^l \leftarrow \max_{\substack{\boldsymbol{y}\in\mathcal{Y} \\ y_2=y_2^u}} y_1$
GLB $\leftarrow \max\{y_1^u y_2^l, y_1^l y_2^u\}$ and GUB $\leftarrow y_1^u y_2^u$ //computing the initial global lower and upper bounds//
if GLB $= y_1^u y_2^l$ **then**
  $\boldsymbol{y}^* \leftarrow (y_1^u, y_2^l)$ //initializing the best solution found//
else
  $\boldsymbol{y}^* \leftarrow (y_1^l, y_2^u)$ //initializing the best solution found//
TREE.*add*$\big((\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u), \text{GUB})\big)$ //creating the first node of the search tree//
while *not Queue.empty*(TREE) *&* GUB $-$ GLB $\geqslant \varepsilon_1$ *&* $\frac{\text{GUB}-\text{GLB}}{\text{GUB}} \geqslant \varepsilon_2$ **do**
  TREE.*PopOut*$\big((\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u), \mathsf{ub})\big)$ //popping out a node from the search tree for exploration//
  $(\hat{w}, \hat{\boldsymbol{y}}) \leftarrow$ DUAL$(\boldsymbol{y}^l, \boldsymbol{y}^u, k^D)$ //computing a dual bound value for the node//
  if $\hat{w} \neq$ *null* **then**
    if $\hat{\boldsymbol{y}} > \boldsymbol{0}$ **then**
      Add the cut $H(\hat{\boldsymbol{y}})$ to $\mathcal{Y}$ //generating and adding the hypotenuse cut//
    if $\hat{y}_1\hat{y}_2 >$ GLB **then**
      GLB $\leftarrow \hat{y}_1\hat{y}_2$
      $\boldsymbol{y}^* \leftarrow \hat{\boldsymbol{y}}$ //updating the best solution found//
    if $\hat{w} -$ GLB $\geqslant \varepsilon_1$ *&* $\frac{\hat{w}-\text{GLB}}{\hat{w}} \geqslant \varepsilon_2$ **then**
      $(\bar{z}, \bar{\boldsymbol{y}}) \leftarrow$ PRIMAL$(\boldsymbol{y}^l, \boldsymbol{y}^u, k^P,$ GLB$)$ //computing a primal bound value for the node//
      if $\bar{z} \neq$ *null* **then**
        if $\bar{\boldsymbol{y}} > \boldsymbol{0}$ **then**
          Add the cut $H(\bar{\boldsymbol{y}})$ to $\mathcal{Y}$ //generating and adding the hypotenuse cut//
        if $\bar{y}_1\bar{y}_2 >$ GLB **then**
          GLB $\leftarrow \bar{y}_1\bar{y}_2$
          $\boldsymbol{y}^* \leftarrow \bar{\boldsymbol{y}}$ //updating the best solution found//
        $\widetilde{\boldsymbol{y}} \leftarrow (y_1^u - \bar{z}, y_2^u - \bar{z})$ //computing the point required for decomposing the node//
        $\boldsymbol{y}^{l,t} \leftarrow (y_1^l, \widetilde{y}_2)$ and $\boldsymbol{y}^{u,t} \leftarrow (\widetilde{y}_1, y_2^u)$ //generating the top box/node//
        $\mathsf{ub}^t \leftarrow \min\{\hat{w}, y_1^{u,t} y_2^{u,t}\}$ //generating the initial dual bound for the top box/node//
        if $\boldsymbol{y}^{l,t} \leqslant \boldsymbol{y}^{u,t}$ *&* $\mathsf{ub}^t -$ GLB $\geqslant \varepsilon_1$ *&* $\frac{\mathsf{ub}^t-\text{GLB}}{\mathsf{ub}^t} \geqslant \varepsilon_2$ **then**
          TREE.*add*$\big((\text{Box}(\boldsymbol{y}^{l,t}, \boldsymbol{y}^{u,t}), \mathsf{ub}^t)\big)$ //adding the top box/node to the tree//
        $\boldsymbol{y}^{l,b} \leftarrow (\widetilde{y}_1, y_2^l)$ and $\boldsymbol{y}^{u,b} \leftarrow (y_1^u, \widetilde{y}_2)$ //generating the bottom box/node//
        $\mathsf{ub}^b \leftarrow \min\{\hat{w}, y_1^{u,b} y_2^{u,b}\}$ //generating the initial dual bound for the bottom box/node//
        if $\boldsymbol{y}^{l,b} \leqslant \boldsymbol{y}^{u,b}$ *&* $\mathsf{ub}^b -$ GLB $\geqslant \varepsilon_1$ *&* $\frac{\mathsf{ub}^b-\text{GLB}}{\mathsf{ub}^b} \geqslant \varepsilon_2$ **then**
          TREE.*add*$\big((\text{Box}(\boldsymbol{y}^{l,b}, \boldsymbol{y}^{u,b}), \mathsf{ub}^b)\big)$ //adding the bottom box/node to the tree//
  Update GUB
return $\boldsymbol{y}^*$

is no parent node, its initial upper bound is basically its trivial upper bound. The algorithm also maintains a global lower (primal) bound GLB and a global upper (dual) bound GUB. The algorithm continues to explore the queue as long as it is non-empty and if the absolute and relative optimality gaps are not smaller than thresholds ε_1 and ε_2, respectively (where $\varepsilon_1, \varepsilon_2 \geqslant 0$ are user-defined parameters) (Line 12). Otherwise, it returns the best feasible point found during its course denoted by $\boldsymbol{y}^*$ as an optimal point (Line 39).

In the initialization phase, the algorithm first computes the endpoints of the nondominated frontier, i.e., (y_1^u, y_2^l) and (y_1^l, y_2^u), through lexicographic operations (Lines 3-5). The endpoints are important since they can create the first box denoted $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$. As mentioned in Section 5.2.2, for the first box, $\boldsymbol{y}^l$ represents the nadir point of the entire nondominated frontier. Similarly, $\boldsymbol{y}^u$ represents the ideal point of the entire nondominated frontier. Therefore, the algorithm uses $\boldsymbol{y}^u$ to initialize GUB, i.e., it sets GUB to $y_1^u y_2^u$ (Line 6). Moreover, since the endpoints are feasible points, the algorithm uses the objective value of the best endpoint to initialize GLB (Line 6). The best endpoint will be also used to initialize $\boldsymbol{y}^*$ (Lines 7-10). The algorithm then initializes the queue, by the node $\big(\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u), \text{GUB}\big)$ (Line 11). Next, we describe the procedure that the algorithm uses to explore each node of the search tree.

In each iteration, the algorithm pops out one of the nodes of the queue (Line 13) and denotes it by $\big(\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u), \mathsf{ub}\big)$. As an aside, we note that in this study, we use the best bound strategy to pick a node from the search tree. Also note that when a node in the queue is picked and popped out then it is fully removed from the queue. The algorithm then immediately calls $\text{DUAL}(\boldsymbol{y}^l, \boldsymbol{y}^u, k^D)$ to compute $(\hat{w}, \hat{\boldsymbol{y}})$, (Line 14). Next, the algorithm checks whether $\hat{w} = \text{null}$ or not (Line 15). Note that if $\bar{z} = \text{null}$ then the dual bound finder was not able to find a feasible solution and so there is no need to explore the node any further (since it is empty), in which case the iteration terminates. However, at the end of each iteration, GUB will be updated (Line 38), i.e., it will be set to the maximum value of upper (dual) bounds associated to nodes available in the search tree.

If $\hat{w} \neq$ null then the algorithm first attempts to add the cut $H(\hat{\boldsymbol{y}})$ to restrict $\mathcal{Y}$. The cut will be added if $\hat{\boldsymbol{y}} > \mathbf{0}$ (Lines 16-17). Afterwards the algorithm updates GLB and $\boldsymbol{y}^*$ if the objective value of $\hat{\boldsymbol{y}}$, i.e., $\hat{y}_1\hat{y}_2$, is better than GLB (Lines 18-20). Next, in Line 21, the algorithm checks whether there is hope to find better GLB in the current node, i.e., the algorithm checks whether the current node can be fathomed due to its corresponding local relative/absolute optimality gaps. If there is no hope then the algorithm stops the search in this iteration and simply updates GUB before starting the next iteration (Line 38). Otherwise, it calls PRIMAL$(\boldsymbol{y}^l, \boldsymbol{y}^u, k^P,$ GLB$)$ to compute $(\bar{z}, \bar{\boldsymbol{y}})$ (Line 22). Next, the algorithm checks whether $\bar{z} =$ null or not (Line 23). Note that if $\bar{z} =$ null then the primal bound finder was not able to find a (better) feasible solution and there is no need to explore the node any further (since it is empty). In such a case, the iteration terminates and again GUB will be updated before starting a new iteration (Line 38).

If $\bar{z} \neq$ null then the algorithm first attempts to add the cut $H(\bar{\boldsymbol{y}})$ to restrict $\mathcal{Y}$. The cut will be added if $\bar{\boldsymbol{y}} > \mathbf{0}$ (Lines 24-25). Afterwards, the algorithm updates GLB and $\boldsymbol{y}^*$ if the objective value of $\bar{\boldsymbol{y}}$, i.e., $\bar{y}_1\bar{y}_2$, is better than GLB (Lines 26-28). Finally, the algorithm applies the proposed branching scheme to complete the iteration. Specifically, it first computes $\widetilde{y}$ (Line 29). It then creates the top node/box by computing $\boldsymbol{y}^{l,t}$ and $\boldsymbol{y}^{u,t}$ (Line 30). Also, it computes the initial upper bound corresponding to the top node, denoted by ub^t (Line 31), by setting it to the minimum of two values: $\hat{w}$ (which is the outcome of applying the sophisticated dual bound finder on the parent node) and $y_1^{u,t}y_2^{u,t}$ (which is the trivial upper bound for the top node). The algorithm then adds the node $\big(\text{Box}(\boldsymbol{y}^{l,t}, \boldsymbol{y}^{u,t}), \mathsf{ub}^t\big)$ to the search tree given it is possible to find better primal bounds than GLB (Lines 32-33). Similar process will be followed to create and add the bottom node/box (Lines 34-37).

Next, we provide a few theoretical results to explain the convergence of the proposed algorithm. For convenience, we first provide a few notations. Recall that $\boldsymbol{y}^N$ and $\boldsymbol{y}^I$ denote the nadir and ideal points of the entire nondominated frontier. We define $L := y_1^I + y_2^I$ and $0 < S \leqslant \min_{\boldsymbol{y} \in \mathcal{Y}_N} y_1 + y_2$. Note that we must have $\min_{\boldsymbol{y} \in \mathcal{Y}_N} y_1 + y_2 > 0$ because otherwise

the optimal objective value of the MIBL-MMP is zero which violates our assumption given in Section 5.2. Overall, by assuming that $y_1^N + y_2^N > 0$, we can simply set $S = y_1^N + y_2^N$.

Proposition 5.4. *Let* $0 \leqslant \varepsilon_1 \leqslant \frac{S^2}{4}$. *When exploring Box*$(\boldsymbol{y}^l, \boldsymbol{y}^u)$ *using Algorithm 4, if child nodes are created then*

$$\bar{z} \geqslant \frac{L - \sqrt{L^2 - 4\varepsilon_1}}{2}.$$

Proof.

Based on Lines 32 and 36 of the proposed algorithm, in order to create any child node we must have

$$y_1^u y_2^u - \bar{y}_1 \bar{y}_2 \geqslant \varepsilon_1.$$

Since by construction,

$$y_1^u y_2^u - (y_1^u - \bar{z})(y_2^u - \bar{z}) \geqslant y_1^u y_2^u - \bar{y}_1 \bar{y}_2,$$

we have

$$y_1^u y_2^u - (y_1^u - \bar{z})(y_2^u - \bar{z}) \geqslant \varepsilon_1 \implies \bar{z}^2 - (y_1^u + y_2^u)\bar{z} + \varepsilon_1 \leqslant 0.$$

Observe that $\bar{z}^2 - (y_1^u + y_2^u)\bar{z} + \varepsilon_1$ is a strictly convex function of $\bar{z}$ since its second derivative is a positive constant, i.e., 1. So, it can take non-positive values only if $\bar{z}$ takes values between its roots:

$$\frac{(y_1^u + y_2^u) \pm \sqrt{(y_1^u + y_2^u)^2 - 4\varepsilon_1}}{2}.$$

Observe too that since $\varepsilon_1 \leqslant \frac{S^2}{4}$, we must have $(y_1^u + y_2^u)^2 - 4\varepsilon_1 \geqslant 0$. These two observations suggest that $\bar{z}$ is bounded below as follows,

$$\bar{z} \geqslant \frac{(y_1^u + y_2^u) - \sqrt{(y_1^u + y_2^u)^2 - 4\varepsilon_1}}{2} \geqslant \frac{L - \sqrt{L^2 - 4\varepsilon_1}}{2}.$$

□

Proposition 5.5. *If* $0 < \varepsilon_1 \leqslant \frac{S^2}{4}$ *then Algorithm 4 terminates after at most* $2^{T+1} - 1$ *iterations where*

$$T := \lfloor \frac{2(L - y_1^N - y_2^N)}{L - \sqrt{L^2 - 4\varepsilon_1}} \rfloor.$$

Proof.

Observe that the total distance between $\boldsymbol{y}^N$ and $\boldsymbol{y}^I$ over both axes is

$$(y_1^I - y_1^N) + (y_2^I - y_2^N) = L - y_1^N - y_2^N.$$

After each iteration, at most two child nodes will be created where in each of them the total distance reduces by $\bar{z}$ (based on Lines 29, 30, and 34). Observe too that based on Proposition 5.4, in order to create child nodes in each iteration, we must have that $\bar{z} \geqslant \frac{L - \sqrt{L^2 - 4\varepsilon_1}}{2}$. So, in order to make the total distance equal to zero, the algorithm needs to explore at most $2^{T+1} - 1$ nodes. □

5.5 A Computational Study

In this section, we conduct a computational study to evaluate and compare the performance of CSBnC with alternative approaches. To this end, first, we investigate the effect of each component of our algorithm and tune its hyperparameters, e.g., k^D and k^P. We then compare CSBnC with a state-of-the-art algorithm [151] and a commercial (mixed integer) SOCP solver, i.e., CPLEX 12.10. We implement the proposed algorithm in C++ and use CPLEX 12.10 to solve its corresponding single-objective mixed integer linear programs arising. All the computational experiments are done on a Dell PowerEdge R640 system with two Intel Xeon (Silver 4116) 2.1GHz 12-core processors, 128GB of memory, and RedHat Enterprise Linux 7.4 operating system, and using only a single thread (unless stated otherwise). For each experiment, a time limit of 3600 seconds is imposed in this study. Our instance generator and codes of this study can be found at https://github.com/Vahidmhn/MIBL-MMP.

5.5.1 Instances

In this computational study, a total number of 400 random MIBL-MMP instances are generated. The instances are divided over two classes, each containing 200 instances. The first class contains only mixed binary instances and the second class contains only mixed integer instances. In each class, exactly half of the variables are continuous. Each class includes 20 subclasses with different number of variables, $n \in \{1000, 1500, 2000, 2500, 3000\}$, and constraints $m = \alpha \times n$, with $\alpha \in \{1, 2, 3, 4\}$. We generate 10 random instances for each subclass. Specifically, for each instance, the entries of A, and vectors C and $\boldsymbol{d}$ are randomly selected from the set $\{1, 2, \ldots, 10\}$. We set the sparsity of the matrices A and C to 50%, i.e., we change 50% of the entries of each matrix to zero. For each $j \in \{1, \ldots, m\}$, we randomly select the value of b_j from the set $\{0, 1, \ldots, \sum_{j=1}^{n} a_{ij}\}$, where a_{ij} is the the entry of A at i-th row and j-th column.

5.5.2 Contribution of the Hypotenuse Cut $H(\bar{\boldsymbol{y}})$

In this section, we investigate the effect of the hypotenuse cut discussed in Section 5.3.4. In order to do so, we assume that $k^D = k^P = 0$ and explore the performance of CSBnC under two settings: activating or deactivating the cut generator. Note that since $k^D = 0$, the operation $\textsc{Dual}(\boldsymbol{y}^l, \boldsymbol{y}^u, k^D)$ is inactive (see Section 5.3.1). This implies that by activating the cut generator only $H(\bar{\boldsymbol{y}})$ can be added and no cut of the form $H(\hat{\boldsymbol{y}})$ can be generated. The performance of the algorithm under both settings for each class of instances can be found in Table 5.1. In the table, the numbers reported under columns labeled 'Time (sec.)' are averages of the solution times in seconds over 10 instances. Also, columns labelled "# of Improvements" show the number of instances that their solution times are improved after activating the cut generator. It is evident that activating the cut generator is crucial and can decrease the solution time of CSBnC between 53.54% and 91.17% for the mixed binary class and between 18.59% and 57.92% for the mixed integer class on average. More importantly, in *at least* 96.25% of all instances, the solution time has improved after activating the cut

generator. Note that we used the term 'at least' because only the instances that are solved to optimality within the limited time are counted.

Table 5.1: The impact of $H(\bar{\boldsymbol{y}})$ when $k^D = k^P = 0$

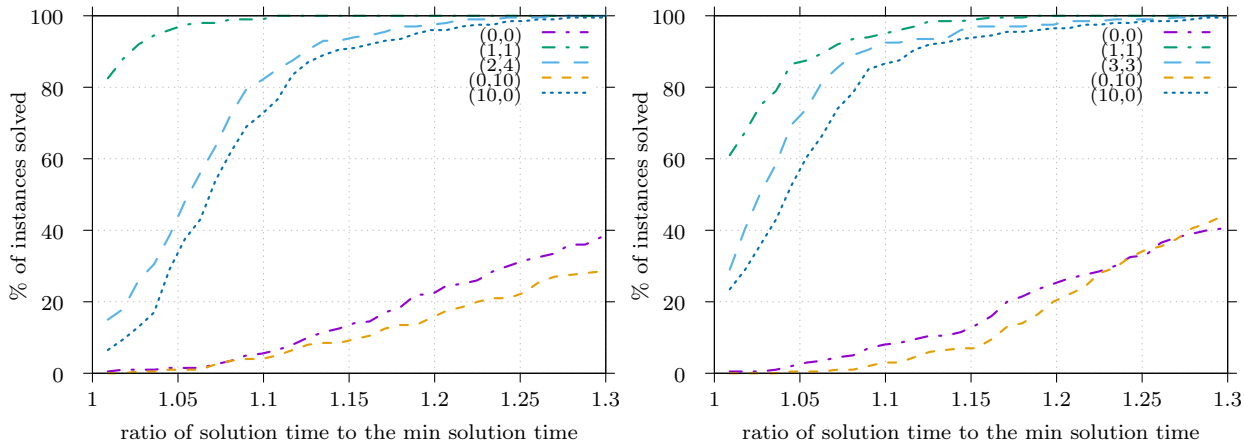
Subclass	**Mixed Binary Class**			**Mixed Integer Class**		
	Time (sec.)		**# of**	**Time (sec.)**		**# of**
	No cut	$H(\bar{\boldsymbol{y}})$	**Improvements**	No cut	$H(\bar{\boldsymbol{y}})$	**Improvements**
1	105.3	44.3	10	110.6	9.8	10
2	728.3	319.8	10	773.9	255.0	10
3	312.1	147.8	10	992.5	179.5	10
4	922.5	446.4	10	1,678.3	598.2	10
5	456.6	231.3	10	661.1	112.7	10
6	1,407.3	783.4	10	1,160.1	280.2	10
7	1,048.0	774.9	10	1,533.5	199.2	10
8	197.9	131.6	10	1,623.7	191.0	10
9	418.6	218.9	10	1,948.8	581.4	10
10	1,689.4	897.0	10	2,207.1	726.7	10
11	766.4	587.3	9	2,720.2	634.8	9
12	1,257.9	904.1	9	3,104.6	566.4	10
13	506.8	292.6	10	1,906.2	587.7	10
14	1,259.0	846.8	9	3,146.9	978.8	9
15	1,108.4	874.8	10	3,536.1	893.3	9
16	1,047.6	692.0	10	3,600.0	1,531.7	7
17	148.6	103.4	10	2,130.7	830.6	8
18	1,385.3	1,129.2	9	3,600.0	820.5	9
19	1,415.8	948.0	10	3,511.0	927.7	9
20	1,494.5	966.9	10	3,600.0	1,673.0	9
Average	883.8	567.0	9.8	2177.3	628.9	9.5

5.5.3 Contribution of the Hypotenuse Cut and Tuning k^D and k^P

In this section, we investigate the importance of $H(\hat{\boldsymbol{y}})$. Note that the cut $H(\hat{\boldsymbol{y}})$ can only be generated if $k^P > 0$. Therefore, in order to explore the importance of the cut, we first tune k^D and k^P. Since the cut generator turned out to be critical in the experiment described in the previous section, we keep it active during the tuning process. This implies that during the tuning process, the algorithm is allowed to add both $H(\bar{\boldsymbol{y}})$ and $H(\hat{\boldsymbol{y}})$. After tuning k^D and k^P, we deactivate $H(\hat{\boldsymbol{y}})$ to measure its impact compared to the case that both $H(\bar{\boldsymbol{y}})$ and $H(\hat{\boldsymbol{y}})$ are active.

In order to tune k^D and k^P, we assumed that $k^D, k^P \in \{0, 1, \ldots, 6\}$ and tested all 49 combinations. In addition to the 49 main scenarios, we considered two extreme scenarios $(k^D = 0, k^P = 10)$ and $(k^D = 10, k^P = 0)$. Recall that setting $k^P = 0$ and $k^D = 0$ would deactivate McCormick in the primal and dual finder operations, respectively. We ran CSBnC

on all 400 instances over all 51 scenarios for k^D and k^P. From the 49 main scenarios, we identified the best, second best, and the worst scenarios for each classes of instances. For the purpose of illustration, we only show the performance of these three scenarios and the two extreme scenarios in this section. To illustrate and compare the performance of these five scenarios, we plot the *performance profile* [127] of their solution times for each class. To construct the solution time performance profile, for each algorithm and for each instance, we first need to compute the ratio of the solution time of the algorithm on the instance and the minimum of the solution times of all algorithms on the instance. The solution time performance profile then shows the ratios on the horizontal axis and, on the vertical axis, for each algorithm, the percentage of instances with a ratio that is smaller than or equal to the ratio on the horizontal axis. Therefore, the values in the upper left-hand corner of the graph indicate the best performance.



(a) Mixed Binary Class (b) Mixed Integer Class

Figure 5.5: Performance profiles of different (k^D, k^P)

Figure 5.5 shows the performance profile for the five selected scenarios of (k^D, k^P) in each class of instances. The graph shows that $(0, 0)$ is one of the worst scenarios. The best scenario is $(1, 1)$ which implies that the best number of pieces for the piecewise McCormick relaxations is just one. Comparing the performance of scenarios $(0, 0)$ and $(1, 1)$ shows that activating (piecewise) McCormick relaxations have significant impacts on improving the solution times. The graph shows that around 60% of instances are solved at least

1.3 times faster under scenario $(1,1)$ compared to $(0,0)$ in both classes of instances. In fact based on our results, on average, the solution time of the proposed algorithm under scenario $(1,1)$ is 9.67 times smaller. This significant impact is mainly because if $k^D = 0$ then the operation $\textsc{Dual}(\boldsymbol{y}^l, \boldsymbol{y}^u, k^D)$ is fully inactive. Note that based on our discussions in Section 5.3.2, if $k^P = 0$ the operation $\textsc{Primal}(\boldsymbol{y}^l, \boldsymbol{y}^u, k^P, \textsc{Glb})$ is still active but its enhancement, i.e., piecewise McCormick relaxation, is disabled. So, the negative impact of $k^D = 0$ is significantly more important than the negative impact of $k^P = 0$.

Another interesting observation is that the optimal value of k^D and k^P is one, which is very small. At first glance, this may look surprising. However, this observation can be explained by the following two reasons: (1) For larger number of pieces, piecewise McCormick relaxation will involve more constraints and binary variables that can make the problem more challenging; and (2) The operation $\textsc{Dual}(\boldsymbol{y}^l, \boldsymbol{y}^u, k^D)$ is active as long as $k \geqslant 1$. If the nondominated frontier trapped in $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$ is so curvy that some parts of it is close to $\boldsymbol{y}^u$ then we expect that the dual bound obtained by $\textsc{Dual}(\boldsymbol{y}^l, \boldsymbol{y}^u, k^D)$ is not going to be significantly better than the trivial dual bound, i.e., $y_1^u y_2^u$. In such a case, it is expected that k^D to be set to small values to reduce the complexity. Moreover, if the nondominated frontier trapped in $\text{Box}(\boldsymbol{y}^l, \boldsymbol{y}^u)$ is not curvy (it is more like a line segment) then it is natural to set k^D to small values as more number of pieces does not have any advantage.

Now that the tuning process is complete we can evaluate the impact of the hypotenuse cut $H(\hat{\boldsymbol{y}})$. Table 5.2 shows the average solution time of the proposed algorithm per subclass when both hypotenuse cuts, i.e., $H(\bar{\boldsymbol{y}})$ and $H(\hat{\boldsymbol{y}})$, are active compared to the case where only $H(\bar{\boldsymbol{y}})$ is active. In the table, columns labelled "# of Improvements" show the number of instances that their solutions times are improved (per subclass) when both hypotenuse cuts are active. We observe that more than 75% of instances are being solved faster when both cuts are active. In fact on average an improvement of 19.03% can be achieved after activating $H(\hat{\boldsymbol{y}})$.

Table 5.2: The impact of $H(\hat{\boldsymbol{y}})$ when $k^D = k^P = 1$

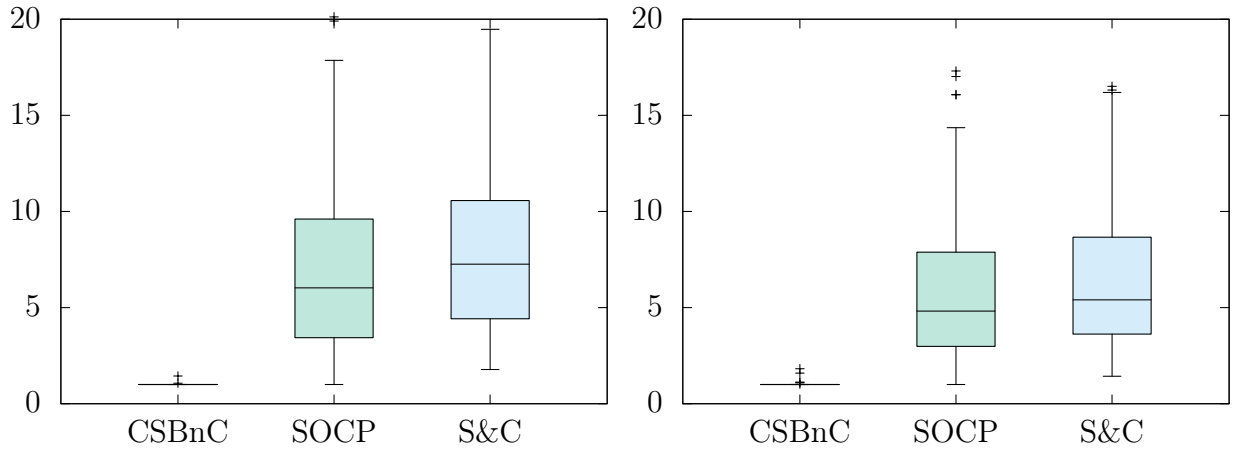
Subclass	**Mixed Binary Class**			**Mixed Integer Class**		
	Time (sec.)		**# of**	**Time (sec.)**		**# of**
	$H(\bar{\boldsymbol{y}})$	$H(\bar{\boldsymbol{y}}) + H(\hat{\boldsymbol{y}})$	**Improvements**	$H(\bar{\boldsymbol{y}})$	$H(\bar{\boldsymbol{y}}) + H(\hat{\boldsymbol{y}})$	**Improvements**
1	4.4	3.2	7	6.3	4.3	10
2	9.2	7.7	5	13.8	10.7	8
3	13.7	12.0	5	24.9	17.2	10
4	20.3	15.4	8	29.6	21.0	8
5	10.9	10.1	4	17.1	12.3	9
6	26.4	23.6	5	33.4	23.0	9
7	37.8	32.2	9	54.0	41.4	9
8	55.2	36.9	10	72.0	51.3	10
9	19.1	16.4	8	29.3	26.2	8
10	55.9	47.9	7	69.5	52.5	9
11	87.1	69.5	8	107.5	77.9	8
12	99.5	79.4	8	123.6	101.6	7
13	38.5	30.2	7	49.4	37.0	9
14	70.3	60.5	6	111.5	75.9	9
15	119.1	96.2	8	161.5	122.5	9
16	168.7	133.6	10	194.5	169.0	7
17	54.6	45.2	7	76.0	49.4	10
18	120.6	110.9	7	140.1	97.4	9
19	171.7	146.1	7	227.9	190.8	7
20	223.1	182.4	7	304.5	278.8	8
Average	70.3	58.0	7.1	92.3	73.0	8.7

5.5.4 CSBnC Algorithm Performance

In this section, we compare the best setting of CSBnC algorithm, i.e., $k^D = k^P = 1$ when all the cuts are active, with the existing state-of-the-art methods. Note that as mentioned in Section 5.2.1, a straightforward multi-objective optimization-based approach to solve an MIBL-MMP would be to first compute the nondominated frontier of its bi-objective optimization counterpart, and then pick the nondominated point that maximizes $y_1 y_2$ as the optimal point of the MIBL-MMP through a simple post-processing operation. The results show that CSBnC is significantly faster as CSBnC employs bounding mechanisms and cuts to avoid generating the entire nondominated frontier. In other words, CSBnC seeks to directly compute an optimal point of an MILP-MMP by searching only the parts of the nondominated frontier where an optimal point possibly lies.

In light of the above, in this section, we only compare the best setting of CSBnC algorithm with the (mixed integer) SOCP solver of CPLEX 12.10 and a state-of-the-art algorithm proposed by [151] which we refer to as S&C. Both CSBnC and S&C use CPLEX to solve optimization problems arising during the process of solving an MIBL-MMP. In all

previous experiments, we employed only a single thread to reduce the variability of solution times. However, in this experiment, we do the comparison under 4 settings: T1, $\cdots$, T4 where T1 and T4 show that one or four threads have been made available to CPLEX, respectively.



(a) Mixed Binary Class

(b) Mixed Integer Class

Figure 5.6: Box plots of solution time ratios for T1

A comparison between the performance of all three algorithms under setting T1 can be found in Figure 5.6 for each class of instances. In order to create the figure, for each instance and algorithm, we first computed the ratio of the solution time of the algorithm on that instance to the minimum solution time of all algorithms on that instance. We then plotted the box plot of the solution time ratios of each method. Note that since the ratios cannot be smaller than one by construction, a box plot that is closer to one is better. Observe that CSBnC is almost flat in both Figures 5.6a and 5.6b. This is because from the total of 400 instances, 393 instances are solved (significantly) faster by CSBnC than the other two methods. Comparing the results reveals that, on average, SOCP is faster than S&C by a factor of 1.27 and 1.21 on mixed binary and mixed integer instances, respectively. As an aside, we note that by construction S&C needs to substantially increase the size of each instance for mixed integer instances. Specifically, S&C needs to transform a mixed integer instance to a mixed binary instance in order to solve it. Overall, we observe that CSBnC

has the best performance. Comparing the medians of the box plots reveals that CSBnC is at least 5 times faster on 50% of instances than SOCP. In fact, our results show that CSBnC is 6.6 times faster than SOCP on average.

Table 5.3: The number of calls on primal and dual bound operations in CSBnC

Subclass	**Mixed Binary Class**		**Mixed Integer Class**	
	# Primal Calls	**# Dual Calls**	**# Primal Calls**	**# Dual Calls**
1	4.7	6.8	4.4	6.6
2	5.2	8.2	5.7	8.4
3	5.3	7.2	5.5	7.8
4	4.7	7	4.8	7.6
5	6.1	8.6	5.3	7.8
6	6.6	9.6	4.8	6.8
7	5.6	7.8	5.6	7.8
8	4.2	6.2	4.7	6.2
9	5	7.2	6.6	9.8
10	7.1	10	5.9	8.4
11	6.3	8.6	5.2	7.6
12	4.6	7.4	4.7	6.6
13	5.9	8	5.9	8.6
14	5	7.4	5.8	7.6
15	5.2	7.2	5.4	7.6
16	5.3	7.6	5.6	7.8
17	5.7	7.4	5.1	7
18	6.5	9	4.5	6
19	5.1	7.6	5.8	8
20	4.6	6.6	6.2	8.6
Average	5.4	7.8	5.4	7.6

The outstanding performance of CSBnC is mainly because of the small number of single-objective mixed integer linear programs that it requires to solve. This can be observed from Table 5.3 that shows the average number of calls on $\textsc{Primal}(\boldsymbol{y}^l, \boldsymbol{y}^u, k^P, \textsc{Glb})$, (see Line 14 of Algorithm 4), and $\textsc{Dual}(\boldsymbol{y}^l, \boldsymbol{y}^u, k^D)$, (see Line 22 of Algorithm 4), per subclass. Note that the number of calls on dual bound finder basically reflects the number of iterations of CSBnC. The table reveals that the average number of iterations per subclass is at most 10 implying that the CSBnC quickly converges to an optimal solution in practice. Note that by construction the primal bound finder will not be necessarily called in each iteration. Consequently, the number of calls on the primal bound finder is significantly less than the number of calls on the dual bound finder.

Tables 5.4 and 5.5 show the average solution time (in seconds) of the three methods under all 4 settings for both classes of instances. The numbers reported in the tables are averages of solutions times over 10 instances (of each subclass). We observe that CSBnC

Table 5.4: Solution time (sec.) of different algorithms on mixed binary class using different number of threads

Subclasses	CSBnC				SOCP				S&C			
	T1	T2	T3	T4	T1	T2	T3	T4	T1	T2	T3	T4
1	3.2	3.2	3.5	3.5	4.7	3.2	2.8	2.3	6.9	6.3	6.3	6.4
2	7.7	8.1	8.1	8.3	26.4	16.5	12.9	10.8	32.1	27.8	28.0	27.8
3	12.0	12.7	12.8	12.5	49.2	29.2	22.1	18.5	62.2	54.8	54.8	55.2
4	15.4	16.4	16.4	16.4	100.8	57.6	42.8	35.2	124.2	110.0	112.2	110.4
5	10.1	10.6	10.9	11.0	26.1	16.7	14.4	12.0	31.3	27.1	27.1	27.0
6	23.6	25.5	25.2	25.4	110.9	65.6	47.2	39.7	126.9	113.8	113.0	111.9
7	32.2	34.7	34.7	34.0	210.4	122.0	85.3	70.1	254.4	224.6	226.0	224.1
8	36.9	38.6	38.8	38.9	307.8	166.5	117.1	96.7	361.3	312.5	324.7	327.3
9	16.4	17.6	17.6	17.6	48.9	31.0	24.9	21.9	60.5	54.7	53.8	53.7
10	47.9	51.4	50.7	50.6	254.9	146.8	106.6	85.3	326.0	280.9	284.0	270.0
11	69.5	72.5	72.3	72.7	492.5	286.6	203.4	156.7	548.2	502.3	511.7	513.8
12	79.4	83.4	82.1	83.8	983.7	566.9	395.5	324.0	1176.1	1083.5	1104.0	1091.7
13	30.2	32.7	31.8	32.8	99.0	63.7	47.6	41.5	123.3	110.0	110.6	110.9
14	60.5	63.1	62.5	62.6	454.4	275.4	193.4	156.8	568.5	502.6	509.9	502.0
15	96.2	101.0	101.2	101.7	981.2	551.6	382.7	301.6	1,102.6	1,002.0	979.1	985.2
16	133.6	141.8	140.1	140.8	1,709.6	1026.5	694.6	560.1	1,931.2	1,766.4	1,761.0	1,677.3
17	45.2	47.3	47.7	47.4	142.1	91.7	67.7	56.5	184.6	166.1	163.9	163.1
18	110.9	116.7	116.6	115.7	800.2	476.0	320.9	262.0	952.3	872.8	862.2	866.8
19	146.1	153.0	152.7	154.1	1,774.3	950.9	700.4	586.9	1,979.4	1,837.5	1,725.4	1,816.3
20	182.4	192.2	191.8	190.5	2,966.6	1,529.5	1,204.0	1,082.7	3,338.8	3,194.9	3,140.8	3,114.1
Average	58.0	61.1	60.9	61.0	577.2	323.7	234.3	196.1	664.5	612.5	604.9	602.8

Table 5.5: Solution time (sec.) of different algorithms on mixed integer class using different number of threads

Subclasses	CSBnC				SOCP				S&C			
	T1	T2	T3	T4	T1	T2	T3	T4	T1	T2	T3	T4
1	4.3	3.8	3.9	3.8	4.7	3.7	2.8	2.5	7.3	7.2	5.7	7.4
2	10.7	9.9	9.6	9.7	27.7	18.0	13.6	11.9	32.4	32.0	29.0	32.3
3	17.2	15.7	15.6	15.5	49.7	29.4	22.7	18.8	56.0	55.9	51.6	56.6
4	21.0	19.8	19.2	19.7	108.8	62.9	46.4	38.4	115.0	115.8	109.1	115.5
5	12.3	11.1	11.2	11.2	21.9	14.9	11.9	10.1	28.1	28.2	23.4	28.6
6	23.0	21.3	21.0	21.3	84.7	52.8	38.0	32.1	96.6	96.2	87.3	96.5
7	41.4	39.0	39.1	39.6	161.5	100.6	69.6	57.3	186.8	186.9	171.2	187.1
8	51.3	48.9	48.8	48.9	348.1	191.4	137.8	109.4	349.7	350.6	327.3	346.5
9	26.2	23.3	23.4	23.6	59.4	37.7	30.9	27.3	77.4	77.7	67.6	78.9
10	52.5	48.5	48.6	48.1	236.7	133.4	102.9	85.1	263.6	259.4	235.1	259.8
11	77.9	73.4	73.2	74.6	482.7	265.2	205.1	154.7	533.4	538.7	501.7	537.0
12	101.6	93.1	95.4	93.9	873.7	467.2	359.9	272.1	931.4	921.5	909.4	925.0
13	37.0	36.4	37.0	37.0	102.0	62.0	50.1	41.6	144.7	145.1	123.4	146.3
14	75.9	76.5	75.9	76.7	497.5	270.5	213.4	162.2	564.3	565.8	542.2	578.5
15	122.5	121.2	121.8	121.3	1,034.0	530.5	429.6	323.4	1,095.5	1,095.3	1,038.0	1,099.6
16	169.0	167.3	167.7	167.9	2,076.5	1,015.3	846.5	692.6	2,175.8	2,176.0	2,110.8	2,183.2
17	49.4	48.8	49.6	48.0	199.9	106.2	90.5	76.8	238.8	241.0	201.2	242.4
18	97.4	95.6	94.3	93.5	760.8	386.7	293.4	249.7	847.6	832.1	731.0	829.3
19	190.8	182.7	183.0	185.8	1,707.0	905.1	647.5	529.6	1,874.0	1,874.6	1,855.3	1,909.3
20	278.8	275.9	276.0	277.9	3,431.2	2,001.3	1,292.5	1,122.2	3,291.3	3,299.7	3,188.9	3,284.1
Average	73.0	70.6	70.7	70.9	613.4	332.7	245.3	200.9	645.5	645.0	615.5	647.2

does not improve when more threads are available to CPLEX. This is partly because in our algorithm CPLEX will be called a few times. In each call, there is a setup cost for enabling parallelization in CPLEX that may not offset the time saving obtained through activating parallelization for solving the subproblem. The same observation is true for S&C and (not surprisingly) no tangible improvement can be seen in the performance of S&C in both classes of instances. For SOCP, we observe that solution times have improved consistently through enabling parallelization in CPLEX. Comparing settings T4 and T1 in SOCP, we observe that an improvement of around 62.69% and 62.20% on average has obtained through enabling parallelization in CPLEX for mixed binary class and mixed integer class, respectively. Again this is not surprising as SOCP pays the initial cost of parallelization in CPLEX only once. Overall, comparing setting T4 of SOCP with T1 of CSBnC reveals that CSBnC is faster than SOCP on average by the factor of 2.53 and 2.03 for mixed binary and mixed integer classes, respectively.

5.6 Conclusion

We studied a class of multiplicative programs, MIBL-MMP, involving both continuous and integer variables through the lens of bi-objective optimization and introduced a generic criterion space branch-and-cut algorithm for it. The proposed algorithm, CSBnC, employs novel dual-bound and primal-bound computing operations based on the concept of piecewise McCormick envelopes. It also employs a highly effective cut-generating mechanism. Through a computational study, we demonstrated the impacts of the main components of our algorithm and numerically showed how its main parameters can be tuned. Our numerical analysis showed that tuning can improve the solution time of our algorithm by a factor of 11.78 on average. Moreover, CSBnC outperforms the mixed integer SOCP solver of CPLEX 12.10 and a state-of-the-art algorithm in the literature by factors 6.54 and 7.54 on average, respectively. The main future research direction of this study is how the proposed solution

method can be customized when the dimension of the criterion space is more than two, i.e., $p > 2$.

Chapter 6: Equitable Workload Allocation in Vehicle Routing Problem with Heterogeneous Driver

In this chapter, we present paper P5. In this work, we introduce an equitable crowd-sourced last-mile delivery model by adopting Nash social welfare solution as the coalition point among different drivers. While the ultimate goal is to maximize the equity and efficiency of drivers, the efficiency of the company is guaranteed by putting a cap on the deviation of company's cost from the least-cost solution value. To solve the proposed new formulation, a column generation method is developed and used to study the behavior of the model.

6.1 Introduction

A solution that maximizes the sum of utilities of all the players might not be implementable, because some of the parties might consider it "unfair" as such a solution may be achieved at the expense of some players [159]. In many environments, fairness might be more important than efficiency. To address fairness, the concept of equity has received a resurgence of interest in various fields over the last decade [160]. Transportation and logistics is not an exception from this trend [161, 162]. Although efficiency-based objectives serve the needs of commercial distribution problems, routing applications in the public/nonprofit sector such as public transportation and humanitarian logistics [163, 164] often require other objectives that capture their non-monetary performance requirements including equitable service provision to their external stakeholders, i.e., the users or customers [165]. In routing problems in the private sector, the most common equity considerations concern internal stakeholders, i.e., the drivers or other personnel providing the service [166]. The aim is

to balance the workload allocation to ensure acceptance of operational plans, to maintain employee satisfaction and morale, to reduce overtime, and to reduce bottlenecks in resource utilization. Practical examples include balancing the workload of service technicians [167], home healthcare professionals [168], and volunteers [169].

Workload assignment equity in applications of the vehicle routing problem (VRP) has been receiving more attention over the past decade [165, 162, 170]. Workload assignment equity is particularly important and often harder to achieve in settings that involve heterogeneous fleets of drivers. One of the most prominent examples of such a setting is the context of crowdshipping. Crowdshipping, relying on the overarching concept of *shared economy*, proposes making the excess capacity of private vehicles on their passenger trips available to delivery operations. Although the business idea of crowdsourced delivery arose in 2013 with the foundation of Postmates [171], followed by ideating the concept by some giant retailers such as Walmart, Amazon, and Home depot [172, 173, 174], it attracted the academic attention first in 2016 [175]. Since then, several papers addressed different aspects of problems that are involved in this idea [176, 177, 178, 179].

In a typical implementation of crowdshipping, a group of ad-hoc/occasional drivers take over the delivery task of one or multiple online orders to online shopper locations [171, 175, 180, 181, 182, 183]. Perhaps the most distinguishing characteristic of crowdshipping from the conventional applications of the VRP in last-mile delivery is the fact that drivers are not employees of the company that provides the delivery service, become available to render service only occasionally, and usually use the spare space on their personal vehicles to make such deliveries on the way to their next destination. In that sense, crowdsourced drivers differ in terms of the time they become available, the amount of time they are willing to dedicate to make deliveries, the capacity of their vehicles and finally their next destinations. In such a setting, an equitable workload assignment is crucial since usually the workload assignment has a direct impact on the compensation paid to and the cost incurred by each driver. These factors play an important role in the overall satisfaction of a

driver from their participation in the delivery platform and a more positive impression would entice a larger group of (returning and new) crowd-based drivers for future participation. Therefore, the main goal of this study is to provide an implementable framework to improve workload equity while maintaining the cost of the commercial distribution platform within an acceptable threshold.

6.1.1 Literature Review

The concept of crowdsourced last mile delivery, although being a quite novel concept, has received up-growing attention recently and therefore the relevant literature is rather rich. On the other hand, the concept of fairness, has been around for a much longer time and once again possesses a rich literature. In this survey of the literature, we mainly focus on the intersection of these two areas; crowdsourced last-mile delivery and workload equity requirements in commercial settings. The existing body of literature on commercial crowdsourced last-mile delivery includes several studies. Some of which explore the pricing of delivery for consumers and compensation rate for drivers [184, 185, 186]. The difference in these studies comes from the authors different perspective to the problem that is cast into a variety of assumptions. For example, some of the papers assume that the occasional drivers are in-store customers who are willing to support delivery operation on their way home [175, 180]; while others consider the occasional drivers as hired couriers with limited time and area coverage [184, 183]. The latter setting is more compatible with most of the commercial crowdsourced last-mile delivery platforms such as Amazon Flex, Walmart Spark Delivery, DoorDash, etc. The dynamic nature of the problem is another assumption that some studies account for [180, 182], beside the ones that only focus on solving a static version of the problem in a specific time period where the demand and available drivers are given [175, 181]. The number of delivery tasks assigned per driver and pick-up points are other features that are addressed differently in the literature. Some of the papers allow more than one delivery task per driver

[181, 182, 184, 180] and multiple pick-up points [182, 183]. For a comprehensive review of all assumptions and methodologies, interested readers may refer to [171].

Although the literature on crowdsourced last-mile delivery is relatively rich, there are very few studies that focus on improving workload equity in the context of commercial crowdsourced last-mile delivery, partly because of the relatively recent emergence of the concept. Existing studies mainly attempted to introduce the concept and highlight its importance/properties rather than proposing an implementable framework to improve workload equity while maintaining the cost of the commercial distribution platform within an acceptable threshold. For example, [187] analyze the effect of different criteria of workload, indicated to as "workload resource", on balanced VRP solutions of some equity measures. By comparing the marginal cost of equity, the authors report that the cost-equity trade-off is more subject to workload resource than the equity measure. In another research, [162] analyze axiomatic properties of some equity measures on some small bi-objective VRP instances where the cost is one objective and the other objective is an equity measure.

In general, in the existing body of literature on fairness in optimization, two measures are popular [159]: Max-Min and Nash social welfare (NSW). Max-Min seeks to improve the profit of the worst player (i.e., driver in VRP) while NSW seeks to improve the profit of all players at the same time. Although Max-Min has been studied in the context of VRP [188, 189, 190], there is no study, to the best of our knowledge, on how NSW can be adopted for improving workload equity in VRP, perhaps due to its nonlinear nature.

6.1.2 Contributions

In this paper, we explore how the workload equity can be improved while maintaining the cost of the commercial distribution platform within an acceptable range from the least-cost solution. We focus on *static* commercial crowdsourced last-mile delivery problem where the available drivers and demand points are given and there is only one pick-up point (depot). Moreover, we assume that multiple delivery tasks can be assigned to one driver as long as the

capacity of the vehicle is not exceeded. The main contributions of this paper are summarized as follows:

- We are the first to introduce a new equitable VRP model that relies on the concept of the NSW, by maximizing the product of drivers' profits (rather than maximizing the sum of drivers' profits) while maintaining the cost of the commercial distribution platform within an acceptable threshold. The main advantage of the NSW-based formulation is that it naturally seeks to simultaneously maximize the total profit of drivers paid by the company and split it among the drivers as equitable as possible. An equitable allocation of profits in an environment that the drivers are heterogeneous is more complex than simply dividing the total profit equally among the drivers, as the profit of each driver must be proportional to his effort as well as the potential maximum profit he can make given his characteristics.

- We discuss different compensation schemes that rely on a combination of per mile and per delivery payments, and show that in a setting with purely per delivery compensations, achieving equitable distribution of tasks and profits is much harder than when the compensation includes both per mile and per delivery payments.

- We develop an exact solution approach based on a branch-and-price framework to solve the proposed NSW-based VRP formulation. We discuss how an exact linear master problem formulation for the proposed NSW-based VRP formulation can be generated. In the proposed decomposed formulation, the subproblems take the form of an elementary shortest path problem with resource constraints (ESPPRC), which is characterized by a nonlinear cost function. We then propose new techniques to adapt the classical Bellman-Ford labeling algorithm to account for the nonlinearity in the cost function.

- We conduct an extensive computational study to evaluate the performance of our proposed models as well as the developed solution approach. Focusing on measures such

as the efficiency of the company, the efficiency of drivers, and the equity among the drivers, we show that the NSW solution can improve the equity in profit allocation among the drivers significantly, while keeping the level of company's efficiency in an "acceptable" range from the non-equitable least-cost solution. Our experiments also show the superiority of the NSW solutions compared to those of the Max-Min formulation.

6.1.3 Paper Structure

The remaining of the paper is organized as follows. First, the problem is introduced, and the required notation is described. The proposed method to solve the problem has two stages, in the first of which a classical VRP is solved to determine the minimum total cost of the company and the selected drivers. In the second stage, the NSW problem is solved to obtain the equitable job assignment among the hired drivers. In Sections 6.3 and 6.4, the elements of the branch-and-price algorithm to solve the first and second stage is described, respectively. Section 6.5 covers the computational study, analyses, and our procedure to generate the used instances. Finally, the concluding remarks and future directions for this research is provided in Section 6.6

6.2 Problem Setting and Formulation

In this section, we describe a last-mile delivery problem that aims at maximizing profit equity (as a proxy for workload assignment equity) among drivers while maintaining cost efficiency for the company. The problem is defined on a graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ and $\mathcal{A}$ represent the set of locations and the arcs within graph $\mathcal{G}$, respectively. A group of customers place orders online and expect deliveries at their home locations. Let $\mathcal{D} \subset \mathcal{N}$ be the home locations of the customers with outstanding orders, where q_j denotes the quantity/size of the delivery at the location of customer $j \in \mathcal{D}$. The travel time between each pair of locations

(j, j'), $j, j' \in \mathcal{N}$ for which an arc exists in $\mathcal{A}$ is given by $t_{jj'}$ while the transportation cost along an arc (j, j'), denoted by $c_{jj'}$, is proportional to $t_{jj'}$.

Online orders are all fulfilled from a single depot o. The deliveries are performed by a set of available occasional drivers, denoted by $\mathcal{V}$. Each driver $v \in \mathcal{V}$ is characterized by their vehicle capacity, Q_v, their origin, s_v, and their destination where they are headed after finishing the delivery task, e_v. Set $\mathcal{N}_v$ represents the subset of nodes associated with diver v: $\mathcal{N}_v = \{o, s_v, e_v\} \cup \mathcal{D} \subseteq \mathcal{N}$, while $\mathcal{A}_v \subseteq \mathcal{A}$ is the set of arcs associated with the subgraph $\mathcal{G}_v$ of driver v. The availability of driver v is given as a function of the travel time associated with a direct trip from their origin to their destination, t_{s_v,e_v}. That is, a driver may be willing to dedicate a maximum of $\gamma_v t_{s_v,e_v}$ minutes to make deliveries, where γ_v is an indication of time flexibility of driver v. Although our proposed approach in this study is generic, for convenience we assume that $\gamma_1 = \gamma_2 = \cdots = \gamma_{|\mathcal{V}|}$ in all numerical examples/experiments in this paper, and use the notation γ to represent the time flexibility of any driver.

Drivers are compensated based on the deliveries they make (the prize received by serving a customer), and the mileage traveled. Let w_j be the prize collected by a driver after serving customer $j \in \mathcal{D}$. In its most generic form, w_j can be defined as $w_j := at_{0j} + bq_j$, where a and b are two weights set by the company. Through this compensation scheme, the company recognizes the extra time and effort required to deliver a larger quantity to a far away location. Additionally, the company compensates the mileage cost associated with trips to make deliveries to the customers.

Let $\mathcal{D}_v \subseteq \mathcal{D}$ be the set of customers assigned to driver v and c'_{r_v} be the mileage cost (variable cost) of the route the driver takes from his origin, to the depot, then to visit the customers in $\mathcal{D}_v$ in the most efficient sequence to finally return to his destination. Moreover, let c'_{s_v,e_v} be the mileage cost that driver v has to pay in order to directly go from his origin to his destination. Logically, the company will (partially) compensate the additional mileage cost to c'_{s_v,e_v} because c'_{s_v,e_v} has to be paid by the driver v anyway. Consequently, the mileage

compensation paid by the company to the driver v for the route r_v is

$$c_{r_v} := \beta(c'_{r_v} - c'_{s_v,e_v}),$$

where parameter $0 \leqslant \beta < 1$, referred to as the *mileage rate*, corresponds to the percentage of the mileage cost incurred by a driver that the company reimburses. Note that $\beta \geqslant 1$ is not a logical assumption because in that case during the process of improving equity, drivers would not seek to automatically reduce the mileage cost of the company; In fact the optimization process can create unfair arbitrage opportunities for them by assigning them non-optimal routes for a given set of tasks while in practice the drivers may choose to take the optimal routes and make additional hidden profits.

Note that for convenience $w_{e_v} := -\beta c'_{s_v,e_v}$ can be viewed as the fixed mileage cost of the driver at his destination, i.e., the cost that should be paid by the driver v when he reaches to his destination. With this in mind, the yield of driver v (the total compensation paid by the company) for the route r_v can be given as

$$y_{r_v} := \sum_{j \in \mathcal{D}_v} w_j + c_{r_v} = \sum_{j \in \mathcal{D}_v \cup \{e_v\}} w_j + \beta c'_{r_v}.$$

Based on this notation, the profit of driver v for the route r_v can be written as

$$p_{r_v} := y_{r_v} - c'_{r_v} = \sum_{j \in \mathcal{D}_v \cup \{e_v\}} w_j + (\beta - 1)c'_{r_v}.$$

It is assumed that drivers become aware of the company's compensation scheme before signing up to the platform. Thus, if after signing up to the platform, a driver is assigned a given delivery task, he typically accepts it. Furthermore, delivery assignments are done in such a way that the last delivery on the driver's route is as near as possible to the driver's next destination. This policy not only reduces the amount of inconvenience for the drivers, it can potentially reduce the cost paid by the company as the company is committed to

partially cover the mileage cost of drivers (indicated by the value of β). The problem is then a variant of the open vehicle routing problem (OVRP) with single depot which although the vehicles are not required to return to the depot, they still have prefixed ending points. The company aims at maximizing workload assignment equity while maintaining certain level of efficiency. In our definition, a workload assignment is said to be equitable when *profit ratios* of the drivers employed are as close as possible to each other. The profit ratio of driver v is defined as $\rho_v = p_v/\overline{p}_v$, where p_v is the profit of employed driver v under the given workload assignment, and $\overline{p}_v$ represents the maximum profit that the employed driver v can possibly earn in a setting where each employed driver must be assigned at least one delivery task. In essence, $\overline{p}_v$ is a representative of the potentials of driver v, given his personal characteristics including his time flexibility, vehicle capacity and the location of their next destination. Based on this definition of equity, our goal is not to compensate all drivers equally, but instead to compensate them fairly based on what they bring to the table, as having ρ_v as close as possible for all employed drivers has little to no implications in terms of closeness of their actual profits p_v.

It is obvious that such an equity comes at a cost. The extra cost paid to improve workload assignment equity in a solution is called the *cost of equity*. From the company's perspective, the costs to pay are the prizes paid to serve customers plus the mileage costs. Since given a set of customers to serve, the total delivery prizes paid is constant regardless of delivery assignments to the drivers, the cost component to optimize is the mileage costs paid to the drivers. We assume that the company is willing to accept a set of routes that involves $\alpha\%$ increase in the total mileage compensation, if an improvement in workload assignment equity can be achieved. In that sense, any solution with a mileage cost within the interval $[z^*, (1+\alpha)z^*]$ is considered *efficient* for the company, where z^* corresponds to the least mileage compensation paid given the set of customers $\mathcal{D}$ and the available fleet of drivers $\mathcal{V}$, ignoring workload assignment equity requirements. That is, in our setting, the cost of equity is caped at αz^*. Therefore, the company's primary objective is to promote workload

assignment equity within the limits of the acceptable efficiency measure, while its secondary objective is to minimize the mileage cost. Notice that the routing solution associated with z^* does not necessarily employ all available drivers in $\mathcal{V}$ and may suggest using only a subset $\mathcal{V}^* \subseteq \mathcal{V}$. Once the subset of drivers $\mathcal{V}^*$ associated with z^* is identified, we then aim at improving workload assignment equity in a solution that only involves drivers in $\mathcal{V}^*$. The following remark about the above-mentioned problem setting is helpful, and it is the main focus of our computational study (in Section 6.5).

Remark 6.1. In our problem under study, there are three main parameters α (the company's willingness to sacrifice efficiency to improve equity), β (the company's willingness to partially compensate the mileage cost incurred), and γ (drivers' time flexibility) that need to be explored in order to measure their impacts on equity when employing our proposed approach for improving equity.

6.2.1 Problem Formulation

Balancing profit ratios has been studied in the context of Nash Social Welfare (NSW). The NSW solution has roots in game theory and pertains, in particular, to a well-known cooperative game problem called bargaining problem. In the solution of a bargaining problem the players, who compete for a higher gain, agree to form a grand coalition [191]. The necessary condition for this coalition is the consent of players on the obtained under-coalition gain. One of the well-known techniques to compute under-coalition gain was first introduced by [129, 192]. In this study, we aim at maximizing the profit ratio of drivers as the players of a bargaining game using Nash's method. In a setting where drivers tend to return to provide services, it is conceivable that they would agree to under-coalition gains, increasing the expected total profit of all drivers in the long run. From the company's perspective, such solutions entice a larger proportion of the drivers to return to participate.

Let $\mathcal{X}$ be the set of all feasible routing solutions that partition orders. The definition of a feasible route is given next.

Definition 6.1. A route is considered *feasible* for driver $v \in \mathcal{V}$ if the following four conditions hold: (a) Departing from his origin, a driver, if employed by the company, would go directly to the depot to collect the items to deliver to a subset of customers, otherwise he goes directly to his destination; (b) The delivery route assigned to a driver must respect vehicle capacity and time availability constraints of the driver; (c) The overall profit of a driver who's assigned a delivery route from executing that route must be non-negative; and (d) The additional profit of serving customer j immediately after customer i by the driver v must be non-negative, i.e.,

$$w_j + (\beta - 1)c'_{ij} \geqslant 0.$$

This is because drivers being gig-workers, have an incentive to decline serving a customer if it results in a loss of profit.

The set of feasible routes $\mathcal{X}$ incorporates the assignment of delivery tasks to the drivers, while specifying the sequence of visits to the customers (routing decision) by each driver. Therefore, $\rho_v(\boldsymbol{x})$ represents the profit ratio of driver v given a routing decision $\boldsymbol{x}$. Suppose also that $g(\boldsymbol{x})$ returns the mileage cost of a given routing solution $\boldsymbol{x}$, i.e., $g(\boldsymbol{x}) = \sum_{v\in\mathcal{V}} c_{r_v}$. Notice that if a driver is not assigned any delivery tasks, his compensation is assumed to be zero.

Inspired by the expanded version of NSW [193], to equitably assign a set of delivery jobs among drivers in $\mathcal{V}^*$, the problem can be formulated as

$$\max \quad \prod_{v\in\mathcal{V}^*} \rho_v(\boldsymbol{x}) \tag{6.1a}$$

$$\text{s.t.} \quad \boldsymbol{x} \in \mathcal{X}, \tag{6.1b}$$

$$g(\boldsymbol{x}) \leqslant (1+\alpha)z^*, \tag{6.1c}$$

where objective function (6.1a) maximizes the Nash social welfare that is the product of drivers' profit ratios, while Constrains (6.1b) guarantee the feasibility of routes assigned to

the drivers. Constrains (6.1c) ensure that any chosen routing schedule remains within the efficiency limits of the company. Notice that Constrains (6.1c) can be embedded into the definition of $\mathcal{X}$, however, for the sake of clarity, we chose to state them explicitly.

Proposition 6.1. *The NSW problem is scale-free: Maximizing* $\prod_{w\in\mathcal{W}} \lambda_w f_w(\boldsymbol{x})$ *is equivalent to maximizing* $\prod_{w\in\mathcal{W}} f_w(\boldsymbol{x})$*, where* $\lambda_w > 0, \forall w \in \mathcal{W}$ *are arbitrary constants.*

Proof.

The proof is rather straightforward and can be found in [194] □

According to Proposition 6.1, one can substitute profit ratios of drivers by their actual profits in objective function (6.1a). Intuitively, since the denominator $\overline{p}_v$ in the profit ratio of all drivers v is constant and independent from any routing decision $\boldsymbol{x} \in \mathcal{X}$, it has no impact on the choice of the optimal routing decision. Consequently, Model (6.1) reduces to

$$\begin{aligned} \max \quad & \textstyle\prod_{v\in\mathcal{V}^*} p_v(\boldsymbol{x}) \\ & \text{(6.1b)} - \text{(6.1c)} \end{aligned} \tag{6.2}$$

Model (6.2) is an extension of the VRP where the objective is to maximize workload assignment equity while keeping the cost of equity in terms of mileage cost under certain threshold. In Appendix 7.4, we present an arc-based formulation of such a VRP. A NSW solution for the introduced delivery problem establishes a balance between the maximum attainable efficiency and workload assignment to drivers. To clarify this, we refer to the following theorem proved by [195].

Theorem 6.1. *For any optimal solution of the following (mixed-integer) linear program, denoted by* $\boldsymbol{x}^*$*,*

$$\boldsymbol{x}^* \in \text{Argmax}\Big\{ \sum_{v\in\mathcal{V}^*} p_v(\boldsymbol{x}) : \boldsymbol{x} \in \mathcal{X} \Big\},$$

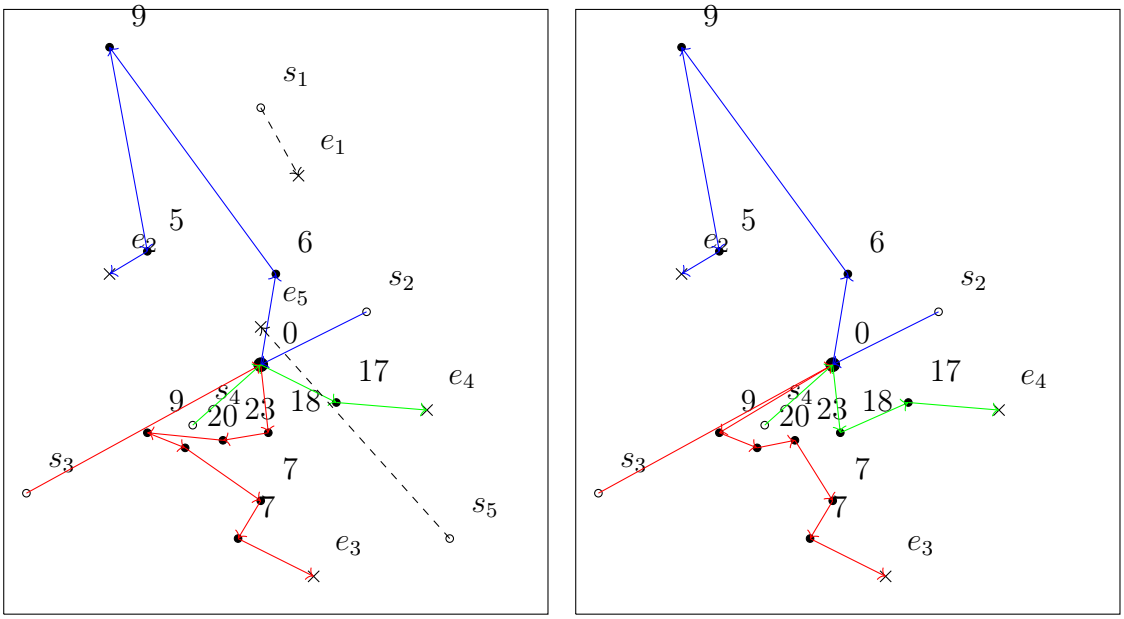
the following inequalities hold.

$$\prod_{v\in\mathcal{V}^*} p_v(\boldsymbol{x}^*) \leqslant \max_{\boldsymbol{x}\in\mathcal{X}} \prod_{v\in\mathcal{V}^*} p_v(\boldsymbol{x}) \leqslant \left(\frac{\sum_{v\in\mathcal{V}^*} p_v(\boldsymbol{x}^*)}{|\mathcal{V}^*|}\right)^{|\mathcal{V}^*|}.$$

The above inequality can be interpreted as follows. If a routing solution maximizes the sum of drivers' profits and all drivers make equal profits, then such a solution coincides the optimal solution of the NSW. Since $\sum_{v\in\mathcal{V}^*} p_v(\boldsymbol{x}^*)$ yields the maximum attainable total profit, no solution can be more equitable than a workload allocation that allows all drivers to make a profit equal to $\frac{\sum_{v\in\mathcal{V}^*} p_v(\boldsymbol{x}^*)}{|\mathcal{V}^*|}$. Note that, because of the discrete nature of most real world problems including the VRP, it is unlikely to exist a solution $\boldsymbol{x}^* \in \mathcal{X}$ so that all the drivers make exactly the same profit. Thus, Theorem 6.1 indicates that NSW solution provides the nearest solution to the fairest possible solution than all the feasible solutions with the maximum total profit. However, to achieve a higher level of fairness, NSW approach may sacrifice the efficiency or vice versa. The multiplicative objective function in this method acts as a super-criterion that compromises the trade-off between efficiency and fairness to a natural balance between them [196, 197]. Interested readers may refer to [**?** 195] and [198] to learn more about the NSW problem and existing solution methods.

6.2.2 An Illustrative Example

We now provide a small example of the proposed approach and analyze the obtained solution by comparing it with that of a classical VRP. In the mentioned example, there are 10 delivery requests and 6 drivers with the vehicle capacity of 80 units. The mileage cost is almost fully compensated, i.e., $\beta = 0.99$, and the drivers' time availability is $\gamma = +\infty$. Figure 6.1 shows the location of depot, delivery requests, and the origin and destination of drivers in this example. The demand size of each delivery request in terms of the unit of capacity (i.e., weight or volume) is reported by a number above its location. First, we solve this problem using the classical VRP where the minimum total cost is obtained 353.5 and



(a) Solution of the classical VRP model (b) Solution of the NSW model ($\alpha = 0.05$)

Figure 6.1: Routes for the example in both classical VRP and NSW models

the company chooses to hire drivers 2 and 3. The route of all drivers is demonstrated in 6.1a. We can observe that the destination of the drivers plays a main role in the assignment of the delivery requests. After plugging $z^* = 353.5$ and restricting the driver set to $\mathcal{V}^*$, we solve the proposed NSW model using the mixed integer Second-Order Cone Programming (SCOP) solver of CPLEX 12.10 for two values of α. Interested readers may refer to [**?**] or [199] to learn how a NSW model can be transformed to a mixed integer SOCP model using the geometric-mean transformation proposed by [200].

Table 6.1: Results of the illustrative example

Players	**Max. Profit**	**Trad. VRP Profit**	**NSW with $\alpha = 0$**		**NSW with $\alpha = 0.05$**	
			Profit	**Ratio**	**Profit**	**Ratio**
$v = 2$	130.64	113.46	113.46	0.87	113.45	0.87
$v = 3$	224.00	204.23	204.23	0.91	170.80	0.76
$v = 4$	53.98	6.20	6.20	0.11	38.45	0.71
Company (cost)	-	61.86	63.04	-	205.20	-

The results are summarized in Figure 6.1b and Table 6.1. The first column of the table reports the maximum revenue of each driver that are obtained by solving separate

classical VRPs when the objective function is the corresponding driver's revenue. Note that the first column is important as it is used to compute profit ratios. Also, the minimum total cost and revenue of both drivers in column "Trad. VRP Profit" is the outcome of solving the classic VRP from the previous step. The rest of the table shows the outcomes of the NSW model. We observe that, the solution of the classical VRP model is the most equitable solution for $\alpha = 0$ as well. However, by allowing a 5% increase in the company's mileage cost, the equitable model finds a much fairer solution where the difference in profit ratios drops to 0.04 (as opposed to 0.48 when $\alpha = 0$).

6.3 Computing the Minimum Total Mileage Cost z^*

In order to compute the minimum total mileage cost (MTMC), i.e., z^*, we propose to employ a branch-and-price approach. Let $\mathcal{R}_v$ be the set of all *feasible* routes for driver $v \in \mathcal{V}$. Note that we assume that all drivers must go from their origin nodes to their destination nodes. If a driver goes directly from his origin node to his destination node, it is implied that the driver has not been selected by the company, rendering a profit of zero for the driver. If a driver is selected by the company, he must first go to the depot to pick up the orders assigned to him, and deliver them on the way to his destination.

Let binary variable $x_{vr} = 1$ if driver $v \in \mathcal{V}$ is selected and is assigned route $r \in \mathcal{R}_v$; 0 otherwise. Suppose binary parameter δ_{jr} equals 1 iff delivery of order $j \in \mathcal{D}$ is performed on route $r \in \mathcal{R}_v$ of driver $v \in \mathcal{V}$. Recall that c_{r_v} is the total mileage compensation that the company should pay to driver $v \in \mathcal{V}$ for taking route $r \in \mathcal{R}_v$. Using these notations, the path-based formulation of the VRP problem can be stated as follows,

$$(\texttt{MTMC}) \quad z^* = \min \quad \sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}_v} c_{r_v} x_{vr} \tag{6.3}$$

$$\text{s.t.} \quad \sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}_v} \delta_{jr} x_{vr} = 1 \quad \forall j \in \mathcal{D}, \tag{6.4}$$

$$\sum_{r \in \mathcal{R}_v} x_{vr} = 1 \qquad \forall v \in \mathcal{V} \tag{6.5}$$

$$x_{vr} \in \{0, 1\} \qquad v \in \mathcal{V}, \forall r \in \mathcal{R}_v, \tag{6.6}$$

where Constraints (6.4)-(6.6) guarantee that each customer is visited exactly once and each vehicle is assigned to exactly one route.

6.3.1 Branch and Price to Solve the MTMC

In the presented model `MTMC`, since the cardinality of $\mathcal{R} := \cup_{v \in \mathcal{V}} \mathcal{R}_v$ is large in practice, the approach used to solve these set partitioning formulation problems is based on a branch-and-price algorithm. In such an approach, one tries to find an integer solution using a branch-and-bound scheme, in which the lower bound of each node of the search tree is calculated by applying a column generation to solve the linear relaxation of the model `MTMC`. The column generation is based on iteratively solving a restricted master problem (`RMP`) and one or more subproblems. The `RMP` consists of the the linear relaxation of the augmented model which has been restricted to a subset of its variables. This subset simply contains those variables which have already been generated through solving the subproblems. Solving the `RMP` using a linear programming solver, usually based on the simplex algorithm, results in a primal and dual solutions. The subproblems, often taking the form of an elementary shortest path problem with resource constraints (ESPPRC), are typically solved using a dynamic programming-based algorithm [see 201, 202]. The negative reduced cost columns, newly generated through solving the subproblems, are then added into the `RMP` and another iteration begins. The process stops when none of the subproblems is able to find new negative reduced cost variables for the `RMP`.

The branch-and-price search tree is properly initialized by adding a set of initial columns into the `RMP`, in the root node. At each node of the tree, the lower bound is calculated through the back-and-forth exchanges between the master problem and subproblems, as described above. If the solution of the `RMP` is integer, it is a feasible solution to the original

problem and it is compared to the current incumbent solution and the current incumbent is updated, if necessary. Otherwise, if the solution of the RMP does not satisfy the integrality condition, a branching procedure is applied to cut off the fractional part of the solution. The optimal solution corresponds to the current incumbent solution, when all the branches have been fathomed.

The RMP contains only a subset of feasible routes $\widetilde{\mathcal{R}}_v \subseteq \mathcal{R}_v$ for each driver v and takes the following form

$$
\begin{aligned}
(\mathsf{RMP}) \quad \min \quad & \sum_{v \in \mathcal{V}} \sum_{r \in \widetilde{\mathcal{R}}_v} c_{r_v} x_{vr} && && (6.7)\\
\text{s.t.} \quad & \sum_{v \in \mathcal{V}} \sum_{r \in \widetilde{\mathcal{R}}_v} \delta_{jr} x_{vr} = 1 && \forall j \in \mathcal{D}, && (6.8)\\
& \sum_{r \in \widetilde{\mathcal{R}}_v} x_{vr} = 1 && \forall v \in \mathcal{V}, && (6.9)\\
& x_{vr} \geqslant 0 && v \in \mathcal{V}, \forall r \in \widetilde{\mathcal{R}}_v. && (6.10)
\end{aligned}
$$

Let π_j and μ_v denote the dual variables associated with Constraints (6.8) and (6.9), respectively. The subproblem associated with each driver $v \in \mathcal{V}$ can be stated as

$$
(\mathsf{SP}_v) \qquad \epsilon_v := \min\{c_{r_v} - \sum_{j \in \mathcal{D}} \delta_{jr} \pi_j - \mu_v : \forall r \in \mathcal{R}_v\}.
$$

The subproblem associated with driver v, SP_v, corresponds to an ESPPRC defined on network $\mathcal{G}_v$. It can, thus, be solved using a labeling algorithm. In this algorithm, a partial path from node s_v to a node $j \in \mathcal{N}_v$ is represented by a tuple $\boldsymbol{\lambda} = (\lambda^{cost}, \lambda^{time}, \lambda^{load}, \left(\lambda^{cust_i}\right)_{i \in \mathcal{C}})$ called a label: λ^{cost} is the path reduced cost; λ^{time} is the arrival time at node j; λ^{load} is the load accumulated in the current trip; and λ^{cust_i}, $i \in \mathcal{D}$, indicates whether or not customer i has been visited along the path to node j.

Starting from the initial label $(-\mu_v, 0, 0, \big(0\big)_{i\in\mathcal{D}})$ associated with node s_v, a labeling algorithm extends the labels in network $\mathcal{G}_v$ to enumerate feasible partial paths, all starting from node s_v. Note that all paths visit the depot o, immediately after departing from the driver's origin s_v and need to finish at the driver's destination e_v. The extension along an arc $(j,k) \in \mathcal{A}_v$ of a label $\boldsymbol{\lambda}_j = (\lambda_j^{cost}, \lambda_j^{time}, \lambda_j^{load}, \big(\lambda_j^{cust_i}\big)_{i\in\mathcal{D}})$ associated with node j yields a new label $\boldsymbol{\lambda}_k = (\lambda_k^{cost}, \lambda_k^{time}, \lambda_k^{load}, \big(\lambda_k^{cust_i}\big)_{i\in\mathcal{D}})$ whose components are computed using the following resource extension functions:

$$\lambda_k^{cost} = \begin{cases} \lambda_j^{cost} + \beta c'_{jk} & \text{if } k = o \\ \lambda_j^{cost} + \beta c'_{jk} + \pi_k & \text{if } k \in \mathcal{D} \\ \lambda_j^{cost} + \beta c'_{jk} + w_{e_v} & \text{if } k = e_v \end{cases} \tag{6.11}$$

$$\lambda_k^{time} = \lambda_j^{time} + t_{jk} \tag{6.12}$$

$$\lambda_k^{load} = \begin{cases} \lambda_j^{load} + q_k & \text{if } k \in \mathcal{D} \\ \lambda_j^{load} & \text{otherwise} \end{cases} \tag{6.13}$$

$$\lambda_k^{cust_i} = \begin{cases} \lambda_j^{cust_i} + 1 & \text{if } i = k \\ \lambda_j^{cust_i} & \text{otherwise} \end{cases} \qquad \forall i \in \mathcal{D} \tag{6.14}$$

Recall that $w_{e_v} = -\beta c'_{s_v,e_v}$. By Definition 6.1. Label $\boldsymbol{\lambda_k}$ is associated with node k and is deemed feasible only if $\lambda_k^{time} + t_{k,e_v} \leqslant \gamma_v t_{s_v,e_v}$, $\lambda_k^{load} \leqslant Q_v$, $\lambda_k^{cust_i} \leqslant 1$ for all $i \in \mathcal{D}$, $p_{\lambda_k} \geqslant 0$ if $k = e_v$, and $w_k - (\beta - 1)c'_{jk} \geqslant 0$ if $k \in \mathcal{D}$. If one of these conditions is violated, label $\boldsymbol{\lambda_k}$ is discarded. To avoid enumerating all feasible $s_v - e_v$ paths, the following dominance rule is applied.

Definition 6.2. For a given vehicle $v \in \mathcal{V}$, let $\boldsymbol{\lambda}^\ell = (\lambda^{\ell,cost}, \lambda^{\ell,time}, \lambda^{\ell,load}, \big(\lambda^{\ell,cust_i}\big)_{i\in\mathcal{D}})$, $\ell = 1, 2$, be two labels associated with (partial) paths ending at the same node in $\mathcal{N}_v$. Label $\boldsymbol{\lambda}^1$ is said to dominate label $\boldsymbol{\lambda}^2$ and is discarded if

$$\lambda^{1,cost} \leqslant \lambda^{2,cost}, \tag{6.15}$$

$$\lambda^{1,time} \leqslant \lambda^{2,time}, \tag{6.16}$$

$$\lambda^{1,load} \leqslant \lambda^{2,load}, \tag{6.17}$$

$$\lambda^{1,cust_i} \leqslant \lambda^{2,cust_i}, \qquad \forall i \in \mathcal{D}, \tag{6.18}$$

$$\begin{cases} \text{either} \quad p_{\boldsymbol{\lambda}_j^2} \leqslant p_{\boldsymbol{\lambda}_j^1}, \\ \text{or} \quad p_{\boldsymbol{\lambda}^1_{j \oplus i * \oplus e_v}} > 0, \end{cases}$$

and at least one of the above inequalities is strictly satisfied.

In our proposed column generation algorithm, for driver $v \in \mathcal{V}$ with $\epsilon_v < 0$, we add all complete paths (labels reached e_v) with a negative reduced cost (i.e., $\lambda_{e_v}^{cost} < 0$) to $\widetilde{\mathcal{R}}_v$ and resolve the RMP. The column generation algorithm stops when $\epsilon_v \geqslant 0$ for all $v \in \mathcal{V}$, in which case no more columns are added and the solution of RMP is optimal. The proposed column generation, however, does not guarantee integrality of the solutions. Therefore, a branching mechanism is put in place. Due to the asymmetric nature of the problem (drivers' routes start and end at different locations), it is likely that the shortest path visiting all the delivery tasks assigned to a given driver to be unique. Thus, we focus on a binary branching scheme that assigns delivery tasks to drivers. That is, in a given fractional solution, if a delivery location is served by more than one driver, we select a (location, driver) pair, and create two child nodes; one enforcing the assignment of the (location, driver), and the other one forbidding it. In rare situations where this branching scheme is not sufficient, i.e., the solution is still fractional while delivery tasks are partitioned among the drivers, we apply the classical branching on flow variables.

6.4 Optimizing the Nash Social Welfare

Recall that $\mathcal{V}^* \subseteq \mathcal{V}$ is the subset of vehicles selected by the company as the result of minimizing its total cost through solving the MTMC problem. In order to compute the NSW solution for improving the workload equity for drivers selected through solving the MTCP problem (referred to as *drivers in the coalition*), we employ a second branch-and-

price approach. Let p_{r_v} be the profit generated for driver $v \in \mathcal{V}^*$ by executing *feasible* route $r \in \mathcal{R}_v$. Using these notations, the path-based formulation of NSW can be stated as follows.

$$
\begin{aligned}
(\texttt{NSW}) \quad \max \quad & \prod_{v \in \mathcal{V}^*} \Big(\sum_{r \in \mathcal{R}_v} p_{r_v} x_{vr} \Big) & & \\
\text{s.t.} \quad & \sum_{v \in \mathcal{V}^*} \sum_{r \in \mathcal{R}_v} \delta_{jr} x_{vr} = 1 & \forall j \in \mathcal{D}, & \quad (6.19) \\
& \sum_{r \in \mathcal{R}_v} x_{vr} = 1 & \forall v \in \mathcal{V}^*, & \quad (6.20) \\
& \sum_{v \in \mathcal{V}^*} \sum_{r \in \mathcal{R}_v} c_{r_v} x_{vr} \leqslant (1+\alpha) z^*, & & \quad (6.21) \\
& x_{vr} \in \{0,1\} & v \in \mathcal{V}^*, \forall r \in \mathcal{R}_v, & \quad (6.22)
\end{aligned}
$$

Observe that NSW is nonlinear. So, in order to employ a branch-and-price approach, it should be linearized. This can be done using the following theorem.

Theorem 6.2. *NSW is equivalent to the following integer linear program,*

$$
\begin{aligned}
(\textit{NSW-L}) \quad \max \quad & \sum_{v \in \mathcal{V}^*} \sum_{r \in \mathcal{R}_v} \log(p_{r_v}) x_{vr} \\
s.t. \quad & (6.19) - (6.22).
\end{aligned}
$$

Proof. Proof. Since by definition of feasible routes (i.e., Definition 6.1), the profit of each driver is nonnegative for any feasible solution, we know that the objective function of NSW-I can be replaced by its log-transformation, i.e.,

$$\sum_{v \in \mathcal{V}^*} \log\Big(\sum_{r \in R_v} p_{r_v} x_{vr} \Big).$$

Now, as based on Constraints (6.20) exactly one route is assigned to each driver, the transformed objective function is equivalent to,

$$\sum_{v \in \mathcal{V}^*} \sum_{r \in R_v} \log(p_{r_v}) x_{vr}.$$

□

6.4.1 Branch and Price to Solve the NSW Program

Theorem 6.2 is important as it uses a transformation trick in order to create an integer linear programming formulation, i.e., `NSW-L`, for optimizing NSW. Therefore, to solve its linear programming relaxation, once again, we can employ column generation. However, the column generation approach presented in this section is substantially different from the one proposed for the MTMC problem because its subproblems are significantly more challenging than those available in literature of classical VRPs. Specifically, we solve the following restricted master problem (`NSW-L-RMP`) where $\widetilde{\mathcal{R}}_v \subseteq \mathcal{R}_v$ for each $v \in \mathcal{V}^*$,

$$\begin{aligned}
(\texttt{NSW-L-RMP}) \quad \max \quad & \sum_{v \in \mathcal{V}^*} \sum_{r \in \widetilde{\mathcal{R}}_v} \log(p_{r_v}) x_{vr} && && (6.23)\\
\text{s.t.} \quad & \sum_{v \in \mathcal{V}^*} \sum_{r \in \widetilde{\mathcal{R}}_v} \delta_{jr} x_{vr} = 1 && \forall j \in \mathcal{D}, && (6.24)\\
& \sum_{r \in \widetilde{\mathcal{R}}_v} x_{vr} = 1 && \forall v \in \mathcal{V}^*, && (6.25)\\
& \sum_{v \in \mathcal{V}^*} \sum_{\widetilde{\mathcal{R}}_v} c_{r_v} x_{vr} \leqslant (1+\alpha) z^*, && && (6.26)\\
& x_{vr} \geqslant 0 && v \in \mathcal{V}^*, \forall r \in \widetilde{\mathcal{R}}_v. && (6.27)
\end{aligned}$$

We use the optimal feasible solution found from solving `MTMC`, i.e., $\boldsymbol{x}^*$, as an initial solution to `NSW-L-RMP`. Let ζ_j, θ_v, and η be the dual variables associated with Constraints

(6.24), (6.25), and (6.26), respectively. With these notations, the subproblem associated with each driver $v \in \mathcal{V}^*$ can be stated as

$$(\texttt{NSW-L-SP}_v) \qquad \epsilon_v = \max\{\log(p_{r_v}) - \sum_{j\in\mathcal{D}} \zeta_j \delta_{jr} - \theta_v - c_{r_v}\eta : \forall r \in \mathcal{R}_v\}. \tag{6.28}$$

We solve the ESPPRC associated with the $\texttt{NSW-L-SP}_v$ using the labeling approach, similar to the one described in Section 6.3.1. However, note that the first term in the objective function of Problem (6.28), $\log(p_{r_v})$, makes it nonlinear, requiring extra care when implementing the dominance rules.

Similar to the approach explained for solving the $\texttt{SP}_v$, a partial path from node s_v to node $j \in \mathcal{N}_v$ is encoded by a tuple $\boldsymbol{\lambda} = (\lambda^{cost}, \lambda^{time}, \lambda^{load}, \left(\lambda^{cust_i}\right)_{i\in\mathcal{C}})$, referred to as a label. Starting from the initial label $(-\theta_v, 0, 0, \left(0\right)_{i\in\mathcal{D}})$ associated with node s_v, the labeling algorithm extends the labels in network $\mathcal{G}_v$, knowing that all paths visit the depot o immediately after departing from the driver's origin s_v and must finish at the driver's destination e_v. The extension along an arc $(j,k) \in \mathcal{A}_v$ of a label $\boldsymbol{\lambda}_j$ associated with node j yields a new label $\boldsymbol{\lambda}_k$ whose components are computed using extension functions (6.12)-(6.14), and (6.29),

$$\lambda_k^{cost} = \begin{cases} \lambda_j^{cost} - \log(p_{\lambda_j}) + \log(p_{\lambda_k}) - \eta\beta c'_{jk} & \text{if } k = o \\ \lambda_j^{cost} - \log(p_{\lambda_j}) + \log(p_{\lambda_k}) - \zeta_k - \eta\beta c'_{jk} & \text{if } k \in \mathcal{D} \\ \lambda_j^{cost} - \log(p_{\lambda_j}) + \log(p_{\lambda_k}) - \eta\beta c'_{jk} + w_{e_v} & \text{if } k = e_v \end{cases} \tag{6.29}$$

where p_{λ_j} denotes the profit of the corresponding driver along the partial path represented by label λ_j. By Definition 6.1, label $\boldsymbol{\lambda_k}$ is deemed feasible and is kept only if $\lambda_k^{time} + t_{k,e_v} \leqslant \gamma_v t_{s_v,e_v}$, $\lambda_k^{load} \leqslant Q_v$, and $\lambda_k^{cust_i} \leqslant 1$ for all $i \in \mathcal{D}$, $p_{\lambda_k} \geqslant 0$ if $k = e_v$, and $w_k - (\beta - 1)c'_{jk} \geqslant 0$ if $k \in \mathcal{D}$. Note that the latter basically implies that we must have that $p_{\lambda_k} \geqslant p_{\lambda_j}$ if $k \in D$. Note too that in Equation (6.29), logarithmic functions can become 'undefined'. Thus, in terms of implementation, 'undefined' can be represented by a large negative number (denoted by $-M$). In other words, we can simply replace $\log(p_{\lambda_j})$ (or $\log(p_{\lambda_k})$) with $-M$ if $p_{\lambda_j} \leqslant 0$ (or

$p_{\lambda_k} \leqslant 0$). However, we avoid checking the domination rules for a partial route with undefined objective value. In fact, due to the nonlinearity of the objective function of the $\texttt{NSW-L-SP}_v$, using the domination rules similar to those given in Definition 6.2 can result in dominating columns that could end up being more profitable than the dominating labels. Therefore, instead, we use Propositions 6.2 and 6.3 to identify and remove dominated labels.

To facilitate the presentation of the propositions and their proofs, we introduce the notation $\oplus$ to denote the *immediate* extension operator for a given label. For example, $\boldsymbol{\lambda}_{j\oplus i}$ is the extension of label $\boldsymbol{\lambda}_j$ to node i as its immediate successor. We also make the following remarks about the immediate extension operator:

- If i is already a node in $\boldsymbol{\lambda}_j$ then immediate extension to i is not valid, i.e., $\boldsymbol{\lambda}_{j\oplus i}$=$\boldsymbol{\lambda}_j$.
- Let σ be a sequence of nodes with no node in common with $\boldsymbol{\lambda}_j$. In this case, $\boldsymbol{\lambda}_{j\oplus\sigma}$ implies that label $\boldsymbol{\lambda}_j$ is extended with sequence σ by connecting the node j (the current node of label $\boldsymbol{\lambda}_j$) to the first node in σ.
- If σ is empty, then $\boldsymbol{\lambda}_{j\oplus\sigma} = \boldsymbol{\lambda}_j$.

Proposition 6.2. *For a given vehicle $v \in \mathcal{V}^*$, let $\boldsymbol{\lambda}_j^\ell = (\lambda_j^{\ell,cost}, \lambda_j^{\ell,time}, \lambda_j^{\ell,load}, \left(\lambda_j^{\ell,cust_i}\right)_{i\in\mathcal{D}})$, $\ell = 1, 2$, be two labels associated with (partial) paths, denoted by 1 and 2, ending at node $j \in \mathcal{N}_v\backslash\{e_v\}$. Let $\mathcal{N}^{\boldsymbol{\lambda}_j^1}$ and $\mathcal{N}^{\boldsymbol{\lambda}_j^2}$ be the set of nodes visited in partial paths 1 and 2, respectively. Label $\boldsymbol{\lambda}_j^1$ is said to dominate label $\boldsymbol{\lambda}_j^2$ and is discarded if the following conditions hold:*

$$(6.16) - (6.19),$$

$$p_{\boldsymbol{\lambda}^1_{j\oplus i^*\oplus e_v}} > 0 \tag{6.30}$$

$$\lambda^{1,cost}_{j\oplus i^*\oplus e_v} \geqslant \lambda^{2,cost}_{j\oplus i^*\oplus e_v} \tag{6.31}$$

where $\overline{\mathcal{D}}^{\boldsymbol{\lambda}_j^2} := \mathcal{D} \backslash \mathcal{N}^{\boldsymbol{\lambda}_j^2}$ *and* $i^* := \arg\min_{i \in \overline{\mathcal{D}}^{\boldsymbol{\lambda}_j^2} \cup \{j\}} p_{\boldsymbol{\lambda}^1_{j \oplus i \oplus e_v}}$*, with* $p_{\boldsymbol{\lambda}^1_{j \oplus i^* \oplus e_v}}$ *being the profit of the complete path obtained following the extension of label* $\boldsymbol{\lambda}_j^1$ *to* i^* *and then to* e_v*.*

Proof. Proof. Proof is rather long and is provided in Appendix 7.4. □

Proposition 6.3. *For a given vehicle* $v \in \mathcal{V}^*$*, let* $\boldsymbol{\lambda}_j^\ell = (\lambda_j^{\ell,cost}, \lambda_j^{\ell,time}, \lambda_j^{\ell,load}, \left(\lambda_j^{\ell,cust_i}\right)_{i \in \mathcal{D}})$*,* $\ell = 1, 2$*, be two labels associated with (partial) paths, denoted by* 1 *and* 2*, ending at node* $j \in \mathcal{N}_v \backslash \{e_v\}$*. Let* $\mathcal{N}^{\boldsymbol{\lambda}_j^1}$ *and* $\mathcal{N}^{\boldsymbol{\lambda}_j^2}$ *be the set of nodes visited in partial paths 1 and 2, respectively. Label* $\boldsymbol{\lambda}_j^1$ *is said to dominate label* $\boldsymbol{\lambda}_j^2$ *if the following conditions hold:*

$$(6.16) - (6.19),$$

$$p_{\boldsymbol{\lambda}^1_{j \oplus e_v}} > 0 \tag{6.32}$$

$$\zeta_i \geqslant 0 \quad \forall i \in \overline{\mathcal{D}}^{\boldsymbol{\lambda}_j^2} \tag{6.33}$$

$$\lambda^{1,cost}_{j \oplus e_v} \geqslant \lambda^{2,cost}_{j \oplus e_v} \tag{6.34}$$

where $\overline{\mathcal{D}}^{\boldsymbol{\lambda}_j^2} := \mathcal{D} \backslash \mathcal{N}^{\boldsymbol{\lambda}_j^2}$*.*

Proof. Proof. Proof is rather lengthy and is provided in Appendix 7.4. □

To identify dominated labels, our proposed approach is to first employ Proposition 6.3, i.e., we first check conditions given in Proposition 6.3; If any of them fails then we check conditions given in Proposition 6.2 as they are more time-consuming. As an aside, we note that when applying Proposition 6.2 or Proposition 6.3, it is possible that two labels dominate each other. In terms of implementation, for such a case, one of the labels can be kept and the other one should be discarded.

In our proposed column generation algorithm, for driver $v \in \mathcal{V}^*$ with $\epsilon_v > 0$, we add all complete paths (labels reached e_v) with a positive reduced cost (i.e., $\lambda_{e_v}^{cost} > 0$) to $\widetilde{\mathcal{R}}_v$ and resolve the `NSW-L-RMP`. The column generation algorithm stops when $\epsilon_v \leqslant 0$ for all $v \in \mathcal{V}^*$, in which case no more columns are added and the solution of `NSW-L-RMP` is optimal. We note that the proposed column generation does not guarantee integrality of the solutions. Therefore, a branching mechanism is put in place. We use a branching scheme similar to the one discussed in Section 6.3.1.

6.5 Computational Study

In this section, we study the performance of the proposed approach through several experiments and sensitivity analysis. Specifically, we study the effect of different parameters on the obtained equity among the selected drivers. We implemented the proposed branch and price algorithm in C++ and used CPLEX 12.10 to solve the linear master problems. All the experiments were run on a Dell PowerEdge R640 system with two Intel Xeon (Silver 4116) 2.1GHz 12-core processors, 128GB of memory, and RedHat Enterprise Linux 7.4 operating system. For each experiment, a time limit of four hours was imposed and CPLEX was limited to use only one thread. For those instance for which the algorithm did not converge to an optimal solution, we use the best found solution for analysis. Our instance generator and codes of this study can be found at `XXX` and `XXX`, respectively. For initialization, we used Google OR-Tools [203] for generating a feasible solution to the `MTMC`. The obtained routes are used to initialize $\widetilde{\mathcal{R}}_v$ for each $v \in \mathcal{V}$. In the following sections we will first compare the results of our method with the ones we obtained from the existing max-min approach in the literature. Then, in a set of comprehensive experiments we study the behavior of the model for different values of parameters α, β, and γ with respect to a series of KPIs, including the summation of profits of the drivers, the actual sacrifice of the company in cost, summation of traveling compensation, and finally the range of profit rations among the drivers.

6.5.1 Instances

In this study, we generated 20+60 random instances by adapting instance "R101" of Solomon's vehicle routing problem benchmark [204] with time windows. The first 20 instances with 10 customers and 5 vehicles are relatively small and can be optimally solved using an arc-based formulation, similar to the one provided in Section 7.4. The other 60 instances are larger and are equally divided into six classes with different number of customers and different vehicle capacities. In these classes, the number of customers $|\mathcal{D}|$ is either 16 or 20, and vehicle capacities are assumed to be equal for all drivers $v \in \mathcal{V}$ and selected from $\{50, 75, 100\}$. Note that since the origin and destination of the drivers are different, the fleet remains heterogeneous. The location of customers as well as the origin and destination of the vehicles are determined by selecting random points from "R101". We also used the information of demand in "R101" to set the quantity of deliveries, q_j. Parameter w_j is computed by setting a and b to one in all the experiments. The other parameters of the problem including α, β, and γ have different values for each experiment that will be stated and discussed in the following sections.

6.5.2 NSW vs. Maxmin

We first compare the outcomes of our proposed approach against those of the max-min (the same as min-max discussed in Section 6.1). The selected drivers are determined similarly in both methods. To make sure the comparison is accurate, we use a set of 20 small instances which are solved to optimality in the considered time limit while $\alpha = 5\%$, $\beta = 50\%$, and $\gamma = 4$. The results are reported in Table 6.2. Column "Cost" in this table, shows the cost of the company (i.e., summation of delivery and mileage compensation). The second column reports the profit ratio of drivers and their average and standard deviation for both approaches. We can observe that the average profit ratio of the drivers in the Max-min approach is mostly less than or equal to the one of NSW, as an indication that NSW tends to find equitable solutions in which the individual profits ratios are higher. By looking at the

Table 6.2: NSW v.s. maxmin

Ins.	Cost		Drivers' Profit Ratio									
	Company		Driver #1		Driver #2		Driver #3		Avg.		STD	
	M-M	NSW	M-M	NSW	M-M	NSW	M-M	NSW	M-M	NSW	M-M	NSW
1	337.98	157.86	0.50	0.63	0.15	0.15	0.83	0.69	0.49	0.49	0.338	**0.294**
2	250.30	145.32	0.77	0.77	0.82	0.82	0.88	0.88	0.83	0.83	0.054	0.054
3	257.43	126.05	0.76	0.76	0.89	0.89	0.70	0.70	0.78	0.78	0.097	0.097
4	282.22	156.16	0.58	0.58	0.82	0.82	0.85	0.85	0.75	0.75	0.145	0.145
5	266.08	132.55	0.69	0.81	0.84	0.67	0.71	0.71	0.75	0.73	0.080	**0.075**
6	274.75	151.92	0.85	0.85	0.60	0.60	0.84	0.84	0.76	0.76	0.143	0.143
7	200.76	104.19	0.70	0.70	0.77	0.77	0.93	0.93	0.80	0.80	0.115	0.115
8	222.74	116.16	0.87	0.87	0.77	0.77	0.80	0.80	0.81	0.81	0.049	0.049
9	303.55	136.74	0.63	0.66	0.56	0.56	0.86	0.84	0.68	0.68	0.157	**0.141**
10	352.07	214.10	0.56	0.56	0.68	0.68	0.69	0.69	0.65	0.65	0.076	0.076
11	264.71	148.26	0.56	0.56	0.35	0.35	0.76	0.76	0.56	0.56	0.209	0.209
12	285.23	189.35	0.76	0.76	0.71	0.71	0.88	0.88	0.78	0.78	0.088	0.088
13	284.88	120.51	0.63	0.63	0.86	0.86	0.60	0.60	0.70	0.70	0.142	0.142
14	274.83	162.65	0.87	0.86	0.80	0.81	0.77	0.77	0.81	0.81	0.051	**0.044**
15	266.39	155.56	0.58	0.81	0.91	0.76	0.23	0.23	0.57	0.60	0.338	**0.322**
16	250.59	105.45	0.63	0.63	0.75	0.75	0.76	0.76	0.71	0.71	0.070	0.070
17	233.68	129.41	0.87	0.87	0.85	0.85	0.15	0.15	0.62	0.62	0.414	0.414
18	313.45	201.07	0.90	0.90	0.70	0.70	0.24	0.24	0.61	0.61	0.335	0.335
19	268.00	148.76	0.62	0.62	0.77	0.77	0.83	0.83	0.74	0.74	0.109	0.109
20	259.39	126.81	0.37	0.37	0.64	0.64	0.86	0.86	0.63	0.63	0.246	0.246
Avg.	272.45	146.44	0.68	0.71	0.71	0.70	0.71	0.70	0.70	0.70	0.163	0.158

standard deviation of the profit ratios among the drivers, lower values of NSW method can be translated into a more equitable distribution of profits among the drivers. Interestingly, the cost of the company is higher in maxmin method. This is because the drivers other than the one that has the least profit are not considered in the optimization and can travel a non-optimal route. This entails a higher cost for the company (because of the mileage compensation) but not necessarily a higher profit for the drivers.

6.5.3 Hypotenuse Cut Evaluation

To evaluate the effect of hypotenuse cut we add for each feasible solution that the branch-and-price algorithm finds for NSW program, we report the computational time for 20 small instances. Table 6.3 shows the total computational time with and without the hypotenuse cut.

Table 6.3: The effect of hypotenuse cut on the computational time

Instance	**Com. Time W. Cut**	**Com. Time W.O. Cut**
1	1.793	1.734
2	2.302	2.167
3	2.006	2.023
4	1.737	1.886
5	1.881	1.882
6	2.238	2.306
7	2.673	2.591
8	1.485	1.481
9	12.409	12.253
10	2.151	2.011
11	2.577	2.773
12	1.372	1.366
13	1.549	1.661
14	8.459	8.466
15	2.422	2.416
16	2.122	2.221
17	2.459	2.358
18	12.171	12.240
19	1.788	1.788
20	2.311	2.372

6.5.4 Sensitivity Analysis on the Company's Allowable Cost of Equity: α

Next, we look at the difference that the flexibility of the company in its cost can make in terms of the level of equity in solutions. To this end, we study the company's efficiency, drivers' efficiency, and the level of equity among the drivers for different levels of α, given set values $\beta = 0.99$ and $\gamma = 4$. Table 6.4 shows the average sum of the drivers profit for each instance class including when the equity requirements are disregarded, i.e., the solution of MTMC. For the sake of simplicity in comparison, we report the percentage of changes in this measure for NSW solution with respect to MTMC. These information are reflected in the last six columns of the table. Columns "# customers" and "Q_v" show the number of customers and vehicle capacity in each class, while the third column shows the average CPU time for `MTMC`. Column "$\mathcal{V}^*$" reports the average number of drivers, and the rest of the table shows the average of total drivers profit. For `MTMC`, we report the absolute value of the average sum of drivers' benefits, while for NSW, the percentage of the change w.r.t. `MTMC` for different values of α is reported. By comparing `MTMC` and NSW when $\alpha = 0$, we observe that, except for the sixth class of instances, total profit of drivers in NSW's remains unchanged. When $\alpha > 0$, more opportunities to improve the equity arise. A better level of equity comes at a cost that is mostly paid by the company. However, a small portion $(1-\beta)$ of the cost of equity is paid by the drivers. This explains the fact that the average sum of drivers' benefits has slightly decreased for the solutions of NSW with $\alpha > 0$ compared to `MTMC`. Another observation from Table 6.4 is that the average sum of drivers' profit drops more as the flexibility of the company (α) increases. A higher α translates into a better chance to achieve a higher level of equity, and consequently a higher cost of equity which will be shared between the company and drivers.

For the same experiments, we studied the variation of company's actual sacrifice made over the maximum flexibility it offers, α. Figure 6.2a shows boxplots for the percentage of the amount that the company is willing to spend $(z^*\alpha)$ that is actually used in the solution of NSW on equity for different thresholds (α). Due to the discrete nature of the VRP, the

Table 6.4: The average sum of drivers' profit for different values of α

# Customers	Q_v	MTMC CPU time	$\mathcal{V}^*$	Summation of profits					
				MTMC	NSW				
					$\alpha = 0\%$	$\alpha = 2.5\%$	$\alpha = 5\%$	$\alpha = 7.5\%$	$\alpha = 10\%$
16	50	591.60	4.2	560.41	0.0000	-0.0077	-0.0248	-0.0340	-0.0489
16	75	264.62	3.4	596.75	0.0000	-0.0063	-0.0169	-0.0209	-0.0311
16	100	452.45	2.8	602.49	0.0000	-0.0069	-0.0120	-0.0162	-0.0205
20	50	3118.47	5.0	694.75	0.0000	-0.0082	-0.0187	-0.0339	-0.0458
20	75	4881.86	4.4	751.86	0.0000	-0.0094	-0.0172	-0.0314	-0.0411
20	100	4395.40	3.2	765.40	0.0190	0.0160	0.0104	0.0069	0.0021

maximum level of equity is not necessarily achieved by a routing solution that maxes out the available equity budget, $z^*\alpha$. On average, 50% of the instances consume 71.60% of the available budget for maximizing equity.

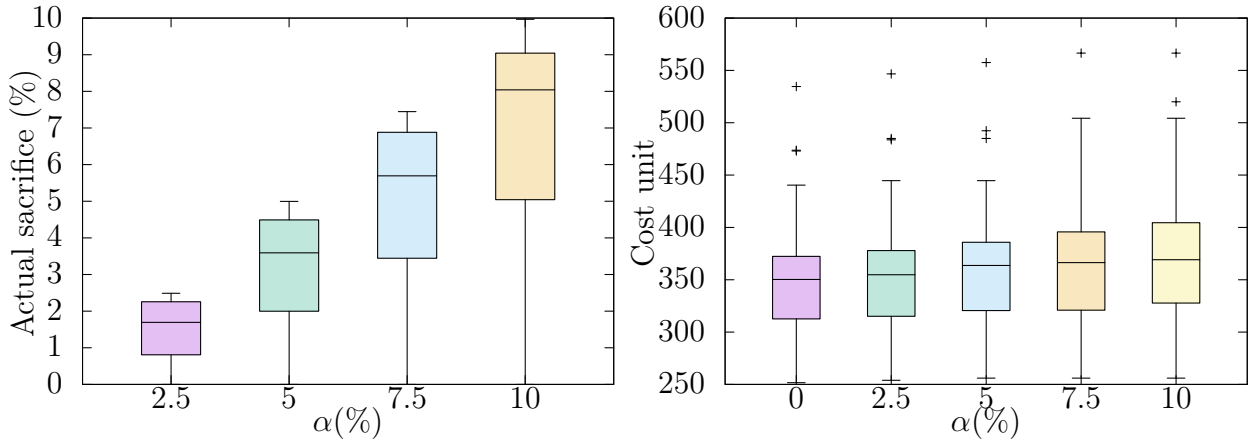
As we discussed in Section 6.2, the compensation of delivery tasks has two parts. One part is the prize of fulfilling the delivery tasks whose summation is fixed, independent from task assignment since all the delivery tasks must be completed. The other part of the compensation corresponds to the mileage cost that depends on delivery task-to-driver assignments and the routing decisions for each driver. When $\beta \approx 1$, the entire mileage cost of the drivers is reimbursed by the company where the sacrifice of the company is being spent. In other words, a part of the company's sacrifice is spent to eliminate the sources of heterogeneity among the drivers, more specifically, their differences in terms of their origins and destinations. Considering the fact that the compensation for fulfilling all the deliveries is fixed, the efficiency of drivers can be better analyzed by summing the compensated traveling costs. Figure 6.2b shows the boxplots of drivers travelling compensation obtained by `NSW` under different flexibility allowance of the company. The horizontal axis in this graph represents α, and the vertical axis is the compensated traveling cost. By comparing the medians of these box-plots, one can observe that the travel compensation increasing with the company's flexibility as more budget becomes available for equity among drivers. From a technical perspective, increasing α relaxes Constraint (6.1c), and makes more solutions with longer and at the same time possibly fairer routes feasible for the model. In this experiment, most of the cost for these longer routes are paid by the company, which leads to the rising

box-plots in Figure 6.2b. As we discussed in Table 6.4, drivers bear a small part of this cost increase, depending on the part of the mileage cost that is nor reimbursed $(1 - \beta)$.

Equity among drivers is also investigated in the obtained solutions. For this purpose, we study the range of drivers profit ratios which is the difference between the maximum and minimum profit ratios among all drivers, i.e., $max_{v \in \mathcal{V}*}\{\rho_v\} - min_{v \in \mathcal{V}*}\{\rho_v\}$. To reflect the effect of company's flexibility in the equitable assignment of delivery tasks, the boxplots for the range of drivers profit ratios are shown in Figure 6.2c. To compute the maximum achievable profit for each driver, we replaced `NSWP-Master`'s objective function by a certain driver's profit. One can observe that this measure has a decreasing trend. This is because the chances to find more equitable solutions (i.e., with closer profit ratios) become higher as the company's flexibility increases. Another interesting observation is that sometimes the variability of this measure increases when the company's flexibility has grown although the median is always dropping. Specifically, the size of the box is not always shrinking. For example, from $\alpha = 2.5\%$ to $\alpha = 5\%$ the difference of the first and third quartiles has increased. It is because NSW objective function optimizes the efficiency and equity at the same time. That is, with a larger α, the model may find solutions that improve more on the efficiency than the equity.

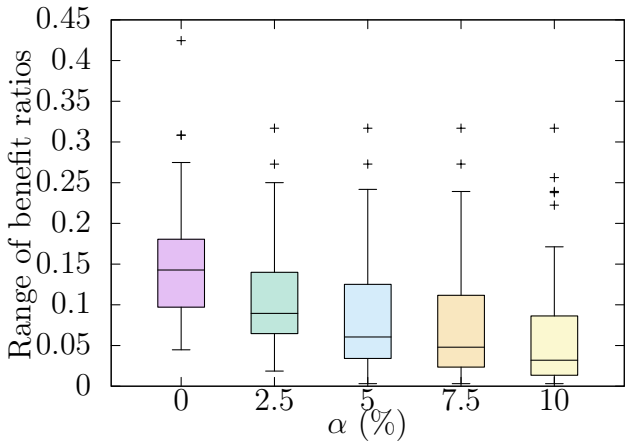
6.5.5 Sensitivity Analysis on β

In this section, we study the effect of variations in the value of parameter β on the model's behaviour. A lower β means less effort from the company to bring about profit equity among the drivers, since it means leaving a higher part of the mileage cost to the drivers (see Figure 6.3b). Specifically, with a lower β, higher portion of the equity cost will be taken over by the drivers. To understand this better, one should remember that, first, in our experiments the source of heterogeneity among drivers originates from their different origins and destinations, and second, a fairer job assignment may involve a higher mileage cost. For example, when $\beta \approx 0$, the solution of NSW will be a pure cooperation of the drivers



(a) Actual sacrifice of the company

(b) Overall traveling compensation of drivers



(c) Drivers profit ratio

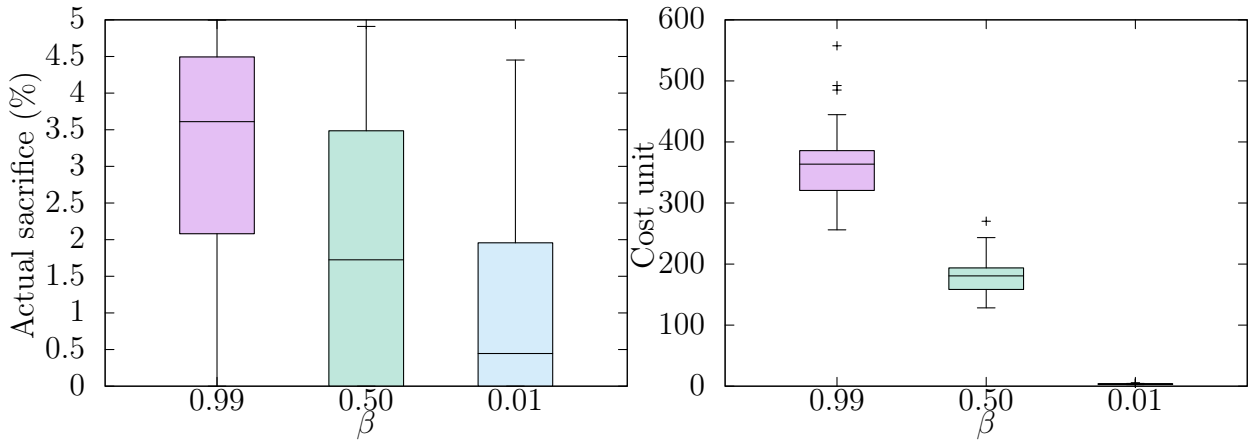
Figure 6.2: Sensitivity analysis for different values of α when $\beta = 0.99$ and $\gamma = 400\%$

to equally distribute the profit opportunity among themselves. Table 6.5 shows the same columns as those in Table 6.4 for three values of $\beta = \{0.99, 0.50, 0.01\}$. As one can notice, the average of the drivers total profit mostly drops as β decreases. In the same vein, Figure 6.3c shows that the range for profit ratios increases, that is, the level of equity deteriorates.

Comparing the boxplots for $\beta = 0.99$ and $\beta = 0.01$ illustrates the effect of company's compensation policy for the mileage cost on equity among drivers. From $\beta = 0.99$ to $\beta = 0.01$, the average range of profit ratios has increased by 91.16%. This leads to the overall conclusion that when α is fixed, with a lower β the chances to find an equitable solution diminish. Therefore, the actual sacrifice of the company has a decreasing trend (Figure 6.3a) too, meaning the available budget to find equitable solutions will no be fully used. We plot the routes for two extreme values of β, i.e., $\beta = 0.99$ and $\beta = 0.01$ in Figure 6.4. In this figure, all the signs and numbers serve similar to Figure 6.4b. By comparing these two cases, we first observe that the change in percentage of mileage cost that is reimbursed by the company results in changes in the assignment of delivery tasks to different drivers. Specifically, we can observe that the farther customers are removed from the routes of drivers that gain less profit when β decreases. For example, Driver 4 (the blue dotted line) loses demand points indicated by 8 and 27 to Driver 7 because Driver 7 will travel significantly shorter route to serve them. In summary, reimbursing a larger portion of the mileage cost by the company (larger β) creates more opportunities to achieve equitable profit equity among the drivers.

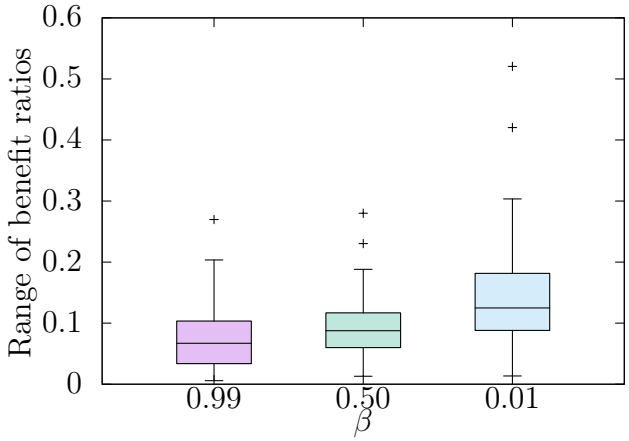
Table 6.5: Average of the drivers profit

# demand	Q_v	$\beta = 0.99$			$\beta = 0.50$			$\beta = 0.01$		
		$\mathcal{V}^*$	MTMC	NSW	$\mathcal{V}^*$	MTMC	NSW	$\mathcal{V}^*$	MTMC	NSW
16	50	4.2	560.388	-0.023	4.2	386.273	-1.098	4.2	212.288	-2.083
16	75	3.5	596.721	-0.019	3.5	440.912	-0.533	3.5	286.305	-1.117
16	100	2.8	602.490	-0.012	2.8	455.164	-0.487	2.8	307.838	-1.256
20	50	5.1	694.676	-0.019	5.1	490.820	-0.846	5.1	286.963	-2.928
20	75	4.4	751.825	-0.019	4.4	565.342	-0.826	4.4	378.859	-1.209
20	100	3.3	761.509	-0.011	3.3	596.173	-0.398	3.3	430.370	-0.038



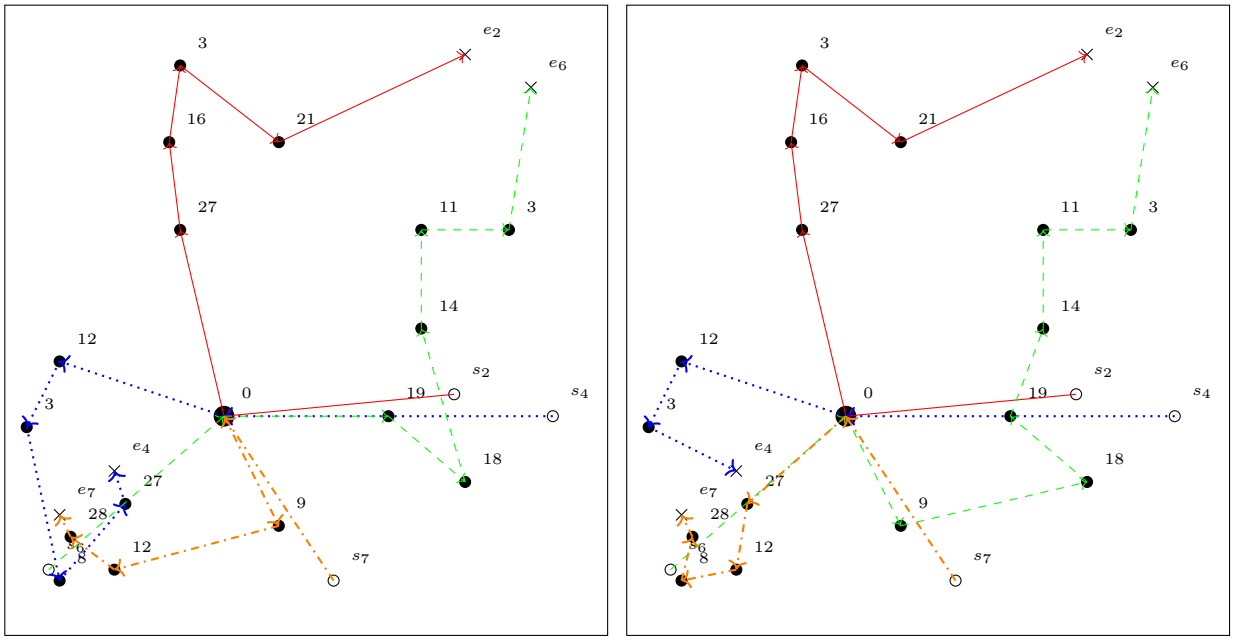
(a) Actual sacrifice of the company

(b) Overall traveling compensation of drivers



(c) Drivers profit ratio

Figure 6.3: Sensitivity analysis for different values of β when $\alpha = 5\%$ and $\gamma = 400\%$



(a) Solution of NSW when $\beta = 0.99$ (b) Solution of NSW when $\beta = 0.01$

Figure 6.4: Solution of NSW for an instance when $\beta = 0.99$ v.s. $\beta = 0.01$

6.5.6 Sensitivity Analysis on γ

Finally, we study the effect of drivers time flexibility, i.e. γ on the same performance metrics when $\alpha = 5\%$ and $\beta = 0.90$. From a modeling perspective, this parameter is a budget for a resource, and the larger it becomes, the better results we must obtain. Table 6.6 reports the average profit of all drivers for `MTMC` and `NSW` (the changes for NSW in particular) under three scenarios for γ, 2, 3, and 4, when $\alpha = 5\%$ and $\beta = 0.50$. We observe that the drivers' total profit of `MTMC` solution rises when γ increases. This is mainly because for smaller values of γ the company has to deploy more drivers.We also observe that the total profit of drivers exhibit a decreasing trend. This is mainly due to the fact that a larger γ results in employing a smaller number of drivers in MTMC. Since in our setting the NSW model is constrained to use the same drivers as the MTMC model, the average traveled distance of each driver would increase, which could result in a lower total profit of the drivers when $\beta < 1$.

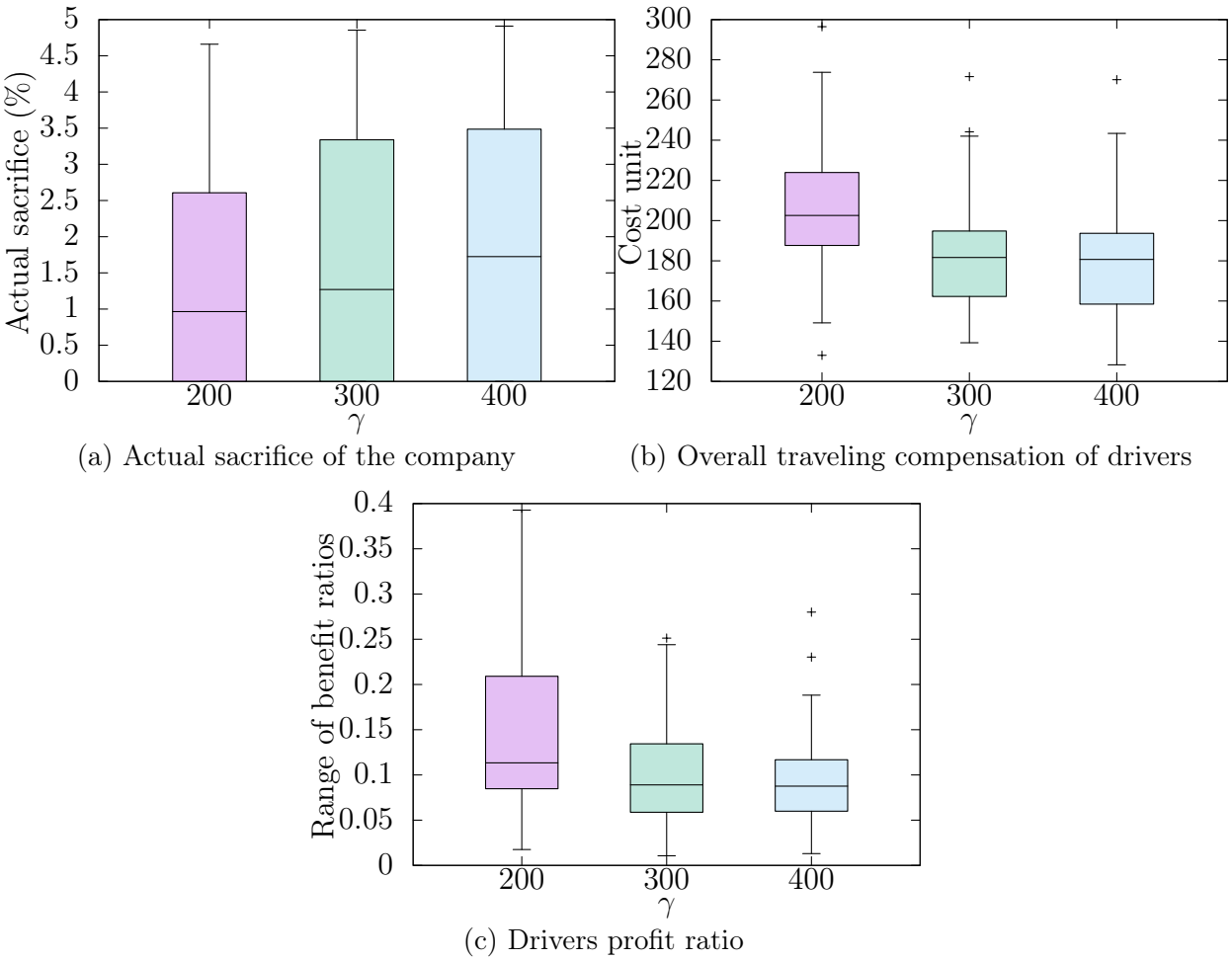
The same conclusion can be drawn from Figure 6.5b where the mileage compensation of drivers exponentially drops as γ increases. However, if we remove the effect of dimension and look at the result proportionally, we observe that the proposed model gives the desired results. For example, Figure 6.5a shows that the model takes the most advantage of the company's flexibility as β increases. This allows to improve the equity among the drivers and can be observed in Figure 6.5c. Specifically, the range for profit ratios of drivers decreases, testifying a higher level of equity, as γ grows.

Table 6.6: Average of the drivers profit

# demand	Q_v	$\gamma = 200\%$			$\gamma = 3$			$\gamma = 4$		
		$\mathcal{V}^*$	MTMC	NSW	$\mathcal{V}^*$	MTMC	NSW	$\mathcal{V}^*$	MTMC	NSW
16	50	5.0	362.256	-0.890	4.3	384.418	-1.309	4.2	386.273	-1.098
16	75	4.0	422.018	-0.406	3.5	438.083	-0.615	3.5	440.912	-0.533
16	100	3.6	420.819	-0.534	2.9	449.800	-0.451	2.8	455.164	-0.487
20	50	5.3	462.481	-0.423	5.0	489.307	-0.574	5.1	490.820	-0.846
20	75	5.0	539.701	-0.612	4.4	561.785	-0.630	4.4	565.342	-0.826
20	100	4.5	561.647	-0.886	3.3	591.767	-0.301	3.3	596.173	-0.398

6.6 Conclusion

In this paper, we studied the problem of improving workload allocation equity in the VRP. The need for creating equal opportunities for profit making among heterogeneous personnel is growing as more companies are adopting crowd-sourced workforce. Any improvement in the level of workload allocation equity usually comes at a cost, referred to as the equity cost. Therefore, the desired equity among the employed (crowd-sourced) employees is achievable only if the employer is willing to deviate from the least cost solution, to cover the equity cost. To this end, we proposed the first formulation of the VRP under the form of a Nash social welfare problem, by imposing a cap on the cost of equity paid by the company. We proposed an exact approach based on branch-and-price framework to solve both the company's baseline and NSW problems. We conducted a series of sensitivity analysis to study the behaviour of the model w.r.t. variations in the level of company's flexibility



(a) Actual sacrifice of the company

(b) Overall traveling compensation of drivers

(c) Drivers profit ratio

Figure 6.5: Sensitivity analysis for different values of γ when $\alpha = 5\%$ and $\beta = 0.9$

in paying for the cost of equity and in the structure of the compensation function. Our results show that the proposed method can effectively improve equity in workload allocation and consequently profit among a fleet of heterogeneous drivers. Due to the exact nature of the proposed algorithm, the size of instances that can be solved within a reasonable amount of time is limited. Therefore, our future work will focus on developing hybrid approaches that rely on the framework of branch-and-price, and can generate "good" solutions within short amount of times. Other future work directions include extending the proposed setting into ones that promote equity in service to service beneficiaries (customers) too.

Chapter 7: Conclusions and Future Research Directions

Here we summarize the conclusions of the presented works in this dissertation and provide directions for future research.

7.1 Rebalancing Problem in Free-Floating Bike Sharing Systems

In this dissertation, we proposed a hybrid rebalancing strategy, i.e., combining an incentive program and an operator-based rebalancing and dynamic hubbing concept, i.e., determining daily hubs (number and locations) to assist hybrid rebalancing. We presented a multi-objective simulation optimization method for solving the proposed strategy for free-floating bike sharing systems by considering two objectives—cost and service level. The optimized outputs are the number and location of hubs, the start time of incentive program, and incentive level. Using the solution algorithm proposed in our previous work, we also obtained the routing of rebalancing trucks and the number of bikes being picked up or dropped off at each stop along the routes.

In future research, one can apply user-based redistribution for rebalancing bikes during the whole day according to demand distribution of the near future (for example, two hours ahead) instead of the following day. Additionally, obtaining and using the offer-acceptance behavior of users by artificial intelligence based on his/her record is another direction of future research.

7.2 Query Batching Optimization in Database Systems

In this dissertation, we studied the query batching problem. This problem aims at partitioning a given set of queries into some batches before retrieving them from a database

system in order to minimize the total retrieving/processing time. This optimization problem is challenging because predicting the time required for processing a given batch of queries is not a trivial task. We developed a generic three-phase approach for solving the problem for any given database system. In the first phase, using a quadratic function, our approach attempts to predict the processing time of any batch of queries for the given database. In the second phase, our approach uses the obtained quadratic function and develops a mixed binary quadratic programming formulation for the query batching problem accordingly. Finally, in the last phase, our proposed approach uses two custom-built heuristic approaches, i.e., RCSA-I and RCSA-II, to quickly solve the obtained formulation in practice. We tested our proposed approach on three well-known database benchmarks by conducting a comprehensive computational study. The results showed that a reduction of up to 61.8% is achievable for the total processing times in the database benchmarks when employing our proposed approach.

We hope that the simplicity, versatility, and performance of our proposed approach encourage practitioners/researchers to consider employing/developing effective query batching optimizers. There are several future research directions that can be considered for this study. One direction can be developing better exact or heuristic solution approaches for solving instances with a larger number of queries. Alternatively, developing effective machine learning methods for predicting the outcome, i.e., an optimal solution, of the proposed mixed binary quadratic program can be an interesting research direction too. Another research direction can be developing some theories and/or methodologies for identifying an optimal amount of time that one should wait to accumulate queries before starting to solve the query batching problem.

7.3 Solving Mixed Integer Linear Minimum/Maximum Multiplicative Programs

We studied a class of single-objective non-convex non-linear (mixed) integer optimization problems, i.e., MIL-mMPs, with applications in different fields of study, e.g., natural resource management. We showed that because the objective function of a MIL-mMP is a

multiplication of p non-negative linear functions, a MIL-mMP can be viewed as the problem of optimization over the efficient set of a multi-objective mixed-integer linear program with p objective functions. Consequently, we developed two novel multi-objective optimization-based algorithms for solving MIL-mMPs. The first algorithm is a decision space search algorithm that relies on the power of (single-objective) linear programming solvers for solving a MIL-mMP. The second algorithm, however, is a criterion space search algorithm that relies on the power of (single-objective) mixed integer linear programming solvers for solving a MIL-mMP. We also developed several enhancement techniques for the proposed algorithms and tested the performance of our algorithms on 960 random instances against the non-linear optimization solver of SCIP. Our numerical results showed that SCIP is significantly outperformed by our proposed algorithms.

We hope that our proposed algorithms inspire practitioners/researchers to develop multi-objective optimization-based algorithms for solving other single-objective optimization problems. There are several future research directions that can be considered for this study. One direction could be exploring whether the proposed algorithms can be combined together through an effective parallelization framework to integrate their advantages. Another research direction is to explore how the proposed methods can be customized effectively for solving the so-called 'generalized' MIL-mMPs in which the only difference is that each term in the objective function may have positive power.

In another work, we focused on a class maximum multiplicative program, MIBL-MMP, involving both continuous and integer variables and introduced a generic criterion space branch-and-cut algorithm for it. The proposed algorithm, CSBnC, employs novel dual-bound and primal-bound computing operations based on the concept of piecewise McCormick envelopes. It also employs a highly effective cut-generating mechanism. Through a computational study, we demonstrated the impacts of the main components of our algorithm and numerically showed how its main parameters can be tuned. Our numerical analysis showed that tuning can improve the solution time of our algorithm by a factor of

11.78 on average. Moreover, CSBnC outperforms the mixed-integer SOCP solver of CPLEX 12.10 and a state-of-the-art algorithm in the literature by factors 6.54 and 7.54 on average, respectively. The main future research direction of this study is how the proposed solution method can be customized when the dimension of the criterion space is more than two, i.e., $p > 2$.

7.4 Equitable Vehicle Routing Problem with Heterogeneous Fleet

Finally, we studied the problem of improving workload allocation equity in the VRP. The need for creating equal opportunities for profit making among heterogeneous personnel is growing as more companies are adopting crowd-sourced workforce. Any improvement in the level of workload allocation equity usually comes at a cost, referred to as the equity cost. Therefore, the desired equity among the employed (crowd-sourced) employees is achievable only if the employer is willing to deviate from the least cost solution, to cover the equity cost. To this end, we proposed the first formulation of the VRP under the form of a Nash social welfare problem, by imposing a cap on the cost of equity paid by the company. We proposed an exact approach based on a branch-and-price framework to solve both the company's baseline and NSW problems. We conducted a series of sensitivity analyses to study the behavior of the model w.r.t. variations in the level of the company's flexibility in paying for the cost of equity and in the structure of the compensation function. Our results show that the proposed method can effectively improve equity in workload allocation and consequently profit among a fleet of heterogeneous drivers. Due to the exact nature of the proposed algorithm, the size of instances that can be solved within a reasonable amount of time is limited. Therefore, a future work can focus on developing hybrid approaches that rely on the framework of branch-and-price, and can generate "good" solutions within a short amount of time. Other future work directions include extending the proposed setting into ones that promote equity in service to service beneficiaries (customers) too.

References

[1] L. Shao and M. Ehrgott, "Primal and dual multi-objective linear programming algorithms for linear multiplicative programmes," *Optimization*, vol. 65, no. 2, pp. 415–431, 2016.

[2] B. Boyacı, K. G. Zografos, and N. Geroliminis, "An integrated optimization-simulation framework for vehicle and personnel relocations of electric carsharing systems with reservations," *Transportation Research Part B: Methodological*, vol. 95, pp. 214–237, 2017.

[3] L. Méndez, D. Hernández, G. Villarrubia, and J. F. de Paz, "Multi-agent system for the control and monitoring of shared bicycle fleets," in *International Symposium on Ambient Intelligence*. Springer, 2017, pp. 187–194.

[4] L. Caggiani, R. Camporeale, M. Marinelli, and M. Ottomanelli, "User satisfaction based model for resource allocation in bike-sharing systems," *Transport Policy*, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0967070X17305322

[5] P. Vogel, J. F. Ehmke, and D. C. Mattfeld, "Cost-efficient allocation of bikes to stations in bike sharing systems," in *International Conference on Computational Logistics*. Springer, 2017, pp. 498–512.

[6] S. Shaheen, S. Guzman, and H. Zhang, "Bikesharing in europe, the americas, and asia: past, present, and future," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2143, pp. 159–167, 2010.

[7] F. Richter. (2018) Bike-sharing clicks into higher gear. [Online]. Available: https://www.statista.com/chart/14542/bike-sharing-programs-worldwide/

[8] H. R. Sayarshad and J. Y. Chow, "A scalable non-myopic dynamic dial-a-ride and pricing problem," *Transportation Research Part B: Methodological*, vol. 81, pp. 539–554, 2015.

[9] A. Pal and Y. Zhang, "Free-floating bike sharing: solving real-life large-scale static rebalancing problems," *Transportation Research Part C: Emerging Technologies*, vol. 80, pp. 92–116, 2017.

[10] J. Y. Chow and H. R. Sayarshad, "Symbiotic network design strategies in the presence of coexisting transportation networks," *Transportation Research Part B: Methodological*, vol. 62, pp. 13–34, 2014.

[11] P. DeMaio, "Bike-sharing: History, impacts, models of provision, and future," *Journal of Public Transportation*, vol. 12, no. 14, pp. 41–56, 2009.

[12] L. Caggiani and M. Ottomanelli, "A dynamic simulation based model for optimal fleet repositioning in bike-sharing systems," *Procedia-Social and Behavioral Sciences*, vol. 87, pp. 203–210, 2013.

[13] H. Sayarshad, S. Tavassoli, and F. Zhao, "A multi-periodic optimization formulation for bike planning and bike utilization," *Applied Mathematical Modelling*, vol. 36, no. 10, pp. 4944–4951, 2012.

[14] T. Raviv, M. Tzur, and I. A. Forma, "Static repositioning in a bike-sharing system: models and solution approaches," *EURO Journal on Transportation and Logistics*, vol. 2, no. 3, pp. 187–229, 2013.

[15] M. Dell'Amico, M. Iori, S. Novellani, and A. Subramanian, "The bike sharing rebalancing problem with stochastic demands," *Transportation research part B: methodological*, vol. 118, pp. 362–380, 2018.

[16] C. Kloimüllner and G. R. Raidl, "Full-load route planning for balancing bike sharing systems by logic-based benders decomposition," *Networks*, vol. 69, no. 3, pp. 270–289, 2017.

[17] Y. Peng and B. Liu, "Research on the pre-peak scheduling problem of public bicycle system with branch-and-price algorithm," *International Journal of Wireless and Mobile Computing*, vol. 12, no. 3, pp. 239–244, 2017.

[18] D. Chemla, F. Meunier, T. Pradeau, R. W. Calvo, and H. Yahiaoui, "Self-service bike sharing systems: simulation, repositioning, pricing," 2013.

[19] C. Fricker and N. Gast, "Incentives and redistribution in homogeneous bike-sharing systems with stations of finite capacity," *Euro journal on transportation and logistics*, vol. 5, no. 3, pp. 261–291, 2016.

[20] L. Li and M. Shan, "Bidirectional incentive model for bicycle redistribution of a bicycle sharing system during rush hour," *Sustainability*, vol. 8, no. 12, p. 1299, 2016.

[21] A. Angelopoulos, D. Gavalas, C. Konstantopoulos, D. Kypriadis, and G. Pantziou, "Incentivized vehicle relocation in vehicle sharing systems," *Transportation Research Part C: Emerging Technologies*, vol. 97, pp. 175–193, 2018.

[22] Z. Haider, A. Nikolaev, J. E. Kang, and C. Kwon, "Inventory rebalancing through pricing in public bike sharing systems," *European Journal of Operational Research*, vol. 270, no. 1, pp. 103–117, 2018.

[23] J. Pfrommer, J. Warrington, G. Schildbach, and M. Morari, "Dynamic vehicle redistribution and online price incentives in shared mobility systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 4, pp. 1567–1578, 2014.

[24] Y. Shen, X. Zhang, and J. Zhao, "Understanding the usage of dockless bike sharing in singapore," *International Journal of Sustainable Transportation*, pp. 1–15, 2018.

[25] K. Kortum and R. B. Machemehl, "Free-floating carsharing systems: innovations in membership prediction, mode share, and vehicle allocation optimization methodologies," Citeseer, Tech. Rep., 2012.

[26] A. Pal, Y. Zhang, and C. Kwon, "Analysis of free-floating bike sharing and insights on system operations or analyzing mobility patterns and imbalance of free floating bike sharing systems," 2018.

[27] L. Caggiani, R. Camporeale, and M. Ottomanelli, "A dynamic clustering method for relocation process in free-floating vehicle sharing systems," *Transportation Research Procedia*, vol. 27, pp. 278–285, 2017.

[28] L. Caggiani, R. Camporeale, M. Ottomanelli, and W. Y. Szeto, "A modeling framework for the dynamic management of free-floating bike-sharing systems," *Transportation Research Part C: Emerging Technologies*, vol. 87, pp. 159–182, 2018.

[29] M. Du, L. Cheng, X. Li, and F. Tang, "Static rebalancing optimization with considering the collection of malfunctioning bikes in free-floating bike sharing system," *Transportation Research Part E: Logistics and Transportation Review*, vol. 141, p. 102012, 2020.

[30] R. Regue and W. Recker, "Proactive vehicle routing with inferred demand to solve the bikesharing rebalancing problem," *Transportation Research Part E: Logistics and Transportation Review*, vol. 72, pp. 192–209, 2014.

[31] D. Zhang, C. Yu, J. Desai, H. Lau, and S. Srivathsan, "A time-space network flow approach to dynamic repositioning in bicycle sharing systems," *Transportation research part B: methodological*, vol. 103, pp. 188–207, 2017.

[32] M. Bierlaire, "Simulation and optimization: A short review," *Transportation Research Part C: Emerging Technologies*, vol. 55, pp. 4–13, 2015.

[33] Y. Carson and A. Maria, "Simulation optimization: methods and applications," in *Proceedings of the 29th conference on Winter simulation*. IEEE Computer Society, 1997, pp. 118–126.

[34] M. C. Fu, F. W. Glover, and J. April, "Simulation optimization: a review, new developments, and applications," in *Proceedings of the Winter Simulation Conference, 2005.* IEEE, 2005, pp. 13–pp.

[35] F. Azadivar, "Simulation optimization methodologies," in *WSC'99. 1999 Winter Simulation Conference Proceedings.'Simulation-A Bridge to the Future'(Cat. No. 99CH37038)*, vol. 1. IEEE, 1999, pp. 93–100.

[36] W. T. de Sousa Junior, J. A. B. Montevechi, R. de Carvalho Miranda, and A. T. Campos, "Discrete simulation-based optimization methods for industrial engineering problems: A systematic literature review," *Computers & Industrial Engineering*, vol. 128, pp. 526–540, 2019.

[37] B. W. Landis, "Bicycle system performance measures," *ITE journal*, vol. 66, no. 2, pp. 18–26, 1996.

[38] L. D. MEng *et al.*, "Implementing bike-sharing systems," *Proceedings of the Institution of Civil Engineers*, vol. 164, no. 2, p. 89, 2011.

[39] I. Frade and A. Ribeiro, "Bicycle sharing systems demand," *Procedia-Social and Behavioral Sciences*, vol. 111, pp. 518–527, 2014.

[40] Y. Xu, S.-L. Shaw, Z. Fang, and L. Yin, "Estimating potential demand of bicycle trips from mobile phone data-an anchor-point based approach," *ISPRS International Journal of Geo-Information*, vol. 5, no. 8, p. 131, 2016.

[41] Z. Yang, J. Hu, Y. Shu, P. Cheng, J. Chen, and T. Moscibroda, "Mobility modeling and prediction in bike-sharing systems," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 2016, pp. 165–178.

[42] L. Lin, Z. He, and S. Peeta, "Predicting station-level hourly demand in a large-scale bike-sharing network: A graph convolutional neural network approach," *Transportation Research Part C: Emerging Technologies*, vol. 97, pp. 258–276, 2018.

[43] Y. Liu, R. Jia, X. Xie, and Z. Liu, "A two-stage destination prediction framework of shared bicycles based on geographical position recommendation," *IEEE Intelligent Transportation Systems Magazine*, vol. 11, no. 1, pp. 42–47, 2018.

[44] Z. Li, J. Zhang, J. Gan, P. Lu, Z. Gao, and W. Kong, "Large-scale trip planning for bike-sharing systems," *Pervasive and Mobile Computing*, vol. 54, pp. 16–28, 2019.

[45] Z. Azarmand and E. Neishabouri, "Location allocation problem," in *Facility location*. Springer, 2009, pp. 93–109.

[46] M. Mahajan, P. Nimbhorkar, and K. Varadarajan, "The planar k-means problem is np-hard," in *International Workshop on Algorithms and Computation*. Springer, 2009, pp. 274–285.

[47] A. Saxena, M. Prasad, A. Gupta, N. Bharill, O. P. Patel, A. Tiwari, M. J. Er, W. Ding, and C.-T. Lin, "A review of clustering techniques and developments," *Neurocomputing*, vol. 267, pp. 664 – 681, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231217311815

[48] R. Z. Farahani and M. Hekmatfar, *Facility location: concepts, models, algorithms and case studies*. Springer, 2009.

[49] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[50] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[51] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary computation*, vol. 2, no. 3, pp. 221–248, 1994.

[52] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[53] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 32–49, 2011.

[54] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[55] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.

[56] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM review*, vol. 59, no. 1, pp. 65–98, 2017. [Online]. Available: https://doi.org/10.1137/141000671

[57] Link, "Global database industry market size 2012-2017," Apr 2015. [Online]. Available: https://www.statista.com/statistics/724611/worldwide-database-market/

[58] K. P. Triantis and C. J. Egyhazy, "An integer programming formulation embedded in an algorithm for query processing optimization in distributed relational database systems," *Computers & Operations Research*, vol. 15, no. 1, pp. 51 – 60, 1988.

[59] J. N. Tsitsiklis and K. Xu, "Flexible queuing architectures," *Operations Research*, vol. 65, no. 5, pp. 1398–1413, 2017.

[60] K. D. Seppi, J. W. Barnes, and C. N. Morris, "A bayesian approach to database query optimization," *ORSA journal on Computing*, vol. 5, no. 4, pp. 410–419, 1993.

[61] M. Eslami, Y. Tu, and H. Charkghard, "A system for batched query processing and optimization," CSE Dept., University of South Florida, Tampa, FL, USA, Tech. Rep. CSE/18-088, URL: www.csee.usf.edu/~tuy/pub/tech18-088.pdf, 2019.

[62] G. Giannikis, G. Alonso, and D. Kossmann, "Shareddb: Killing one thousand queries with one stone," *Proc. VLDB Endow.*, vol. 5, no. 6, pp. 526–537, 2012.

[63] S. Arumugam, A. Dobra, C. M. Jermaine, N. Pansare, and L. Perez, "The datapath system: A data-centric analytic processing engine for large data warehouses," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '10, 2010, pp. 519–530.

[64] T. K. Sellis, "Multiple-query optimization," *ACM Trans. Database Syst.*, vol. 13, no. 1, pp. 23–52, Mar. 1988. [Online]. Available: http://doi.acm.org/10.1145/42201.42203

[65] S. Harizopoulos, V. Shkapenyuk, and A. Ailamaki, "Qpipe: A simultaneously pipelined relational query engine," in *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '05. ACM, 2005, pp. 383–394.

[66] G. Wäscher, *Order Picking: A Survey of Planning Problems and Methods*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 323–347. [Online]. Available: https://doi.org/10.1007/978-3-540-24815-6_15

[67] M. Yu and R. B. de Koster, "The impact of order batching and picking area zoning on order picking system performance," *European Journal of Operational Research*, vol. 198, no. 2, pp. 480 – 490, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221708007601

[68] H. D. Ratliff and A. S. Rosenthal, "Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem," *Operations Research*, vol. 31, no. 3, pp. 507–521, 1983.

[69] R. A. Ruben and F. R. Jacobs, "Batch construction heuristics and storage assignment strategies for walk/ride and pick systems," *Management Science*, vol. 45, no. 4, pp. 575–596, 1999.

[70] L. Pansart, N. Catusse, and H. Cambazard, "Exact algorithms for the order picking problem," *Computers & Operations Research*, vol. 100, pp. 117 – 127, 2018.

[71] E. F. Codd, "A relational model of data for large shared data banks," *Commun. ACM*, vol. 13, no. 6, pp. 377–387, Jun. 1970. [Online]. Available: http://doi.acm.org/10.1145/362384.362685

[72] TPC, "Tpc benchmark by transaction processing performance council," 2001. [Online]. Available: http://www.tpc.org

[73] M. Poess, R. O. Nambiar, and D. Walrath, "Why you should run tpc-ds: A workload analysis." in *VLDB*, vol. 7, 2007, pp. 1138–1149.

[74] V. Leis, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann, "How good are query optimizers, really?" *Proc. VLDB Endow.*, vol. 9, no. 3, pp. 204–215, Nov. 2015. [Online]. Available: http://dx.doi.org/10.14778/2850583.2850594

[75] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems (5th Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2006.

[76] H. Garcia-Molina, J. D. Ullman, and J. Widom, *Database Systems: The Complete Book*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2008.

[77] C. Coronel and S. Morris, *Database systems: design, implementation, & management.* Cengage Learning, 2016.

[78] G. Oh, S. Kim, S.-W. Lee, and B. Moon, "Sqlite optimization with phase change memory for mobile applications," *Proc. VLDB Endow.*, vol. 8, no. 12, pp. 1454–1465, Aug. 2015. [Online]. Available: http://dx.doi.org/10.14778/2824032.2824044

[79] G. J. Ramakrishnan, Raghu, *Database management systems.* McGraw Hill, 2000.

[80] N. Roussopoulos, "View indexing in relational databases," *ACM Trans. Database Syst.*, vol. 7, no. 2, pp. 258–290, 1982.

[81] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*, 7th ed. Pearson, 2015.

[82] J. A. Hoffer, V. Ramesh, and H. Topi, *Modern database management.* Upper Saddle River, NJ: Prentice Hall,, 2011.

[83] M. S. Kester, M. Athanassoulis, and S. Idreos, "Access path selection in main-memory optimized data systems: Should i scan or should i probe?" in *Proceedings of the 2017 ACM International Conference on Management of Data*, ser. SIGMOD '17. ACM, 2017, pp. 715–730.

[84] D. Kuhn, S. R. Alapati, and B. Padfield, *Expert Oracle Indexing and Access Paths: Maximum Performance for Your Database.* Apress, 2016.

[85] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price, "Access path selection in a relational database management system," in *Proceedings of the 1979 ACM SIGMOD international conference on Management of data.* ACM, 1979, pp. 23–34.

[86] G. Giannikis, D. Makreshanski, G. Alonso, and D. Kossmann, "Workload optimization using shareddb," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '13. ACM, 2013, pp. 1045–1048.

[87] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[88] H. D. Sherali and J. C. Smith, "Improving discrete model representations via symmetry considerations," *Management Science*, vol. 47, no. 10, pp. 1396–1407, 2001.

[89] M. Pöss and C. Floyd, "New TPC benchmarks for decision support and web commerce," *SIGMOD Record*, vol. 29, no. 4, pp. 64–71, 2000. [Online]. Available: https://doi.org/10.1145/369275.369291

[90] M. Ehrgott and X. Gandibleux, "Bound sets for biobjective combinatorial optimization problems," *Computers & Operations Research*, vol. 34, no. 9, pp. 2674 – 2694, 2007.

[91] M. Özlen and M. Azizoğlu, "Multi-objective integer programming: A general approach for generating all non-dominated solutions," *European Journal of Operational Research*, vol. 199, pp. 25–35, 2009.

[92] Ö. Özpeynirci and M. Köksalan, "An exact algorithm for finding extreme supported nondominated points of multiobjective mixed integer programs," *Management Science*, vol. 56, no. 12, pp. 2302–2315, 2010.

[93] K. Dächert, J. Gorski, and K. Klamroth, "An augmented weighted Tchebycheff method with adaptively chosen parameters for discrete bicriteria optimization problems," *Computers & Operations Research*, vol. 39, pp. 2929–2943, 2012.

[94] B. Lokman and M. Köksalan, "Finding all nondominated points of multi-objective integer programs," *Journal of Global Optimization*, vol. 57, no. 2, pp. 347–365, 2013.

[95] A. Eusébio, J. Figueira, and M. Ehrgott, "On finding representative non-dominated points for bi-objective integer network flow problems," *Computers & Operations Research*, vol. 48, pp. 1 – 10, 2014.

[96] G. Kirlik and S. Sayın, "Computing the nadir point for multiobjective discrete optimization problems," *Journal of Global Optimization*, vol. 62, no. 1, pp. 79–99, May 2015.

[97] B. Soylu and G. B. Yıldız, "An exact algorithm for biobjective mixed integer linear programming problems," *Computers & Operations Research*, vol. 72, pp. 204–213, 2016.

[98] N. Boland, H. Charkhgard, and M. Savelsbergh, "The quadrant shrinking method: A simple and efficient algorithm for solving tri-objective integer programs," *European Journal of Operational Research*, vol. 260, no. 3, pp. 873 – 885, 2017.

[99] A. Przybylski and X. Gandibleux, "Multi-objective branch and bound," *European Journal of Operational Research*, vol. 260, no. 3, pp. 856 – 872, 2017.

[100] L. Shao and M. Ehrgott, "An objective space cut and bound algorithm for convex multiplicative programmes," *Journal of Global Optimization*, vol. 58, no. 4, pp. 711–728, 2014.

[101] H. Charkhgard, M. Savelsbergh, and M. Talebian, "A linear programming based algorithm to solve a class of optimization problems with a multi-linear objective function and affine constraints," *Computers & Operations Research*, vol. 89, pp. 17 – 30, 2018.

[102] P. G. Saghand, H. Charkhgard, and C. Kwon, "A branch-and-bound algorithm for a class of mixed integer linear maximum multiplicative programs: A bi-objective optimization approach," *Computers & Operations Research*, vol. 101, pp. 263–274, 2019.

[103] H. Konno and T. Kuno, "Linear multiplicative programming," *Mathematical Programming*, vol. 56, no. 1-3, pp. 51–64, 1992.

[104] H. Benson and G. Boger, "Multiplicative programming problems: analysis and efficient point search heuristic," *Journal of Optimization Theory and Applications*, vol. 94, no. 2, pp. 487–510, 1997.

[105] T. Kuno, "A finite branch-and-bound algorithm for linear multiplicative programming," *Computational Optimization and Applications*, vol. 20, no. 2, pp. 119–135, 2001.

[106] N. T. B. Kim, N. T. Le Trang, and T. T. H. Yen, "Outcome-space outer approximation algorithm for linear multiplicative programming," *East-West Journal of Mathematics*, vol. 9, no. 1, 2007.

[107] N. Van Thoai, "A global optimization approach for solving the convex multiplicative programming problem," *Journal of Global Optimization*, vol. 1, no. 4, pp. 341–357, 1991.

[108] T. Kuno, Y. Yajima, and H. Konno, "An outer approximation method for minimizing the product of several convex functions on a convex set," *Journal of Global optimization*, vol. 3, no. 3, pp. 325–335, 1993.

[109] H. P. Benson, "An outcome space branch and bound-outer approximation algorithm for convex multiplicative programming," *Journal of Global Optimization*, vol. 15, no. 4, pp. 315–342, 1999.

[110] Y. Gao, G. Wu, and W. Ma, "A new global optimization approach for convex multiplicative programming," *Applied Mathematics and Computation*, vol. 216, no. 4, pp. 1206–1218, 2010.

[111] R. M. Oliveira and P. A. Ferreira, "A convex analysis approach for convex multiplicative programming," *Journal of Global Optimization*, vol. 41, no. 4, pp. 579–592, 2008.

[112] H. Benson and G. Boger, "Outcome-space cutting-plane algorithm for linear multiplicative programming," *Journal of Optimization Theory and Applications*, vol. 104, no. 2, pp. 301–322, 2000.

[113] E. Nicholson and H. P. Possingham, "Objectives for multiple-species conservation planning," *Conservation Biology*, vol. 20, no. 3, pp. 871–881, 2006.

[114] H. P. Benson, "Optimization over the efficient set," *Journal of Mathematical Analysis and Applications*, vol. 98, no. 2, pp. 562–580, 1984.

[115] N. Boland, H. Charkhgard, and M. Savelsbergh, "A new method for optimizing a linear function over the efficient set of a multiobjective integer program," *European Journal of Operational Research*, vol. 260, no. 3, pp. 904 – 919, 2017.

[116] J. M. Jorge, "An algorithm for optimizing a linear function over an integer efficient set," *European Journal of Operational Research*, vol. 195, no. 1, pp. 98–103, 2009.

[117] S. Sayın, "Optimizing over the efficient set using a top-down search of faces," *Operations Research*, vol. 48, no. 1, pp. 65–72, 2000.

[118] A. Sierra Altamiranda and H. Charkhgard, "A new exact algorithm to optimize a linear function over the set of efficient solutions for biobjective mixed integer linear programs," *INFORMS Journal on Computing*, vol. 31, no. 4, pp. 823–840, 2019.

[119] P.-C. Chen, P. Hansen, and B. Jaumard, "On-line and off-line vertex enumeration by adjacency lists," *Operations Research Letters*, vol. 10, no. 7, pp. 403 – 409, 1991. [Online]. Available: http://www.sciencedirect.com/science/article/pii/016763779190042N

[120] M. Ehrgott, *Multicriteria optimization*, 2nd ed. New York: Springer, 2005.

[121] Y. P. Aneja and K. P. K. Nair, "Bicriteria transportation problem," *Management Science*, vol. 27, pp. 73–78, 1979.

[122] A. Pal and H. Charkhgard, "FPBH: A feasibility pump based heuristic for multi-objective mixed integer linear programming," *Computers & Operations Research*, vol. 112, p. 104760, 2019.

[123] N. Boland, H. Charkhgard, and M. Savelsbergh, "A criterion space search algorithm for biobjective mixed integer programming: The triangle splitting method," *INFORMS Journal on Computing*, vol. 27, no. 4, pp. 597–618, 2015.

[124] K. Dächert, K. Klamroth, R. Lacour, and D. Vanderpooten, "Efficient computation of the search region in multi-objective optimization," *European Journal of Operational Research*, vol. 260, no. 3, pp. 841 – 855, 2017.

[125] G. Kirlik and S. Sayın, "A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems," *European Journal of Operational Research*, vol. 232, no. 3, pp. 479 – 488, 2014.

[126] N. Boland, H. Charkhgard, and M. Savelsbergh, "A criterion space search algorithm for biobjective integer programming: The balanced box method," *INFORMS Journal on Computing*, vol. 27, no. 4, pp. 735–754, 2015.

[127] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Mathematical Programming*, vol. 91(2), pp. 201–213, 2002.

[128] P. G. Saghand and H. Charkhgard, "Exact solution approaches for integer linear generalized maximum multiplicative programs through the lens of multi-objective optimization," *Preprint*, 2019. [Online]. Available: http://www.optimization-online.org/DB_FILE/2019/10/7435.pdf

[129] J. F. Nash, "The bargaining problem," *Econometrica: Journal of the econometric society*, pp. 155–162, 1950.

[130] H. Charkhgard, K. Keshanian, R. Esmaeilbeigi, and P. Charkhgard, "The magic of nash social welfare in optimization: Do not sum, just multiply!" *Preprint http://www. optimization-online. org/DB_ FILE/2019/01/7031. pdf*, 03 2020.

[131] I. Caragiannis, D. Kurokawa, H. Moulin, A. D. Procaccia, N. Shah, and J. Wang, "The unreasonable fairness of maximum nash welfare," *ACM Transactions on Economics and Computation (TEAC)*, vol. 7, no. 3, pp. 1–32, 2019.

[132] R. Dai, H. Charkhgard, and F. Rigterink, "A robust biobjective optimization approach for operating a shared energy storage under price uncertainty," *International Transactions in Operational Research*, 2020, available Online.

[133] K. A. Melendez, V. Subramanian, T. K. Das, and C. Kwon, "Empowering end-use consumers of electricity to aggregate for demand-side participation," *Applied Energy*, vol. 248, pp. 372 – 382, 2019.

[134] A. Sierra-Altamiranda, H. Charkhgard, M. Eaton, J. Martin, S. Yurek, and B. J. Udell, "Spatial conservation planning under uncertainty using modern portfolio theory and nash bargaining solution," *Ecological Modelling*, vol. 423, p. 109016, 2020.

[135] J. Mendoza-Alonzo, J. L. Zayas-Castro, and H. Charkhgard, "Office-based and home-care for older adults in primary care: A comparative analysis using the nash bargaining solution," *Socio-Economic Planning Sciences*, vol. 69, p. 100710, 2020.

[136] J. A. Acuna, J. L. Zayas-Castro, and H. Charkhgard, "Ambulance allocation optimization model for the overcrowding problem in us emergency departments: A case study in Florida," *Socio-Economic Planning Sciences*, vol. 71, p. 100747, 2020.

[137] J. Arsenyan, G. Büyüközkan, and O. Feyzioğlu, "Modeling collaboration formation with a game theory approach," *Expert Systems with Applications*, vol. 42, no. 4, pp. 2073 – 2085, 2015.

[138] M. Nagarajan and G. Sošić, "Game-theoretic analysis of cooperation among supply chain agents: Review and extensions," *European Journal of Operational Research*, vol. 187, no. 3, pp. 719 – 745, 2008.

[139] M. Nagarajan and Y. Bassok, "A bargaining framework in supply chains: The assembly problem," *Management Science*, vol. 54, no. 8, pp. 1482–1496, 2008.

[140] B. Hezarkhani and W. Kubiak, "A coordinating contract for transshipment in a two-company supply chain," *European Journal of Operational Research*, vol. 207, no. 1, pp. 232 – 237, 2010.

[141] D. Chakrabarty, N. Devanur, and V. V. Vazirani, "New results on rationality and strongly polynomial time solvability in eisenberg-gale markets," in *International Workshop on Internet and Network Economics*. Springer, 2006, pp. 239–250.

[142] K. Jain and V. V. Vazirani, "Eisenberg-gale markets: Algorithms and structural properties," in *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, 2007, pp. 364–373.

[143] V. V. Vazirani, "The notion of a rational convex program, and an algorithm for the arrow-debreu nash bargaining game," *Journal of the ACM (JACM)*, vol. 59, no. 2, pp. 1–36, 2012.

[144] ——, "Rational convex programs and efficient algorithms for 2-player nash and non-symmetric bargaining games," *SIAM Journal on Discrete Mathematics*, vol. 26, no. 3, pp. 896–918, 2012.

[145] M. Abouei Ardakan, M. Sima, A. Zeinal Hamadani, and D. W. Coit, "A novel strategy for redundant components in reliability–redundancy allocation problems," *IIE Transactions*, vol. 48, no. 11, pp. 1043–1057, 2016.

[146] M. Feizabadi and A. E. Jahromi, "A new model for reliability optimization of series-parallel systems with non-homogeneous components," *Reliability Engineering & System Safety*, vol. 157, pp. 101–112, 2017.

[147] E. Zhang and Q. Chen, "Multi-objective reliability redundancy allocation in an interval environment using particle swarm optimization," *Reliability Engineering & System Safety*, vol. 145, pp. 83–92, 2016.

[148] A. Fattahi and M. Turkay, "ϵ-oa for the solution of bi-objective generalized disjunctive programming problems in the synthesis of nonlinear process networks," *Computers & Chemical Engineering*, vol. 72, pp. 199–209, 2015.

[149] G. Eichfelder, P. Kirst, L. Meng, and O. Stein, "A general branch-and-bound framework for continuous global multiobjective optimization," *Journal of Global Optimization*, 2021, available online.

[150] J. Niebling and G. Eichfelder, "A branch–and–bound-based algorithm for nonconvex multiobjective optimization," *SIAM Journal on Optimization*, vol. 29, no. 1, pp. 794–821, 2019.

[151] P. G. Saghand and H. Charkhgard, "A criterion space search algorithm for mixed integer linear maximum multiplicative programs: A multi-objective optimization approach," *Preprint*, 2019. [Online]. Available: http://www.optimization-online.org/DB_FILE/2019/01/7031.pdf

[152] T. Perini, N. Boland, D. Pecin, and M. Savelsbergh, "A criterion space method for biobjective mixed integer programming: The boxed line method," *INFORMS Journal on Computing*, vol. 32, no. 1, pp. 16–39, 2020.

[153] S. A. B. Rasmi, A. Fattahi, and M. Türkay, "SASS: slicing with adaptive steps search method for finding the non-dominated points of tri-objective mixed-integer linear programming problems," *Annals of Operations Research*, 2019. [Online]. Available: https://doi.org/10.1007/s10479-019-03422-9

[154] J. Hooker, *Logic-based methods for optimization: combining optimization and constraint satisfaction.* John Wiley & Sons, 2011, vol. 2.

[155] V. Mahmoodian, H. Charkhgard, and Y. Zhang, "Multi-objective optimization based algorithms for solving mixed integer linear minimum multiplicative programs," *Computers & Operations Research*, vol. 128, p. 105178, 2021.

[156] G. P. McCormick, "Computability of global solutions to factorable nonconvex programs: Part i—convex underestimating problems," *Mathematical programming*, vol. 10, no. 1, pp. 147–175, 1976.

[157] M. L. Bergamini, P. Aguirre, and I. Grossmann, "Logic-based outer approximation for globally optimal synthesis of process networks," *Computers & chemical engineering*, vol. 29, no. 9, pp. 1914–1933, 2005.

[158] R. Karuppiah and I. E. Grossmann, "Global optimization for the synthesis of integrated water systems in chemical processes," *Computers & Chemical Engineering*, vol. 30, no. 4, pp. 650–673, 2006.

[159] D. Bertsimas, V. F. Farias, and N. Trichakis, "The price of fairness," *Operations Research*, vol. 59, no. 1, pp. 17–31, 2011.

[160] Ö. Karsu and A. Morton, "Inequity averse optimization in operational research," *European journal of operational research*, vol. 245, no. 2, pp. 343–359, 2015.

[161] W. Ogryczak, H. Luss, M. Pióro, D. Nace, and A. Tomaszewski, "Fair optimization and networks: A survey," *Journal of Applied Mathematics*, vol. 2014, 2014.

[162] P. Matl, R. F. Hartl, and T. Vidal, "Workload equity in vehicle routing problems: A survey and analysis," *Transportation Science*, vol. 52, no. 2, pp. 239–260, 2018.

[163] E. Erkut, "Inequality measures for location problems." *Computers & Operations Research*, 1993.

[164] M. T. Marsh and D. A. Schilling, "Equity measurement in facility location analysis: A review and framework," *European journal of operational research*, vol. 74, no. 1, pp. 1–17, 1994.

[165] B. Balcik, S. M. Iravani, and K. Smilowitz, "A review of equity in nonprofit and public sector: a vehicle routing perspective," *Wiley encyclopedia of operations research and management science*, 2010.

[166] T. Vidal, G. Laporte, and P. Matl, "A concise guide to existing and emerging vehicle routing problem variants," *European Journal of Operational Research*, 2019.

[167] F. Blakeley, B. Argüello, B. Cao, W. Hall, and J. Knolmajer, "Optimizing periodic maintenance operations for schindler elevator corporation," *Interfaces*, vol. 33, no. 1, pp. 67–79, 2003.

[168] S. Liu, "A hybrid population heuristic for the heterogeneous vehicle routing problems," *Transportation Research Part E: Logistics and Transportation Review*, vol. 54, pp. 67–78, 2013.

[169] J. C. Goodson, "Election day routing of rapid response attorneys," *INFOR: Information Systems and Operational Research*, vol. 52, no. 1, pp. 1–9, 2014.

[170] P. Matl, R. F. Hartl, and T. Vidal, "Workload equity in vehicle routing: The impact of alternative workload resources," *Computers & Operations Research*, vol. 110, pp. 116–129, 2019.

[171] A. Alnaggar, F. Gzara, and J. H. Bookbinder, "Crowdsourced delivery: A review of platforms and academic literature," *Omega*, p. 102139, 2019.

[172] S. Banjo, "Home depot to offer same day delivery," *Wall Street Journal*, 2013. [Online]. Available: https://www.wsj.com/articles/home-depot-to-offer-same-day-delivery-1386789390

[173] A. Barr and J. Wohl, "Exclusive:Walmart may get customers to deliver packages to online buyers," *Reuters*, 2013.

[174] G. Bensinger, "Amazon's next delivery drone: You," *Wall Street Journal*, 2015. [Online]. Available: https://www.wsj.com/articles/amazon-seeks-help-with-deliveries-1434466857

[175] C. Archetti, M. Savelsbergh, and M. G. Speranza, "The vehicle routing problem with occasional drivers," *European Journal of Operational Research*, vol. 254, no. 2, pp. 472–480, 2016.

[176] V. E. Castillo, J. E. Bell, W. J. Rose, and A. M. Rodrigues, "Crowdsourcing last mile delivery: strategic implications and future research directions," *Journal of Business Logistics*, vol. 39, no. 1, pp. 7–25, 2018.

[177] A. Devari, A. G. Nikolaev, and Q. He, "Crowdsourcing the last mile delivery of online orders by exploiting the social networks of retail store customers," *Transportation Research Part E: Logistics and Transportation Review*, vol. 105, pp. 105–122, 2017.

[178] A. Giret, C. Carrascosa, V. Julian, M. Rebollo, and V. Botti, "A crowdsourcing approach for sustainable last mile delivery," *Sustainability*, vol. 10, no. 12, p. 4563, 2018.

[179] K. Huang and M. N. Ardiansyah, "A decision model for last-mile delivery planning with crowdsourcing integration," *Computers & Industrial Engineering*, vol. 135, pp. 898–912, 2019.

[180] I. Dayarian and M. Savelsbergh, "Crowdshipping and same-day delivery: Employing in-store customers to deliver online orders," *Production and Operations Management*, vol. 29, no. 9, pp. 2153–2174, 2020.

[181] G. Macrina, L. D. P. Pugliese, F. Guerriero, and D. Laganà, "The vehicle routing problem with occasional drivers and time windows," in *International Conference on Optimization and Decision Science*. Springer, 2017, pp. 577–587.

[182] A. M. Arslan, N. Agatz, L. Kroon, and R. Zuidwijk, "Crowdsourced delivery—a dynamic pickup and delivery problem with ad hoc drivers," *Transportation Science*, vol. 53, no. 1, pp. 222–235, 2019.

[183] D. Soto Setzke, C. Pflügler, M. Schreieck, S. Fröhlich, M. Wiesche, and H. Krcmar, "Matching drivers and transportation requests in crowdsourced delivery systems," 2017.

[184] W. Qi, L. Li, S. Liu, and Z.-J. M. Shen, "Shared mobility for last-mile delivery: Design, operational prescriptions, and environmental impact," *Manufacturing & Service Operations Management*, vol. 20, no. 4, pp. 737–751, 2018.

[185] S. Fatehi and M. Wagner, "Crowdsourcing last-mile deliveries," *Available at SSRN 3434625*, 2019.

[186] K. Gdowska, A. Viana, and J. P. Pedroso, "Stochastic last-mile delivery with crowdshipping," *Transportation research procedia*, vol. 30, pp. 90–100, 2018.

[187] P. Matl, R. F. Hartl, and T. Vidal, "Workload equity in vehicle routing: The impact of alternative workload resources," *Computers & Operations Research*, vol. 110, pp. 116–129, 2019.

[188] W. Ogryczak and T. Śliwiński, "On direct methods for lexicographic min-max optimization," in *International Conference on Computational Science and Its Applications*. Springer, 2006, pp. 802–811.

[189] S. Saliba, "Heuristics for the lexicographic max-ordering vehicle routing problem," *Central European Journal of Operations Research*, vol. 14, no. 3, pp. 313–336, 2006.

[190] F. Lehuédé, O. Péton, and F. Tricoire, "A lexicographic minimax approach to the vehicle routing problem with route balancing," *European Journal of Operational Research*, vol. 282, no. 1, pp. 129–147, 2020.

[191] R. Serrano, "Fifty years of the Nash program, 1953-2003," *Investigaciones Economicas*, vol. 29, no. 2, pp. 219–258, May 2005. [Online]. Available: https://ideas.repec.org/a/iec/inveco/v29y2005i2p219-258.html

[192] J. F. Nash, "Two-person cooperative games," *Econometrica: Journal of the Econometric Society*, pp. 128–140, 1953.

[193] E. Kalai, "Nonsymmetric nash solutions and replications of 2-person bargaining," *International Journal of Game Theory*, vol. 6, no. 3, pp. 129–133, 1977.

[194] H. Charkhgard, K. Keshanian, R. Esmaeilbeigi, and P. Charkhgard, "The magic of nash social welfare in optimization: Do not sum, just multiply," *Optimization Online*, 2020.

[195] P. G. Saghand and H. Charkhgard, "Exact solution approaches for integer linear generalized maximum multiplicative programs through the lens of multi-objective optimization," *Preprint*, 2019, http://www.optimization-online.org/DB_FILE/2019/10/7435.pdf.

[196] R. Cole and V. Gkatzelis, "Approximating the nash social welfare with indivisible items," *SIAM Journal on Computing*, vol. 47, no. 3, pp. 1211–1236, 2018.

[197] H. Moulin, *Fair division and collective welfare*. MIT press, 2004.

[198] V. Mahmoodian, I. Dayarian, P. G. Saghand, Y. Zhang, and H. Charkhgard, "A new algorithm for mixed integer bi-linear maximum multiplicative programs," *INFORMS Journal on Computing*, 2021, to appear.

[199] P. G. Saghand, H. Charkhgard, and C. Kwon, "A branch-and-bound algorithm for a class of mixed integer linear maximum multiplicative programs: A bi-objective optimization approach," *Computers & Operations Research*, vol. 101, pp. 263 – 274, 2019.

[200] A. Ben-Tal and A. Nemirovski, "On polyhedral approximations of the second-order cone," *Mathematics of Operations Research*, vol. 26, no. 2, pp. 193–205, 2001.

[201] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen, "An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems," *Networks*, vol. 44, no. 3, pp. 216–229, 2004.

[202] S. Irnich and G. Desaulniers, *Shortest Path Problems with Resource Constraints*, ser. GERAD 25th Anniversary Series. Springer, 2005, ch. 2, pp. 33–65.

[203] L. Perron and V. Furnon, "Or-tools," Google. [Online]. Available: https://developers.google.com/optimization/

[204] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations research*, vol. 35, no. 2, pp. 254–265, 1987.

Appendix A: Proof of Theorem 3.1

Theorem .1. *The query batching problem when employing the proposed processing time prediction function* (3.1) *is NP-hard.*

Proof. In order to prove that the QBP is NP-hard, it is sufficient to show that its corresponding decision problem, `QBP`, is NP-complete. Note that `QBP` is obviously in NP and an instance of `QBP` can be shown by

$$[m, n, U, \beta_0, \beta_1, \beta_2 \log_2(s_1), \ldots, \beta_{m+1} \log_2(s_m), Q_1, \ldots, Q_m].$$

In the remaining, we show that `Partition-Problem` is polynomially reducible to `QBP`, where `Partition-Problem` is defined as follows: given a set $\{c_1, \ldots, c_{m'}\}$ of positive integers, does there exist a subset $A \subseteq \{1, \ldots, m\}$ such that

$$\sum_{i \in A} c_i = \sum_{i \in A \setminus \{1,\ldots,m'\}} c_i.$$

Observe that an integer programming formulation for `Partition-Problem` can be stated as follows (the second constraint is redundant but we keep it for the sake of the proof),

$$\begin{aligned}
& -\frac{\sum_{i=1}^{m'} c_i}{2} + \sum_{i=1}^{m'} c_i x_i = 0 \\
& -\frac{\sum_{i=1}^{m'} c_i}{2} + \sum_{i=1}^{m'} c_i (1 - x_i) = 0 \\
& x_i \in \{0, 1\} \qquad\qquad \forall i \in \{1, 2, \ldots, m'\}
\end{aligned}$$

Now, given an arbitrary instance of `Partition-Problem`, denoted by $[m, c_1, \ldots, c'_m]$, we construct an instance of `QBP` by setting,

- $m = n = m'$
- $U = 0$
- $\beta_0 = -\frac{\sum_{t=1}^{m'} c_t}{2}$
- $\beta_1 = 0$
- $\beta_{t+1} log(s_t) = c_t$ for all $t \in \{1, \ldots, m'\}$
- $Q_t = \{t\}$ for all $t \in \{1, \ldots, m'\}$

Observe that the created instance of `QBP` is valid in a sense that it can be constructed in polynomial time and each of its queries involves at least one table. Now, we show that an instance of `Partition-Problem`, $[m, c_1, \ldots, c'_m]$, is a Yes-instance if an only if its corresponding `QBP` instance is a Yes-instance. For doing so, we use the formulation proposed for the `QBP` and we show that when we plug the constructed instance into it, the formulation will be simplified/equivalent to the formulation of `Partition-Problem`.

In light of the above, observe that because $\beta_1 = 0$ and $n_j = \sum_{i=1}^{n} x_{ij}$, the formulation of `QBP` can be simplified to,

$$\sum_{j=1}^{n} \Big[\beta_0 z_j + \sum_{t \in T} \beta_{t+1} \log_2(s_t) y_{jt}\Big]^2 \leqslant U$$

$$y_{jt} \leqslant \sum_{i \in Q_t} x_{ij} \qquad \forall j \in \{1, 2, \ldots, n\},\ \forall t \in T$$

$$z_j \leqslant \sum_{i=1}^{n} x_{ij} \qquad \forall j \in \{1, 2, \ldots, n\}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i \in \{1, 2, \ldots, n\}$$

$$\sum_{i \in Q_t} x_{ij} \leqslant n y_{jt} \qquad \forall j \in \{1, 2, \ldots, n\},\ \forall t \in T$$

$$\sum_{t\in T} y_{jt} \leqslant m z_j \qquad \forall j \in \{1,2,\ldots,n\}$$

$$x_{ij}, y_{jt}, z_j \in \{0,1\} \qquad \forall i \in \{1,2,\ldots,n\}, \forall j \in \{1,2,\ldots,n\},\ \forall t \in T.$$

Moreover since $Q_t = \{t\}$ for all $t \in \{1,\ldots,m'\}$ the term $\sum_{i\in Q_t} x_{ij}$ is equivalent to x_{tj}. So, the formulation can be simplified to,

$$\sum_{j=1}^{n} \left[\beta_0 z_j + \sum_{t\in T} \beta_{t+1} \log_2(s_t) y_{jt}\right]^2 \leqslant U$$

$$y_{jt} \leqslant x_{tj} \qquad \forall j \in \{1,2,\ldots,n\},\ \forall t \in T$$

$$z_j \leqslant \sum_{i=1}^{n} x_{ij} \qquad \forall j \in \{1,2,\ldots,n\}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i \in \{1,2,\ldots,n\}$$

$$x_{tj} \leqslant n y_{jt} \qquad \forall j \in \{1,2,\ldots,n\},\ \forall t \in T$$

$$\sum_{t\in T} y_{jt} \leqslant m z_j \qquad \forall j \in \{1,2,\ldots,n\}$$

$$x_{ij}, y_{jt}, z_j \in \{0,1\} \qquad \forall i \in \{1,2,\ldots,n\}, \forall j \in \{1,2,\ldots,n\},\ \forall t \in T.$$

Note that a constraint of the form $x_{tj} \leqslant n y_{jt}$ is equivalent to $x_{tj} \leqslant y_{jt}$ since both x_{ij} and y_{ij} are binary variables. Therefore the formulation simplifies to,

$$\sum_{j=1}^{n} \left[\beta_0 z_j + \sum_{t\in T} \beta_{t+1} \log_2(s_t) y_{jt}\right]^2 \leqslant U$$

$$y_{jt} = x_{tj} \qquad \forall j \in \{1,2,\ldots,n\},\ \forall t \in T$$

$$z_j \leqslant \sum_{i=1}^{n} x_{ij} \qquad \forall j \in \{1,2,\ldots,n\}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i \in \{1,2,\ldots,n\}$$

$$\sum_{t\in T} y_{jt} \leqslant m z_j \qquad \forall j \in \{1,2,\ldots,n\}$$

$$x_{ij}, y_{jt}, z_j \in \{0,1\} \qquad \forall i \in \{1,2,\ldots,n\}, \forall j \in \{1,2,\ldots,n\},\ \forall t \in T.$$

Consequently, the decision variable y_{jt} can be replaced by x_{tj} since $m = n = m'$. So, the formulation can be simplified further to,

$$\sum_{j=1}^{m'} \big[\beta_0 z_j + \sum_{i=1}^{m'} \beta_{i+1} \log_2(s_i) x_{ij}\big]^2 \leqslant U$$

$$z_j \leqslant \sum_{i=1}^{m'} x_{ij} \qquad \forall j \in \{1,2,\ldots,m'\}$$

$$\sum_{j=1}^{m'} x_{ij} = 1 \qquad \forall i \in \{1,2,\ldots,m'\}$$

$$\sum_{i=1}^{m'} x_{ij} \leqslant m' z_j \qquad \forall j \in \{1,2,\ldots,m'\}$$

$$x_{ij}, z_j \in \{0,1\} \qquad \forall i \in \{1,2,\ldots,m'\}, \forall j \in \{1,2,\ldots,m'\}.$$

Now, since $U = 0$, $\beta_0 = -\frac{\sum_{t=1}^{m'} c_t}{2}$, and $\beta_{t+1} log(s_t) = c_t$ for all $t \in \{1,\ldots,m'\}$, the formulation can be rewritten as,

$$\sum_{j=1}^{m'} \big[-\frac{\sum_{i=1}^{m'} c_i}{2} z_j + \sum_{i=1}^{m'} c_i x_{ij}\big]^2 \leqslant 0$$

$$z_j \leqslant \sum_{i=1}^{m'} x_{ij} \qquad \forall j \in \{1,2,\ldots,m'\}$$

$$\sum_{j=1}^{m'} x_{ij} = 1 \qquad \forall i \in \{1,2,\ldots,m'\}$$

$$\sum_{i=1}^{m'} x_{ij} \leqslant m' z_j \qquad \forall j \in \{1,2,\ldots,m'\}$$

$$x_{ij}, z_j \in \{0,1\} \qquad \forall i \in \{1,2,\ldots,m'\}, \forall j \in \{1,2,\ldots,m'\}.$$

Since the sum of squares is not positive, each square must be zero. So, the formulation can be simplified to,

$$-\frac{\sum_{i=1}^{m'} c_i}{2} z_j + \sum_{i=1}^{m'} c_i x_{ij} = 0 \qquad \forall j \in \{1, 2, \ldots, m'\} \tag{1}$$

$$z_j \leqslant \sum_{i=1}^{m'} x_{ij} \qquad \forall j \in \{1, 2, \ldots, m'\} \tag{2}$$

$$\sum_{j=1}^{m'} x_{ij} = 1 \qquad \forall i \in \{1, 2, \ldots, m'\} \tag{3}$$

$$\sum_{i=1}^{m'} x_{ij} \leqslant m' z_j \qquad \forall j \in \{1, 2, \ldots, m'\} \tag{4}$$

$$x_{ij}, z_j \in \{0, 1\} \qquad \forall i \in \{1, 2, \ldots, m'\}, \forall j \in \{1, 2, \ldots, m'\}. \tag{5}$$

Observe that Constraints (2) and (4) guarantee that $z_j = 1$ if an only if $\sum_{i=1}^{m'} x_{ij} > 0$. This combined with Constraint (1) ensure that for any $j \in \{1, \ldots, m'\}$, $\sum_{i=1}^{m'} c_i x_{ij} = \frac{\sum_{i=1}^{m'} c_i}{2}$ if and only if $z_j = 1$ (or $\sum_{i=1}^{m'} x_{ij} > 0$). This itself combined with Constraint (3), which is defined for partitioning the set of queries, guarantee that (if the instance is feasible then) there will be exactly two batches $j, j' \in \{1, \ldots, m'\}$ with $j \neq j'$ such that $\sum_{i=1}^{m'} c_i x_{ij} = \frac{\sum_{i=1}^{m'} c_i}{2}$ and $\sum_{i=1}^{m'} c_i x_{ij'} = \frac{\sum_{i=1}^{m'} c_i}{2}$. Note that the remaining batches must be empty since by assumptions $c_1, \ldots, c_{m'} > 0$. So, the above formulation is equivalent to,

$$-\frac{\sum_{i=1}^{m'} c_i}{2} + \sum_{i=1}^{m'} c_i x_{ij} = 0 \qquad \forall j \in \{1, 2\}$$

$$\sum_{j=1}^{2} x_{ij} = 1 \qquad \forall i \in \{1, 2, \ldots, m'\}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in \{1, 2, \ldots, m'\}, \forall j \in \{1, 2\}$$

This itself is equivalent to,

$$
\begin{aligned}
& -\frac{\sum_{i=1}^{m'} c_i}{2} + \sum_{i=1}^{m'} c_i x_i = 0 \\
& -\frac{\sum_{i=1}^{m'} c_i}{2} + \sum_{i=1}^{m'} c_i (1 - x_i) = 0 \\
& x_i \in \{0, 1\} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \forall i \in \{1, 2, \ldots, m'\}
\end{aligned}
$$

This is precisely an integer programming formulation of `Partition-Problem`. Therefore, the result follows. □

Appendix B: Online Supplement on Chapter 6

B.1 Arc-based Formulation

In this Appendix, we provide the arc-based formulation of the introduced vehicle routing problem with equity considerations. Model `EXP-MTMC` represents the expanded formulation to solve the MTMC problem, namely the problem that that minimizes the company's total cost and yields z^* and $\mathcal{V}^*$. Table B.1 introduces the notation used for this formulation.

Table B.1 Notation of expanded models

Notation	Description
Sets	
$\mathcal{V}$	set of all available drivers $\mathcal{V} = \{1, 2, \ldots, m\}$
$\mathcal{D}$	set of customers (delivery locations)
o	depot
o'_v	dummy copy of depot for driver $v \in \mathcal{V}$
s_v	origin of driver $v \in \mathcal{V}$
e_v	destination of driver v
$\mathcal{N}_v$	the set of nodes that driver $v \in \mathcal{V}$ can visit, $\mathcal{N}_v = \mathcal{D} \cup \{o, s_v, e_v\}$
Parameters	
c'_{ij}	the mileage cost incurred along arc $(i, j), \forall i, j \in \mathcal{N}_v$
β	the mileage rate, i.e., the percentage of the mileage cost incurred by a driver that the company reimburses.
t_{ij}	the travel time of arc $(i, j), \forall i, j \in \mathcal{N}$
w_j	the prize collected by a driver after serving customer $j \in \mathcal{D}$
w_{e_v}	the fixed mileage cost of the driver at his destination, i.e., $-\beta c'_{s_v, e_v}$.
q_j	the quantity/size of the delivery at the location of customer $j \in \mathcal{D}$
Q_v	the vehicle capacity of driver $v \in \mathcal{V}$
L_v	the longest route that the driver $v \in \mathcal{V}$ would accept to travel, $L_v = \gamma t_{s_v e_v}$
z^*	the minimum cost of company for delivering all the demands without equity
α	the maximum ratio that the company is willing to increase its total mileage cost by.
Variables	
x_{ijv}	1 if vehicle $v \in \mathcal{V}$ travels from $i \in \mathcal{N}_v$ to $j \in \mathcal{N}_v$; 0 otherwise.
u_i	A Miller–Tucker–Zemlin (dummy nonnegative) variable for capturing the time indicator of visiting $i \in \cup_{v \in \mathcal{V}} \{o'_v, s_v, e_v\} \cup \mathcal{D}$

$$\text{(EXP-MTMC)} \quad \min \quad \sum_{v\in\mathcal{V}}\sum_{i\in\mathcal{N}_v}\sum_{j\in\mathcal{N}_v}\beta c'_{ij}x_{ijv} + \sum_{v\in\mathcal{V}} w_{e_v} \tag{6}$$

$$\text{s.t.} \quad \sum_{v\in\mathcal{V}}\sum_{i\in\mathcal{N}_v} x_{ijv} = 1 \qquad \forall j \in \mathcal{D} \tag{7}$$

$$x_{s_v o v} + x_{s_v e_v v} = 1 \qquad \forall v \in \mathcal{V} \tag{8}$$

$$\sum_{i\in\mathcal{D}\cup\{s_v\}} x_{ie_v v} = 1 \qquad \forall v \in \mathcal{V} \tag{9}$$

$$x_{s_v j v} = 0 \qquad \forall v \in \mathcal{V}, j \in \mathcal{D} \tag{10}$$

$$x_{i s_v v} = 0 \qquad \forall v \in \mathcal{V}, i \in \mathcal{N}_v \backslash \{s_v\} \tag{11}$$

$$x_{e_v i v} = 0 \qquad \forall v \in \mathcal{V}, i \in \mathcal{N}_v \backslash \{e_v\} \tag{12}$$

$$x_{i o v} = 0 \qquad \forall v \in \mathcal{V}, i \in \mathcal{D} \tag{13}$$

$$\sum_{i\in\mathcal{N}_v} x_{ijv} = \sum_{i\in\mathcal{N}_v} x_{jiv} \qquad \forall v \in \mathcal{V}, j \in \mathcal{D}\cup\{o\} \tag{14}$$

$$\sum_{i\in\mathcal{D}\cup\{o\}}\sum_{j\in\mathcal{D}} q_j x_{ijv} \leqslant Q_v \qquad \forall v \in \mathcal{V} \tag{15}$$

$$\sum_{i\in\mathcal{N}_v}\sum_{j\in\mathcal{N}_v} t_{ij}x_{ijv} \leqslant L_v \qquad \forall v \in \mathcal{V} \tag{16}$$

$$\sum_{i\in\mathcal{D}\cup\{o\}} \Big(w_j + (\beta-1)c'_{ij}\Big)x_{ijv} \geqslant 0 \qquad \forall v \in \mathcal{V}, \forall j \in D \tag{17}$$

$$\sum_{i\in\mathcal{N}_v}\sum_{j\in\mathcal{D}\cup\{e_v\}} w_j x_{ijv} + \sum_{i\in\mathcal{N}_v}\sum_{j\in\mathcal{N}_v}(\beta-1)c'_{ij}x_{ijv} \geqslant -c'_{s_v e_v}x_{s_v,e_v} \qquad \forall v \in \mathcal{V} \tag{18}$$

$$u_{s_1} = 1 \tag{19}$$

$$u_{s_v} - u_{o'_v} + 1 \leqslant 0 \qquad \forall v \in \mathcal{V} \tag{20}$$

$$u_{e_v} - u_{s_{v+1}} + 1 \leqslant 0 \qquad \forall v \in \mathcal{V}\backslash\{m\} \tag{21}$$

$$u_{o'_v} - u_j + 1 \leqslant M(1 - x_{ojv}) \qquad \forall j \in D, v \in \mathcal{V} \tag{22}$$

$$u_i - u_j + 1 \leqslant M(1 - \sum_{v\in\mathcal{V}} x_{ijv}) \qquad \forall i \in \mathcal{D}, j \in \mathcal{D} \tag{23}$$

$$u_i - u_{e_v} + 1 \leqslant M(1 - x_{ie_vv}) \qquad v \in \mathcal{V}, \forall i \in \mathcal{D} \cup \{s_v\}, \quad (24)$$

$$x_{ijv} \in \{0, 1\} \qquad \forall v \in \mathcal{V}, i \in \mathcal{N}_v, j \in \mathcal{N}_v \backslash \{i\} \quad (25)$$

$$0 \leqslant u_i \leqslant M \qquad \forall i \in \cup_{v \in \mathcal{V}} \{o'_v, s_v, e_v\} \cup \mathcal{D} \quad (26)$$

where $M = |\mathcal{D}| + 3m$. In this model, the objective function (6) minimizes the total cost of the company. Note that the second term of this function deducts the cost of traveling from s_v to e_v by driver v, since the company does not compensate this distance, regardless if the driver is employed or not. Constraints (7) guarantee that each customer is visited only once. Constraints (8)-(14) enforce each driver to start a route from his origin and finish the route at his destination. A route may include a delivery task where the vehicle has to visit the depot immediately after the origin. While, if no delivery task is assigned to a vehicle, Constraints (8) and (9) force the driver to move from his origin straight to the destination. The latter case means that the driver is not employed in the system, and therefore, the cost of travel on arc (s_v, e_v) is canceled out by the second term in the objective function. Capacity and the longest travel time limits are imposed on assigned delivery tasks to each driver by (15) and (16), respectively. Also, possible subtours are eliminated using Miller–Tucker–Zemlin method by adding Constraints (19)-(24). In this method of subtour elimination, we only need to build an order of node visits independent from the vehicles. Therefore, we created m copies of depot node corresponding to each vehicle and combined all the routes by connecting the destination of one route to the start node of another route (Constraint (21)) as if only one vehicle sequentially travels through all the routes. Finally, the type of the variables are declared in Constraints (25) and (26).

After solving `EXP-MTMC` and obtaining the values of z^* and the optimal set of drivers $\mathcal{V}^*$, we can solve the arc-based NSW `(EXP-NSW)` model. For the sake of brevity, we use the common constraints between traditional and equitable model by a simple transformation in

parameters. For this purpose, we rename the drivers in $\mathcal{V}^*$ to $1, 2, \ldots, |\mathcal{V}^*|$, and accordingly, rearrange all the affected parameters by setting $V := \mathcal{V}^*$ and $m := |\mathcal{V}^*|$. With these transformation in parameters and sets, the EXP-NSW problem can be formulated as follows:

$$\text{(EXP-NSW)} \max \prod_{v\in\mathcal{V}} \Bigg[\sum_{i\in\mathcal{N}_v} \sum_{j\in\mathcal{D}\cup\{e_v\}} w_j x_{ijv} + \sum_{i\in\mathcal{N}_v} \sum_{j\in\mathcal{N}_v} (\beta-1) c'_{ij} x_{ijv} \Bigg] \tag{27}$$

$$\text{s.t. } (7),\ (11)\text{–}(24),\ (25),\ (26),$$

$$x_{s_v ov} = 1 \qquad \forall v \in \mathcal{V} \tag{28}$$

$$\sum_{i\in\mathcal{D}} x_{ie_v v} = 1 \qquad \forall v \in \mathcal{V} \tag{29}$$

$$x_{s_v jv} = 0 \qquad \forall v \in \mathcal{V}, j \in \mathcal{D}\cup\{e_v\} \tag{30}$$

$$\sum_{i\in\mathcal{N}_v} \sum_{j\in\mathcal{D}\cup\{e_v\}} w_j x_{ijv} + \sum_{i\in\mathcal{N}_v} \sum_{j\in\mathcal{N}_v} (\beta-1) c'_{ij} x_{ijv} \geqslant 0 \quad \forall v \in \mathcal{V} \tag{31}$$

$$\sum_{v\in\mathcal{V}} \sum_{i\in\mathcal{N}_v} \sum_{j\in\mathcal{N}_v} \beta c'_{ij} x_{ijv} + \sum_{v\in\mathcal{V}} w_{e_v} \leqslant (1+\alpha) z^*. \tag{32}$$

where, the objective function (27) is the multiplication of drivers' profits. Also, Constraints (28)–(30) are slightly changed form of their counterparts in EXP-MTMC. The small changes in the mentioned constraints is because at least one delivery task will be assigned to each vehicle in EXP-NSW. Constraints (31) guarantee a nonnegative profit for each vehicle and Constraint (32) restricts the company's total mileage cost to $(1+\alpha)z^*$.

B.2 Proof of Proposition 6.2

For a given vehicle $v \in \mathcal{V}^*$, let $\boldsymbol{\lambda}_j^\ell = (\lambda_j^{\ell,cost}, \lambda_j^{\ell,time}, \lambda_j^{\ell,load}, \left(\lambda_j^{\ell,cust_i}\right)_{i\in\mathcal{D}})$, $\ell = 1, 2$, be two labels associated with (partial) paths, denoted by 1 and 2, ending at node $j \in \mathcal{N}_v \backslash \{e_v\}$. Let $\mathcal{N}^{\boldsymbol{\lambda}_j^1}$ and $\mathcal{N}^{\boldsymbol{\lambda}_j^2}$ be the set of nodes visited in partial paths 1 and 2, respectively. Label $\boldsymbol{\lambda}_j^1$ is said to dominate label $\boldsymbol{\lambda}_j^2$ and is discarded if the following conditions hold:

$$(6.16) - (6.19),$$

$$p_{\boldsymbol{\lambda}^1_{j\oplus i^*\oplus e_v}} > 0, \tag{33}$$

$$\lambda^{1,cost}_{j\oplus i^*\oplus e_v} \geqslant \lambda^{2,cost}_{j\oplus i^*\oplus e_v}, \tag{34}$$

where $\overline{\mathcal{D}}^{\boldsymbol{\lambda}^2_j} := \mathcal{D}\backslash\mathcal{N}^{\boldsymbol{\lambda}^2_j}$ and $i^* := \arg\min_{i\in\overline{\mathcal{D}}^{\boldsymbol{\lambda}^2_j}\cup\{j\}} p_{\boldsymbol{\lambda}^1\oplus i\oplus e_v}$, with $p_{\boldsymbol{\lambda}^1_{j\oplus i^*\oplus e_v}}$ being the profit of the complete path obtained following the extension of label $\boldsymbol{\lambda}^1_j$ to i^* and then to e_v.

Proof. Proof. Conditions (6.16)-(6.19) are required based on Definition 6.2. Conditions (33) and (34), on the other hand, are necessary because of the nonlinearity of the objective function of the $\texttt{NSW-L-SP}_v$. Specifically, we need them to show that for any feasible common extension of labels 1 and 2 all the way to e_v, the objective value associated with the extended path 1 is not worse than the one for the extended path 2. Let σ be an arbitrary sequence of nodes in $\overline{\mathcal{D}}^{\boldsymbol{\lambda}^2_j}$ (possibly an empty sequence). To prove the assertion, we need to show that

$$\lambda^{1,cost}_{j\oplus\sigma\oplus e_v} \geqslant \lambda^{2,cost}_{j\oplus\sigma\oplus e_v},$$

for any arbitrary *feasible* sequence of nodes σ in $\overline{\mathcal{D}}^{\boldsymbol{\lambda}^2_j}$, i.e., the routes corresponding to $\lambda^1_{j\oplus\sigma\oplus e_v}$ or $\lambda^2_{j\oplus\sigma\oplus e_v}$ should not violate conditions in Definition 6.1. To that end, we first note that the objective function of the $\texttt{NSW-L-SP}_v$ consists of two parts: a linear function and a logarithmic function. So, for simplicity the objective function can be represented as $\log(\textsc{Profit}) + L(\textsc{Path})$ where $L(.)$ captures the linear part of the objective function. Using this notation, we would like to show that

$$\log(p_{\boldsymbol{\lambda}^1_{j\oplus\sigma\oplus e_v}}) + L^1(j\oplus\sigma\oplus e_v) \geqslant \log(p_{\boldsymbol{\lambda}^2_{j\oplus\sigma\oplus e_v}}) + L^2(j\oplus\sigma\oplus e_v).$$

for any arbitrary feasible sequence of nodes σ in $\overline{\mathcal{D}}^{\boldsymbol{\lambda}^2_j}$. We note that due to linearity, the contribution of $\sigma \oplus e_v$ in $L^1(j \oplus \sigma \oplus e_v)$ and $L^2(j \oplus \sigma \oplus e_v)$ is the same. Thus, the inequality can be further simplified to:

$$\begin{aligned}
&\log(p_{\boldsymbol{\lambda}^1_{j\oplus\sigma\oplus e_v}}) + L^1(j) \geqslant \log(p_{\boldsymbol{\lambda}^2_{j\oplus\sigma\oplus e_v}}) + L^2(j) \iff \\
&\log(p_{\boldsymbol{\lambda}^1_{j\oplus\sigma\oplus e_v}}) - \log(p_{\boldsymbol{\lambda}^2_{j\oplus\sigma\oplus e_v}}) \geqslant L^2(j) - L^1(j) \iff \\
&\log(p_{\boldsymbol{\lambda}^1_j} + p_{j,\sigma} + p_{\sigma,e_v}) - \log(p_{\boldsymbol{\lambda}^2} + p_{j,\sigma} + p_{\sigma,e_v}) \geqslant L^2(j) - L^1(j),
\end{aligned}$$

where $p_{j,\sigma}$ is the profit gained by moving from the current node of label 1, j, to the start node of sequence σ and completing sequence σ. We also define p_{σ,e_v} as the profit obtained by moving from the end of sequence σ to node e_v. Note that if sequence σ is empty then by the definition of the immediate extension operator, $j \oplus \sigma \oplus e_v = j \oplus e_v$. So, in summary, we need to show that

$$\log(p_{\boldsymbol{\lambda}^1_j} + p_{j,\sigma} + p_{\sigma,e_v}) - \log(p_{\boldsymbol{\lambda}^2_j} + p_{j,\sigma} + p_{\sigma,e_v}) \geqslant L^2(j) - L^1(j), \tag{35}$$

for any arbitrary feasible sequence σ (in $\overline{\mathcal{D}}^{\boldsymbol{\lambda}^2_j}$) in the remaining of the proof.

We start our proof by showing that $p_{j,\sigma} + p_{\sigma,e_v} \geqslant p_{j,i^*} + p_{i^*,e_v}$. Note that by definition of the immediate extension operator, we know that if $i^* = j$ then $j \oplus i^* \oplus e_v = j \oplus e_v$, hence we set $p_{j,i^*} = 0$ if $i^* = j$. Moreover, we know by condition (d) in Definition 6.1 that a route is not feasible if there is an arc on the route with endpoints in $\mathcal{D} \cup \{o\}$ and negative additional profit. Therefore, it follows that

$$p_{j,\sigma} + p_{\sigma,e_v} \geqslant \min_{i \in \overline{\mathcal{D}}^{\boldsymbol{\lambda}^2_j} \cup \{j\}} p_{j,i} + p_{i,e_v} = p_{j,i^*} + p_{i^*,e_v} \tag{36}$$

for any arbitrary feasible sequence σ (we also know that $p_{j,i^*} + p_{i^*,e_v} \leqslant p_{j,e_v} \leqslant 0$). This combined with Conditions (6.19) and (33), i.e., $p_{\boldsymbol{\lambda}^2} \geqslant p_{\boldsymbol{\lambda}^1}$ and $p_{\boldsymbol{\lambda}^1_{j\oplus i^*\oplus e_v}} > 0$ imply that

$$\begin{aligned} p_{\boldsymbol{\lambda}^2_{j\oplus\sigma\oplus e_v}} = p_{\boldsymbol{\lambda}^2_j} + p_{j,\sigma} + p_{\sigma,e_v} \geqslant p_{\boldsymbol{\lambda}^1_j} + p_{j,\sigma} + p_{\sigma,e_v} = p_{\boldsymbol{\lambda}^1_{j\oplus\sigma\oplus e_v}} \geqslant p_{\boldsymbol{\lambda}^1_{j\oplus i^*\oplus e_v}} = \\ p_{\boldsymbol{\lambda}^1_j} + p_{j,i^*} + p_{i^*,e_v} > 0. \end{aligned} \tag{37}$$

for any arbitrary feasible sequence σ. Therefore, we can conclude that the function $\log(p_{\boldsymbol{\lambda}^1_j} + p_{j,\sigma} + p_{\sigma,e_v}) - \log(p_{\boldsymbol{\lambda}^2_j} + p_{j,\sigma} + p_{\sigma,e_v})$ in the left side of Equation (35) is not undefined for any for any arbitrary feasible sequence σ, that is the terms in logs are positive. Next, we show that this function is non-decreasing with respect to $C^\sigma := p_{j,\sigma} + p_{\sigma,e_v}$. The derivative of $\log(p_{\boldsymbol{\lambda}^1_j} + C^\sigma) - \log(p_{\boldsymbol{\lambda}^2_j} + C^\sigma)$ with respect to C^σ is

$$\frac{1}{p_{\boldsymbol{\lambda}^1_j} + C^\sigma} - \frac{1}{p_{\boldsymbol{\lambda}^2_j} + C^\sigma} = \frac{1}{p_{\boldsymbol{\lambda}^1_{j\oplus\sigma\oplus e_v}}} - \frac{1}{p_{\boldsymbol{\lambda}^2_{j\oplus\sigma\oplus e_v}}} \geqslant 0.$$

Note that the derivative is non-negative for any sequence σ because of Equation (37). Thus, $\log(p_{\boldsymbol{\lambda}^1_j} + C^\sigma) - \log(p_{\boldsymbol{\lambda}^2_j} + C^\sigma)$ is non-decreasing with respect to C^σ. This is an important property when considering Condition (34). Based on this condition, $\lambda^{1,cost}_{j\oplus i^*\oplus e_v} \geqslant \lambda^{2,cost}_{j\oplus i^*\oplus e_v}$ which can be simplified as follows

$$\begin{aligned} &\log(p_{\boldsymbol{\lambda}^1_{j\oplus i^*\oplus e_v}}) + L^1(j\oplus i^*\oplus e_v) \geqslant \log(p_{\boldsymbol{\lambda}^2_{j\oplus i^*\oplus e_v}}) + L^2(j\oplus i^*\oplus e_v) \iff \\ &\log(p_{\boldsymbol{\lambda}^1_j} + p_{j,i^*} + p_{i^*,e_v}) - \log(p_{\boldsymbol{\lambda}^2_j} + p_{j,i^*} + p_{i^*,e_v}) \geqslant L^2(j) - L^1(j). \end{aligned}$$

The last inequality combined with the fact that $\log(p_{\boldsymbol{\lambda}^1_j} + C^\sigma) - \log(p_{\boldsymbol{\lambda}^2_j} + C^\sigma)$ is non-decreasing with respect to C^σ, and Equation (36) imply that

$$\begin{aligned} &\log(p_{\boldsymbol{\lambda}^1_j} + p_{j,\sigma} + p_{\sigma,e_v}) - \log(p_{\boldsymbol{\lambda}^2_j} + p_{j,\sigma} + p_{\sigma,e_v}) \geqslant \\ &\log(p_{\boldsymbol{\lambda}^1_j} + p_{j,i^*} + p_{i^*,e_v}) - \log(p_{\boldsymbol{\lambda}^2_j} + p_{j,i^*} + p_{i^*,e_v}) \geqslant L^2(j) - L^1(j). \end{aligned}$$

which is precisely what we sought to show (see Equation (35).) □

B.3 Proof of Proposition 6.3

For a given vehicle $v \in \mathcal{V}^*$, let $\boldsymbol{\lambda}_j^\ell = (\lambda_j^{\ell,cost}, \lambda_j^{\ell,time}, \lambda_j^{\ell,load}, \left(\lambda_j^{\ell,cust_i}\right)_{i\in\mathcal{D}})$, $\ell = 1, 2$, be two labels associated with (partial) paths, denoted by 1 and 2, ending at node $j \in \mathcal{N}_v \backslash \{e_v\}$. Let $\mathcal{N}^{\boldsymbol{\lambda}_j^1}$ and $\mathcal{N}^{\boldsymbol{\lambda}_j^2}$ be the set of nodes visited in partial paths 1 and 2, respectively. Label $\boldsymbol{\lambda}_j^1$ is said to dominate label $\boldsymbol{\lambda}_j^2$ if the following conditions hold:

$$(6.16) - (6.19),$$

$$p_{\boldsymbol{\lambda}_{j\oplus e_v}^1} > 0, \tag{38}$$

$$\zeta_i \geqslant 0 \quad \forall i \in \overline{\mathcal{D}}^{\boldsymbol{\lambda}_j^2} \tag{39}$$

$$\lambda_{j\oplus e_v}^{1,cost} \geqslant \lambda_{j\oplus e_v}^{2,cost} \tag{40}$$

where $\overline{\mathcal{D}}^{\boldsymbol{\lambda}_j^2} := \mathcal{D} \backslash \mathcal{N}^{\boldsymbol{\lambda}_j^2}$.

Proof. Proof. Let σ be an arbitrary sequence of nodes in $\overline{\mathcal{D}}^{\boldsymbol{\lambda}_j^2}$. We explore two cases in this proof:

- Case 1: $p_{j,\sigma} + p_{\sigma,e_v} \leqslant p_{j,e_v}$,
- Case 2: $p_{j,\sigma} + p_{\sigma,e_v} > p_{j,e_v}$,

where $p_{j,\sigma}$ is the profit gained by moving from node j to the start node of sequence σ and completing sequence σ. We also define p_{σ,e_v} as the profit obtained by moving from the end of sequence σ to node e_v.

First consider Case 1. We claim that $\lambda^{1,cost}_{j\oplus e_v} \geqslant \lambda^{1,cost}_{j\oplus\sigma\oplus e_v}$ and $\lambda^{2,cost}_{j\oplus e_v} \geqslant \lambda^{1,cost}_{j\oplus\sigma\oplus e_v}$ under Case 1. This can be derived by exploring the objective function of $\texttt{NSW-L-SP}_v$. Note that we must have that $\eta > 0$ by the duality theory. This implies that in $\texttt{NSW-L-SP}_v$, the value of $c_{r_v}\eta$ for longer routes is a smaller negative value, i.e., results in a smaller objective value. Moreover, we know that δ_v is a constant and has no impact in the process of optimization in $\texttt{NSW-L-SP}_v$. Additionally, by Inequality (39), we know that visiting more nodes from $\overline{\mathcal{D}}^{\boldsymbol{\lambda}^2_j}$ cannot improve the objective function of $\texttt{NSW-L-SP}_v$. Therefore, the only way to improve the value of the objective function in $\texttt{NSW-L-SP}_v$ is to increase the profit, which cannot happen because in Case 1 we have that $p_{j,\sigma} + p_{\sigma,e_v} \leqslant p_{j,e_v}$. Therefore, the claim is true and this combined by Inequality (40) imply that $\lambda^{1,cost}_{j\oplus e_v}$ dominates $\lambda^{2,cost}_{j\oplus e_v}$ and all possible extensions that fall into category of Case 1.

Next, consider Case 2. It is sufficient to show that

$$\lambda^{1,cost}_{j\oplus\sigma\oplus e_v} \geqslant \lambda^{2,cost}_{j\oplus\sigma\oplus e_v},$$

for any arbitrary (and possibly empty) sequence of nodes σ in $\overline{\mathcal{D}}^{\boldsymbol{\lambda}^2_j}$ that falls into category of Case 2. To that end, we first note that the objective function of the $\texttt{NSW-L-SP}_v$ consists of two parts: a linear function and a logarithmic function. So, for simplicity the objective function can be represented as $\log(\textsc{Profit}) + L(\textsc{Path})$ where $L(.)$ captures the linear part of the objective function. Using this notation, we would like to show that

$$\log(p_{\boldsymbol{\lambda}^1_{j\oplus\sigma\oplus e_v}}) + L^1(j\oplus\sigma\oplus e_v) \geqslant \log(p_{\boldsymbol{\lambda}^2_{j\oplus\sigma\oplus e_v}}) + L^2(j\oplus\sigma\oplus e_v).$$

for any arbitrary sequence of nodes σ in $\overline{\mathcal{D}}^{\boldsymbol{\lambda}^2_j}$ that falls into category of Case 2. We note that due to linearity, the contribution of $\sigma\oplus e_v$ in $L^1(j\oplus\sigma\oplus e_v)$ and $L^2(j\oplus\sigma\oplus e_v)$ is the same. Thus, the inequality can be further simplified to:

$$\log(p_{\boldsymbol{\lambda}^1_{j\oplus\sigma\oplus e_v}}) + L^1(j) \geqslant \log(p_{\boldsymbol{\lambda}^2_{j\oplus\sigma\oplus e_v}}) + L^2(j) \iff$$

$$\log(p_{\boldsymbol{\lambda}^1_{j\oplus\sigma\oplus e_v}}) - \log(p_{\boldsymbol{\lambda}^2_{j\oplus\sigma\oplus e_v}}) \geqslant L^2(j) - L^1(j) \iff$$
$$\log(p_{\boldsymbol{\lambda}^1_j} + p_{j,\sigma} + p_{\sigma,e_v}) - \log(p_{\boldsymbol{\lambda}^2} + p_{j,\sigma} + p_{\sigma,e_v}) \geqslant L^2(j) - L^1(j),$$

Note that if sequence σ is empty then by the definition of the immediate extension operator, $j \oplus \sigma \oplus e_v = j \oplus e_v$. So, in summary, we need to show that

$$\log(p_{\boldsymbol{\lambda}^1_j} + p_{j,\sigma} + p_{\sigma,e_v}) - \log(p_{\boldsymbol{\lambda}^2_j} + p_{j,\sigma} + p_{\sigma,e_v}) \geqslant L^2(j) - L^1(j), \tag{41}$$

for any arbitrary sequence σ in $\overline{\mathcal{D}}^{\boldsymbol{\lambda}^2_j}$ that falls into category of Case 2 in the remaining of the proof. We first note that under Case 2 and because of Inequalities (6.19) and (38), we have that

$$p_{\boldsymbol{\lambda}^2_{j\oplus\sigma\oplus e_v}} = p_{\boldsymbol{\lambda}^2_j} + p_{j,\sigma} + p_{\sigma,e_v} \geqslant p_{\boldsymbol{\lambda}^1_j} + p_{j,\sigma} + p_{\sigma,e_v} = p_{\boldsymbol{\lambda}^1_{j\oplus\sigma\oplus e_v}} \geqslant p_{\boldsymbol{\lambda}^1_{j\oplus e_v}} = p_{\boldsymbol{\lambda}^1_j} + p_{j,e_v} > 0. \tag{42}$$

Therefore, we can conclude that the function $\log(p_{\boldsymbol{\lambda}^1_j}+p_{j,\sigma}+p_{\sigma,e_v})-\log(p_{\boldsymbol{\lambda}^2_j}+p_{j,\sigma}+p_{\sigma,e_v})$ in the left side of Equation (41) is not undefined (i.e., the terms in logs are positive) for any arbitrary sequence σ that falls into category of Case 2. Next, we show that this function is non-decreasing with respect to $C^\sigma := p_{j,\sigma} + p_{\sigma,e_v}$. The derivative of $\log(p_{\boldsymbol{\lambda}^1_j} + C^\sigma) - \log(p_{\boldsymbol{\lambda}^2_j} + C^\sigma)$ with respect to C^σ is

$$\frac{1}{p_{\boldsymbol{\lambda}^1_j} + C^\sigma} - \frac{1}{p_{\boldsymbol{\lambda}^2_j} + C^\sigma} = \frac{1}{p_{\boldsymbol{\lambda}^1_{j\oplus\sigma\oplus e_v}}} - \frac{1}{p_{\boldsymbol{\lambda}^2_{j\oplus\sigma\oplus e_v}}} \geqslant 0.$$

Note that the derivative is non-negative for any sequence σ because of Equation (42). Thus, $\log(p_{\boldsymbol{\lambda}^1_j} + C^\sigma) - \log(p_{\boldsymbol{\lambda}^2_j} + C^\sigma)$ is non-decreasing with respect to C^σ. This is an important property when considering Condition (40). Based on this condition, $\lambda^{1,cost}_{j\oplus e_v} \geqslant \lambda^{2,cost}_{j\oplus e_v}$ which can be simplified as follows

$$\log(p_{\boldsymbol{\lambda}^1_{j\oplus e_v}}) + L^1(j \oplus e_v) \geqslant \log(p_{\boldsymbol{\lambda}^2_{j\oplus e_v}}) + L^2(j \oplus e_v) \iff$$

$$\log(p_{\boldsymbol{\lambda}_j^1} + p_{j,e_v}) - \log(p_{\boldsymbol{\lambda}_j^2} + p_{j,e_v}) \geqslant L^2(j) - L^1(j).$$

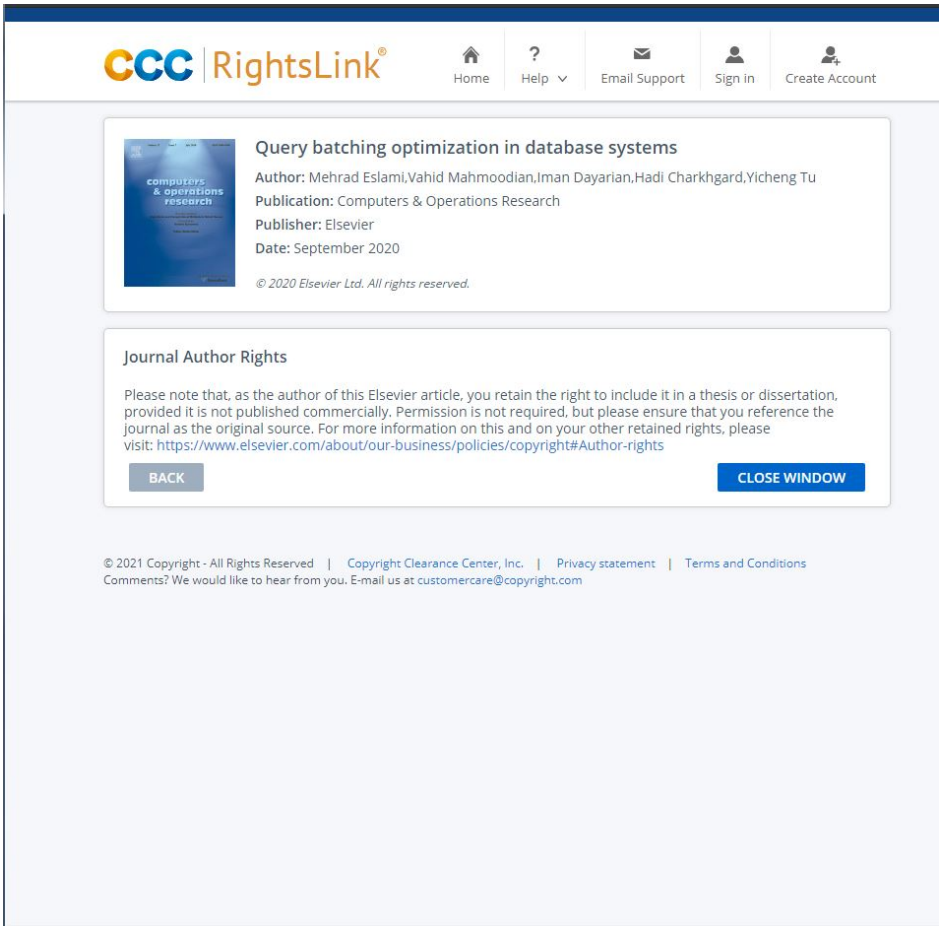
The last inequality combined with the fact that $\log(p_{\boldsymbol{\lambda}_j^1} + C^\sigma) - \log(p_{\boldsymbol{\lambda}_j^2} + C^\sigma)$ is non-decreasing with respect to C^σ, and the definition of Case 2, imply that

$$\log(p_{\boldsymbol{\lambda}_j^1} + p_{j,\sigma} + p_{\sigma,e_v}) - \log(p_{\boldsymbol{\lambda}_j^2} + p_{j,\sigma} + p_{\sigma,e_v}) \geqslant$$
$$\log(p_{\boldsymbol{\lambda}_j^1} + p_{j,e_v}) - \log(p_{\boldsymbol{\lambda}_j^2} + p_{j,e_v}) \geqslant L^2(j) - L^1(j).$$

which is precisely what we sought to show (see Equation (41).)

Appendix C: Copyright Permissions

C.1 Reprint Permission for Chapter 3



C.2 Reprint Permission for Chapter 4