Université de Montréal

Towards the Reduction of Greenhouse Gas Emissions: Models and Algorithms for Ridesharing and Carbon Capture and Storage

par

Gabriel André Homsi

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée en vue de l'obtention du grade de
Philosophiæ Doctor (Ph.D.)
en Informatique

June 15, 2023

Université de Montréal

Faculté des arts et des sciences

Cette thèse intitulée

Towards the Reduction of Greenhouse Gas Emissions: Models and Algorithms for Ridesharing and Carbon Capture and Storage

présentée par

Gabriel André Homsi

a été évaluée par un jury composé des personnes suivantes :

Margarida Carvalho
(président-rapporteur)

Sanjay Dominik Jena
(directeur de recherche)

Fabian Bastin
(membre du jury)

Antoine Legrain
(examinateur externe)

Margarida Carvalho
(représentant du doyen de la FESP)

Résumé

Avec la ratification de l'Accord de Paris, les pays se sont engagés à limiter le réchauffement climatique bien en dessous de 2, de préférence à 1,5 degrés Celsius, par rapport aux niveaux préindustriels. À cette fin, les émissions anthropiques de gaz à effet de serre (GES, tels que CO_2) doivent être réduites pour atteindre des émissions nettes de carbone nulles d'ici 2050. Cet objectif ambitieux peut être atteint grâce à différentes stratégies d'atténuation des GES, telles que l'électrification, les changements de comportement des consommateurs, l'amélioration de l'efficacité énergétique des procédés, l'utilisation de substituts aux combustibles fossiles (tels que la bioénergie ou l'hydrogène), le captage et le stockage du carbone (CSC), entre autres. Cette thèse vise à contribuer à deux de ces stratégies : le covoiturage (qui appartient à la catégorie des changements de comportement du consommateur) et la capture et le stockage du carbone. Cette thèse fournit des modèles mathématiques et d'optimisation et des algorithmes pour la planification opérationnelle et tactique des systèmes de covoiturage, et des heuristiques pour la planification stratégique d'un réseau de captage et de stockage du carbone.

Dans le covoiturage, les émissions sont réduites lorsque les individus voyagent ensemble au lieu de conduire seuls. Dans ce contexte, cette thèse fournit de nouveaux modèles mathématiques pour représenter les systèmes de covoiturage, allant des problèmes d'affectation stochastique à deux étapes aux problèmes d'empaquetage d'ensembles stochastiques à deux étapes qui peuvent représenter un large éventail de systèmes de covoiturage. Ces modèles aident les décideurs dans leur planification opérationnelle des covoiturages, où les conducteurs et les passagers doivent être jumelés pour le covoiturage à court terme. De plus, cette thèse explore la planification tactique des systèmes de covoiturage en comparant différents modes de fonctionnement du covoiturage et les paramètres de la plateforme (par exemple, le partage des revenus et les pénalités). De nouvelles caractéristiques de problèmes sont étudiées, telles que l'incertitude du conducteur et du passager, la flexibilité de réappariement et la réservation de l'offre de conducteur via les frais de réservation et les pénalités. En particulier, la flexibilité de réappariement peut augmenter l'efficacité d'une plateforme de covoiturage, et la réservation de l'offre de conducteurs via les frais de réservation et les pénalités peut augmenter la satisfaction des utilisateurs grâce à une compensation garantie si un covoiturage

n'est pas fourni. Des expériences computationnelles détaillées sont menées et des informations managériales sont fournies.

Malgré la possibilité de réduction des émissions grâce au covoiturage et à d'autres stratégies d'atténuation, des études macroéconomiques mondiales montrent que même si plusieurs stratégies d'atténuation des GES sont utilisées simultanément, il ne sera probablement pas possible d'atteindre des émissions nettes nulles d'ici 2050 sans le CSC. Ici, le CO_2 est capturé à partir des sites émetteurs et transporté vers des réservoirs géologiques, où il est injecté pour un stockage à long terme. Cette thèse considère un problème de planification stratégique multipériode pour l'optimisation d'une chaîne de valeur CSC. Ce problème est un problème combiné de localisation des installations et de conception du réseau où une infrastructure CSC est prévue pour les prochaines décennies. En raison des défis informatiques associés à ce problème, une heuristique est introduite, qui est capable de trouver de meilleures solutions qu'un solveur commercial de programmation mathématique, pour une fraction du temps de calcul. Cette heuristique comporte des phases d'intensification et de diversification, une génération améliorée de solutions réalisables par programmation dynamique, et une étape finale de raffinement basée sur un modèle restreint. Dans l'ensemble, les contributions de cette thèse sur le covoiturage et le CSC fournissent des modèles de programmation mathématique, des algorithmes et des informations managériales qui peuvent aider les praticiens et les parties prenantes à planifier des émissions nettes nulles.

Mots-clés : affectation de covoiturage; captage et stockage du carbone; programmation stochastique; heuristiques.

Abstract

With the ratification of the Paris Agreement, countries committed to limiting global warming to well below 2, preferably to 1.5 degrees Celsius, compared to pre-industrial levels. To this end, anthropogenic greenhouse gas (GHG) emissions (such as CO_2) must be reduced to reach net-zero carbon emissions by 2050. This ambitious target may be met by means of different GHG mitigation strategies, such as electrification, changes in consumer behavior, improving the energy efficiency of processes, using substitutes for fossil fuels (such as bioenergy or hydrogen), and carbon capture and storage (CCS). This thesis aims at contributing to two of these strategies: ridesharing (which belongs to the category of changes in consumer behavior) and carbon capture and storage. This thesis provides mathematical and optimization models and algorithms for the operational and tactical planning of ridesharing systems, and heuristics for the strategic planning of a carbon capture and storage network.

In ridesharing, emissions are reduced when individuals travel together instead of driving alone. In this context, this thesis provides novel mathematical models to represent ridesharing systems, ranging from two-stage stochastic assignment problems to two-stage stochastic set packing problems that can represent a wide variety of ridesharing systems. These models aid decision makers in their operational planning of rideshares, where drivers and riders have to be matched for ridesharing on the short-term. Additionally, this thesis explores the tactical planning of ridesharing systems by comparing different modes of ridesharing operation and platform parameters (e.g., revenue share and penalties). Novel problem characteristics are studied, such as driver and rider uncertainty, rematching flexibility, and reservation of driver supply through booking fees and penalties. In particular, rematching flexibility may increase the efficiency of a ridesharing platform, and the reservation of driver supply through booking fees and penalties may increase user satisfaction through guaranteed compensation if a rideshare is not provided. Extensive computational experiments are conducted and managerial insights are given.

Despite the opportunity to reduce emissions through ridesharing and other mitigation strategies, global macroeconomic studies show that even if several GHG mitigation strategies are used simultaneously, achieving net-zero emissions by 2050 will likely not be possible without CCS. Here, CO_2 is captured from emitter sites and transported to geological reservoirs, where

it is injected for long-term storage. This thesis considers a multiperiod strategic planning problem for the optimization of a CCS value chain. This problem is a combined facility location and network design problem where a CCS infrastructure is planned for the next decades. Due to the computational challenges associated with that problem, a slope scaling heuristic is introduced, which is capable of finding better solutions than a state-of-the-art general-purpose mathematical programming solver, at a fraction of the computational time. This heuristic has intensification and diversification phases, improved generation of feasible solutions through dynamic programming, and a final refining step based on a restricted model. Overall, the contributions of this thesis on ridesharing and CCS provide mathematical programming models, algorithms, and managerial insights that may help practitioners and stakeholders plan for net-zero emissions.

Keywords: ridesharing matching; carbon capture and storage; stochastic programming; heuristics.

Contents

List of tables

List of figures

List of abbreviations

2S-SDMP	2-stage Stochastic Driver Matching Problem
BECCS	Bioenergy with carbon capture and storage
CCS	Carbon capture and storage
CDR	Carbon dioxide removal
CO_2	Carbon dioxide
EVP	Expected value problem
GHG	Greenhouse gas
H_2	Hydrogen
HOV	High-occupancy vehicle
MILP	Mixed-integer linear programming
SAA	Sample average approximation

SS	Slope scaling
VCO	Value chain optimization

Acknowledgments

I would like to thank professors Sanjay Dominik Jena and Bernard Gendron for their guidance. I would also like to extend my gratitude to Mathieu Lozeau and Netlift for providing indispensable domain knowledge and data for the ridesharing-related projects on this thesis. I also wish to thank Étienne Ayotte-Sauvé and CanmetENERGY Varennes for their mentoring, support, and involvement in the carbon capture and storage project of this thesis. Thanks also go to Richard Middleton and the SimCCS Gateway team, in particular Ryan Kammer at Indiana University, for sharing carbon capture and storage data for Alberta oil sands and SimCCS Gateway case studies (respectively) with the CanmetENERGY team.

I would also like to thank the following institutions for their support throughout this doctorate: the Université de Montréal, the Centre interuniversitaire de recherche sur les reseaux d'entreprise, la logistique et le transport (CIRRELT), the Département d'informatique et de recherche opérationnelle (DIRO, Université de Montréal), CanmetENERGY Varennes (Government of Canada), and the Chaire en transformation du transport (CTT, Université de Montréal/Polytechnique Montréal), which is funded by the ministère de l'Économie, de l'Innovation et de l'Énergie du Québec. Thanks also go to Calcul Québec and the Digital Research Alliance of Canada, which provided valuable computing resources throughout this doctorate.

Chapter 1

Introduction

A net-zero emissions target has to be met by 2050 to reach the objective of the Paris Agreement: to limit global warming to 2 degrees Celsius (preferably 1.5 degrees Celsius) compared to pre-industrial levels. This requires a radical reduction of anthropogenic greenhouse gas (GHG) emissions. Out of all GHG emissions, CO_2 is the most prevalent one. Emissions can be mitigated in several ways, such as electrification, changes in user behavior, improving the energy efficiency of processes, using substitutes for fossil fuels such as biomass and hydrogen, and carbon capture and storage (CCS). All strategies listed above have an impact on the reduction of GHG emissions, and the cement industry is an example that shows that a combination of mitigation strategies is needed to sufficiently reduce emissions. In this industry, about 60% of the CO_2 emissions in a cement plant come from the calcination reaction, and not from the burning of fossil fuels [Strunge et al., 2022]. Thus, even if fossil fuels are completely replaced in the cement industry, a considerable amount of emissions will be left unaddressed. These unaddressed emissions can be mitigated with CCS. Further, the replacement of fossil fuels with, for example, biomass, poses additional challenges: biomass is a limited resource, and its demand will grow with the decarbonization of industrial sectors. Thus, the supply chain of biomass will have to be adapted, and a growth in agricultural production will be needed to satisfy the ever-increasing needs for biomass [Popp et al., 2021]. Overall, macroeconomic studies around the world show that several GHG mitigation strategies have to be used in parallel to sufficiently reduce emissions, and that the net-zero target is unlikely to be met without CCS [Lane et al., 2021, Riahi et al., 2017, IEA, 2017].

To address the need for implementing several GHG mitigation strategies in parallel, this thesis focuses on the operational, tactical, and strategic planning of two mitigation strategies. The first strategy concerns the operational and tactical planning of ridesharing systems as a means to reduce the total distance traveled by individuals when commuting. Ridesharing falls into the scope of changes in user behavior GHG mitigation strategies. The second strategy concerns the strategic planning of CCS, where CO_2 is captured from emitter sites, and

transported into geological reservoirs (inshore or offshore) where it is injected for long-term storage. CCS allows for the mitigation of emissions in sectors of the economy where, for example, CO_2 is a byproduct of chemical reactions, or in sectors where replacing fossil fuels is not attainable in the short to medium term. In this thesis, the planning of ridesharing and CCS is done by the use of Operations Research tools. More specifically, mathematical programming models and algorithms for optimizing these models. With better planning, the overall performance of ridesharing and CCS may improve, which may reduce operating costs and improve the GHG mitigation potential of these systems. In the following, further details are provided on the mathematical models behind the planning problems associated with these mitigation strategies.

1.1. Ridesharing systems

Ridesharing systems are agencies that match drivers and riders such that they can travel together to fulfill their itineraries, rather than driving alone. Individuals may rideshare for a multitude of reasons, such as reducing their transportation costs, reducing their carbon footprint, lack of access to a vehicle, or unreliable access to mass transit. In the Operations Research literature, ridesharing systems are often modeled as one-to-one bipartite matching problems [e.g., Agatz et al., 2011, Wang et al., 2018], as below:

$$\max \sum_{i \in D} \sum_{j \in R} s_{ij} x_{ij} \tag{1.1.1}$$

$$\text{s.t.} \sum_{j \in R} x_{ij} \leq 1, \forall i \in D \tag{1.1.2}$$

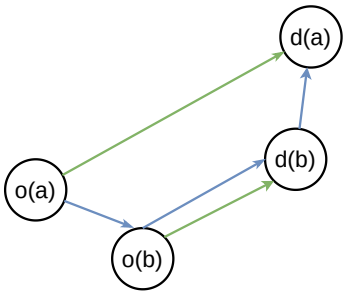
$$\sum_{i \in D} x_{ij} \leq 1, \forall j \in R \tag{1.1.3}$$

$$x_{ij} \in \{ 0, 1 \}, \tag{1.1.4}$$

where D is a set of drivers and R is a set of riders available for ridesharing. For each driver $i \in D$ and rider $j \in R$, let x_{ij} be a binary variable valued to one if and only if driver i rideshares with rider j. The value generated when driver i rideshares with rider j is represented by s_{ij}. This value can represent, for example, the distance savings generated by this rideshare (illustrated in Figure 1.1). Thus, the objective function of the model above maximizes the total value generated by rideshares, while constraints (1.1.2) and (1.1.3) ensure that a driver can be matched to at most one rider and that a rider can be matched to at most one driver, respectively. The model above is a mixed-integer programming problem, but the integrality constraints (1.1.4) can be relaxed, as the constraint matrix of this formulation is totally unimodular and therefore the optimal solution of the resulting linear programming relaxation is integral [Edmonds and Johnson, 2003]. Thus, the bipartite matching problem can be represented as a linear program, which is solvable in polynomial time. The bipartite

matching model has been adapted to represent ridesharing systems with different attributes, such as matching stability [so that participants have little incentive to leave the ridesharing platform, see Wang et al., 2018]. Different types of ridesharing systems are reviewed in Chapter 2.

Fig. 1.1. Distance savings generated by a rideshare between driver a and rider b, where $o(a)$ and $o(b)$ correspond to the origin coordinates of a and b, respectively. Parameters $d(a)$ and $d(b)$ correspond to the destination coordinates of a and b, respectively. Individual travel distances are in green, and shared travel distances are in blue.



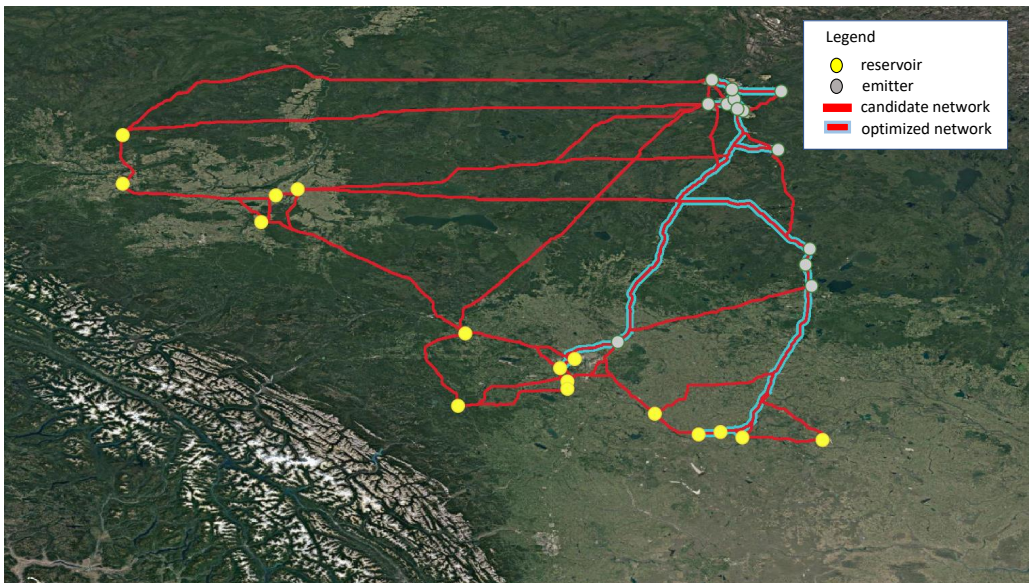
The ridesharing papers of this thesis concern both operational and tactical planning of ridesharing systems: in operational planning, rideshares for the short-term are planned, and in tactical planning, different ridesharing system attributes are evaluated to identify the trade-off of different modes of operation. The first two articles of this thesis (Chapter 3 and Chapter 4) extend the bipartite matching model to a multiperiod and stochastic one-to-one matching problem for ridesharing. The problem allows for the dynamic matching of drivers and riders, as well as for the rematching of drivers and riders such that ridesharing agreements can be rearranged to improve the performance of the ridesharing system. The rematching operation comes at a cost: the system has to pay a penalty, which may represent monetary compensation to participants. The third article of this thesis (Chapter 5) continues to focus on ridesharing under uncertainty, but allows for rideshares with more than one rider (many-to-one matching). The problem is modeled as a two-stage stochastic set-packing model, and is general enough to represent different ridesharing modes of operation. Overall, the ridesharing papers of this thesis explore how user incentives (or compensation due to inconveniences) can be implemented in ridesharing systems and what impact these incentives have on the system profitability. These papers also explore how ridesharing systems can be optimized under uncertainty. Based on computational experiments, insights on incentive policies are given, as well as insights on optimization under uncertainty for ridesharing.

1.2. Carbon capture and storage

In CCS, CO_2 emissions are first captured from emitter sites and then transported to geological reservoirs where it is injected for long-term storage. Emitter sites can be industrial

facilities in different sectors, such as steel, cement, pulp and paper, bioenergy, fertilizer and chemicals, combustion-based power, oil and gas, and hydrogen generation. Geological reservoirs can be, for example, saline aquifers and depleted oil reservoirs. The transportation of CO_2 can be done through different modes, such as pipelines, ships, trucks, and trains. The deployment of a CCS infrastructure may span several decades and require billions of dollars of investments. To help stakeholders assess the costs of CCS, strategic planning models for the optimization of a CCS value chain can be used. Such models have combined facility location and network design decisions, and may contain a single time period or multiple time periods, depending on the level of accuracy required when assessing CCS costs. In these models, sources correspond to emitter sites, sinks correspond to geological reservoirs, and a network corresponds to the CO_2 transportation infrastructure (e.g., pipelines, trains, ships). An optimized CCS infrastructure is illustrated in Figure 1.2.

Fig. 1.2. Example of a CCS network from an Alberta oil sands case study [Middleton and Brandt, 2013]. The data necessary to build this case study was provided by Richard Middleton.



The multiperiod CCS pipeline value chain optimization model studied in Chapter 6 is based on the model of Jones et al. [2022]. In this model, an objective function minimizes (for the whole planning horizon) the costs of activating and operating sources (emitter sites), activating and operating sinks (geological reservoirs), and activating a network (pipeline network). The operating costs are associated with the CO_2 capture and injection rates at sources and sinks. Sources and sinks can be opened only once during the planning horizon, and CO_2 can only be captured from sources and injected at sinks if these sites were previously activated. Each source and sink has a maximum CO_2 capture rate (capacity constraints). Furthermore, sinks have two levels of decisions: the activation of geological reservoirs, and

the digging of injection wells at these reservoirs. Each well has a maximum CO_2 injection rate, and each reservoir also has a maximum CO_2 injection rate as well as a maximum number of injection wells that can be dug. Finally, each reservoir has a lifetime injection capacity. The activation of pipelines is a discrete decision. The capacity of pipelines is a continuous decision, and the total cost of activating a pipeline at a certain capacity is represented by a piecewise linear function. A constraint specifies that, for each time period, a certain amount of CO_2 has to be captured from wells and injected at reservoirs. Together with this constraint, mass balance constraints ensure the flow of CO_2 from sources to sinks. Due to the combination of multiperiod facility location decisions and multiperiod network design decisions, solving this model poses considerable computational challenges. The CCS paper of this thesis addresses a computational need of analysts: high-quality solutions for strategic CCS planning problems have to be generated quickly, such that analysts can perform sensitivity studies that may contain hundreds or thousands of experiments. To address this need, this thesis provides a novel slope scaling heuristic that generates (on average) better solutions than a state-of-the-art mathematical programming solver, at a fraction of the time.

1.3. Organization of this thesis

This thesis provides mathematical programming models and solution methodologies for operational and strategic planning problems with greenhouse gas emission mitigation potential. In Chapter 2, a literature review on GHG emissions mitigation strategies is conducted, with a focus on ridesharing and CCS as potential strategies. The main scientific contributions of this thesis are given by four articles (corresponding to Chapter 3, Chapter 4, Chapter 5, and Chapter 6). In Chapter 3, a stochastic one-to-one multiperiod ridesharing problem with matching and rematching is introduced, along with two-stage stochastic programming models and reduction techniques to reduce the number of variables of multiperiod models. In Chapter 4, the model of Chapter 3 is adapted and empirically evaluated on a rolling horizon framework, and different system parameters are evaluated in order to understand their impact on system performance. Namely, penalties for unmatching and also penalties for delayed matches. In Chapter 5, a one-to-many two-stage stochastic ridesharing problem is proposed. In this problem, drivers must be booked in advance for a fee, and a penalty may be paid if drivers are not assigned to any rideshare. Such problem is general enough to represent different ridesharing modes of operation, which are discussed and computationally evaluated. The computational results are then analyzed, leading to managerial insights on these operation modes. In Chapter 6, a slope scaling heuristic for a multiperiod CCS problem is proposed. This problem has characteristics of combined facility location problems and network design problems. Computational experiments show that the slope scaling heuristic generates better solutions than a commercial mixed-integer linear programming solver, at a

fraction of the computational time. Chapter 7 concludes this thesis and provides possible directions for future work. Appendix A provides supplementary material for Chapter 5.

Chapter 2

Literature Review

This chapter first conducts a general review of the literature on greenhouse gas (GHG) mitigation strategies in Section 2.1. Then, two specific GHG mitigation strategies that are studied throughout this thesis are reviewed: ridesharing systems, reviewed in Section 2.2, and carbon capture and storage (CCS), reviewed in Section 2.3. For further reviews on ridesharing, we refer to the literature reviews written in Chapters 3, 4, and 5. For a further review of CCS, we refer the literature review in Chapter 6.

2.1. Review on greenhouse gas mitigation strategies

To sufficiently reduce anthropogenic GHG as an effort to meet the targets of the Paris Agreement, combining several GHG mitigation strategies is necessary [Riahi et al., 2017]. We give an overview of some of these main strategies. Namely, electrification, replacements to fossil fuels, changes in consumer behavior, and CCS.

2.1.1. Replacements to fossil fuels

Carbon-intensive fossil fuels can be replaced by cleaner alternatives [Riahi et al., 2017]. For example, in the shipping industry, liquefied natural gas can be used instead of conventional marine fuels such as heavy fuel oil and marine gas oil, as it generates fewer CO_2 emissions [Comer and Sathiamoorthy, 2022]. Alternatively, renewable energy can be used instead of fossil fuels, such as solar and wind energy [Granovskii et al., 2007], and bioenergy, where biomass is used as a fossil fuel replacement to, for example, heat boilers [Saidur et al., 2011]. Furthermore, hydrogen can be used as another fossil fuel replacement, as it does not release CO_2 when it is used to produce energy [Razi and Dincer, 2022].

2.1.2. Electrification

Electrification may contribute to reducing emissions in several sectors that traditionally depend on fossil fuels for energy, such as agriculture [e.g., with the electrification of agricultural

machinery, see Scolaro et al., 2021], transportation [e.g., the electrification of buses, see Xylia et al., 2019], and in commercial and residential buildings [e.g., the electrification of space heating and water heating, see Dennis, 2015].

2.1.3. Changes in consumer behavior

Changes in consumer habits can contribute to the mitigation of GHG emissions. In Girod et al. [2014], the authors review the carbon footprint of products in the five main consumption categories: food, shelter, travel, goods, and services. For each category, the authors identify consumption options compatible with the Paris Agreement target to limit global warming. In Matasci et al. [2021], the authors conduct an analysis of the consumer environmental impact (both direct and indirect) on different sectors of the Swiss economy. The potential for mitigating agricultural GHG emissions through consumer behavior together with biotechnology and organic systems is studied in Del Grosso and Grant [2011].

In the field of transportation, researchers seek to understand the factors that influence the choice of transport mode by individuals, with the goal of minimizing car-based trips and therefore minimizing emissions [Comendador et al., 2014]. The efficient planning of ridesharing systems can influence consumer choice, leading to more drivers and riders choosing ridesharing over driving alone.

2.1.4. Carbon capture and storage

Finally, CCS [Smit et al., 2014] can be used to address the emissions that have not been reduced by other strategies. In CCS, CO_2 is captured from emitter sites and transported to underground geological reservoirs (inshore or offshore), where it is injected for long-term storage. CCS is part of a group called carbon dioxide removal (CDR) technologies and practices [Terlouw et al., 2021]. Other CDR methods include afforestation and reforestation [Trabucco et al., 2008], soil carbon sequestration [Lal, 2004], biochar [Lehmann et al., 2021], enchanced rock weathering [Beerling et al., 2020], bioenergy with CCS [Fridahl and Lehtveer, 2018], direct air CCS [Gambhir and Tavoni, 2019], and ocean fertilization [Lampitt et al., 2008].

2.2. Review on ridesharing

In ridesharing, individuals travel together (e.g., from home to a common work location) to fulfill their itineraries rather than driving alone. Individuals may engage in ridesharing for a multitude of reasons. For example: to reduce transportation costs, to reduce their environmental footprint, or because they do not have access to a car or to a reliable form of mass transit. Ridesharing leads to higher vehicle occupancy and reduced vehicle miles traveled, and therefore reductions in fuel consumption and GHG emissions [Shaheen et al., 2018,

Coulombel et al., 2019, Yan et al., 2020]. Minimizing the carbon footprint of transportation systems [e.g., in freight transportation Bektaş et al., 2019] is part of the field of green logistics [see Dekker et al., 2012]. A particular field within green logistics is the field of shared mobility, which includes the sharing of transportation equipment such as car sharing [Jung and Koo, 2018], bike sharing [Kou et al., 2020], and the sharing of the transportation itself, such as ridesharing [Teal, 1987].

The practice of ridesharing is not new, and dates as back as WWII. A growing number of ridesharing systems benefited from the introduction of mobile phones and the Internet [Chan and Shaheen, 2012, Dailey et al., 1999]. Chronologically, ridesharing had several phases, such as the ridesharing clubs during WWII, ridesharing as a response to the 1970s oil crisis, and today's technology-enabled ridesharing systems where individuals that do not know each other can be matched for ridesharing [Chan and Shaheen, 2012]. Despite the practice of ridesharing not being new, ridesharing systems still have a large potential for growth: in 2017, the United States Department of Transportation estimated that the average vehicle occupancy rate in the United States is about 1.67 person per vehicle, and the occupancy rate for vehicles used for work-related trips is even smaller [McGuckin and Fucci, 2018]. Thus, there is a large fleet of vehicles that are operating under capacity, and the individuals that are driving these vehicles for commuting purposes could engage in ridesharing. Some benefits of a growth in ridesharing participation include the reduction of GHG emissions [Shaheen et al., 2018, Coulombel et al., 2019, Yan et al., 2020], and the reduction of traffic congestion and demand for parking infrastructure [Shaheen et al., 2018].

2.2.1. Surveys on ridesharing and related shared mobility systems

There are several reviews on ridesharing and related shared mobility systems in the literature. In Agatz et al. [2012], the authors outline the optimization challenges in dynamic ridesharing and survey related Operations Research models in the literature. Some optimization challenges outlined by the authors are modeling related, such as:

(1) the number of participants in a rideshare (single versus multiple riders);
(2) the possibility of transfers between multiple rideshares;
(3) the dynamic arrival of ridesharing requests;
(4) reoptimization policies upon the release of new information;
(5) anticipation of future requests;
(6) deviations from planned trips.

This thesis provides contributions to points (1), (3), (4), (5), and (6). Further, the authors identify key characteristics of dynamic ridesharing systems:

- rideshares can be established on short notice;
- drivers that engage in ridesharing are independent;

- trip-related costs on a rideshare are distributed among participants such that it is financially beneficial for them to rideshare;
- trips in dynamic ridesharing are non-recurrent, as opposed to traditional ridesharing systems that require a long-term commitment among participants;
- trips are prearranged: participants agree in advance to rideshare;
- the matching of drivers and riders is automated.

In Furuhata et al. [2013], the authors highlight that despite the convenience of modern communication systems, the widespread use of ridesharing still faces challenges. Three key challenges highlighted by the authors are: the design of pricing mechanisms that are attractive to drivers and riders, a ride arrangement that takes into consideration user preferences and other system attributes, and building trust between unknown drivers and riders in ridesharing systems. The authors present a classification of 39 ridesharing companies and introduce a framework to help identify key challenges in the widespread adoption of ridesharing. On a wider scope, Mourad et al. [2019], survey models and algorithms for optimizing shared mobility systems. The authors classify shared mobility systems in two groups: systems where people share rides (e.g., ridesharing, vanpooling, dial-a-ride), and systems where parcel transportation and people transportation are combined (e.g., combined delivery, share-a-ride, crow-sourcing). A more recent survey [Martins et al., 2021] reviews ridesharing and prearranged carpooling optimization problems. The authors classify optimization papers according to solution methodology, identify ridesharing and prearranged carpooling challenges in smart and sustainable cities, and discuss algorithmic venues for the generation of real-time solutions, which may be required in dynamic ridesharing systems.

2.2.2. Casual and organized ridesharing

Ridesharing can be either casual or organized. Casual ridesharing (also called flexible carpooling, casual carpooling, or slugging) is user-run, where impromptu ridesharing groups are formed in meeting points such as park-and-ride facilities or public transit centers [Chan and Shaheen, 2012]. On the other hand, in organized ridesharing, an intermediate platform acts as a broker that matches drivers and riders. Such platforms can simply advertise the itineraries of drivers and riders. Users can then communicate with each other to plan rideshares, or the platform can suggest users a specific rideshare. These platforms allow for the creation of rideshares between individuals with no previous historical involvements [Dailey et al., 1999], and usually charge a small fee from participants or take a cut of the monetary savings generated by the rideshare.

2.2.3. The value of ridesharing

A common assumption in the literature is that individuals will consider engaging in ridesharing only if it brings financial benefit to them [e.g., Agatz et al., 2011]. For example, individuals may rideshare if the cost of ridesharing is smaller than the cost of some alternative form of transportation, such as the cost of driving alone. Additionally, riders that participate in ridesharing benefit from not having to pay the upfront costs and maintenance costs of owning a car. The scientific literature on ridesharing often assumes that the monetary value associated with a rideshare equates to the amount of travel distance savings generated by the trip when compared to the individual trips for each participant [Agatz et al., 2011, Wang et al., 2018]. This is referred to the *distance savings* generated by a rideshare, and this metric can help estimate the GHG emissions mitigation potential of ridesharing. Such savings have monetary value, and this value can be distributed among the participants and the ridesharing company when calculating the ridesharing trip costs. The literature on Shapley values for ridesharing discusses how these distance savings (or trip costs) can be distributed among ridesharing participants [e.g., see Levinger et al., 2020].

2.2.4. Incentives for the adoption of ridesharing

The adoption of ridesharing can be driven by several incentives. Ridesharing not only benefits the participating individuals: as ridesharing leads to fewer cars on the road and therefore to less congestion, it brings benefits to mass transit systems (e.g., buses), taxis, and individuals who are driving alone. Thus, policymakers may have an interest in promoting ridesharing, and may achieve that by implementing incentive policies. For example, in order to make ridesharing more appealing than driving alone, reserved parking may be offered to individuals who rideshare [Brownstone and Golob, 1992]. Further, high-occupancy vehicle (HOV) lanes can be implemented, where one or more lanes of a road or highway are reserved for rideshares, taxis, and mass transit [Giuliano et al., 1990]. Similar to HOV lanes, high-occupancy toll lanes can be introduced, where ridesharing vehicles and other vehicles (e.g., buses) can drive toll-free, but all other users of the lane (e.g., solo drivers) have to pay a toll [Kon, 2010]. Another way of making ridesharing user-friendly is to guarantee rides back home if individuals rideshare to work [Brownstone and Golob, 1992]. Ridesharing can also be promoted through carbon-emission reduction certification and carbon trading programs, where users of the platform can participate in carbon trading due to the carbon emission reductions generated when ridesharing [Si et al., 2022].

2.2.5. Different types of ridesharing systems

A wide variety of ridesharing systems with different problem characteristics and assumptions is studied in the literature. Ridesharing involves a driver picking up one or more

passengers, driving to each passenger's destination, and finally driving to the driver's destination. Further, participants may have time constraints when ridesharing. Thus, ridesharing can be viewed as a form of pickup and delivery problem with time windows [see Parragh et al., 2008a,b, for surveys on different pickup and delivery problems].

In its simplest form, one-to-one ridesharing, a driver picks up only one passenger. These systems are therefore often modeled as bipartite matching problems, which are solvable in polynomial time. This is the case of the dynamic driver-rider matching problem by Agatz et al. [2011]. Alternatively, in one-to-many ridesharing, a driver picks up one or more passengers. These systems therefore require modeling complex vehicle routing decisions through, for example, three-index flow formulations or set partitioning formulations, which are known to be NP-hard. This is the case in Baldacci et al. [2004], who introduced a dial-a-ride problem for ridesharing. In this problem, a driver is able to pick up multiple riders, and all participants have the same destination (e.g., the workplace). To solve this problem, the authors propose an exact method.

Some studies take into account fairness when planning rideshares in order to encourage a loyal user base. For example, Wang et al. [2018] study a ridesharing problem with matching stability, such that participants are less likely to quit the system and arrange rideshares by themselves. This study was later extended by Peng et al. [2022] to consider a taxi-sharing setting with stability constraints. Both studies propose a one-to-many problem and model them either via a set-packing or a set-partitioning formulation. In Peng et al. [2022], the authors propose a branch-and-price algorithm to solve their one-to-many taxi-sharing problem with stability constraints.

To improve the efficiency of ridesharing, [Stiglic et al., 2015] evaluate the impact of meeting points, where participants agree to be picked up and dropped off at meeting points within a certain distance of their origins and destinations. The authors conclude that meeting points can significantly increase the system-wide distance savings and the number of matched participants. Meeting points imply geographical flexibility. Alternatively, temporal flexibility is evaluated by Stiglic et al. [2016], where flexible departure times or detour times can significantly increase the likelihood that drivers and riders find a match for ridesharing.

Some studies focus on the integration of ridesharing with mass transit systems. For example, in Stiglic et al. [2018], the authors evaluate the performance of this integration according to driver flexibility, park-and-ride matches (i.e., the driver parks and takes mass transit after dropping off riders at a transit station), number of participants, and mass transit system parameters such as train speed and departure frequency. When considering integration with mass transit, participants are transferring from one transportation mode to another. When considering only rideshares as transportation modes, Masoud and Jayakrishnan [2017] consider a problem setting where participants can transfer from one rideshare to another. Finally, a system containing dedicated drivers is studied by Lee and Savelsbergh [2015]. The

purpose of this study is to understand the value of employing a small number of dedicated drivers, such that these can service riders that would remain unmatched if no dedicated drivers were available.

2.3. Review on carbon capture and storage

In CCS, CO_2 is captured at emitter sites and transported to geological reservoirs (inshore or offshore) where it is injected for long-term storage. Emitter sites can be industrial facilities in different sectors, such as steel, cement, pulp and paper, bioenergy, fertilizer and chemicals, combustion-based power, oil and gas, and hydrogen generation. The transportation of CO_2 can be done through different modes, such as pipelines, ships, trucks, and trains.

Around the world, studies indicate that net-zero emissions by 2050 will not be achievable without using a combination of GHG mitigation strategies, including CCS [IEA, 2017, Lane et al., 2021, Riahi et al., 2017]. The cement industry is a good example for the need of implementing CCS: in this industry, about two-thirds of the CO_2 emissions are a byproduct of a chemical reaction, and not from the burning of fossil fuels [Strunge et al., 2022]. Not using CCS in the cement industry would mean that the majority of CO_2 emissions in this sector would not be addressed, unless cement is replaced by other materials, which is unlikely to be feasible within the timeframe of the Paris Accords.

2.3.1. Surveys on models for CCS planning problems

Huang et al. [2013] review network design models for the transportation of CO_2 and energy expansion planning with a focus on CCS. The authors provide a classification of models according to objective function type (single-objective and multi-objective), objective function terms, types of constraints, model type, and solution methodology. Tapia et al. [2018] conduct a review of optimization and decision-making models for the planning of CCS systems, as well as systems that combine CCS with CO_2 utilization. The authors identify that most planning techniques in the literature are either mathematical models, pinch-based methods, or miscellaneous approaches such as numerical simulations and metaheuristics. In Zhang et al. [2022], the authors survey 16 source-sink matching models for CCS and classify them according to six key attributes: mitigation targets, carbon sources, carbon sinks, transportation networks, utilization, and integration and synergy of models (e.g., modeling uncertainty, policies, risk assessment, and integration within a negative-carbon technology system).

2.3.2. Characteristics of a CCS network

This section provides some details on the emitters, geological reservoirs and modes of transportation that may be part of a CCS network.

2.3.2.1. Emitters. Emitters may be industrial sites in various sectors, such as steel, cement, pulp and paper, bioenergy, fertilizer and chemicals, combustion-based power, oil and gas, as well as fossil fuel based hydrogen generation. Different technologies can be used to capture CO_2 from emitter sites. These technologies fall in three broad groups: post-combustion, pre-combustion, and oxyfuel combustion [read more in Chapter 6 and Bui et al., 2018].

2.3.2.2. Geological reservoirs. CO_2 can be injected in underground geological formations for long-term storage. Some geological reservoirs that are fit for CO_2 storage are, for example, saline aquifers and depleted oil reservoirs [Cauchois et al., 2021]. In North America, it is estimated that saline aquifers correspond to 95% of the available CO_2 storage capacity [Middleton et al., 2020]. To successfully deploy CCS, hundreds or thousands of potential storage sites have to be identified and screened, and particularities and uncertainties of each site may impact CO_2 injection capacity and costs [Keating et al., 2011, Middleton et al., 2020]. For example, some attributes that impact CO_2 injection rates and associated costs are the reservoir depth, thickness, permeability, porosity, and temperature [Middleton et al., 2020]. As geological reservoir storage is uncertain, given the financial and time scale of ramping-up CCS, this uncertainty provides considerable risks to the successful deployment of CCS [Lane et al., 2021]. Compared to a scenario where reservoir capacity is widely available for CO_2 storage, a scenario where storage capacity is limited would require relying on GHG mitigation technologies that are at an earlier stage of development, and that are more expensive than CCS [IEA, 2019].

2.3.2.3. Transportation modes. CO_2 can be transported from emitter sites to geological reservoirs by a mix of pipelines, ships, trucks, and trains [IPCC, 2005, Han and Lee, 2011]. Each transportation mode poses a different tradeoff, and may require the transportation of CO_2 in a specific phase (e.g., gaseous, supercritical). For example, pipelines can transport pressurized CO_2 in supercritical phase, which allows for the transportation of large volumes of CO_2 (which may be a requirement to achieve net-zero emissions). The CO_2 pressure has to be maintained throughout the pipeline network, which requires the installation of recompression stations [Svensson et al., 2005]. Alternatively, repurposed oil and gas pipelines can be used to transport CO_2, usually in gaseous phase. Reusing oil and gas infrastructure can reduce the deployment costs of CCS [Cauchois et al., 2021]. However, gaseous phase CO_2 is at a lower density than supercritical phase CO_2, which means that oil and gas pipelines can be less attractive when transporting large volumes of CO_2 is a priority. CO_2 transportation by ship and truck is more flexible than pipeline transportation: ships and trucks can be deployed faster than pipelines and they can adapt more quickly to routing changes and changes in capacity requirements [Svensson et al., 2005]. For CO_2 transportation by ships, purpose-built CO_2 tankers can be used. Alternatively, other types of ships can be repurposed to allow

for the transportation of CO_2 [Orchard et al., 2021]. Transportation of CO_2 by train can take advantage of existing railway networks, but cannot adapt to routing changes as easily as ships and trucks. Despite the advantages of transporting CO_2 by ships, trucks or trains, these transportation modes are less efficient at transporting large volumes of CO_2, in contrast to supercritical phase pipelines.

2.3.3. Technologies related to CCS

We now list some technologies related to CCS, namely, direct air carbon capture and storage, bioenergy with CCS, and CCS with CO_2 utilization. In direct air carbon capture and storage, CO_2 is captured from air and stored in geological reservoirs. This technology has, however, considerably higher costs when compared to other GHG mitigation strategies: the concentration of CO_2 in the air is about 300 times more dilute than the concentration of CO_2 in flue gas streams found in industrial emitter sites, which considerably increases the costs of separating CO_2 [Ranjan and Herzog, 2011]. In bioenergy with CCS [BECCS, Fridahl and Lehtveer, 2018], bioenergy is extracted from biomass (e.g., trees and crops). The CO_2 absorbed when growing these trees and crops is then captured when the biomass is used (e.g., combustion, fermentation). Thus, BECCS has the potential to achieve net-negative CO_2 emissions. Also, some products can be made using the CO_2 obtained with CCS [Aresta and Dibenedetto, 2010]. For example, in enhanced oil recovery, CO_2 is injected in depleted oil fields to extract oil that would not be possible to be extracted otherwise, and has the potential to reduce the carbon footprint of oil production [Middleton, 2013].

2.3.4. Optimization models for a CCS value chain

Deploying a CCS infrastructure during the next decades would require investments in the order of billions of dollars, and CCS stakeholders can benefit from having access to tools that aid them in planning for the deployment of CCS and assessing costs. To this end, several strategic planning problems have been proposed to optimize the value chain of CCS. We refer to this as CCS value chain optimization (VCO). These problems are often modeled as combined facility location and network design problems [e.g., Middleton and Bielicki, 2009, Jones et al., 2022], where facilities (sources and sinks) correspond to emitters and geological reservoirs, and the network corresponds to the CO_2 transportation infrastructure between emitters and geological reservoirs.

Models for the CCS VCO have different levels of detail when representing CCS infrastructure and costs. For example, networks that connect sources and sinks may be explicitly represented in the model, or this network may be simplified to yield a simpler model. For example, in Diamante et al. [2013], the authors propose a bipartite matching problem where

sources and sinks are directly connected. In contrast, models such as SimCCS [a widely-recognized single-period VCO model for pipeline-based CCS, see Middleton and Bielicki, 2009] explicitly consider complex pipeline networks connecting sources and sinks, and therefore provide a more accurate cost estimation of CCS costs. Other models [e.g. Han and Lee, 2011, d'Amore and Bezzo, 2017] consider more than one transportation mode (e.g., pipelines, trains, trucks, and ships).

Models for CCS VCO can also be divided according to the temporal aspect of decisions. The deployment of CCS infrastructure will likely span several decades. The underlying planning problem therefore has multiple time periods. Single-period CCS models such as SimCCS [Middleton and Bielicki, 2009] simplify the temporal aspect of decisions, and may therefore overestimate CCS costs by more than 50% [see Middleton et al., 2012b]. A benefit of using single-period models is that solving the underlying optimization problem is computationally more tractable. To better represent the real costs associated with a CCS infrastructure, SimCCS was extended to a multiperiod context [Middleton et al., 2012b, Jones et al., 2022], referred to as SimCCS-Time.

To improve the tractability of SimCCS, Lobo [2017] proposes valid inequalities for a single-period CCS problem. In Whitman et al. [2021], the authors propose three heuristics for a single-period CCS model. Namely, a constructive heuristic, a slope scaling heuristic, and a hybrid combination of the first two heuristics. In Middleton [2013], the author proposes a piecewise linear approximation for pipeline costs, which renders a model that is, on average, two to three orders of magnitude faster to solve than traditional models with discrete pipeline capacities. Given the importance of solving real-world sized instances of the multiperiod planning problem, Chapter 6 builds on the proposed slope scaling heuristic to develop an efficient heuristic capable of finding high-quality solutions within a matter of minutes.

Chapter 3

Dynamic and Stochastic Rematching for Ridesharing Systems: Formulations and Reductions

This chapter introduces a one-to-one dynamic and stochastic matching problem for ridesharing. A novelty characteristic of this problem is the possibility of unmatching previously-matched riders and drivers. This allows the system to rearrange rideshares in the case where more profitable rideshares become available. To promote user-friendliness, unmatching is associated with a penalty (e.g., compensation to unmatched users), leading to a trade-off between not unmatching and unmatching to obtain a more profitable rideshare while paying a penalty. Both driver and rider release are uncertain. We therefore propose a stochastic optimization model. As the stochastic model has to represent different realizations of driver and rider release, its number of variables and constraints may pose computational challenges. To address this issue, this chapter introduces model reduction techniques based on platform assumptions commonly found in ridesharing systems. The main contributions of this chapter are theoretical, and Chapter 4 extends the contributions of this chapter by empirically evaluating on a rolling horizon framework different models for the here proposed problem. The contributions of the student are:

- Introduction of a novel ridesharing problem;
- Deterministic and two-stage stochastic mathematical programming models for that problem;
- Reduction techniques to reduce the number of variables and constraints on these models.

The contents of this chapter were published in the *Lecture Notes in Computer Science*. This publication can be cited as follows: Homsi, G., Gendron, B., Jena, S.D. (2020). Dynamic and Stochastic Rematching for Ridesharing Systems: Formulations and Reductions. In: Baïou, M., Gendron, B., Günlük, O., Mahjoub, A.R. (eds) Combinatorial Optimization.

ISCO 2020. Lecture Notes in Computer Science, vol 12176. Springer, Cham. https://doi.org/10.1007/978-3-030-53262-8_16.

Dynamic and Stochastic Rematching for Ridesharing Systems: Formulations and Reductions

Gabriel Homsi [1], Bernard Gendron [1], Sanjay Dominik Jena [2]

[1] Department of Computer Science and Operations Research, Université de Montréal and CIRRELT, Canada

[2] Department of Management and Technology, École des Sciences de la Gestion, Université du Québec à Montréal and CIRRELT, Canada

Abstract

We introduce a dynamic and stochastic rematching problem with applications in request matching for ridesharing systems. We propose three mathematical programming formulations that can be used in a rolling horizon framework to solve this problem. We show how these models can be simplified provided that specific conditions that are typically found in practice are met.

Keywords: ridesharing, request matching, stochastic programming.

3.1. Introduction

Ridesharing systems are matching agencies for drivers and riders that are interested in sharing commute expenses. Over time, these systems receive requests from their customers corresponding to the intent of engaging in ridesharing as a driver or as a rider for a certain itinerary. In this context, an itinerary is composed of an origin, a destination, the desired departure time, and the desired arrival time. A common goal of ridesharing systems is to create matches that generate profit and that promote customer engagement. We use the term *match* to refer to the pairing of two requests, meaning that the customers behind these requests are assigned to travel together to fulfill their corresponding itineraries. A driver request may be matched to a rider request if their itineraries are compatible and if the corresponding ridesharing trip generates value for the participants. The value of a ridesharing trip is often assumed to be the amount of travel distance savings generated by the trip when compared to the individual trips for each participant [Agatz et al., 2011, Wang et al., 2018].

In this work, we study a ridesharing system that matches requests that arrive dynamically and may unmatch requests whose corresponding rides have not yet started. We assume to have access to forecasts on the probability that future requests exist. To avoid compromising customer engagement, we allow for defining a penalty on unmatch operations, which may

correspond to a discount on future trips offered to the unmatched customers. These penalties may depend, for example, on the amount of time since the requests have been created. Unmatching a request that was released much earlier would therefore lead to a large penalty. Nevertheless, unmatching requests may be desirable if a new request becomes available such that it allows for a highly profitable match with a currently-matched request.

Research on optimization for ridesharing started gaining traction with the work of Agatz et al. [2011], where the authors studied a dynamic driver-rider matching problem. Several further studies explored specific attributes of ridesharing systems. To name a few, Stiglic et al. [2015] studied the impact of meeting points in ridesharing systems, Stiglic et al. [2016] studied the impact of participant time flexibility in ridesharing, Stiglic et al. [2018] studied the integration of a ridesharing system with public transit, and Wang et al. [2018] investigated the impact of matching stability on a dynamic ride-matching system. For surveys on ridesharing and related shared mobility systems, we refer the reader to Agatz et al. [2012], Furuhata et al. [2013] and Mourad et al. [2019]. Our work extends the ridesharing request matching literature by addressing the stochasticity in requests, the unmatching of requests, and the time-dependency of matching profits and unmatching penalties. In summary, our contributions are threefold: 1) we introduce a new dynamic and stochastic rematching problem 2) we propose three mathematical programming formulations to solve this problem, and 3) we show how these formulations can be simplified under specific but realistic conditions.

3.2. Problem Definition

Requests in a ridesharing system arrive continuously over a planning horizon $T = \{1, 2, \ldots, h\}$. Let $G = (V, E)$ be a bipartite graph where V is the set of requests and E is the set of edges between compatible requests. The set of requests is partitioned into a set of driver requests D and a set of rider requests R, such that $V = D \cup R$, $D \cap R = \emptyset$, and $E \subseteq D \times R$. A pair of requests $(ij) \in D \times R$ does not belong to E if the itineraries of i and j are incompatible or if the corresponding ridesharing trip does not generate value for its participants.

At each time period $t \in T$, a pair of requests $(ij) \in E$ can be matched for a profit of c_{ij}^t and unmatched for a cost of d_{ij}^t. We assume that $c_{ij}^t \leq d_{ij}^t$, i.e., it is never profitable to unmatch a pair of requests and then match it in the same time period. A pair of requests (ij) is said to be active at the beginning of the time period t if it was not unmatched since the last period it has been matched. For each request $i \in V$, let r_i be its release time and b_i be its latest possible match time. The latest possible match time may correspond to the desired departure time or to the latest time period that a customer is willing to wait for a match. The pair of requests $(ij) \in E$ can only be matched or unmatched at time period $t \in T$ if both requests have already been released and if they are still available for matching. Hence,

matching and unmatching are only possible if $t \in W_{ij}$, where

$$W_{ij} = \{\, t \in T \mid \max(r_i, r_j) \leq t \leq \min(b_i, b_j) \,\}.$$

If the condition above is not met, we interdict matches and unmatches by assigning $c_{ij}^t = -M$ and $d_{ij}^t = M$, where M is a sufficiently big constant such that an optimal solution never has the pair of requests (ij) matched or unmatched outside W_{ij}.

The objective of the ridesharing system is to dynamically match and unmatch driver and rider requests such that the net profit over the planning horizon is maximized. In the following, we present three mathematical programming formulations that can be used in a rolling horizon framework to provide matches and rematches at specific time periods throughout the planning horizon. We first present a myopic formulation that can be used when no forecasts on future demand are available. Then, we present a static formulation that can be used when the information for all time periods is known in advance, or when a sufficiently accurate forecast is available. Finally, we present a stochastic formulation that can be used when sufficient historical information is known to accurately generate multiple scenarios that are representative of future demand.

3.3. The Myopic Problem

When no forecasts on future demand are available, a myopic optimization problem can be defined. For each pair $(ij) \in E$, let x_{ij}^t be a binary variable equal to 1 if and only if the pair (ij) is matched at time period t, y_{ij}^t be a binary variable equal to 1 if and only if the pair (ij) is unmatched at time period t, and a_{ij}^t be a binary constant equal to 1 if and only if (ij) is active at the beginning of t. The myopic problem of matching and unmatching requests such that the net profit at time period $t \in T$ is maximized can be formulated as below

$$f_{\text{MYO}}(t, a^t) \coloneqq \max \sum_{(ij) \in E} (c_{ij}^t x_{ij}^t - d_{ij}^t y_{ij}^t) \tag{3.3.1}$$

$$\text{s.t.} \sum_{(ij) \in \delta(v)} (x_{ij}^t - y_{ij}^t) \leq 1 - \sum_{(ij) \in \delta(v)} a_{ij}^t \qquad \forall v \in V \tag{3.3.2}$$

$$y_{ij}^t \leq a_{ij}^t \qquad \forall (ij) \in E \tag{3.3.3}$$

$$x_{ij}^t, y_{ij}^t \in \{\, 0, 1 \,\} \qquad \forall (ij) \in E. \tag{3.3.4}$$

The objective function (3.3.1) maximizes the net profit of matching and unmatching requests at time period t. Constraints (3.3.2) ensure that requests can only be matched if they are inactive. Constraints (3.3.3) ensure that requests can only be unmatched if they are active.

The formulation above can be rewritten as a maximum-weight bipartite matching problem. Let $E_0 = \{\, (ij) \in E \mid a_{ij}^t = 0 \,\}$ and $E_1 = E \setminus E_0$. For each pair $(ij) \in E_0$, let x_{ij}^t be a binary variable equal to 1 if and only if the inactive pair (ij) is matched at time period t. For each pair $(ij) \in E_1$, let z_{ij}^t be a binary variable equal to 1 if and only if the active pair (ij) is not

unmatched at time period t. The bipartite matching reformulation is defined below

$$f'_{\text{MYO}}(t) \coloneqq \max \sum_{(ij)\in E_0} c_{ij}^t x_{ij}^t + \sum_{(ij)\in E_1} d_{ij}^t (z_{ij}^t - 1) \tag{3.3.5}$$

$$\text{s.t.} \sum_{(ij)\in\delta(v)\cap E_0} x_{ij}^t + \sum_{(ij)\in\delta(v)\cap E_1} z_{ij}^t \leq 1 \qquad \forall v \in V. \tag{3.3.6}$$

The objective function (3.3.5) maximizes the net profit of matching and unmatching requests at time period t. If $z_{ij}^t = 0$, then (ij) is unmatched, which yields a penalty of $-d_{ij}^t$ in the objective function. Constraints (3.3.6) ensure that requests can be matched at most once, either in E_0 or in E_1. Despite having two sets of variables, the formulation above is equivalent to a classic bipartite matching formulation. Nevertheless, we have decided to use distinct variable names to highlight the differences between matches in E_0 and matches in E_1.

3.4. The Static Problem

When the requests released throughout all time periods are known in advance, a multi-period static problem that provides matches and unmatches for the whole planning horizon can be formulated. This model can be used as a benchmark for other models evaluated in the rolling horizon framework, and is defined below

$$\max \sum_{t\in T} \sum_{(ij)\in E} (c_{ij}^t x_{ij}^t - d_{ij}^t y_{ij}^t) \tag{3.4.1}$$

$$\text{s.t.} \sum_{\ell=1}^{t} \sum_{(ij)\in\delta(v)} (x_{ij}^\ell - y_{ij}^\ell) \leq 1 \qquad \forall t \in T, v \in V \tag{3.4.2}$$

$$y_{ij}^t \leq \sum_{\ell=1}^{t-1} (x_{ij}^\ell - y_{ij}^\ell) \qquad \forall t \in T, (ij) \in E \tag{3.4.3}$$

$$x_{ij}^t, y_{ij}^t \in \{0, 1\} \qquad \forall t \in T, (ij) \in E. \tag{3.4.4}$$

The objective function (3.4.1) maximizes the net profit over the full planning horizon. Constraints (3.4.2) ensure that requests can only be matched if they are inactive. Constraints (3.4.3) ensure that requests can only be unmatched if they are active.

Assumption 1. The profit of matching a pair of requests is never bigger than the cost of unmatching it, i.e.,

$$c_{ij}^t \leq d_{ij}^k \qquad \forall t \in T, k \in T, t \leq k.$$

Proposition 1. If Assumption 1 holds for all pairs of requests, then it is never necessary to unmatch in an optimal solution for (3.4.1)–(3.4.4).

Proof 1. Let $(\bar{x}, \bar{y})$ be an optimal solution for (3.4.1)–(3.4.4) and $z(\bar{x}, \bar{y})$ its objective function value. Assume that there exists a pair $(ij) \in E$ and time periods $t \in T$ and $k \in T$ with $t \leq k$ such that $\bar{x}_{ij}^t = 1$ and $\bar{y}_{ij}^k = 1$. As $c_{ij}^t - d_{ij}^k \leq 0$, there exists a solution $(\hat{x}, \hat{y})$ similar to $(\bar{x}, \bar{y})$,

except for $\hat{x}_{ij}^t = 0$ and $\hat{y}_{ij}^k = 0$. Consequently, $z(\hat{x}, \hat{y}) \geq z(\bar{x}, \bar{y})$, which either contradicts the optimality of $(\bar{x}, \bar{y})$, or shows that both solutions have the same objective function value. □

If Assumption 1 holds, by Proposition 1, it follows that the static problem can be reduced by removing all unmatching variables, which gives the formulation below

$$\max \sum_{t \in T} \sum_{(ij) \in E} c_{ij}^t x_{ij}^t \tag{3.4.5}$$

$$\text{s.t.} \sum_{t \in T} \sum_{(ij) \in \delta(v)} x_{ij}^t \leq 1 \qquad \forall v \in V \tag{3.4.6}$$

$$x_{ij}^t \in \{ 0, 1 \} \qquad \forall t \in T, (ij) \in E. \tag{3.4.7}$$

As each pair $(ij) \in E$ can be matched at most once, it is more profitable to match (ij) at the time period that maximizes $c_{ij}^t, \forall t \in T$. Thus, the multiple time periods can be represented implicitly and the formulation above can be reduced to a maximum-weight bipartite matching problem, as below

$$\max \sum_{(ij) \in E} \hat{c}_{ij} x_{ij} \tag{3.4.8}$$

$$\text{s.t.} \sum_{(ij) \in \delta(v)} x_{ij} \leq 1 \qquad \forall v \in V, \tag{3.4.9}$$

where $\hat{c}_{ij} = \max \{ c_{ij}^t \mid t \in T \}$. Let $\bar{x}$ be an optimal solution for Eqs. (3.4.8)–(3.4.9). If $\bar{x}_{ij} = 1$, then (ij) is matched in time period $t \in \arg\max_{t \in T} c_{ij}^t$.

The static formulation can be adapted to be used in a rolling horizon framework if a forecast such as the expected future demand is available. The static model that matches and unmatches requests for a time period $k \in T$ while taking into consideration a forecast for periods $(k+1), \ldots, h$ is defined below

$$f_{\text{STAT}}(k, a^k) \coloneqq \max \sum_{t=k}^{h} \sum_{(ij) \in E} (c_{ij}^t x_{ij}^t - d_{ij}^t y_{ij}^t) \tag{3.4.10}$$

$$\text{s.t.} \sum_{\ell=k}^{t} \sum_{(ij) \in \delta(v)} (x_{ij}^\ell - y_{ij}^\ell) \leq 1 - \sum_{(ij) \in \delta(v)} a_{ij}^k \qquad \forall t = k, \ldots, h, v \in V \tag{3.4.11}$$

$$y_{ij}^t \leq a_{ij}^k + \sum_{\ell=1}^{t-1} (x_{ij}^\ell - y_{ij}^\ell) \qquad \forall t = k, \ldots, h, (ij) \in E \tag{3.4.12}$$

$$x_{ij}^t, y_{ij}^t \in \{ 0, 1 \} \qquad \forall t = k, \ldots, h, (ij) \in E. \tag{3.4.13}$$

The formulation above is similar to the single-scenario case of the two-stage stochastic programming formulation defined next.

3.5. The Stochastic Problem

To address the uncertainty on future demand, we introduce a two-stage stochastic programming formulation. In the first stage, matching and unmatching decisions are given for requests available at the current time period $t \in T$. In the second stage, these decisions are made for the remainder of the planning horizon, given the first-stage decisions and a sample realization of future requests.

Let S be the set of second-stage scenarios and p_s the probability associated with each scenario $s \in S$. We assume that each scenario $s \in S$ contains a full realization of the planning horizon since the time period $t+1$. For each request $i \in V$ and scenario $s \in S$, let ξ_i^s be a random variable defined below

$$\xi_i^s = \begin{cases} 1 \text{ if request } i \text{ is available for matching in scenario } s, \\ 0 \text{ otherwise.} \end{cases}$$

For each pair $(ij) \in E$ and time period $k = (t+1), \ldots, h$, if $\xi_i^s = 0$ or $\xi_j^s = 0$, then $c_{ij}^{ks} = -M$ and $d_{ij}^{ks} = M$. Otherwise, $c_{ij}^{ks} = c_{ij}^k$ and $d_{ij}^{ks} = d_{ij}^k$. Moreover, if a request $i \in V$ is released before the second stage, i.e., $r_i \leq t$, then we assume that $\xi_i^s = 1, \forall s \in S$. The two-stage stochastic programming formulation is defined as below

$$f_{\text{STO}}(t) \coloneqq \max \sum_{(ij)\in E_0} c_{ij}^t x_{ij}^t + \sum_{(ij)\in E_1} d_{ij}^t (z_{ij}^t - 1) + \sum_{s\in S} p_s Q(t+1, s, a^{t+1}) \tag{3.5.1}$$

$$\text{s.t.} \sum_{(ij)\in E_0 \cap \delta(v)} x_{ij}^t + \sum_{(ij)\in E_1 \cap \delta(v)} z_{ij}^t \leq 1 \qquad \forall v \in V \tag{3.5.2}$$

$$a_{ij}^{t+1} = \begin{cases} x_{ij}^t \text{ if } (ij) \in E_0, \\ z_{ij}^t \text{ otherwise.} \end{cases} \qquad \forall (ij) \in E \tag{3.5.3}$$

$$x_{ij}^t \in \{0, 1\} \qquad \forall (ij) \in E_0 \tag{3.5.4}$$

$$z_{ij}^t \in \{0, 1\} \qquad \forall (ij) \in E_1. \tag{3.5.5}$$

The objective function (3.5.1) maximizes the net profit at time period t plus the expect net profit for time periods $(t+1), \ldots, h$. Constraints (3.5.2) match and unmatch requests for period t. Equations (3.5.3) define the values of $a_{ij}^{t+1}, \forall (ij) \in E$. The second-stage problem is a multiperiod problem over the time periods $(t+1), \ldots, h$, and is defined for each scenario

$s \in S$ as below

$$Q(k, s, a^k) \coloneqq \max \sum_{t=k}^{h} \sum_{(ij)\in E} (c_{ij}^{ts} x_{ij}^{ts} - d_{ij}^{ts} y_{ij}^{ts}) \tag{3.5.6}$$

$$\text{s.t.} \sum_{\ell=k}^{t} \sum_{(ij)\in\delta(v)} (x_{ij}^{\ell s} - y_{ij}^{\ell s}) \leq 1 - \sum_{(ij)\in\delta(v)} a_{ij}^{k} \qquad \forall t = k, \ldots, h, v \in V \tag{3.5.7}$$

$$y_{ij}^{ts} \leq a_{ij}^{k} + \sum_{\ell=k}^{t-1} (x_{ij}^{\ell s} - y_{ij}^{\ell s}) \qquad \forall t = k, \ldots, h, (ij) \in E \tag{3.5.8}$$

$$x_{ij}^{ts}, y_{ij}^{ts} \in \{0, 1\} \qquad \forall t = k, \ldots, h, (ij) \in E. \tag{3.5.9}$$

We now show how the two-stage formulation can be reduced if some realistic conditions are met.

3.5.1. Reductions based on the system environment

Motivated by the fact that unmatching close to the departure time is typically not user-friendly, we study how to reduce the stochastic model if the penalty of unmatching does not become less expensive over time.

Assumption 2. For each $(ij) \in E$, the unmatching costs are non-decreasing in W_{ij}, i.e.,

$$d_{ij}^{t} \leq d_{ij}^{k} \qquad \forall t \in W_{ij}, k \in W_{ij}, t < k$$

Proposition 2. If Assumption 2 holds, then for each $(ij) \in E$, d_{ij}^{t} is non-decreasing in $\{ t \in T \mid \min(W_{ij}) \leq t \leq h \}$.

Proof 2. For each $t \in T$, if $t \in W_{ij}$, then $d_{ij}^{t} \leq M$. Otherwise, if $\max(W_{ij}) < t \leq h$, then $d_{ij}^{t} = M$. $\square$

Proposition 3. If Assumptions 1 and 2 hold, then the two-stage problem can be reduced such that the second stage has a single period.

Proof 3. Let $(\bar{x}, \bar{a}, \bar{y})$ be an optimal solution for the two-stage problem. For each $(ij) \in E$ where $\bar{a}_{ij}^{k} = 1$, the corresponding time period t for when (ij) was last matched must be in W_{ij}. Moreover, by Proposition 2, it follows that the second-stage penalties for unmatching (ij) are non-decreasing. Thus, k is the best period to unmatch (ij) in the second stage. Together with Proposition 1, it follows that an optimal solution will never unmatch the same pair more than once in the second stage. Consequently, the second-stage unmatching variables $y_{ij}^{ts}, \forall t = (k+1), \ldots, h, (ij) \in E$ can be set to 0. As a result, the time periods for the second-stage matching variables can be represented implicitly, which gives us the

single-period second-stage problem below

$$Q'(k,s,a^k) \coloneqq \max \sum_{(ij)\in E} (\hat{c}^s_{ij} x^s_{ij} - d^{ks}_{ij} y^{ks}_{ij}) \tag{3.5.10}$$

$$\text{s.t.} \sum_{(ij)\in\delta(v)} (x^s_{ij} - y^{ks}_{ij}) \le 1 - \sum_{(ij)\in\delta(v)} a^k_{ij} \qquad \forall v \in V \tag{3.5.11}$$

$$y^{ks}_{ij} \le a^k_{ij} \qquad \forall (ij) \in E \tag{3.5.12}$$

$$x^s_{ij}, y^{ks}_{ij} \in \{0,1\} \qquad \forall (ij) \in E, \tag{3.5.13}$$

where $\hat{c}^s_e = \max\{ c^{ts}_{ij} \mid t = k,\dots,h \}$. Together, Eqs. (3.5.1)–(3.5.5) and (3.5.10)–(3.5.13) form a reduced two-stage model. □

3.5.2. Reductions based on the first-stage solution structure

In certain situations, it is important to efficiently solve the second-stage problem given a first-stage solution. For example, when the two-stage problem is solved independently within a mathematical decomposition method. We provide two reductions for the second-stage problem, based on the first-stage solution.

3.5.2.1. No matches before the second stage. If $a^k_{ij} = 0, \forall (ij) \in E$, then the second-stage problem is independent of the first stage, and is defined as follows

$$Q''(k,s,\mathbf{0}) \coloneqq \max \sum_{t=k}^{h} \sum_{(ij)\in E} (c^{ts}_{ij} x^{ts}_{ij} - d^{ts}_{ij} y^{ts}_{ij}) \tag{3.5.14}$$

$$\text{s.t.} \sum_{\ell=k}^{t} \sum_{(ij)\in\delta(v)} (x^{\ell s}_{ij} - y^{\ell s}_{ij}) \le 1 \qquad \forall t = k,\dots,h, v \in V \tag{3.5.15}$$

$$y^{ts}_{ij} \le \sum_{\ell=k}^{t-1} (x^{\ell s}_{ij} - y^{\ell s}_{ij}) \qquad \forall t = k,\dots,h, (ij) \in E \tag{3.5.16}$$

$$x^{ts}_{ij}, y^{ts}_{ij} \in \{0,1\} \qquad \forall t = k,\dots,h, (ij) \in E, \tag{3.5.17}$$

which has the same structure as the static problem defined in Eqs. (3.4.1)–(3.4.4). It follows that if Assumption 1 holds, then the formulation above can be rewritten as a maximum-weight bipartite matching problem.

3.5.2.2. Matches before the second stage. If Assumption 1 holds, then the second-stage unmatching of pairs in $\{ (ij) \in E \mid a^k_{ij} = 0 \}$ is never profitable, even if $\{ (ij) \in E \mid a^k_{ij} = 1 \} \neq \emptyset$. Thus, the variables $y^{ts}_{ij}, \forall t = k,\dots,h, (ij) \in E, a^k_{ij} = 0$ can be set to 0.

Although the second-stage problem can be reduced in such cases, these reductions do not apply to the full two-stage problem defined in Eqs. (3.5.1)–(3.5.9).

3.6. Conclusions and Future Work

We have introduced a matching and rematching problem with applications in request matching for ridesharing systems. We have presented three mathematical formulations that can be used in a rolling horizon framework. We discussed how to reduce these formulations provided that specific conditions that are typically found in practice are met. In some cases, these formulations can be reduced to a simple maximum-weight bipartite matching formulation. Some opportunities for future work are the evaluation of the proposed models on a rolling horizon framework and the development of efficient decomposition methods that exploit the proposed model reduction techniques.

Chapter 4

Rolling Horizon Strategies for a Dynamic and Stochastic Ridesharing Problem with Rematches

In this chapter, myopic and stochastic models for the problem presented in Chapter 3 are empirically evaluated within a rolling horizon framework. The studied problem is a one-to-one dynamic driver and rider matching problem with stochastic drivers and riders. Additionally, this problem allows for the rematching of previously-matched requests, such that the ridesharing system can rearrange decisions if it is advantageous. As the contributions of Chapter 3 are of theoretical nature, this chapter extends these contributions through the analysis of computational results. The computational results allow the assessment of model performance, the value of rematching under different penalty levels, and the impact of other system parameters. The contributions of the student are:

- An expected value approximation for a two-stage stochastic matching and rematching problem for ridesharing;
- Empirical evaluation of this ridesharing problem;
- Managerial insights on the value of rematching and other system attributes;
- Insights on the performance of stochastic model approximations according to different system attributes.

The contents of this chapter were submitted to Discrete Applied Mathematics, and this submission is currently under review. A technical report is available in https://www.cirrelt.ca/documentstravail/cirrelt-2021-20.pdf.

Rolling Horizon Strategies for a Dynamic and Stochastic Ridesharing Problem with Rematches

Gabriel Homsi [1], Bernard Gendron [1], Sanjay Dominik Jena [2]

[1] Department of Computer Science and Operations Research, Université de Montréal and CIRRELT, Canada

[2] Department of Management and Technology, École des Sciences de la Gestion, Université du Québec à Montréal and CIRRELT, Canada

Abstract

We study a dynamic matching and rematching problem with applications in ridesharing systems. Transportation requests for riders and passengers arrive dynamically and are represented as nodes of a bipartite graph, connected by edges that correspond to compatible requests. Matching requests by selecting the corresponding edge earns a profit. We further allow for unmatching previously matched requests. While this increases the system's flexibility to adjust to new matching opportunities, unmatching may degrade customer experience and therefore implies penalty costs. We evaluate myopic and stochastic multi-period mixed-integer programming models in a rolling horizon framework. All models are compared against a static model that has perfect knowledge about future requests, using a novel and extensive benchmark set of realistic instances. Our results demonstrate the value of being able to unmatch, as well as the benefits of the stochastic strategies over the myopic strategy.

Keywords: ridesharing; bipartite matching; stochastic optimization; rolling horizon; rematching.

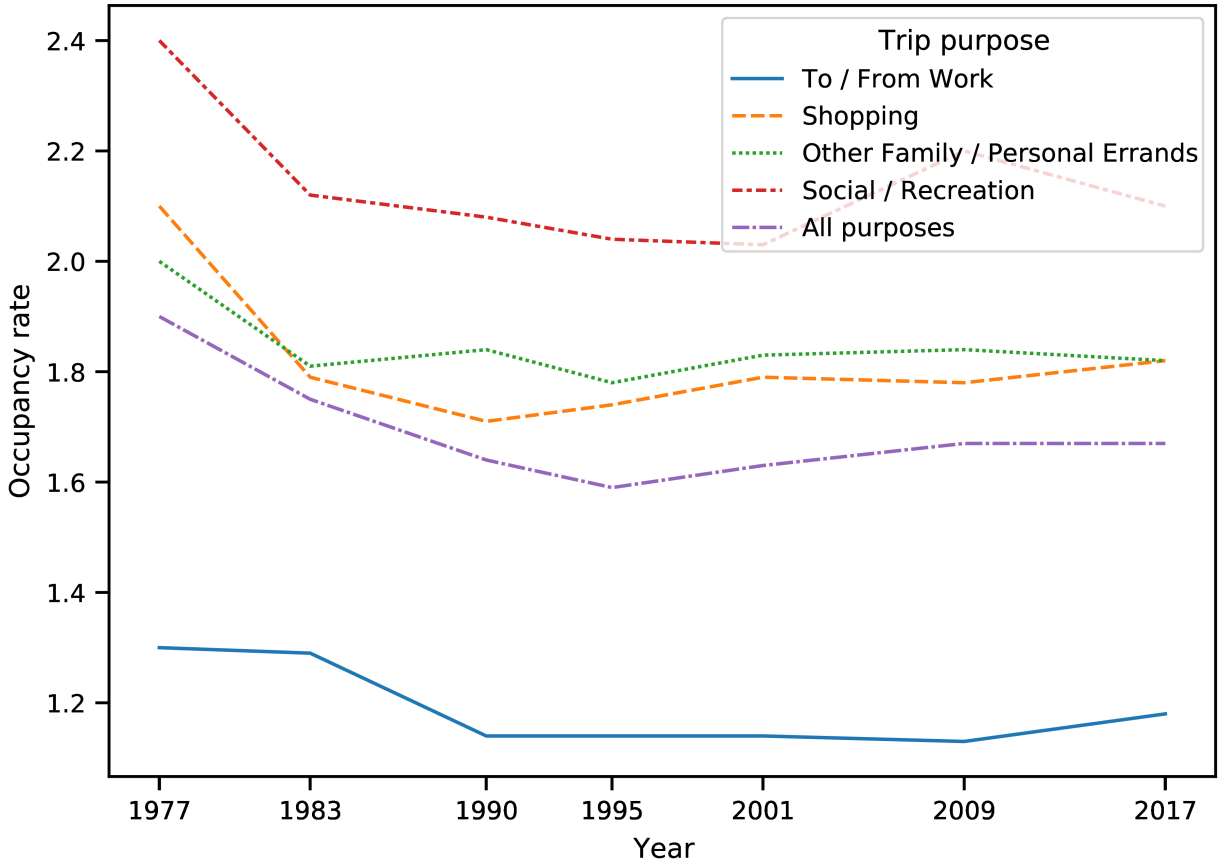
4.1. Introduction

Traffic congestion and air pollution are common problems in metropolitan areas. In addition to the environmental impact, these issues have an immediate impact on urban quality of life. To mitigate these issues, cities may encourage individuals to engage in ridesharing.

The act of ridesharing consists of traveling together to share trip expenses, which contributes to the reduction of vehicle emissions. The practice of ridesharing is not new, and dates as back as WWII and the 1970s oil crisis [Chan and Shaheen, 2012]. Nevertheless, ridesharing still has a huge growth potential: the average vehicle occupancy rate in the United States is estimated to be about 1.67 persons per vehicle [McGuckin and Fucci, 2018].

Additionally, the occupancy rate of vehicles used for work-related trips is even smaller (see Figure 4.1). Those vehicles, operating under capacity, could potentially engage in ridesharing. Note that there is no consensus in the literature on the differences between carpooling and ridesharing [Neoh et al., 2017]. We therefore use these terms interchangeably.

Fig. 4.1. Vehicle occupancy rate by trip purpose, source: [McGuckin and Fucci, 2018].



In this work, we revisit a matching and rematching problem with applications in ridesharing systems, previously defined in Homsi et al. [2020]. Ridesharing systems are matching agencies for drivers and riders that are interested in sharing commute expenses. Over time, these systems receive requests from their customers corresponding to the intent of engaging in ridesharing for a certain itinerary, either as a driver or as a rider. In this context, an itinerary is composed of an origin, a destination, the earliest departure time, and the latest arrival time. A common goal of ridesharing systems is to create matches that generate profit and that promote customer engagement. We use the term *match* to refer to the pairing of two requests, meaning that the customers behind these requests are assigned to travel together to fulfill their corresponding itineraries. Requests may not only be matched, but may also be unmatched afterwards (for a certain penalty cost) such that they can be reassigned to better rideshares.

Note that there are no guarantees that all requests will be provided service. Matching all driver and rider requests may not only be infeasible in practice, but is also likely to be unprofitable from a system operator's perspective. As such, the here considered planning problem assumes that the ridesharing operator wants to maximize profit, which is in line with current ridesharing systems (e.g., Netlift and Amigo Express), where driver supply is often not equal to rider demand.

This problem has been considered in Homsi et al. [2020], who proposed a static formulation spanning the entire planning horizon, as well as a myopic and a two-stage stochastic model that can be integrated in a rolling horizon framework. In this paper, we extend these developments in order to evaluate the usefulness of a variety of models proposed for this problem in practice.

Contributions. The contributions of this paper can be summarized as follows. In addition to the previously introduced models, we here propose an expected value model, which optimizes over the average demand of the considered scenarios. We then evaluate the performance of all models in extensive computational experiments on novel problem instances generated based on trip data from an industrial collaborator. Specifically, we simulate a rolling horizon planning, where new requests are dynamically released into the system and the different models are used to dynamically generate matching and unmatching decisions. Finally, we also investigate different matching profit and unmatching penalty functions, providing valuable insights on how different corporate strategies may affect profitability and customer satisfaction.

4.2. Related Work

Ridesharing has been the focus of extensive research in the literature. For surveys on ridesharing and related shared mobility systems, we refer the reader to Agatz et al. [2012], Furuhata et al. [2013] and Mourad et al. [2019]. In Agatz et al. [2012] components and challenges of ride-matching optimization for dynamic ridesharing systems are discussed. Shortly after, Furuhata et al. [2013] presented a broad taxonomy for 39 ridesharing matching agencies and identified challenges in the mass adoption of ridesharing. A more general survey on shared mobility systems was recently conducted by Mourad et al. [2019], which not only includes problems where vehicles carry passengers, but also parcels next to passengers.

Ridesharing studies usually focus either on operational-level decisions or on policy studies. Studies focusing on operational planning typically formulate an optimization model to find efficient rideshares for the participants. Policy-based studies explore the impact of pricing and incentive policies on ridesharing, and focus less on individual-based decisions and more on a macro-level analysis.

4.2.1. Operational planning studies

In operational planning studies, ridesharing problems are usually formulated as vehicle routing problems (VRPs) or as bipartite matching problems.

4.2.1.1. Vehicle routing based problems. One of the first studies to model a ridesharing problem as a VRP formulation is the work of Baldacci et al. [2004]. The authors solve a carpooling problem with a common workplace destination. The authors propose exact and heuristic methods based on Lagrangean column generation. In Lee and Savelsbergh [2015], the authors study the impact of dedicated drivers on ridesharing systems. A ridesharing problem with transfers is studied in Masoud and Jayakrishnan [2017]. In Riley et al. [2019], the authors study a dynamic VRP where the objective function has a penalty term proportional to the amount of time that customers are left unserved.

4.2.1.2. Bipartite matching based problems. When a bipartite matching formulation is used, one side of the graph usually corresponds to driver requests, and the other side usually corresponds to rider requests. These studies often focus on one-to-one matches, that is, when a driver carries exactly one passenger. Agatz et al. [2011] were among the first to study a dynamic bipartite matching problem with applications to ridesharing. The authors solved this problem under a rolling horizon simulation framework. Subsequent studies focus on specific attributes present in ridesharing operations such as meeting points [Stiglic et al., 2015], participant time flexibility [Stiglic et al., 2016], integration with mass transit systems [Stiglic et al., 2018], and matching stability [Wang et al., 2018]. Recently, Homsi et al. [2020] first defined a rematching problem with applications in ridesharing. In this work, we further investigate this problem.

4.2.1.3. Online matching. Dynamic ridesharing systems have strong connections to online matching problems. Online matching studies usually focus on the competitive ratio of deterministic and randomized online matching algorithms. In Mehta [2005], the author studies the generalized online matching problem and its applications in search engine advertisement. In Chen et al. [2009], the authors study an iterative matching problem where each edge of the graph has a probability of existing. Edges with a probability of existing are useful when the compatibility between two nodes is uncertain, for example, in kidney exchange settings. In our problem, the nodes of our graph are uncertain. The existence or not of an edge between two nodes is certain, as it depends only on the profitability and on the time-window feasibility of the rideshare associated with these nodes.

4.2.2. Policy studies

Pricing strategies and incentive policies may impact the viability of ridesharing systems. In some cases, subsidies must be allocated to encourage adoption. In Brownstone and Golob

[1992], the authors study the impact of incentive policies such as reserved parking and guaranteed rides home. To evaluate the impact of these policies, the authors built a discrete choice model. The impact of high occupancy vehicle (HOV) lanes is studied in Giuliano et al. [1990]. Later, Kon [2010] study the impact of introducing tolls for solo drivers that decide to take HOV lanes, referred to as high occupancy toll lanes. To motivate rider participation in ridesharing, Masoud and Jayakrishnan [2017] propose a mechanism design where riders are matched on a first-come, first-served basis, but are also offered the opportunity to buy the itinerary of a previously-matched rider. Our study focuses less on policy and more on the daily operational planning of a ridesharing company.

4.3. Problem Definition

We now formally define the setting of our ridesharing matching problem. Requests in a ridesharing system arrive dynamically over a planning horizon $T = \{1, 2, \ldots, h\}$. Let $G = (V, E)$ be a bipartite graph where V is the set of requests that may be released and E is the set of edges between compatible requests. The set of requests is partitioned into a set D of drivers requesting a passenger and a set R of riders requesting a driver, such that $V = D \cup R$, $D \cap R = \emptyset$, and $E \subseteq D \times R$. For each request $v \in V$, let $o_v \in \mathbb{R}^2$ be its origin coordinates, $d_v \in \mathbb{R}^2$ be its destination coordinates, $r_v \in T$ be its release time, $a_v \in \mathbb{R}$ be its earliest departure time, and $b_v \in \mathbb{R}$ be its latest arrival time. Let E_t be the set of pairs of requests that can be matched or unmatched at time period $t \in T$, with $E = \bigcup_{t \in T} E_t$. At each time period $t \in T$, a pair of released requests $(ij) \in E_t$ can be matched for a profit of p_{ij}^t and unmatched for a cost of c_{ij}^t. We assume that

$$p_{ij}^t \leq c_{ij}^t,$$

i.e., it is never profitable to unmatch a pair of requests and then match it in the same time period. A pair of requests (ij) is said to be *active* at the beginning of the time period t if it was matched before t and not unmatched ever since. The objective of the problem is to match and unmatch requests such that the net profit over the planning horizon is maximized.

4.3.1. Release of requests

Whether a request $v \in V$ is released or not is uncertain, and is represented as a random binary variable $\mathbf{q_v}$ equal to 1 if and only if v is released. The value of $\mathbf{q_v}$ is observed at period r_v, i.e., at the corresponding release time of v. We write q when referring to a possible realization of $\mathbf{q}$, and we write q' when referring to the actual realization of $\mathbf{q}$ within our rolling horizon simulation framework.

4.3.2. Request compatibility

A pair of requests (ij) is in E_t if and only if

- both requests have already been released, i.e., $\max\{r_i, r_j\} \leq t$;
- the departure time of both requests is not in the past, i.e., $t \leq \min\{a_i, a_j\}$;
- the rideshare generates distance savings for its participants;
- and the rideshare is time-window feasible.

The value of a ridesharing trip is often assumed to be the amount of travel distance savings generated by the trip when compared to the individual trips for each participant [Agatz et al., 2011, Wang et al., 2018]. Let $d(o, d)$ be the distance (in km) between two points o and d. The distance savings s_{ij} (in km) generated by a rideshare $(ij) \in E_t$ is the difference between the distance of the individual trips and the distance of the rideshare trip, as below

$$s_{ij} = d(o_i, d_i) + d(o_j, d_j) - [d(o_i, o_j) + d(o_j, d_j) + d(d_j, d_i)].$$

To determine if a rideshare (ij) is time-window feasible, we check if the departure and arrival times of the participants are compatible. Let $t(p_1, p_2)$ be the travel time (in periods) between any two points p_1 and p_2. The following conditions must be met to ensure time-window feasibility:

$$\max\{a_i + t(o_i, o_j), a_j\} + t(o_j, d_j) \leq b_j$$

for the rider, and

$$\max\{a_i + t(o_i, o_j), a_j\} + t(o_j, d_j) + t(d_j, d_i) \leq b_i$$

for the driver. Next, we give a formal definition of the static problem that generates decisions for the entire planning horizon.

4.3.3. The static problem definition

When all real realizations q'_v of requests $v \in V$ are known in advance, a static problem can be defined, taking optimal decisions for the full planning horizon. The optimal objective function value of this problem gives an upper bound that can be used to evaluate the performance of strategies that generate matches and unmatches dynamically, as described in Section 4.4. Let

$$\delta_t(v) = \{\, (ij) \in E_t \mid v \in (ij) \,\}$$

be the adjacency of each node $v \in V$ for each period $t \in T$ and

$$W_{ij} = \{\, t \in T \mid (ij) \in E_t \,\}$$

be the periods where i and j can be matched. For each pair $(ij) \in E_t$, let x_{ij}^t be a binary variable equal to 1 if and only if the pair (ij) is matched at time period t, y_{ij}^t be a binary variable equal to 1 if and only if the pair (ij) is unmatched at time period t, and z_{ij}^t be a binary

variable equal to 1 if and only if (ij) is active at the beginning of period t. Without loss of generality, we here assume that all requests are initially unmatched, that is, $z_{ij}^1 = 0, \forall (ij) \in E$. The static model is defined as:

$$f_{\text{STAT}}(z^1) := \max \sum_{t=1}^{h} \sum_{(ij)\in E_t} (p_{ij}^t x_{ij}^t - c_{ij}^t y_{ij}^t) \tag{4.3.1}$$

$$\text{s.t.} \sum_{(ij)\in\delta_t(v)} (x_{ij}^t - y_{ij}^t) \leq q'_v - \sum_{(ij)\in\delta_t(v)} z_{ij}^t \qquad \forall t = 1,\dots,h, v\in V \tag{4.3.2}$$

$$y_{ij}^t \leq z_{ij}^t \qquad \forall t = 1,\dots,h, (ij)\in E_t \tag{4.3.3}$$

$$z_{ij}^{t+1} = z_{ij}^t + \begin{cases} x_{ij}^t - y_{ij}^t \text{ if } (ij) \in E_t \\ 0 \text{ otherwise} \end{cases} \qquad \forall t = 1,\dots,h, (ij)\in E \tag{4.3.4}$$

$$x_{ij}^t, y_{ij}^t \in \{0, 1\} \qquad \forall t = 1,\dots,h, (ij)\in E_t \tag{4.3.5}$$

$$z_{ij}^{t+1} \in \{0, 1\} \qquad \forall t = 1,\dots,h, (ij)\in E. \tag{4.3.6}$$

Objective (4.3.1) maximizes the net profit over the full planning horizon. Constraints (4.3.2) ensure that requests can only be matched if they are released and inactive. Constraints (4.3.3) ensure that requests can only be unmatched if they are active. Constraints (4.3.4) define the values of z for the next time period. As the static problem assumes perfect knowledge of q', it is not used in a dynamic environment, but solved only once for the entire planning horizon. Note that, as the objective of this model is to maximize profit, it is not guaranteed that released requests will always be matched. As previously mentioned, it may neither be feasible nor profitable to match all requests. To tackle the dynamic arrival of requests, we next introduce a rolling horizon simulation framework to evaluate the myopic and stochastic decision strategies.

4.4. The Rolling Horizon Framework

We now describe the rolling horizon framework. Such frameworks are common, for example, in dynamic vehicle routing problems [Mitrović-Minić et al., 2004]. In this framework, information on released requests (i.e., the realization of uncertain requests) arrives dynamically, time is discretized into τ periods per hour, and new decisions can be taken at each time period. At period t, the system only knows the values of $q'_v, \forall v \in V, r_v \leq t$, and may or may not have access to information about the distribution of $\mathbf{q}$ for subsequent time periods. Previous matching decisions can only be modified if the requests are unmatched. Thus, decisions taken at each time period may impact the profitability of future decisions, and different matching strategies may have different performances throughout the planning horizon. The rolling horizon framework is outlined in Algorithm 4.1.

Algorithm 4.1: Rolling horizon simulation framework.

1 **for** $t \coloneqq 1, \ldots, h$ **do**
2 | observe $q'_v, \forall v \in V, r_v = t$;
3 | obtain some estimate on $q'_v, \forall v \in V, r_v > t$;
4 | decide on matches and unmatches for period t;
5 **end**

We now introduce three strategies that can generate decisions (as in line 4 of Algorithm 4.1) within the rolling horizon framework. The first one is a myopic strategy that assumes no knowledge on future information. The other strategies are stochastic, each assuming different levels of knowledge about the distribution of $\mathbf{q}$.

4.4.1. The myopic strategy

A simple approach to generate matches and unmatches in a rolling horizon framework when forecasts for $\mathbf{q}$ are unavailable consists in formulating a myopic strategy. At each time period t, the strategy optimizes only for the current time period. Therefore, decisions may not be optimal when considering the full planning horizon. Given the smaller model size, this strategy may be used when decisions are required quickly. Moreover, the performance of the myopic strategy can be used as a benchmark to assess the performance of the stochastic strategies, described in the next sections. We formulate the myopic problem of matching and unmatching requests such that the net profit at time period $k \in T$ is maximized as below:

$$f_{\text{MYO}}(k, z^k) \coloneqq \max \sum_{(ij)\in E_k} (p^k_{ij} x^k_{ij} - c^k_{ij} y^k_{ij}) \tag{4.4.1}$$

$$\text{s.t.} \sum_{(ij)\in\delta_k(v)} (x^k_{ij} - y^k_{ij}) \leq q'_v - \sum_{(ij)\in\delta_k(v)} z^k_{ij} \qquad \forall v \in V, r_v \leq k \tag{4.4.2}$$

$$y^k_{ij} \leq z^k_{ij} \qquad \forall (ij) \in E_k \tag{4.4.3}$$

$$z^{k+1}_{ij} = z^k_{ij} + \begin{cases} x^k_{ij} - y^k_{ij} \text{ if } (ij) \in E_k \\ 0 \text{ otherwise} \end{cases} \qquad \forall (ij) \in E \tag{4.4.4}$$

$$x^k_{ij}, y^k_{ij} \in \{0, 1\} \qquad \forall (ij) \in E_k \tag{4.4.5}$$

$$z^{k+1}_{ij} \in \{0, 1\} \qquad \forall (ij) \in E. \tag{4.4.6}$$

The objective (4.4.1) maximizes the net profit of matching and unmatching requests at time period k. The constraints (4.4.2) ensure that only inactive requests can be matched. The constraints (4.4.3) ensure that only active requests can be unmatched. The constraints (4.4.4) define the values of z for the next time period. Note that this formulation only accesses the values of q'_v for the requests released up to the current period k ($r_v \leq k$).

4.4.2. The stochastic strategies

The decisions generated at period k impact the decisions for subsequent time periods $k+1, \ldots, h$. It is therefore beneficial to develop strategies that go beyond the myopic view and explicitly consider the impact of decisions made at period k on subsequent time periods. While the static model may be used based on forecasts for the requests, such point-estimates may not well represent their stochastic nature. We therefore adapt the static model to a two-stage stochastic programming model that exploits information on the distribution of $\mathbf{q}$.

Two-stage stochastic programming models have two sets of decisions: first-stage and second-stage decisions. Variable coefficients associated with the first-stage decisions are assumed to be known and certain. On the other hand, some coefficients associated with the second-stage decisions are uncertain, but it is assumed that their distributions can be sufficiently well estimated. First-stage decisions are optimized based on the objective associated with the first stage decisions and their anticipated impact on the second-stage problem. This impact is referred to as the *second-stage value function*, and quantifies the impact of the first stage decisions over all possible realizations of the second-stage coefficients. For further reading on stochastic programming, we refer the reader to Birge and Louveaux [2011].

In our case, first-stage decisions are given for the requests available at the current time period. In the second stage, these decisions are made for the remainder of the planning horizon, considering all possible outcomes for $\mathbf{q}$. This leads to first-stage decisions that maximize the expected net profit over the remainder of the planning horizon. The stochastic program is defined as follows.

$$f_{\text{STO}}(k, z^k) := \max \sum_{(ij)\in E_k} (p_{ij}^k x_{ij}^k - c_{ij}^k y_{ij}^k) + \mathscr{Q}(k+1, z^{k+1}) \tag{4.4.7}$$

$$\text{s.t.} \quad \sum_{(ij)\in\delta_k(v)} (x_{ij}^k - y_{ij}^k) \leq q_v' - \sum_{(ij)\in\delta_k(v)} z_{ij}^k \qquad \forall v \in V, r_v \leq k \tag{4.4.8}$$

$$y_{ij}^k \leq z_{ij}^k \qquad \forall (ij) \in E_k \tag{4.4.9}$$

$$z_{ij}^{k+1} = z_{ij}^k + \begin{cases} x_{ij}^k - y_{ij}^k \text{ if } (ij) \in E_k \\ 0 \text{ otherwise} \end{cases} \qquad \forall (ij) \in E \tag{4.4.10}$$

$$x_{ij}^k, y_{ij}^k \in \{0, 1\} \qquad \forall (ij) \in E_k \tag{4.4.11}$$

$$z_{ij}^{k+1} \in \{0, 1\} \qquad \forall (ij) \in E, \tag{4.4.12}$$

where

$$\mathscr{Q}(k+1, z^{k+1}) = \mathbb{E}_q[Q(k+1, z^{k+1}, q)]$$

is the expected second-stage value function. The objective (4.4.7) maximizes the first-stage net profit (for period k) plus the expected second-stage net profit (for periods $k+1, \ldots, h$).

The constraints (4.4.8) and (4.4.9) match and unmatch requests for period k. The constraints (4.4.10) define the values of z for the next time period. The first-stage problem has a structure similar to the one of the myopic problem (4.4.1)–(4.4.5). For a specific realization q of $\mathbf{q}$ (also called scenario), the second-stage value function $Q(k+1, z^{k+1}, q)$ is a multiperiod problem over the time periods $(k+1), \ldots, h$, defined below.

$$Q(k, z^k, q) \coloneqq \max \sum_{t=k}^{h} \sum_{(ij)\in E_t} (p_{ij}^t x_{ij}^t - c_{ij}^t y_{ij}^t) \tag{4.4.13}$$

$$\text{s.t.} \sum_{(ij)\in\delta_t(v)} (x_{ij}^t - y_{ij}^t) \leq q_v' - \sum_{(ij)\in\delta_t(v)} z_{ij}^k \qquad \forall t = k, \ldots, h, v\in V, r_v < k \tag{4.4.14}$$

$$\sum_{(ij)\in\delta_t(v)} (x_{ij}^t - y_{ij}^t) \leq q_v - \sum_{(ij)\in\delta_t(v)} z_{ij}^k \qquad \forall t = k, \ldots, h, v\in V, r_v \geq k \tag{4.4.15}$$

$$y_{ij}^t \leq z_{ij}^t \qquad \forall t = k, \ldots, h, (ij)\in E_t \tag{4.4.16}$$

$$z_{ij}^{t+1} = z_{ij}^t + \begin{cases} x_{ij}^t - y_{ij}^t \text{ if } (ij) \in E_t \\ 0 \text{ otherwise} \end{cases} \qquad \forall t = k, \ldots, h, (ij)\in E \tag{4.4.17}$$

$$x_{ij}^t, y_{ij}^t \in \{0, 1\} \qquad \forall t = k, \ldots, h, (ij)\in E_t \tag{4.4.18}$$

$$z_{ij}^{t+1} \in \{0, 1\} \qquad \forall t = k, \ldots, h, (ij)\in E, \tag{4.4.19}$$

and has a structure similar to the one of the static problem (4.3.1)–(4.3.6). We emphasize that this two-stage stochastic program maximizes the expected profit over all possible realizations. However, given that in the rolling horizon simulation a specific realization q' will occur, the planning is not guaranteed to be optimal for q'. Additionally, our stochastic program is limited to two stages, which only approximates the true multistage structure of our problem.

4.4.2.1. The expected value strategy. Solving the stochastic program is a challenging task, as the number of all possible realizations of $\mathbf{q}$ is exponential in the number of requests. We are thus interested in alternative models that approximate $\mathscr{Q}(k+1, z^{k+1})$. A first approach to approximate the second-stage value function is to replace the random variables $\mathbf{q}$ with their expected values $\mathbb{E}[q]$. We refer to this problem as the expected value problem (EVP). As we can not reasonably assume perfect knowledge of $\mathbb{E}[q]$ in a real-life setting, we build the EVP based on the expected value over N independent samples $q^1, \ldots, q^N$ of $\mathbf{q}$ (which may stem from historical observations). To build the EVP, we replace the expected second-stage value function $\mathscr{Q}(k+1, z^{k+1})$ by

$$\bar{\mathscr{Q}}(k+1, z^{k+1}) = Q(k+1, z^{k+1}, \bar{q}),$$

where

$$\bar{q} = \frac{1}{N} \sum_{j=1}^{N} q^j$$

is the expected value of the samples $q^1, \ldots, q^N$. The EVP is defined below.

$$f_{\text{EVP}}(k, z^k) := \max \sum_{(ij) \in E_k} (p_{ij}^k x_{ij}^k - c_{ij}^k y_{ij}^k) + \bar{\mathscr{Q}}(k+1, z^{k+1}) \tag{4.4.20}$$

$$\text{s.t. } (4.4.8)\text{–}(4.4.12). \tag{4.4.21}$$

As the expected values of our random variables can be fractional and the second-stage decision variables are binary, most of them will assume value zero to satisfy the matching constraints. Therefore, the first-stage decisions would likely be identical or similar to the ones generated by the myopic model, leading to an ill-defined EVP. To circumvent this issue, we relax the integrality constraints of all second-stage variables.

4.4.2.2. The sample average approximation strategy. The EVP may not generate good first-stage decisions, and solving the full stochastic program may be necessary. However, enumerating all $2^{|V|}$ realizations of $\mathbf{q}$ along with their probability distribution may be infeasible in practice and computationally intractable. We consider instead an approximation of $\mathscr{Q}(k+1, z^{k+1})$, where we sample $N \ll 2^{|V|}$ samples $q^1, \ldots, q^N$ of $\mathbf{q}$. Specifically, samples are generated by independent random sampling from the binary distributions associated with the release of each request. Such samples are also referred to as *scenarios*, and we aim to build a model that optimizes second-stage decisions for each scenario, while keeping the same first-stage decisions. In the stochastic programming literature, this link between the set of first-stage decisions and the multiple sets of second-stage decisions is referred to as non-anticipativity [Birge and Louveaux, 2011]. Thus, the objective function of such model would quantify the expected profitability of first-stage decisions under several possible realizations of the random variables. To write this objective function, we replace $\mathscr{Q}(k+1, z^{k+1})$ by

$$\mathscr{Q}_N(k+1, z^{k+1}) = \frac{1}{N} \sum_{j=1}^{N} Q(k+1, z^{k+1}, q^j),$$

and solve the so-called sample average approximation (SAA) problem:

$$f_{\text{SAA}}(k, z^k) := \max \sum_{(ij) \in E_k} (p_{ij}^k x_{ij}^k - c_{ij}^k y_{ij}^k) + \mathscr{Q}_N(k+1, z^{k+1}) \tag{4.4.22}$$

$$\text{s.t. } (4.4.8)\text{–}(4.4.12). \tag{4.4.23}$$

Both the SAA and EVP models can be easily used within a rolling horizon framework like the one outlined in Algorithm 4.1. At each optimization step, i.e., at each moment in time when the optimization model is executed, the input data of the model is set according to requests that have already been occurred, as well as the past matching decisions. Requests that have already been released are included in the first stage of the two-stage stochastic model, and sample realizations (or mean values, if the EVP is used) of potential future requests are present in the second stage of the model.

4.5. Computational Study

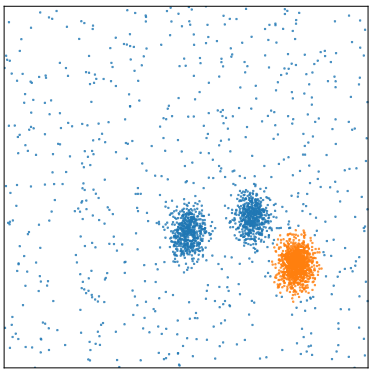
In this section, we evaluate the performance of the dynamic strategies. We generate an extensive set of benchmark instances based on data obtained from an industrial partner. We then repeatedly simulate each strategy in the rolling horizon framework.

4.5.1. Benchmark instances

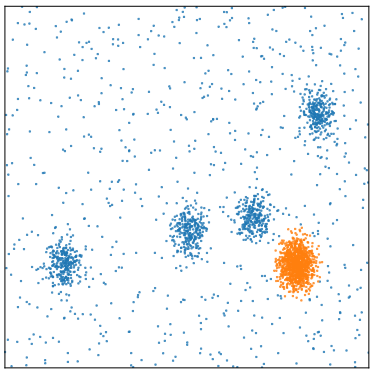
To ensure that we consider multiple realistic set-ups for ridesharing systems, we generate an extensive benchmark set of 240 instances based on the demand patterns of Netlift, a Montreal-based ridesharing company. We based our instances on their ridesharing patterns observed from January to June 2019. A problem instance consists of a planning horizon T, a set of requests V, the random vector $\mathbf{q}$, the probabilities $P(\mathbf{q_v} = 1), \forall v \in V$, and a real realization q' of $\mathbf{q}$. We consider a planning horizon $T = \{1, \ldots, 72\}$, where each period corresponds to an interval of 20 minutes. The first period corresponds to 8pm and the last period corresponds to the same time 24 hours later. We also consider that $|V| = 150 \cdot |T|$, where 77% of the requests are requests from drivers. We generate instances with different characteristics: patterns of origins and destinations, request groups, release times, and proportion of recurrent requests. We generate 10 instances for each combination of such characteristics, which are described in the following.

4.5.1.1. Patterns of origins and destinations. Origins and destinations are restricted to the greater Montreal region and are distributed according to three different patterns: 3g, 5g, and 7g. We define the greater Montreal region with the following longitude and latitude bounding box: -73.9058, 45.4146, -73.4769, 45.7029. All patterns have one demand center representing downtown Montreal (coordinates -73.56154389, 45.49721524). The other demand centers represent different regions of interest in the Montreal metropolitan area, such as La Petite-Italie (-73.61233988, 45.53537754), Côte-Vertu (-73.6882723, 45.52320286), Dollard-des-Ormeaux (-73.836727, 45.49915694), Montréal-Est (-73.53648307, 45.61656685), Rosemère (-73.81177627, 45.63200686), and Montréal-Nord (-73.63821111, 45.59797576). In pattern 3g, points are sampled from three demand centers. In pattern 5g, points are sampled from five demand centers. Finally, in pattern 7g, points are sampled from seven demand centers. The coordinates for each demand center are obtained by sampling from Gaussian distributions with variance 10^{-4}. We have opted to generate origins and destinations with Gaussian distributions (as opposed to uniform distributions), given that they can more realistically represent demand hot spots, with higher demand in the center and lower demand as the distance from the center gets larger. Additionally, to represent requests outside common demand centers, all patterns have some points that are generated uniformly over the greater Montreal region bounding box (described in the next section). The different patterns are illustrated in Figure 4.2, with the downtown region highlighted in orange.

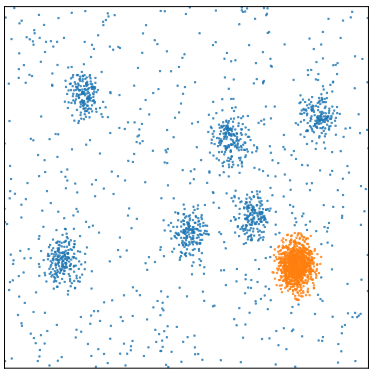
Fig. 4.2. The distribution of points for different patterns. Orange points refer to downtown.



(a) Pattern 3g.



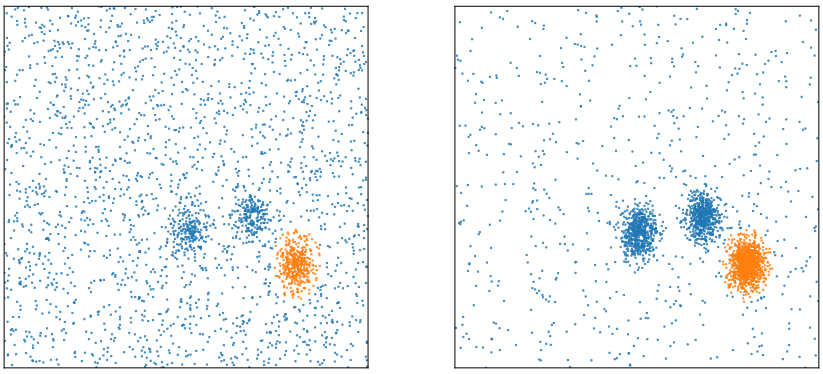
(b) Pattern 5g.



(c) Pattern 7g.

4.5.1.2. Request groups. Requests belong to one of the following groups: *central* or *random*. The central requests have a downtown origin and a non-downtown destination, or a non-downtown origin and a downtown destination. These origins and destinations are generated from the Gaussian distributions of each pattern. The random requests have origins and destinations uniformly sampled from the greater Montreal region. We use a centrality parameter ω_c to control the proportion of central requests (ω_c) and the proportion of random requests ($1 - \omega_c$), where $\omega_c \in \{\, 25\%, 75\% \,\}$. Figure 4.3 exemplifies how the distribution of points changes for two different values of ω_c. Note that when ω_c approaches zero, the distribution of points becomes entirely uniform.

Fig. 4.3. Distribution of points for different values of ω_c.

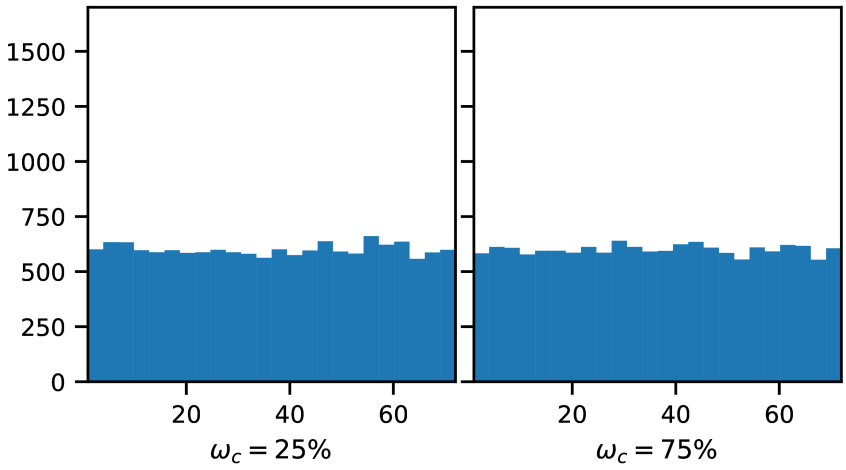


(a) $\omega_c = 25\%$. **(b)** $\omega_c = 75\%$.

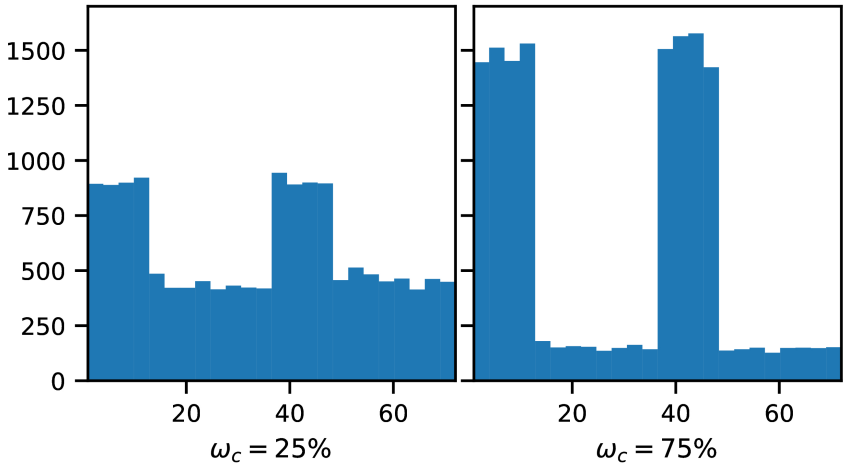
4.5.1.3. Release times. The release times of requests are either *uniform* or *clustered*. In the uniform case, for each $v \in V$, r_v is a period in T chosen uniformly at random. In the clustered case, if a request belongs to the random group, then its release time is also a randomly chosen period in T. Otherwise, if the request belongs to the central group, then the release time is a number randomly chosen from $\{1, \ldots, 12\}$ (8 pm to midnight) if its destination is downtown, or from $\{36, \ldots, 48\}$ (next day, 8 am to noon) if its destination is outside downtown.

Figure 4.4 shows the distribution of uniform and clustered release times for different values of ω_c. For uniform release times, the value of ω_c does not affect the shape of the distribution. However, for clustered release times, smaller values of ω_c lead to more evenly distributed request releases.

Fig. 4.4. Distribution of release times for different values of ω_c.



(a) Uniform release times.



(b) Clustered release times.

4.5.1.4. Random variables and recurrent requests. For each request $v \in V$, the probability of it being released ($q_v = 1$) is a random number in $[20\%, 50\%]$. Some requests may be *recurrent*. Recurrent requests represent customers that use the ridesharing system regularly. For these requests, we assign higher probabilities, randomly selected from $[80\%, 100\%]$. We control the proportion of recurrent requests in V with a trip-recurrence parameter ω_r and generate instances with $\omega_r \in \{\, 5\%, 10\% \,\}$.

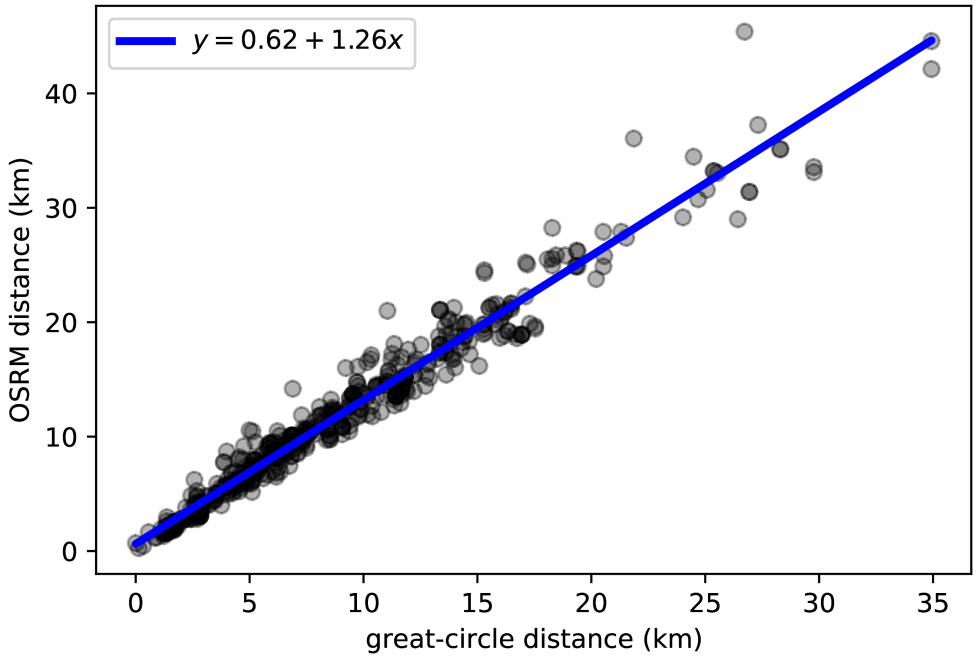
4.5.1.5. Travel distance and time. To evaluate the profitability and the time-window feasibility of matches, it is necessary to define the distances between origins and destinations. One option is to use a dedicated routing software such as OSRM [Luxen and Vetter, 2011] to find real route distances. Another more suitable option for preliminary analyses is to use

a simple distance metric $\bar{d}(a, b)$ as an approximation for the real distance and to adjust it with coefficients found by fitting a regression model. We use this second option, using the great-circle distance as an approximation for the real distance. We express the adjusted distance $d(a, b)$ as a function of the great-circle distance $\bar{d}(a, b)$, as shown below:

$$d(a, b) = \beta_0 + \beta_1 \bar{d}(a, b).$$

To obtain the coefficients β_0 and β_1, we built a linear least squares regression model with 100 distances for trips within the Greater Montreal area, illustrated in Figure 4.5. For each of those 100 trips, we obtained the true distances from OSRM and fitted the linear regression, yielding $\beta_0 = 0.62$ and $\beta_1 = 1.26$ (rounded to two decimal places). Finally, based on travel distance, we calculate the travel time $t(a, b)$ assuming a constant speed of $40km/h$, i.e., $t(a, b) = d(a, b)/40$.

Fig. 4.5. The regression model.



4.5.1.6. Departure and arrival times. We generate the requests departure times based on their release times plus a delay. For each request $v \in V$ with release time r_v, we randomly select a period in $r_v, \ldots, r_v + 30$ for its departure time a_v. Departure times are therefore requested on average five hours after their corresponding release times. The arrival time b_v is $a_v + t_v(o_v, d_v)$.

4.5.1.7. Objective function. We define the profit coefficient p and the penalty coefficient c based on the distance savings generated by the rideshare and on the number of time periods

between request release and request matching or unmatching:

$$p_{ij}^t \coloneqq s_{ij} - \frac{\lambda_p}{\tau \cdot 100} \Delta_{ij}^t s_{ij} \qquad \forall t \in T, (ij) \in E_t,$$

$$c_{ij}^t \coloneqq s_{ij} + \frac{\lambda_c}{\tau \cdot 100} \Delta_{ij}^t s_{ij} \qquad \forall t \in T, (ij) \in E_t.$$

The coefficient τ is the number of periods per hour (in our case, $\tau = 3$) and

$$\Delta_{ij}^t = \frac{(t - r_i) + (t - r_j)}{2} \qquad \forall t \in T, (ij) \in E_t,$$

is the average number of periods between t and the release time of the two requests. The parameters λ_p and λ_c are penalty coefficients that represent the hourly decrease of matching profit and the hourly increase of unmatching penalty. We generally assume that $(\lambda_p, \lambda_c) = (5, 2)$, such that matching profits are penalized 5% per hour and unmatching costs are increased 2% per hour. We conduct further experiments with different values for λ_p and λ_c in Section 4.5.2.8.

4.5.2. Computational experiments

In this section, we evaluate the performance of the dynamic strategies when simulated in a rolling horizon framework. We implemented the strategies in Python 3.7 and used CPLEX 12.10 to solve the mathematical programming models. We limited CPLEX to one thread and used its default parameters. The experiments were run on a cluster with CPUs ranging from 2.1 to 2.4 GHz.

When solving the instances with the stochastic strategies, we consider 5 to 40 scenarios. We always perform 5 rolling horizon simulations for these strategies, each of which considers new samples of $\mathbf{q}$. We used the model reduction techniques proposed in Homsi et al. [2020] to reduce the time needed to build and solve the models.

4.5.2.1. Performance over all instances. We first analyze the average performance of each strategy over our instance benchmark set. The set contains a total of 240 instances for all combinations of patterns 3g, 5g and 7g, $\omega_c \in \{ 25\%, 75\% \}$, $\omega_r \in \{ 5\%, 10\% \}$, and clustered and uniform release times. Average results over the instances are shown in Table 4.1. The column "gap" is the percentage gap to the revenue generated by the static model. The column "std" is the gap standard deviation. The column "T(min)" is the average sum of computing time (in minutes) required to build and solve the 72 models with CPLEX. The last row contains the average over all rows (except for row STAT).

Table 4.1. Rolling horizon results averaged over all instances.

algo	$\|S\|$	gap	std	T(min)
STAT	–	–	–	0.01
MYO	–	4.05	1.04	0.08
EVP	5	2.77	0.90	1.58
	10	2.72	0.90	1.75
	20	2.69	0.89	1.75
	40	2.69	0.90	1.66
SAA	5	2.71	0.90	4.87
	10	2.66	0.91	10.01
	20	2.66	0.91	20.37
	40	2.65	0.92	39.45
mean	–	2.84	0.92	9.06

The results show that both stochastic strategies outperform the myopic strategy. Furthermore, increasing the number of scenarios tends to reduce the gap of the stochastic strategies. Using the same set of scenarios, the SAA constantly outperforms the EVP. However, the difference between the gaps found by the two models is rather small. On the other hand, the SAA is about 3 to 24 times slower than the EVP. In fact, the SAA computing time increases with the number of scenarios, while the EVP computing time remains stable. Note again that this is the total computing time required for 72 optimization runs carried out during a planning horizon of 24 hours. The time required for each run is rather negligible. We elaborate on the computing time per optimization run in Section 4.5.2.7.

4.5.2.2. Performance for different patterns. We evaluate the performance of each strategy on the patterns 3g, 5g, and 7g. The average results are shown in Table 4.2.

Table 4.2. Rolling horizon results for different patterns.

		3g			**5g**			**7g**		
algo	$\|S\|$	gap	std	T(min)	gap	std	T(min)	gap	std	T(min)
STAT	–	–	–	0.01	0.00	0.00	0.01	0.00	0.00	0.01
MYO	–	4.08	1.03	0.10	3.83	0.96	0.07	4.23	1.08	0.08
EVP	5	2.81	0.89	1.81	2.69	0.91	1.40	2.81	0.89	1.51
	10	2.78	0.90	2.02	2.66	0.91	1.55	2.72	0.89	1.68
	20	2.74	0.89	2.03	2.63	0.89	1.54	2.71	0.89	1.68
	40	2.72	0.88	1.91	2.64	0.91	1.45	2.70	0.89	1.61
SAA	5	2.72	0.87	5.89	2.63	0.92	4.10	2.76	0.91	4.61
	10	2.69	0.89	12.18	2.57	0.90	8.35	2.70	0.93	9.50
	20	2.67	0.88	24.71	2.59	0.90	16.99	2.71	0.95	19.42
	40	2.66	0.88	47.80	2.56	0.91	32.91	2.71	0.95	37.64
mean	–	2.88	0.90	10.94	2.76	0.91	7.60	2.89	0.93	8.64

The results indicate that the stochastic strategies outperform the myopic strategy for all patterns. For patterns $3g$ and $5g$, the SAA strategy outperforms the EVP strategy. For pattern $7g$, both strategies have a similar performance.

4.5.2.3. Performance for different values of centrality parameter ω_c. We now evaluate how the performance of each strategy changes for different values of ω_c. The higher the value of ω_c, the more to-downtown and from-downtown trips are present in the set of potential requests. We test $\omega_c \in \{\, 25\%, 75\% \,\}$. Results are summarized in Table 4.3.

Table 4.3. Results for different values of centrality ω_c.

		25%			**75%**		
algo	$\|S\|$	gap	std	T(min)	gap	std	T(min)
STAT	–	–	–	0.01	0.00	0.00	0.01
MYO	–	4.45	0.74	0.04	3.65	1.13	0.13
EVP	5	3.37	0.59	0.91	2.17	0.75	2.24
	10	3.35	0.55	0.99	2.09	0.72	2.51
	20	3.31	0.55	0.99	2.07	0.71	2.51
	40	3.31	0.55	0.93	2.06	0.72	2.39
SAA	5	3.27	0.57	2.02	2.14	0.81	7.72
	10	3.24	0.58	4.07	2.08	0.80	15.95
	20	3.24	0.56	8.36	2.07	0.81	32.38
	40	3.23	0.57	15.76	2.06	0.81	63.14
mean	–	3.42	0.59	3.79	2.27	0.81	14.33

The results indicate that increasing ω_c leads to smaller gaps for all strategies. This suggests that the higher geographical density of downtown trips and the wider availability of information on periods associated with central trips reduces uncertainty. However, increasing ω_c also leads to a denser graph and therefore bigger models, which increases the CPU time.

4.5.2.4. Performance for different values of trip-recurrence ω_r. We now investigate the impact of the proportion of recurrent requests on each strategy. Bigger values of ω_r may represent the fact that the company has a loyal customer base. We here test $\omega_r \in \{\, 5\%, 10\% \,\}$. Results are shown in Table 4.4.

Table 4.4. Results for different values of trip-recurrence ω_r.

		5%			10%		
algo	$\|S\|$	gap	std	T(min)	gap	std	T(min)
STAT	–	–	–	0.01	0.00	0.00	0.01
MYO	–	4.01	0.99	0.08	4.08	1.08	0.09
EVP	5	2.86	0.91	1.50	2.68	0.88	1.65
	10	2.82	0.90	1.68	2.62	0.88	1.82
	20	2.79	0.88	1.71	2.59	0.89	1.80
	40	2.79	0.89	1.60	2.59	0.89	1.72
SAA	5	2.82	0.92	4.39	2.59	0.87	5.35
	10	2.76	0.92	9.11	2.56	0.89	10.91
	20	2.76	0.93	18.68	2.56	0.88	22.07
	40	2.74	0.94	35.92	2.55	0.88	42.98
mean	–	2.93	0.92	8.30	2.76	0.90	9.82

The results indicate that as ω_r increases, the myopic strategy gap and standard deviation increase, and the stochastic strategies gaps and standard deviations decrease. This result is expected because the myopic strategy has no knowledge that some requests are highly likely to be released, and therefore it is subject to generate many suboptimal rideshares. On the other hand, the stochastic strategies can capture this information and generate better decisions. The results also show that as ω_r increases, the CPU time increases, this is due to having a denser graph, as more requests are released.

4.5.2.5. Performance on clustered and uniform release times. We now evaluate the impact of clustered and uniform release times. Results are shown in Table 4.5.

Table 4.5. Results for clustered and uniform release times.

		clustered			uniform		
algo	$\|S\|$	gap	std	T(min)	gap	std	T(min)
STAT	–	–	–	0.01	0.00	0.00	0.01
MYO	–	3.44	0.96	0.12	4.65	0.70	0.05
EVP	5	2.34	0.93	1.90	3.20	0.63	1.25
	10	2.31	0.92	2.11	3.13	0.64	1.39
	20	2.28	0.92	2.13	3.11	0.63	1.38
	40	2.27	0.91	2.03	3.11	0.64	1.29
SAA	5	2.26	0.92	6.51	3.15	0.62	3.23
	10	2.20	0.92	13.37	3.11	0.63	6.65
	20	2.20	0.93	27.32	3.11	0.61	13.42
	40	2.19	0.94	53.16	3.11	0.61	25.75
mean	–	2.39	0.93	12.07	3.30	0.64	6.04

The results suggest that instances with clustered release times take longer to be solved. This is due to denser graphs for the periods where requests are released more often. For the instances with uniform release times, all gaps are larger. This is due to the randomness in

release times, which causes less information to be available at each period and makes the prediction on the release of future requests harder for the stochastic strategies.

4.5.2.6. The impact of forbidding unmatching. We now evaluate the performance of each strategy when unmatching is forbidden. To do so, we assign a big value for λ_c. As shown in Homsi et al. [2020], the static model never unmatches previously matched requests given that the rematching profits are never bigger than the unmatching costs. Therefore, forbidding unmatching does not impact the optimal objective function value of the static model and we can directly compare the performance of the various models under these two problem variants. Results are shown in Table 4.6.

Table 4.6. Results with allowed and forbidden unmatching.

		allowed			**forbidden**		
algo	$\|S\|$	gap	std	T(min)	gap	std	T(min)
STAT	–	0.00	0.00	0.01	0.00	0.00	0.01
MYO	–	4.05	1.04	0.08	14.89	4.41	0.08
EVP	5	2.77	0.90	1.58	3.48	1.26	1.57
	10	2.72	0.90	1.75	3.18	1.11	1.75
	20	2.69	0.89	1.75	3.10	1.07	1.75
	40	2.69	0.90	1.66	3.07	1.04	1.66
SAA	5	2.71	0.90	4.87	3.50	1.21	4.80
	10	2.66	0.91	10.01	3.34	1.11	9.89
	20	2.66	0.91	20.37	3.34	1.09	19.97
	40	2.65	0.92	39.45	3.38	1.11	38.16
mean	–	2.84	0.92	9.06	4.59	1.49	8.85

When unmatching is forbidden, the gap of the myopic strategy is about 4 times higher, and the gap of the stochastic strategies is about 1.5 times higher. Thus, the flexibility of unmatching is beneficial to all strategies and may increase the profits of ridesharing companies. This is particularly the case for the myopic strategy, given that the lack of predictions for future requests may lead to bad premature decisions which can only be corrected if unmatching is allowed.

The results also allow us to draw interesting conclusions on the relationship between the value of unmatches and the value of stochastic information, and how unmatching or stochastic information can be used in isolation to find solutions with similar gaps.

No stochastic information. When stochastic information is not available, unmatches can be used to correct past decisions. In this case, the myopic strategy gap is only slightly larger than the best gap of the stochastic strategies with forbidden unmatches (4.05% vs. 3.07%).

No unmatches. Otherwise, when unmatching is forbidden, stochastic information can be exploited to generate matches that are less likely to benefit from rematching. In this case, increasing the number of scenarios reduces the stochastic strategies gaps from 3.50% to 3.07%.

Due to that, stochastic strategies with forbidden unmatches attain a gap slightly smaller than the myopic strategy gap with allowed unmatches (3.07% vs. 4.05%).

Everything is available. Finally, when unmatches are allowed and stochastic information is available, the average gap of the stochastic strategies is even smaller than the myopic strategy gap with allowed unmatches (2.65% vs. 4.05%). However, increasing the number of scenarios decreases the gap at a slower pace. As it is unlikely that good forecasts on future information will be always available during daily practice, the possibility of unmatching becomes a valuable asset in a decision-maker toolset.

4.5.2.7. Solution characteristics. The goal of this section is to understand the structure and characteristics of the solutions generated from each strategy by investigating solution attributes besides profit, such as the number of matches and unmatches. For this analysis, we consider only the subset of experiments where unmatches are allowed. Average results are shown in Table 4.7. Columns "matches" and "unmatches" give the number of matches and unmatches generated in the rolling horizon, respectively. Column "net" represents the difference between both values, i.e., the number of active matches at the end of the planning horizon. Column "prop" gives the proportion of unmatches over all matches and unmatches. Column "match delay" shows the average number of periods between release times and matches. Finally, column T_{max} indicates the highest computing time (in minutes) encountered in the different optimization runs throughout the day.

Table 4.7. Statistics on matches and unmatches.

algo	$\|S\|$	matches	unmatches	net	prop	match delay	T_{max}
STAT	–	270.52	0.00	270.52	0.00	3.29	0.01
MYO	–	395.73	126.30	269.43	24.19	3.16	0.00
EVP	5	279.48	10.00	269.48	3.46	4.55	0.04
	10	276.90	7.38	269.52	2.60	4.63	0.04
	20	276.07	6.53	269.55	2.31	4.65	0.04
	40	275.85	6.30	269.55	2.23	4.66	0.04
SAA	5	283.39	13.97	269.42	4.70	4.44	0.11
	10	280.71	11.24	269.48	3.85	4.53	0.23
	20	279.18	9.65	269.53	3.34	4.60	0.47
	40	278.47	8.95	269.52	3.11	4.63	0.93
mean	–	291.75	22.26	269.50	5.53	4.43	0.21

Unmatch proportion. The results highlight the greediness of the myopic strategy. Even if it has a net match balance similar to all other strategies, it has a large unmatch proportion of 24.19%. It matches far more than the static model, which indicates that many matches are created prematurely and are unmatched as soon as new information arrives. Alternatively, the stochastic strategies have a smaller unmatch proportion. Increasing the number of scenarios reduces this proportion even more, while keeping the net match balance stable.

The EVP unmatch proportion. The EVP strategy has a slightly smaller unmatch proportion than the SAA strategy. In Homsi et al. [2020], it is shown that the unmatching of requests in the isolated second-stage problem is never profitable, provided that matching profits are smaller than unmatching costs. The same assumption is also present in our problem definition, and the results on unmatching profitability trivially extend to our EVP model with continuous second-stage variables and fractional right hand side coefficients. This suggests that the first-stage decisions generated by the EVP strategy are the ones that are less likely to require unmatching upon the arrival of new information, which may explain the smaller unmatch proportion of the EVP strategy.

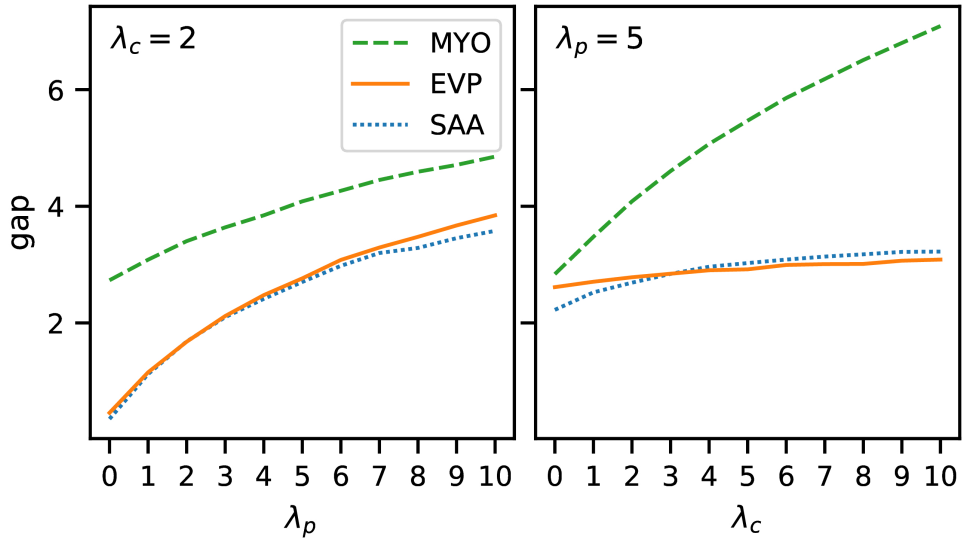
Matching delay. The myopic strategy has a matching delay similar to the one found in the static model (3.16 and 3.29 periods, i.e., about 60 minutes). The stochastic strategies have a longer delay (about 4.5 periods, i.e., about 1h30min), and increasing the number of scenarios slightly increases this delay. The SAA strategy has a slightly smaller matching delay than the EVP strategy, which may explain its slightly bigger unmatch proportion, as early decisions lead to more unmatches. The fact that the stochastic strategies have a delay of only 30 minutes longer shows that the solutions generated by these strategies are not only more profitable, but they also do not much degrade the customer experience.

Time per iteration. Among all models, the SAA requires the longest computing times. With 40 scenarios, it takes about one minute to solve the model. Nevertheless, given that each time period corresponds to an interval of 20 minutes, the SAA with up to 40 scenarios is solved sufficiently fast for a dynamic use in practice. If time periods correspond to shorter intervals (e.g., 1 to 5 minutes), having real-time decisions may be necessary. In this case, the EVP may become an interesting alternative, given that it takes at most 0.04 minute (2.4 seconds) per optimization run, and its solution time does not depend on the number of scenarios.

4.5.2.8. Performance under different corporate preferences. Ridesharing companies may have different objectives when it comes to the desired customer experience. A company may therefore adjust the profit and penalty parameters λ_p and λ_c to best fit their corporate strategy. The goal of this section is to assess the performance of the dynamic strategies when the objective function penalties λ_p and λ_c change. We conduct two sets of experiments on the subset of instances with pattern 3g, $\omega_c \in \{ 25\%, 75\% \}$, and $\omega_r \in \{ 5\%, 10\% \}$. In the first set of experiments, we fix $\lambda_c = 2$ and test values for λ_p from 0 to 10. In the second set of experiments, we fix $\lambda_p = 5$ and test values for λ_c from 0 to 10. We limited the stochastic strategies to five scenarios, and for each instance, we conducted five runs with different random samples of $\mathbf{q}$. Results are illustrated in Figure 4.6. The vertical axis represents the average percentage gap between the obtained profit (for each strategy) and the optimal profit

(obtained from the static model). The horizontal axis represents the different values of λ_p and λ_c.

Fig. 4.6. Gap for different values of λ_p and λ_c.



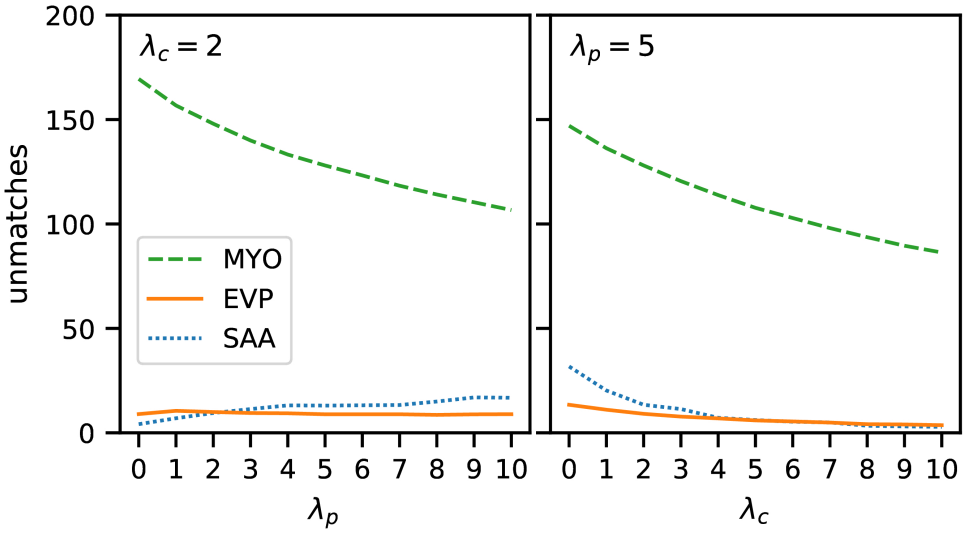
The results indicate that the stochastic strategies always outperform the myopic strategy. Below we discuss the results for interesting values for λ_p and λ_c.

Inexpensive vs. expensive delayed matching. When the profit parameter λ_p decreases, the gap between the stochastic strategies and the myopic strategy widens, as matching early is less important. Given that the stochastic strategies have forecasts for future requests, they may avoid premature matches, favoring delayed matching without the need of frequent unmatching. In contrast, when λ_p increases, the gap between the stochastic strategies and the myopic strategy narrows, as matching as soon as possible becomes more important, and therefore the stochastic strategies start to behave similarly to the myopic strategy.

Inexpensive vs. expensive unmatches. When the penalty parameter λ_c reduces, the gap between all strategies reduces, as there are seemingly no disadvantages for matching myopically and unmatching for little or no cost when necessary. Still, when unmatching is free ($\lambda_c = 0$), the myopic strategy is still the worst performer. This suggests that even when unmatching is free, matching as soon as possible may not be the best decision, as some matches may permanently block unmatches (e.g. when a rideshare starts) and restrict better matches in the future. In contrast, when $\lambda_c > 3$, the EVP strategy slightly outperforms the SAA strategy. This may be explained by the results in Table 4.7: as the unmatch proportion of the EVP is smaller than the one of the SAA, the EVP is more profitable when the unmatching costs increase.

We now analyze how the number of unmatches for each strategy changes as we vary λ_p and λ_c. Results are illustrated in Figure 4.7.

Fig. 4.7. Unmatches for different values of λ_p and λ_c.



The results indicate that as λ_p increases, the myopic strategy unmatches less. Contrarily, the SAA tends to unmatch more as λ_p increases, and the number of unmatches of the EVP is relatively stable regardless of the value of λ_p. Quite understandably, an increase of λ_c leads to a reduction in the number of unmatches for all strategies.

4.6. Conclusions and Future Work

In this paper, we have revisited a matching and rematching problem with applications in request matching for ridesharing systems. In addition to the existing myopic and two-stage stochastic models, we have proposed an EVP variant of the problem. Given that the demand averaged over all scenarios is typically fractional, a classical implementation of this model would tend to prevent any matches in the second-stage. We therefore relax the integrality for the second-stage decisions.

We then conducted computational experiments for all models on an extensive set of benchmark instances representative of different ridesharing settings. Our results demonstrate the value of unmatches and the value of stochastic information: the average gap for the myopic strategy is 4.05% and the average gap for the stochastic strategies is 2.69%. When unmatching is forbidden, the benefits of stochastic information are clear, resulting in significantly lower average gaps (4.59% vs. 14.89%). These benefits are less pronounced when unmatches are allowed, which indicates that the possibility to unmatch and rematch is a good substitute for when reliable forecasts on future requests are difficult to obtain. Furthermore, the SAA strategy generates slightly better solutions than the EVP strategy: it has an average gap of 2.67%, while the EVP strategy has an average gap of 2.72%. We believe that our results can

provide valuable insights to ridesharing operators, such as understanding in which scenarios unmatching may provide benefits, how much effort should be allocated to forecast future information, and how to define the penalties for delayed matches and unmatches.

Acknowledgements

We would like to thank Netlift and Mathieu Lozeau for their thoughtful input, domain knowledge, and for providing us with access to data. The work of the third author was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada under grant 2017-05224. We would also like to thank Calcul Québec and Compute Canada for providing the computing infrastructure necessary to conduct their computational experiments.

Chapter 5

Two-Stage Stochastic One-to-Many Driver Matching for Ridesharing

In the previous chapters, a one-to-one ridesharing problem was studied. This imposes an artificial constraint on the number of passengers in a rideshare, which limits the emission mitigation potential of the ridesharing system. This chapter relaxes this assumption and studies a one-to-many ridesharing problem. While the one-to-one problem studied in the previous chapters was modeled based on a bipartite matching formulation, the one-to-many problem of this chapter is modeled using a set-packing formulation, which allows for the representation of ridesharing routes with more than one rider. The here considered problem has stochastic riders and a two-stage structure: all drivers are known in the first stage, and riders are released in the second stage. This chapter explores the possibility of booking drivers in advance (in the first stage) to improve user-friendliness through better predictability: booked drivers are paid a fee in advance, or are provided compensation in case they are not assigned to a rideshare. In the second stage, the released riders are observed and can be assigned for ridesharing with booked drivers. The problem introduced in this chapter is general enough to represent a wide range of ridesharing systems. The results of extensive computational experiments are analyzed to identify the advantages and disadvantages of different types of ridesharing systems and stochastic programming model approximations. The contributions of the student are:

- The introduction of a novel one-to-many ridesharing problem with driver booking;
- Deterministic and two-stage stochastic mathematical programming models for that problem;
- Approximations to the two-stage stochastic programming model, including a relaxation of the classical sample average approximation model;
- Empirical evaluation of this ridesharing problem on four different ridesharing modes of operation;

- Managerial insights on different ridesharing modes of operation, and recommendations on the best mode of operation along with recommendations on system parameters.

The contents of this chapter have been accepted for publication in *Networks*. A technical report is available in https://www.cirrelt.ca/documentstravail/cirrelt-2022-36.pdf.

Two-Stage Stochastic One-to-Many Driver Matching for Ridesharing

Gabriel Homsi [1,3], Bernard Gendron [1,3], Sanjay Dominik Jena [2,3]

[1] Department of Computer Science and Operations Research, Université de Montréal, Canada

[2] Department of Analytics, Operations and Information Technology, École des Sciences de la Gestion, Université du Québec à Montréal, Canada

[3] Centre interuniversitaire de recherche sur les reseaux d'entreprise, la logistique et le transport (CIRRELT)

Abstract

We introduce a modeling framework for stochastic rider-driver matching in many-to-one ridesharing systems, in which drivers have to be selected before the exact rider demand is known. The modeling framework allows for the use of driver booking fees and penalties for unmatched drivers, therefore supporting different system operating modes. We model this problem as a two-stage stochastic set packing problem. To tackle the intractability of the stochastic problem, we introduce three model approximations and evaluate them on a large set of benchmark instances for three different system operating modes. Our computational experiments show the superiority of some model approximations over others and provide valuable insights on the impact of penalties and booking fees on the system's profitability and user satisfaction.

Keywords: 2-stage stochastic programming; ridesharing matching; mixed-integer programming.

5.1. Introduction

Ridesharing agencies match drivers and riders to jointly fulfill their itineraries. As more than half of the world population lives in urban areas [UN, 2018], the potential benefits of ridesharing are numerous. For instance, the reduction of road congestion and CO_2 emissions, improved access to transportation in regions with limited mass transit, and the reduction of transportation costs. Individuals may participate in ridesharing for several reasons [see, e.g., Teal, 1987]. Reasons may be, for example: environmental awareness, economical savings, and the lack of access to mass transit systems. Furthermore, cities may encourage ridesharing through the introduction of high-occupancy vehicle lanes [Giuliano et al., 1990], and governments may implement carbon-reduction regulations to promote ridesharing [Si et al., 2022].

The practice of ridesharing is not new, and dates as early as WWII and the 1970s oil crisis. Furthermore, the expansion of telecommunication systems, particularly the Internet, contributed to the development of numerous ridesharing systems [Chan and Shaheen, 2012]. The potential scale of ridesharing markets is evidenced by surveys conducted by the US government on the occupancy rate of vehicles. These surveys show that the average occupancy rate of vehicles involved in work-related trips has been smaller than 1.4 since the 1970s [McGuckin and Fucci, 2018]. Thus, provided the presence of a large user-base, ridesharing systems can promptly access a large fleet of vehicles with spare capacity for transportation. The presence of such user-base is therefore of key importance for the success of a ridesharing system.

Ridesharing has similarities to other shared mobility practices, such as ridehailing (also known as ridesourcing, e.g., Uber and Lyft), taxi sharing (e.g., UberPOOL and Lyft Line), carsharing, and even practices that do not involve cars, such as bike sharing. In each of these cases, a resource must be shared between individuals to fulfill their transportation objectives. For carsharing and bike sharing, the shared resources are vehicles and bikes, respectively: participants can rent a car or a bike for a short amount of time to fulfill their transportation needs. In contrast, for ridesharing, ridehailing (or ridesourcing), and taxi sharing, participants share the same resource (i.e., the same vehicle) to fulfill a transportation trip. In ridehailing and taxi sharing, drivers typically are professionals who receive a salary for their work. In ridesharing, drivers typically are not professionals, but individuals with a planned trip from an origin to a destination (e.g., from their home to their work location). Such drivers may wish to reduce transportation costs by sharing the ride with other individuals with similar origins and destinations. For a comprehensive classification of shared mobility systems, we refer the reader to Shaheen and Chan [2016].

Different ridesharing platforms exist across the globe (e.g., BlaBlaCar and Kangaride). Factors such as sociodemographics and user behavior, demand and pricing, supply and user incentives, and platform operations and reliability influence the success of ridesharing systems. Analysis of such factors, among others, are summarized in several surveys [see, e.g., Agatz et al., 2012, Wang and Yang, 2019]. Specifically, several works have identified the target groups and the required conditions that encourage users to engage in ridesharing [see, e.g., Gargiulo et al., 2015, Liu and Li, 2017, Sprei, 2018]. Our paper proposes models to match drivers and riders in a variety of ridesharing systems and further complements the works above in the sense that it empirically explores the impact of the system's configuration on its performance.

Driver booking and rideshare assignment. Ridesharing systems are two-sided markets: the ridesharing platform serves as an intermediary agency that proposes rideshares (matches) to drivers and riders. These drivers and riders are free to accept or reject a rideshare. Therefore, focusing on user-friendliness is a priority for ridesharing companies.

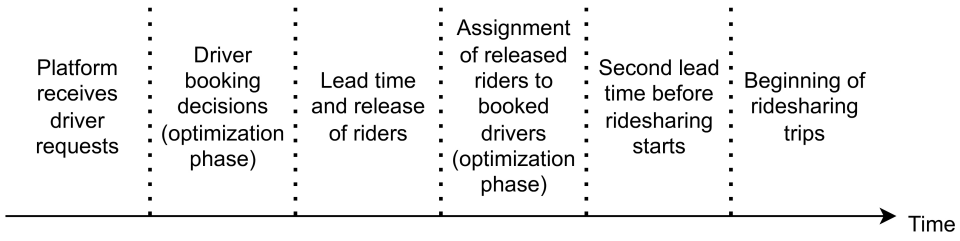
Ridesharing works best when participants have similar itineraries. For example, when commuting from residential to commercial areas or vice versa. Even if ridesharing systems limit themselves to rideshares that are geographically compatible, these rideshares may not be user-friendly enough to convince the participants to commit to the rideshares. Encouraging user participation can be achieved in several ways. One way of increasing user participation is, for example, to offer incentives such as service guarantees to participants. Furthermore, applying Operations Research tools to ridesharing may improve the quality of the generated rideshares and the overall performance of the system.

Given the importance of available drivers within the system that can be matched to ride requests, it is of utmost importance that drivers remain satisfied users of the system. It is therefore in the best interest of the operator to respond as quickly as possible to driver requests. Such timely response is not only a matter of user-friendliness, but may be a requirement for many drivers in order to prepare for the ridesharing trip. The ridesharing matching problem here studied therefore assumes that the operator has to respond to driver requests in a timely manner. As such, drivers have to be selected, typically, before all compatible rider requests are known. The operator has two possibilities to reassure selected (i.e., booked) drivers: either by paying an a priori booking fee or by paying compensation in case a selected driver is not matched to any rideshare.

The information and decision flow of the resulting planning problem can hence be organized into six stages, as illustrated in Figure 5.1. The platform first receives driver requests for the upcoming planning period (e.g., throughout the evening). The system then considers all received driver requests and decides which drivers will be booked for the upcoming ridesharing period. This step has to occur sufficiently early (e.g., at the end of the day) to give drivers a reasonably quick response and, in our case, will be optimized, using stochastic information on potential future rider requests. Meanwhile, rider requests may come in, which will be considered up to a specific moment. Within a second optimization step, the platform then assigns booked drivers and rider requests to rideshares. This assignment has to be carried out sufficiently early (e.g., early next morning) before the actual start of the trip to give participants the time to prepare for the rideshare (or look for alternative transportation modes, in case a rider was not matched).

The underlying planning problem is a two-stage many-to-one dynamic ridematching problem, in which drivers can be booked on different types in the first stage, while the rider requests are still uncertain. In the second stage, the actual rider requests are observed and assigned to booked drivers. Assignment decisions between drivers and riders correspond to planned ridesharing trips that may take place in the near future. The objective is to book drivers and assign riders to booked drivers such that the planned rideshares maximize profit.

Fig. 5.1. Information and decision flow for our planning problem.



Contributions. The contributions of this paper are as follows: 1) We introduce a novel many-to-one ride-matching problem that allows for booking drivers on different contract types under rider request uncertainty. The representation of stochasticity in the context of driver booking, as well as allowing for booking on different contract types extend the existing literature. Our work thus expands the Operations Research toolset available for researchers and practitioners in shared mobility. 2) We propose a two-stage stochastic set-packing modeling framework, instantiating three different ridesharing system operating modes. Such models provide a flexible starting point for operators that can be tailored to their specific contexts. This planning problem is NP-hard, implying a significant computational effort to solve the mathematical models. 3) We show how to efficiently enumerate feasible ridesharing routes and integrate them into the formulation. Using pruning techniques to consider only time-feasible trips, our exact enumeration procedure reduces the computing time to a fraction of the time required when naively enumerating all routes. 4) We propose models that approximate the second-stage value function of the two-stage stochastic model, specifically, a sample average approximation model and an expected value problem model. To further improve the computational tractability, we introduce a relaxation of the sample average approximation model where the integrality constraints of the second-stage problem are relaxed. Such developments allow us to solve the problem in computing times sufficiently fast in practice. 5) Finally, we evaluate all models on a new set of benchmark instances that represent a wide range of ridesharing operations. Specifically, we analyze the performance of the solutions generated by each model and quantify the economic importance to explicitly consider the uncertainty in the problem. Further, we analyze the impact of booking fees, penalties, and the use of multiple booking types, and provide managerial insights, allowing operators to identify system configurations that are both profitable and result in high user satisfaction.

Outline. This paper is structured as follows. In Section 5.2, we conduct a literature review and distinguish our work with respect to previous research in the field of ridesharing. In Section 5.3, we define the considered planning problem. In Section 5.4, we introduce mathematical formulations to solve the problem, namely, a deterministic and a two-stage stochastic

formulation. We also provide approximations for the two-stage stochastic formulation. In Section 5.5, we conduct computational studies where we evaluate the two-stage stochastic approximations on instances inspired by data from an industrial partner. We also summarize the managerial insights drawn from such results. In Section 5.6, we present the conclusions of our study and identify potential future research directions.

5.2. Related Work

Ridesharing planning is usually modeled either as one-to-one bipartite matching [Agatz et al., 2011] or as vehicle routing [Baldacci et al., 2004]. The bipartite matching formulation allows for a simple modeling of the decision problem, and its network structure allows it to be solved in polynomial time. Different characteristics of ridesharing operations were studied before. For instance, Baldacci et al. [2004] study a same-destination dial-a-ride problem for carpooling with multiple passengers per vehicle. Wang et al. [2018] and Peng et al. [2022] study ridesharing and taxi-sharing problems with stability constraints. The motivation for studying matching stability is that users would be less likely to arrange rideshares outside a rideshare platform that provides stable matches to their users. When considering multiple riders per vehicle, both studies employ a set-packing or set-partitioning formulation (which are known to be NP-Hard) similar to our problem, but without rider stochasticity, booking fees, and penalties for not finding a match. In particular, Peng et al. [2022] propose a branch-and-price algorithm to solve a many-to-one stable matching problem. To find feasible rideshares, the authors design the pricing subproblem as several knapsack problems, one for each driver. As the authors consider routes with at most two riders, this pricing subproblem can be solved efficiently. For surveys on ridesharing and related problems, we refer the reader to Agatz et al. [2012], Furuhata et al. [2013], Mourad et al. [2019], and Martins et al. [2021].

The issue of information flow and uncertainty concerning user requests has received mixed attention in the ridesharing literature. Several studies [e.g. Agatz et al., 2011, Wang et al., 2018] assume that information is released dynamically within a rolling horizon framework. In other words, in ridesharing contexts, information is rarely fully known in advance. However, tackling this uncertainty has often been overlooked in the literature on ridesharing and related fields. For example, the two studies cited above myopically generate decisions within a rolling horizon framework. Further, Bertsimas et al. [2019] study an online taxi dispatching problem, and propose a myopic matching policy that has no access to predictions on future demand. Myopic approaches are likely to underperform approaches that explicitly acknowledge uncertainty. To the best of our knowledge, Homsi et al. [2020, 2021] are the only studies that address demand uncertainty in a dynamic context. Homsi et al. [2020] introduce a two-stage matching model with rematches and stochastic rider requests for ridesharing, while Homsi et al. [2021] evaluate this model within a rolling horizon framework. Our work

is inspired by these works in the sense that we consider stochastic rider requests. We further extend these works by allowing for multiple riders in the same rideshare. However, we do not consider the possibility of rematching in our study. Instead, our work focuses on the interplay between the ridesharing decisions and the booking fees and penalties.

Furthermore, to the extent of our knowledge, there are no ridesharing studies that consider the possibility of different service type agreements between participants and the ridesharing system. We address this gap in the literature by allowing for different types of driver booking, which may be associated with different booking fees and penalties, and impact the profitability and feasibility of potential rideshares. Finally, previous studies have suggested penalties based on the response time, i.e., the time of occurrence of requests and the time the match is generated [e.g. Riley et al., 2019, Homsi et al., 2020], or penalties for riders that were not serviced [Lee and Savelsbergh, 2015]. However, to the best of our knowledge, no attention was given to studying penalties associated with the failure to provide a rideshare to drivers once an agreement (i.e., a booking) between the system and the drivers is established. Given that a ridesharing system depends on the presence of a large driver user-base to attract potential riders, our study explores the use of penalties for unmatched drivers. These penalties promote a higher level of service stability to users, as the rideshare operator has a monetary risk associated with decisions that are likely to leave drivers without a rideshare. Furthermore, prioritizing drivers may be an effective strategy to bootstrap a ridesharing system, as the presence of drivers is needed in order to attract a rider user-base.

From an Operations Research perspective, the here considered problem has connections to several classical optimization problems. As drivers have to pick up riders at their origin and drop them off at their destinations, the problem resembles an extension of the Pickup and Delivery Problem defined on a graph [see, e.g. Parragh et al., 2008a], in which the vehicles have to be selected before the commodities are known. From the matching perspective, the problem structurally can also be seen as a stochastic set-packing problem [see, e.g. Escudero et al., 2011], where the objective function coefficients are uncertain. In our case, the right-hand-side coefficients of some constraints are uncertain. Our problem also has connections to capacitated facility location problems, which can be modeled as a set-partitioning problem [Baldacci et al., 2002]. In this case, driver bookings correspond to the activation of facilities, and the rideshares correspond to binary assignments of customers to facilities. These three classic Operations Research problems (Pickup and delivery, Set-packing and Capacitated facility location) are NP-hard. Given that our problem generalizes these three classic problems, solving it is at least as difficult as solving these problems. Furthermore, our problem is stochastic, which further increases its difficulty.

5.3. Planning Problem and Operating Modes

We now define the planning problem and the variants considered in this paper. Section 5.3.1 introduces the general planning problem, based on which Section 5.3.2 then instantiates three problem variants corresponding to different rideshare operating modes.

5.3.1. Problem Definition

A ridesharing platform receives transportation requests for a specific time window (also referred to as the planning period), indicating the willingness of drivers and riders to engage in ridesharing with other users on the platform. When creating requests, a user specifies its origin, destination, earliest departure time, latest required arrival time at the destination, and whether it wants to participate as a driver or a rider. Based on this information, the platform plans ridesharing groups such that participants can fulfill their itineraries while traveling together to save fuel. As such, a ridesharing group is defined as a planned trip composed of one driver and one or more riders.

Driver and rider requests. Drivers are defined as individuals that possess a vehicle and have a planned itinerary consisting of an origin, a destination, an earliest departure time, and a latest arrival time at which the driver has to arrive at her destination. Drivers may participate in a rideshare with riders, i.e., a driver may pick up riders at their origin locations and drop them off at their specified destinations, as long as the specified time windows are respected. In most ridesharing systems that involve commuting or intra-city transportation, driver requests must be made and selected in a timely manner before the actual rider requests are known. As such, the matching planning problem considers driver and rider requests with itineraries that fall into the window of a specific planning period, e.g., the morning commuting period of a weekday. Specifically, this planning period spans the time period starting at the earliest departure time among all drivers and ending at the latest arrival time among all drivers. Once the driver requests for the planning period are known, a subset of these requests has to be *booked* in advance (e.g., the evening before) before rider requests are known, such that these booked drivers can be assigned to rider requests once they have occurred.

The occurrence of specific rider requests is uncertain. It is assumed that the platform is able to characterize this uncertainty through probability distributions or historical observations (e.g., rider requests observed throughout the last weekdays). The planning problem therefore resolves into a two-stage structure: in the first stage, driver requests are released and must be booked; in the second stage, rider requests are known and must be assigned to booked drivers. As mentioned above, such two-stage structure may represent a context where driver requests are made and selected in the evening. Rider requests are then gradually released until the morning commuting period of the planning day and assigned to the selected drivers.

Contract types. Given that drivers have to be selected before the actual rider requests are known, it may be possible that a booked driver is not assigned to any actual rideshare. In order to provide an incentive to drivers and encourage an expanding driver user-base, the system operator may therefore book a driver according to different *contract types*, which can be seen as a type of service agreement or booking policy: either a predefined fee is paid to a selected driver, no matter whether the driver is actually assigned to a rideshare, or a penalty is paid to drivers that have been selected, but not assigned to a rideshare. These contract types, along with their specified booking fees and penalty values, may be specified for different service characteristics. For example, the operator may decide to pay higher booking fees for drivers that are available during larger time windows. While the costs to book drivers may increase, larger availability windows also offer higher flexibility to match potential riders. As a result, the final choice of such contract types may impact several relevant performance measures considered by the system operator, such as the profitability of the rideshare, the satisfaction of drivers (reflected by the level of compensation when not matched) and the satisfaction of riders (reflected by the ability to match them).

While drivers are typically compensated for making available their vehicles, riders typically pay fees in order to be matched with the prospect of saving due to the rideshare. Incentives to participate in a rideshare are therefore monetary. Riders pay for the service, because it is cheaper than driving alone, or because they do not have access to a car or other forms of transportation. The drivers pick up riders in exchange for compensation because it is also cheaper than driving alone. Therefore, both drivers and riders are potentially saving money when ridesharing. We can characterize the benefit associated with a rideshare by, for example, the amount of distance savings that it generates for its participants. Specifically, such savings may be computed as the difference between the total rideshare distance and the individual travel distances in case each participant drives alone. However, the general problem, as well as our modeling framework are general enough to allow for benefits that are based on other performance measures. Note that we do not make any assumption on how the fees are distributed among participants. For further reading on this subject, we refer the reader to literature on, for example, Shapley values for ridesharing [Levinger et al., 2020].

The objective of the problem is to book drivers and then assign them to riders to build rideshares with maximal value. Note that this problem does not require that all rider's requests are served. In fact, this would be an unreasonable requirement when demand is uncertain. While, instead, this problem focuses on the profitability of the system, our computational experiments will allow us to study the conditions under which the proportion of satisfied rider requests is high. In the following, we will refer to this problem as the 2-Stage Stochastic Driver Matching Problem (2S-SDMP). Next, we describe how this problem can represent different operating modes of ridesharing services.

5.3.2. Different Operating Modes

The above introduced notions of contract types, booking fees and penalties allow for various ridesharing system configurations. In the following, we will focus on three specific problem variants, which represent realistic operating modes in practice and will allow us to explore the impact of the system parameters on the performance measures relevant to the operator:

- **Multiple contract types with penalties** (PEN-k): Each driver can be booked at one of k different contract types (i.e., $k = |L|$ denotes the cardinality of the set of contract types L). Booking a driver is free of cost. However, a penalty is paid if a booked driver is not assigned to a rideshare. Each contract type specifies a different driver availability window and a penalty. Assigning rider itineraries to the route of a driver results in overall savings (both in terms of used vehicles and in terms of distance), which is shared among the participants. This variant corresponds to traditional ridesharing systems, with an additional penalty mechanism to attract drivers to its user-base. For $k = 1$, this variant represents ridesharing operations where the system only offers one type of contract to its users. For $k > 1$, it represents a more flexible system that offers drivers different types of contracts.
- **Multiple contract types with booking fees** (FEE-k): Booking a driver involves a fee, but there are no additional penalties in case drivers are not assigned to rideshares. Drivers can be booked at one of k contract types. While drivers have a specific origin (i.e., a location where they will start their trip), they do not have a fixed destination. This variant may represent any context where fees are paid in advance by the platform to reserve driver capacity, for example, taxi-like sharing platforms where drivers are independent contractors that are booked in advance to transport riders. After completing the trip, they are free to take new trips or to leave the platform. Here, benefits from rideshares mostly come from the cost savings when several riders share the same taxi, as opposed to paying a separate taxi for each ride.
- **Penalties and booking fees** (PEN-FEE): Drivers can be booked on one of two contract types: either without a booking fee, but with a penalty if the driver is not assigned to a rideshare, or vice-versa (i.e., with a booking fee, but without penalties). This variant may fit a traditional ridesharing system, which has a smaller driver user-base and would like to attract more riders by providing more driver resources. If a driver is willing to be booked on either contract type, this provides an additional opportunity to optimize profits and improve the overall service level.

As mentioned earlier, a ridesharing operator is not only concerned with the profitability of the system but also with its long-term success, often reflected by its attractiveness to the driver and rider user-base (reflected by, e.g., booking fees, penalties, matching probability).

The first problem variant, PEN-k, will allow us to explore the trade-off between driver availability windows, rideshare profitability, and penalties. Large driver availability windows allow for more matching flexibility, but their attractiveness depend on the price. The second problem variant, FEE-k, will allow us to study how the system's profitability is affected by an increasing compensation of the drivers. Finally, studying the hybrid variant PEN-FEE may help understand under which penalty values are acceptable compared to the booking fees. Note that, while the here presented planning problem involves only a single time period, the problem can be used in a dynamic multi-period setting. We elaborate on the relation to the multi-period and multi-stage counterpart in Appendix A.1.

5.4. Mathematical Models

In this section, we formally describe the problem that we are studying through mathematical programming models that are general enough to encompass all problem variants discussed above (see Section 5.3.2). In Section 5.4.1, we first introduce a deterministic model, which assumes that the actual rider requests are known in advance. While this is unrealistic in practice (unless requests can be accurately predicted), the performance of the deterministic model operating under perfect information can be used as a benchmark. We then define a two-stage stochastic model in Section 5.4.2, explicitly addressing rider request uncertainty. Finally, we discuss several approximations of the two-stage model in Section 5.4.3.

5.4.1. The deterministic formulation

Let D be the set of driver requests and R be the set of rider requests, which jointly form the set of transportation requests with $D \cap R = \emptyset$. Let L be the set of contract types on which a driver can be booked. In order to be available to participate in a rideshare, a driver $i \in D$ has to be booked at some contract type $\ell \in L$, implying a booking cost (e.g., a booking fee) of f_i^ℓ. A rideshare is a route that starts at the driver's origin location, then picks up and delivers one or more riders to their destination, and ends at the driver's destination.

Each booked driver and each rider can participate in at most one rideshare. Let $\mathbb{P}(R)$ be the power set of riders R, i.e., a set containing all subsets of R. For a given driver $i \in D$ booked at contract type $\ell \in L$, let $\Omega_i^\ell \subseteq \mathbb{P}(R)$ represent the set of feasible rider subsets that may share a ride with driver i. Any rider subset $\omega \in \Omega_i^\ell$ therefore represents a possible rideshare, and we here assume that the exact route for this rideshare is the most economical one, generating a total system revenue $r_{i\omega}^\ell$. The set of feasible rideshares Ω_i^ℓ, along with the corresponding revenues, can be enumerated beforehand. If a booked driver is not assigned to any rideshare, a penalty p_i^ℓ is paid for compensation.

Let z_i^ℓ be a binary variable that takes value 1, if and only if driver $i \in D$ is booked at type $\ell \in L$. For each driver $i \in D$, type $\ell \in L$ and riders $\omega \in \Omega_i^\ell$, let $y_{i\omega}^\ell$ be a binary variable

that takes value 1, if and only if the riders in ω share a ride with driver $i \in D$, booked at type $\ell \in L$.

Further, for each rider $j \in R$, let b'_j be a binary constant equivalent to 1, if and only if rider request j occurs, i.e., rider j is available ridesharing. For a given driver $i \in D$, contract type $\ell \in L$ and rider subset $\omega \in \Omega_i^\ell$, let $a_{j\omega}$ denote a binary constant equivalent to 1, if and only if $j \in \omega$. A deterministic matching model can be written as:

$$\max \sum_{i \in D} \sum_{\ell \in L} -f_i^\ell z_i^\ell + \sum_{i \in D} \sum_{\ell \in L} \Big(\sum_{\omega \in \Omega_i^\ell} r_{i\omega}^\ell y_{i\omega}^\ell - p_i^\ell (z_i^\ell - \sum_{\omega \in \Omega_i^\ell} y_{i\omega}^\ell) \Big) \tag{5.4.1}$$

$$\sum_{\ell \in L} z_i^\ell \leq 1 \qquad \forall i \in D \tag{5.4.2}$$

$$\sum_{\omega \in \Omega_i^\ell} y_{i\omega}^\ell \leq z_i^\ell \qquad \forall i \in D, \ell \in L \tag{5.4.3}$$

$$\sum_{i \in D} \sum_{\ell \in L} \sum_{\omega \in \Omega_i^\ell} a_{j\omega} y_{i\omega}^\ell \leq b'_j \qquad \forall j \in R \tag{5.4.4}$$

$$z_i^\ell \in \{0, 1\} \qquad \forall i \in D, \ell \in L \tag{5.4.5}$$

$$y_{i\omega}^\ell \in \{0, 1\} \qquad \forall i \in D, \ell \in L, \omega \in \Omega_i^\ell. \tag{5.4.6}$$

Objective (5.4.1) maximizes the total system profit, given by the difference between the revenue generated by the rideshares, the fees for booked drivers, and the penalties for drivers that have been booked, but have not been assigned to any rideshare. Constraints (5.4.2) ensure that drivers cannot be booked on more than one type. Constraints (5.4.3) ensure that a driver can serve at most one rideshare, and only if it is booked. Finally, Constraints (5.4.4) ensure that a released rider can rideshare with at most one driver. This model has a set-packing structure with the addition of driver booking variables z_i^ℓ. As such, the problem is NP-hard.

Note that, instead of using Constraints (5.4.4) no matter the outcome of constants b'_j, the model can be slightly modified such that these constraints are defined exclusively for the set of released riders. However, when using model (5.4.1)–(5.4.6), the presolving step of modern mixed-integer programming solvers will remove constraints where $b'_j = 0$ and adjust the bounds of the associated y-variables accordingly. To keep the model notation consistent with the stochastic models that we will introduce next, we have decided to explicitly include the constants $b'_j, \forall j \in R$ in the deterministic model.

Further note that this problem formulation makes no assumption on how route feasibility is defined, but is flexible enough to accommodate many feasibility requirements while enumerating Ω_i^ℓ. The definition of feasibility may depend on the ridesharing system and the objectives of the operator. For the sake of our numerical analyses, we will focus on

two intuitive feasibility criteria: vehicle capacity and time-window feasibility. The latter is even more important if the planning period spans a long time (e.g., several hours). We give further details on route feasibility when explaining our route enumeration procedure in Section 5.5.1.4.

While the formulation above is based on set-packing, the problem can theoretically also be modeled as a compact arc-flow formulation [see, e.g., Baldacci et al., 2004]. Such arc-flow formulation would require to explicitly model all driver routes, including their feasibility requirements, on an (almost) complete graph. As such, the problem can be seen as a stochastic extension of the Pickup and Delivery Problem with Time Windows (see Parragh et al. [2008a] and Parragh et al. [2008b] for extensive surveys), but with an excessively large number of vehicles. This problem generalizes the vehicle routing problem, which is known to be NP-hard [Lenstra and Kan, 1981]. The corresponding formulations would therefore likely be intractable for general-purpose MIP solvers and the explicit representation of route constraints (e.g. time window constraints) may lead to a formulation that provides worse bounds than a set-packing formulation [see, e.g., Costa et al., 2019]. Next to the issue of solving such formulations, they also limit the possible definitions for rideshare feasibility and revenue functions. While a set-packing formulation allows for the application of complex business rules and feasibility criteria during the route enumeration process, an arc-flow formulation is limited to linear additive rules.

5.4.2. The two-stage stochastic formulation

We now no longer assume that we have an a priori knowledge of whether a rider is released or not. Instead, we assume that we can characterize the uncertainty on the release of riders through probability distributions or historical observations. Thus, instead of having binary constants b'_j for the release of rider $j \in R$ as in the deterministic model, we represent this uncertainty through random binary variables $\tilde{b}_j, \forall j \in R$.

We are concerned with finding a set of booked drivers that allows for maximum profit as averaged over all (or most) realizations of $\tilde{\mathbf{b}}$. To this end, we formulate a two-stage stochastic problem that maximizes an expected second-stage value function. In the first stage, the model decides which drivers to book. Based on these decisions, one second-stage problem is solved for each realization of $\tilde{\mathbf{b}}$, where booked drivers and released riders are assigned to rideshares. The two-stage stochastic model can be written as follows:

$$\max \sum_{i \in D} \sum_{\ell \in L} -f_i^\ell z_i^\ell + \mathscr{Q}(\mathbf{z}) \tag{5.4.7}$$

$$\sum_{\ell \in L} z_i^\ell \leq 1 \qquad \forall i \in D \tag{5.4.8}$$

$$z_i^\ell \in \{\, 0, 1 \,\} \qquad \forall i \in D, \ell \in L, \tag{5.4.9}$$

where

$$\mathscr{Q}(\mathbf{z}) = \mathbb{E}_{\tilde{\mathbf{b}}}[Q(\mathbf{z}, \mathbf{b})]$$

is the expected second-stage value function. Objective (5.4.7) maximizes the expected profit of building rideshares in the second stage, minus the costs associated with driver booking in the first stage. For a given realization (e.g. a scenario) $\mathbf{b}$ of $\tilde{\mathbf{b}}$, the second-stage value function $Q(\mathbf{z}, \mathbf{b})$ is defined as:

$$Q(\mathbf{z}, \mathbf{b}) \coloneqq \max \sum_{i \in D} \sum_{\ell \in L} \Big(\sum_{\omega \in \Omega_i^\ell} r_{i\omega}^\ell y_{i\omega}^\ell - p_i^\ell (z_i^\ell - \sum_{\omega \in \Omega_i^\ell} y_{i\omega}^\ell) \Big) \tag{5.4.10}$$

$$\sum_{\omega \in \Omega_i^\ell} y_{i\omega}^\ell \leq z_i^\ell \qquad \forall i \in D, \ell \in L \tag{5.4.11}$$

$$\sum_{i \in D} \sum_{\ell \in L} \sum_{\omega \in \Omega_i^\ell} a_{j\omega} y_{i\omega}^\ell \leq b_j \qquad \forall j \in R \tag{5.4.12}$$

$$y_{i\omega}^\ell \in \{0, 1\} \qquad \forall i \in D, \ell \in L, \omega \in \Omega_i^\ell. \tag{5.4.13}$$

Objective (5.4.10) maximizes the profit of the selected rideshares, minus the costs associated with booked drivers that were not assigned to any rideshare. The second-stage problem is a set-packing problem and, as such, is NP-hard itself. Solving the two-stage model for all realizations of $\tilde{\mathbf{b}}$ would require solving a formulation with an exponential number of variables, which would be computationally intractable for most probability distributions. For that reason, we now introduce three formulations that approximate the second-stage value function.

5.4.3. Approximations for the second-stage value function

The two-stage problem may be challenging to solve, as the number of all possible rideshares and the number of all possible realizations for $\tilde{\mathbf{b}}$ are both exponential on the number of riders, which can lead to a very large model. To address this, we propose three approximations for the second-stage value function.

5.4.3.1. The sample average approximation problem. We first introduce a sample average approximation (SAA) for the two-stage stochastic formulation [Birge and Louveaux, 2011]. To this end, we generate a set of scenarios $S = \{b^1, \ldots, b^{|S|}\}$ randomly sampled from $\tilde{\mathbf{b}}$, where $|S| \ll 2^{|R|}$. Alternatively, S can be obtained from historical observations. These scenarios are then used to approximate the second-stage value function $\mathscr{Q}(\mathbf{z})$ by

$$\mathscr{Q}_{\text{SAA}}(\mathbf{z}) = \sum_{s=1}^{|S|} \frac{1}{|S|} Q(\mathbf{z}, \mathbf{b}^s).$$

The corresponding model is solved under the same set of Constraints (5.4.8) and (5.4.9). Note that, if only a single scenario is used to build the SAA (i.e., $|S| = 1$), then the model is structurally identical to the deterministic model (5.4.1)–(5.4.6).

5.4.3.2. The sample average approximation problem with relaxed second stage. If the integrality gap of the previously defined SAA model is small, then relaxations of the second-stage function may still generate high-quality integer solutions. The relaxation of integrality constraints in stochastic programming has been applied to other problems. For example, Ahmed et al. [2003] study a multi-stage stochastic programming model for capacity expansion under uncertainty. The authors relax the integrality constraints and then use a heuristic to transform the continuous relaxation solution into an integer solution.

We here pursue a similar approach. We relax the integrality constraints (5.4.13) of the second-stage problems (i.e., $y^{\ell}_{i\omega} \in [0,1], \forall i \in D, \ell \in L, \omega \in \Omega^{\ell}_i$). First-stage decisions, however, remain binary. We refer to this relaxation as the SAA-R. Such a relaxation is likely to dramatically reduce the computing time required to solve the model, while still maintaining first-stage binary decisions. Such formulation would also allow for the use of a Benders decomposition algorithm to solve the model, if needed. In Section 5.5.2.2, we empirically evaluate if the second-stage value function approximation of the SAA-R is sufficiently strong to replace the SAA model, and provide insights on the performance of the SAA-R.

5.4.3.3. The expected value problem. When computational resources are limited and solutions are required in nearly real-time, it may be desirable to use an approximation of the second-stage value function that does not depend on multiple scenarios. Instead of considering the scenarios $S = \{b^1, \ldots, b^{|S|}\}$ explicitly in the formulation, we may use a formulation with a single sample mean realization:

$$\bar{\mathbf{b}} = \sum_{s=1}^{|S|} \frac{1}{|S|} \mathbf{b}^s.$$

The second-stage value function $\mathscr{Q}(\mathbf{z})$ is then replaced by

$$\mathscr{Q}_{\text{EVP}}(\mathbf{z}) = Q(\mathbf{z}, \bar{\mathbf{b}}).$$

This problem is commonly referred to as the expected value problem (EVP) [Birge and Louveaux, 2011]. Similarly to Homsi et al. [2020], the EVP has binary variables on the left-hand side and may have fractional coefficients on the right-hand side of Constraints (5.4.12). This may render the formulation too restrictive or even infeasible, which therefore requires us to also relax the integrality constraints of the y variables.

5.5. Computational Study

In this section, we conduct extensive computational experiments on the proposed models. To provide insights on a wide range of ridesharing operating modes (see Section 5.3.2), we

evaluate four problem variants: PEN-1, PEN-2, FEE-2 and PEN-FEE. We first introduce the set of benchmark instances used throughout our experiments in Section 5.5.1. We then evaluate the proposed models in Section 5.5.2, focusing on three types of key insights: the benefits of using one problem variant over another; an understanding of why certain models out- or underperform in specific settings; and, the impact of the input parameter values on the expected user satisfaction and system profitability.

5.5.1. Benchmark instances

We generate a benchmark set of 6 base instances following the generation process of Homsi et al. [2021], based on data from a Montreal ridesharing company. Then, we expand this base benchmark set by varying the parameters of these instances to assess the impact on the platform profit and user satisfaction, while keeping the same network topology. Namely, we vary the values of the booking fees, revenues, and penalties. As such, we have 39 instances for variant PEN-1, 6 instances for variant PEN-2, 6 instances for variant FEE-k, and 6 instances for variant PEN-FEE. An instance is composed of:

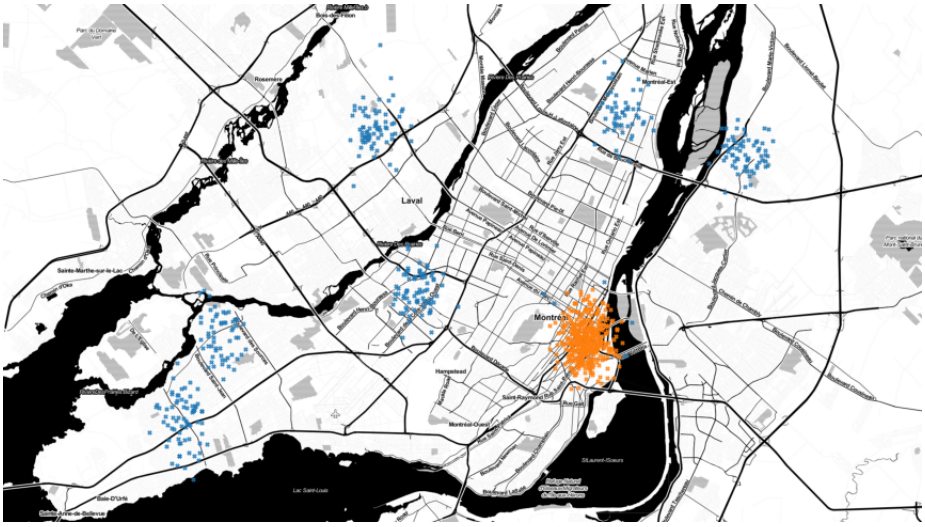
- a set D of 200 drivers, each with a specified origin, destination, and latest arrival time;
- a set R of 200 riders, each with a specified origin, destination, latest arrival time, as well as the probability that its request occurs;
- a set of booking types L, the booking fees coefficients $f_i^\ell, \forall i \in D, \ell \in L$, and the penalty coefficients $p_i^\ell, \forall i \in D, \ell \in L$;
- the availability window scaling parameters τ_{R} and $\tau_{\text{D}}^\ell, \forall \ell \in L, d \in D$;
- a set of feasible rideshares Ω_i^ℓ for each driver and booking type, that is constructed based on τ_{R} and $\tau_{\text{D}}^\ell, \forall \ell \in L, i \in D$;
- the revenues $r_{i\omega}^\ell$ associated with each rideshare $\omega \in \Omega_i^\ell, \forall \ell \in L, i \in D$;
- a ground truth set of 2000 scenarios, each containing a realization of $\tilde{\mathbf{b}}$, to validate the profit of the first-stage decisions generated by our models.

Note that the availability window scaling parameters mentioned above adjust the size of the time window that drivers are available for ridesharing. Further details are given in Section 5.5.1.3. Next, we describe how these instances are generated.

5.5.1.1. Origin and destination locations. Inspired by Homsi et al. [2021], we randomly generate points based on seven different demand clusters around the metropolitan region of Montreal (see Figure 5.2). These demand clusters correspond to residential and downtown regions. Due to time-window restrictions and the requirement to generate rideshares with positive distance savings, many of the feasible rideshares satisfying these conditions have origin or destination requests within the same local region. Thus, such complex instances can be often separated into multiple instances and solved individually to provide close to

optimal approximations for the larger instance. We therefore focus our study on the case where requests originate in different residential regions and have destinations in the downtown region. This represents, for example, the case of a daily morning commute.

Fig. 5.2. The geographical region considered around the metropolitan region of Montreal, along with the clusters for origins and destinations. Orange points correspond to the downtown region.



5.5.1.2. Probability of request occurrence. The probability that rider $j \in R$ submits a rider request for the upcoming planning (i.e., the probability that the random variable $\tilde{b}_j$ takes value 1) is a real number sampled uniformly from $[0, 1]$. We therefore sample realizations of $\tilde{\mathbf{b}}$ to generate independent request scenarios used in our models.

5.5.1.3. Latest arrival times and availability windows. We randomly generate the latest required arrival times for each request $i \in D \cup R$. Its latest arrival time t^i_{A} is a real number sampled uniformly from $[8, 10]$, representing the time range from 8am to 10am. The latest arrival times represent, for example, the latest time individuals may want to arrive at work. We hence assume that individuals are not willing to arrive later than the specified latest arrival time. Instead, they are willing to leave home earlier to accommodate for the delays involved in ridesharing. Thus, based on the latest arrival time, we calculate the earliest departure time t^i_{D} based on the travel time between an individual's origin $u_i \in \mathbb{R}^2$ and destination $v_i \in \mathbb{R}^2$ (in longitude and latitude), as follows:

$$t^i_{\text{D}} = t^i_{\text{A}} - \tau \cdot \frac{d(u_i, v_i)}{s}, \forall i \in D \cup R,$$

where $d(u_i, v_i)$ is the approximate routing distance in kilometers between u_i and v_i, $s = 40\ km/h$ is a constant speed assumed for all participants, and τ is an availability window scaling factor. Therefore, $d(u_i, v_i)/s$ represents the estimated individual travel time of user i if she traveled alone. When referring to a driver at booking type ℓ, we use the notation τ^ℓ_{D}.

When referring to riders, we use the notation τ_{R}. When ridesharing, we assume that riders are willing to accept travel times up to 30% longer than individual travel times, i.e., their earliest departure time is computed based on $\tau_{\text{R}} = 1.3$. As in Homsi et al. [2021], we estimate the true routing distance between two points with the aid of a regression model trained with OSRM routing data [Luxen and Vetter, 2011], resulting in the following formula:

$$d(u_i, v_i) = 1.57 + 81.91 \cdot \bar{d}(u_i, v_i),$$

where $\bar{d}(u_i, v_i)$ is the Manhattan distance between u_i and v_i.

In problem variant FEE-k, drivers have no fixed destination, and their time windows simply represent a time block during which they are available to provide transportation. For each type-1 driver $i \in D$, the earliest departure time t^i_{D} is set to as 45 minutes before t^i_{A} (i.e., $t^i_{\text{A}} = t^i_{\text{D}} + 45/60$). For a type-2 driver $i \in D$, t^i_{D} is set to one hour before t^i_{A} (i.e., $t^i_{\text{A}} = t^i_{\text{D}} + 1$). We uniformly sample t^i_{D} from the interval $[7, 9]$. When calculating the distance savings, we disregard the arc between the driver's origin and the first pick-up. Note, however, that this distance is still relevant to determine whether a rideshare is time-feasible or not.

5.5.1.4. Route enumeration and route feasibility. We a priori enumerate feasible ridesharing routes with a depth-first search algorithm that incrementally builds routes. This algorithm (implemented in C++) evaluates partial routes with different sizes, and extends routes by adding new riders. The extension of a route stops as soon as the route is no longer time-window feasible (i.e., when it violates the earliest departure time and latest arrival time of participants). Enumerating all time-feasible routes is an exponential procedure and a naive implementation may result in impractically large computing times. It is therefore important to prune ridesharing routes that are not time-window feasible as early as possible, considering the specified time-windows of both the driver and the riders. Specifically, at a certain node (either a pickup or a delivery) that is being evaluated to extend a partial route, we prune our route extension if at least one of the two criteria below is true:

- If the current node is a delivery and the total travel time of the partial route exceeds the latest arrival time for the delivery;
- If the current node is a pickup and the total travel time of the partial route plus the travel time required to go directly from that node's origin to its destination exceeds the latest arrival time for the delivery.

To further speed up the enumeration, we restrict the set of possible routes to those that are user-friendly and that make sense in practice for ridesharing. Specifically, we limit each route to at most three riders, which accurately represents the capacity of most vehicles that have ridesharing-related availability window restrictions. As all destinations are in the downtown region, it is unlikely that routes with additional pick-ups after deliveries would be more advantageous than routes with all pick-ups before all deliveries. We therefore also limit

the enumeration to routes that start dropping off riders to their destinations only after all riders have been picked up from their origins. Our depth-first search algorithm is exact and generates all routes that satisfy the criteria above. These criteria can be relaxed or modified according to the needs of the ridesharing operator. A pseudo-code for the depth-first search route enumeration algorithm is outlined in Appendix A.2.

Table 5.1 shows average statistics for the enumeration procedure, grouped by problem variant. The first columns show the problem variant and the average number of routes enumerated for each problem variant. The last column shows the maximum CPU time in seconds among all problem instances required to enumerate these routes.

Table 5.1. Statistics for the route enumeration procedure by problem variant.

Variant	Routes	Routes (1 rider)	Routes (2 riders)	Routes (3 riders)	max T (sec)
PEN-1	2,410	1,177	1,192	41	0.75
PEN-2	3,177	1,947	1,211	19	0.77
FEE-2	9,862	3,823	5,853	186	11.19
PEN-FEE	1,996	1,518	478	0	0.65

Most problem variants can be enumerated in less than a second. For variant FEE-2, the enumeration time is the largest (11.19 seconds) due to the higher level of flexibility that drivers have within this problem variant (i.e., fixed and large availability windows, and no final destination). Experiments naively enumerating the trip routes without the second pruning criteria have shown that the maximum enumeration time increases from 11.19 seconds to 3,450 seconds, which highlights the importance of using efficient pruning criteria. As such, the enumeration procedure is sufficiently fast to be used even in real-time planning. While we expect that the enumeration procedure scales well even if significantly larger instances were to be considered, we here outline two potential approaches to further reduce the computing time: First, the depth-first search could be easily parallelized to reduce the enumeration time. Second, the procedure may be split into two parts: the enumeration of rider routes and the matching of such routes to drivers. The former is based on the available rider-request scenarios and therefore only needs to be updated once more refined probabilities (i.e., more scenarios) are available (e.g., once per month). The latter has to be carried out at each planning, but has a computational complexity linear in the number of drivers. As such, it is likely to be extremely fast.

5.5.1.5. The revenue of a rideshare. We assume that the revenue $r^{\ell}_{i\omega}$ the system operator earns for each rideshare $\ell \in L, i \in D, \omega \in \Omega^{\ell}_{i}$ is computed as a share of the total distance savings $s_{i\omega}$ generated by the rideshare. A scaling parameter $\gamma^{\ell}_{\mathrm{R}}$ converts the total distance savings into a revenue value allocated to the ridesharing operator:

$$r^{\ell}_{i\omega} = \gamma^{\ell}_{\mathrm{R}} \cdot s_{i\omega}.$$

This allows us to adjust the value of a rideshare individually for each driver booking type.

5.5.1.6. Booking penalties. We define the booking penalties $p_i^\ell, \forall i \in D, \ell \in L$ as the product of a penalty scaling parameter γ_{P}^ℓ and the distance between driver origin and destination $d(u_i, v_i)$, as below:

$$p_i^\ell = \gamma_{\text{P}}^\ell \cdot d(u_i, v_i).$$

We can therefore represent situations where the system subsidizes a proportion of the driver's journey in case it is unable to find riders to form a rideshare. As penalties are indexed by driver and booking type, more complex penalty functions could be used without changing the structure of our models. For example, the penalty scaling parameter could also be driver-dependent.

5.5.1.7. Parameters used for different problem variants. We now describe the values of the parameters that are specific to each variant:

- **Single contract type with penalties** (PEN-1): We consider only one level of booking per driver (i.e., $L = \{1\}$) and no cost associated to driver booking (i.e., $f_i^1 = 0, \forall i \in D$). Studying this variant, we aim at understanding the benefits of increasing the size of the availability window, which results in a trade-off between profitability and user-friendliness. To this end, we evaluate instances with different driver availability windows by varying $\tau_{\text{D}}^1 \in \{1.3, 1.4, 1.5\}$. Furthermore, we test the following values for the revenue and penalty scaling coefficients (γ_{R}^1 and γ_{P}^1):
  (1) We fix $\gamma_{\text{R}}^1 = 0.3$, and vary $\gamma_{\text{P}}^1 \in \{0.1, 0.3, 0.5, 0.7, 0.9, 1.1, 1.3, 1.5\}$;
  (2) We fix $\gamma_{\text{P}}^1 = 0.3$, and vary $\gamma_{\text{R}}^1 \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$.
- **Multiple contract types with penalties** (PEN-2): We consider two booking types (i.e., $L = \{1, 2\}$) and no cost associated with driver booking (i.e., $f_i^1 = 0, \forall i \in D$). For type-1 drivers, we assume that $\tau_{\text{D}}^1 = 1.3$, $\gamma_{\text{R}}^1 = 0.2$ and $\gamma_{\text{P}}^1 = 0.3$. For type-2 drivers, we assume that $\tau_{\text{D}}^2 = 1.4$, $\gamma_{\text{R}}^2 = 0.2$ and vary $\gamma_{\text{P}}^2 \in \{0.5, 1, 2, 4, 8, 16\}$. Studying this variant, we aim at understanding the trade-off between availability window size and penalties.
- **Multiple contract types with booking fees** (FEE-2): We consider two booking levels. All booked drivers are paid a fee in advance, and there are no penalties if they are not matched in the second stage (i.e., $\gamma_{\text{P}}^\ell = 0, \forall \ell \in L$). For type-1 drivers, we fix $\gamma_{\text{R}}^1 = 0.3$ and $f_i^1 = 1, \forall i \in D$. For type-2 drivers, we fix $\gamma_{\text{R}}^2 = 0.3$ and vary the booking fees $f_i^2 \in \{1.5, 2, 2.5, 3, 3.5, 4\}, \forall i \in D$. Type-1 drivers are available for 45 minutes, and type-2 drivers are available for 1 hour. We study this variant to explore which booking fees can be justified when increasing the availability window.
- **Penalties and booking fees** (PEN-FEE): The operator has the choice of selecting a driver either with a booking fee, or with a penalty as in variant PEN-2. Both types have the same availability window $\tau_{\text{D}}^1 = \tau_{\text{D}}^2 = 1.3$ and revenue share $\gamma_{\text{R}}^1 = \gamma_{\text{R}}^2 = 0.3$.

Type-1 drivers can be booked at no cost (i.e., $f_i^1 = 0, \forall i \in D$), but penalties are paid if the booked drivers are not assigned to any rideshare on the second stage. To this end, we vary $\gamma_{\text{P}}^1 \in \{ 0.5, 1, 1.5, 2, 2.5, 3 \}$. Type-2 drivers have no penalties (i.e., $\gamma_{\text{P}}^2 = 0$), but a fee $f_i^2 = 1, \forall i \in D$ is paid to book them. We study this variant in order to gain an understanding of the interplay between the use of the two different driver booking types.

5.5.2. Computational results

In this section, we evaluate the solutions provided by the proposed models on our benchmark instances. All experiments have been executed on a machine with an AMD 3970X processor and 256 GB of memory. Models have been solved using CPLEX version 22.1. We limited CPLEX to one thread, set its execution time limit to one hour, and used Python to build all models.

5.5.2.1. Validation of first-stage decisions. To estimate the unbiased performance of the first-stage decisions (i.e., the values of the $\mathbf{z}$ variables) generated by each model, we evaluate these decisions on the ground truth validation set with 2,000 samples for $\tilde{\mathbf{b}}$. Let $\tilde{f}(\mathbf{z})$ be a model's optimal objective function value over these validation samples. We compare $\tilde{f}(\mathbf{z})$ against the objective function value of the *wait-and-see* problem [Madansky, 1960], which consists of solving several independent single-scenario models, one for each sample of $\tilde{\mathbf{b}}$. First-stage decisions are therefore tailored to each sample of $\tilde{\mathbf{b}}$, allowing us to obtain the best possible profit in case the planner has the ability to accurately predict the future realization of $\tilde{\mathbf{b}}$. While this is unrealistic in practice, the wait-and-see solution can be used as an upper bound to evaluate the quality of the solutions obtained from the other models. The wait-and-see objective f_{WS} is the average of the objective function values of each independent problem, and is an upper bound for $\tilde{f}(\mathbf{z})$. Thus, to estimate the quality of the first-stage solutions z we calculate the relative gap between $\tilde{f}(\mathbf{z})$ and f_{WS} as:

$$gap_{\text{WS}} = \frac{f_{\text{WS}} - \tilde{f}(\mathbf{z})}{f_{\text{WS}}}.$$

It is interesting to note that, as the first-stage decisions of the wait-and-see problem are specific to each scenario and are aware of the actual rider requests, the wait-and-see solution will never book drivers that are not matched in the second-stage, given that this would incur unnecessary (and sub-optimal) penalties or booking fees.

5.5.2.2. Performance for different models on all problem variants. We now analyze the difficulty of solving the various models for each of the problem variants. Specifically, for each of the four problem variants, we evaluate the benefits of using a scenario-based approach such as the SAA over a simpler model such as the EVP. We also explore the

tradeoff between representing the second-stage problem of the SAA formulation as a binary set-packing problem, or as a relaxation.

Table 5.2 summarizes the results averaged over the problem instances for each of the four problem variants (PEN-1, PEN-2, FEE-2, PEN-FEE) and each of the three models: the conventional SAA, the SAA with relaxed second-stage (SAA-R) and the EVP. Column "$|S|$" indicates the number of scenarios used in each model. Column "$\text{profit}_{\text{GT}}$" represents the average profit associated with the first-stage decisions and evaluated on the the ground truth validation set. For each instance, the first-stage decisions of all models are evaluated on the same validation set. Column "gap_{WS} (%)" indicates the average percentage gap of the objective function value to the optimal objective function value of the wait-and-see solution. Column "T (sec)" represents the average CPU time (in seconds) required by CPLEX to solve the models. To assess the potential benefits of having more precise information on the occurrence of riders, we test our models with different numbers of scenarios (specifically, with 25, 50, 100 and 200 scenarios).

Table 5.2. Results for models on all problem variants.

model	$\|S\|$	$\text{profit}_{\text{GT}}$	gap_{WS} (%)	T (sec)	model	$\|S\|$	$\text{profit}_{\text{GT}}$	gap_{WS} (%)	T (sec)
PEN-1					PEN-2				
SAA	25	517.44	8.73	1.96	SAA	25	234.33	20.50	43.13
	50	518.70	8.46	7.37		50	255.66	13.29	53.76
	100	519.66	8.25	33.21		100	258.06	12.48	530.64
	200	519.95	8.16	164.08		200	262.47	10.99	1,618.65
SAA-R	25	517.44	8.73	0.75	SAA-R	25	237.74	19.35	7.99
	50	518.74	8.46	2.53		50	254.19	13.79	8.02
	100	519.65	8.25	9.54		100	258.20	12.43	45.14
	200	519.90	8.17	44.62		200	262.72	10.91	356.84
EVP	25	500.62	12.89	0.49	EVP	25	78.91	73.00	1.85
	50	499.60	13.07	0.87		50	92.66	68.36	1.83
	100	506.12	11.48	0.36		100	86.45	70.46	1.26
	200	506.50	11.42	0.53		200	82.60	71.76	1.77
FEE-2					PEN-FEE				
SAA	25	113.22	17.37	1,287.37	SAA	25	355.92	12.61	0.55
	50	113.19	17.39	2,893.44		50	355.95	12.60	3.45
	100	113.84	16.88	3,600.00		100	359.28	11.78	27.10
	200	113.78	16.92	3,600.00		200	360.74	11.43	449.30
SAA-R	25	112.91	17.61	450.89	SAA-R	25	355.90	12.61	0.43
	50	113.18	17.36	2,068.08		50	355.87	12.62	1.49
	100	113.63	17.02	3,459.31		100	359.44	11.74	9.86
	200	113.70	16.98	3,600.00		200	360.69	11.44	16.71
EVP	25	110.72	19.21	2,657.70	EVP	25	238.56	41.34	0.10
	50	110.83	19.05	3,600.00		50	247.95	39.04	0.63
	100	111.33	18.70	2,383.40		100	270.84	33.44	0.97
	200	110.55	19.24	2,016.35		200	270.54	33.51	0.94

The results highlight the benefits of using a stochastic model: for variant PEN-1 at 200 scenarios, the gaps of the SAA and the SAA-R are, on average, about 3 percentage points smaller than the gap of the EVP (8.16% and 8.17% versus 11.42%). Moreover, the results

show that increasing the number of scenarios reduces the gap for all models, except for the EVP at problem variant PEN-2, where the average gaps are large (68.36% to 73.00%) and do not appear to improve consistently with the number of scenarios. In contrast, for that same problem variant, the average gaps of the SAA and SAA-R at 200 scenarios are 10.99% and 10.91%, respectively.

On the SAA-R performance. The results show that relaxing the second-stage problem for the SAA formulation can be an efficient strategy: for example, for variant PEN-2 with 200 scenarios, the model can be solved in about 20% of the original computing time (from 1,618.65 seconds to 356.84 seconds), while the gaps are similar to those of the original SAA. We observe similar reductions in computing time and similar gaps for all other problem variants, except for variant FEE-k where both models exceed the computing time limit, while the gaps are similar. This may suggest two further advantages associated with relaxing the second-stage problem: as the relaxed second-stage problems are easier to solve, we may be able to better approximate the second-stage value function by increasing the number of samples. Further, in the case where the original SAA formulation becomes too hard to solve, relaxing the second-stage allows us to apply Benders decomposition (which requires continuous subproblems).

To better understand the different levels of computational effort required to solve the SAA and the SAA-R, we conduct further experiments where we compare the optimal solution (or best known solution) value of each model with the solution value of its linear programming relaxation. Average results for the SAA and SAA-R with 200 scenarios are shown in Table 5.3. Column "profit_{200}" indicates the profit (objective function value) of the models over the 200 scenarios used to find the first-stage decisions. Column "int. gap (%)" shows the integrality gap between the optimal (or best known) solution and the linear programming relaxation. Column "opt. gap (%)" indicates the optimality gap as reported by CPLEX.

Table 5.3. Integrality and optimality gaps for the SAA and the SAA-R with 200 scenarios.

variant	model	profit_{200}	int. gap (%)	opt. gap (%)	T (sec)
PEN-1	SAA	521.05	0.15	0.00	164.08
	SAA-R	521.69	0.01	0.00	44.62
PEN-2	SAA	268.91	0.15	0.01	1,618.65
	SAA-R	269.23	0.03	0.01	356.84
FEE-2	SAA	115.77	4.12	1.18	3,600.00
	SAA-R	119.30	1.26	0.71	3,600.00
PEN-FEE	SAA	364.62	0.12	0.00	449.30
	SAA-R	365.04	0.00	0.00	16.71

As expected, the SAA-R model has smaller integrality gaps (often zero or close to zero) than the SAA model, which results in faster solution times. For cases where optimality is not proven (FEE-2), we observe that the optimality gaps of the SAA-R are still smaller than those of the SAA.

As the SAA-R is a relaxation of the SAA, the optimal value of the objective function of the SAA-R will be an upper bound to the optimal SAA objective. Column profit_{200} shows that this upper bound is relatively tight. For all variants except for FEE-2, the SAA-R objectives have a gap of at most 0.12% to the SAA objective. For variant FEE-2, none of the models have been able to prove optimality, and the average gap between the best known solutions of the SAA-R and the SAA is 2.96% (115.77 and 119.30). This gap could be further improved by strengthening the relaxed second-stage set-packing formulation through valid inequalities [see, e.g. Borndörfer, 1998].

In an effort to gain an understanding of why the SAA-R provides such tight approximations for the second-stage value function, we inspect the proportion of the second-stage y variables that take value 1, as opposed to variables that take a fractional value greater than zero. Upon further inspection of a PEN-1, a PEN-2 and a FEE-2 instance solved by the SAA-R with 200 scenarios, we have observed that y variables with value 1 represent about 91% (for PEN-1), 77% (for PEN-2) and 50% (for FEE-2) of the total number of variables with non-zero values (i.e., variables with value 1, and variables with non-zero fractional values). The fact that more SAA-R variables are fractional for variant FEE-2 explains the larger gap between the SAA-R and SAA objectives (2.96%, as opposed to 0.12% for other problem variants). We furthermore observed that the ratio between rideshares (non-zero y variables) and activated drivers is 1.01 (for PEN-1), 1.12 (for PEN-2) and 1.21 (for FEE-2). These ratios are significantly smaller than those in the EVP solutions (3.03, 3.22 and 4.92, respectively). This suggests that, even if the second-stage variables are relaxed, a significant proportion of the variables assume integer values, which leads to a better approximation of the second-stage value function, and explains the good performance of the SAA-R. Given the high solution quality induced by the SAA-R model on the ground truth and its small integrality and optimality gaps when compared to the SAA model, we restrict the following experiments to the SAA-R model using 200 scenarios.

On the EVP performance. Despite its short computing times, the EVP model does not reliably provide a reasonable approximation to the original SAA solution. This becomes evident, in particular, for variants PEN-2 and PEN-FEE, where its gaps are significantly worse than those of the other models. Instead, it may be more effective to solve the other models with a smaller number of scenarios, which likely generates solutions of higher quality and CPU times comparable to the EVP. The bad performance of the EVP is due to the combination of fractional right-hand-side coefficients (for the mean sample realization $\bar{\mathbf{b}}$ of $\tilde{\mathbf{b}}$) and continuous second-stage variables, as $b_j \in [0,1], \forall j \in R$ are likely to be fractional, Constraints (5.4.12) will likely limit the creation of rideshares to fractional values (i.e., $y^{\ell}_{i\omega} < 1$). As a consequence, the EVP solution may partially open multiple rideshares for the same driver, such that the cost of the booking fee is offset. Indeed, upon further inspection of some instances, we have observed that drivers are partially assigned to, on average, 3.03 different rideshares for PEN-1,

3.22 for PEN-2, and 4.92 for FEE-2. The first-stage decisions are therefore not optimized under the assumption that drivers can be assigned to at most one rideshare, and consequently they may not be as profitable when evaluated on the ground truth. The performance of the EVP for variant PEN-2 is further discussed in Section 5.5.2.5.

5.5.2.3. PEN-1: impact of driver availability windows. In this and the following sections, we explore how the platform profit and user satisfaction are impacted under the different problem variants and parameter settings. With the overall objective to gain an understanding which parameter values are desirable such that both the platform and the participants benefit, we will examine several performance indicators: the platform profit, the proportion of booked drivers, the proportion of failed bookings, the proportion of satisfied riders, and the cost of the rideshare compared to driving alone.

We first focus on the impact of the driver availability window. We limit these experiments to variant PEN-1 and test availability window scaling coefficients $\tau_{\text{D}}^1 \in \{ 1.3, 1.4, 1.5 \}$, along with different configurations of the revenue share scaling parameter γ_{R}^1 and penalties scaling parameter γ_{P}^1, namely:

- we fix the penalty coefficients to $\gamma_{\text{P}}^1 = 0.3$, and vary the share of savings collected by the ridesharing platform $\gamma_{\text{R}}^1 \in \{ 0.1, 0.3, 0.5, 0.7, 0.9 \}$, representing contexts where the platform passes almost all revenue to the users ($\gamma_{\text{R}}^1 = 0.1$) up to contexts where the platform keeps most of the revenue, and only little for its users ($\gamma_{\text{R}}^1 = 0.9$);
- we fix the coefficient for the share of savings collected by the ridesharing platform to $\gamma_{\text{R}}^1 = 0.3$, and vary the penalties $\gamma_{\text{P}}^1 \in \{ 0.1, 0.3, 0.5, 0.7, 0.9, 1.1, 1.3, 1.5 \}$. This represents contexts where the platform subsidizes only 10% of a booked driver's trip in case it is not assigned to a rideshare ($\gamma_{\text{P}}^1 = 0.1$) up to contexts where drivers are highly compensated through penalties if they are not assigned to any rideshare ($1.1 \leq \gamma_{\text{P}}^1 \leq 1.5$).

We expect larger availability windows to contribute to a higher number of successful matches, and hence to more profitable solutions. However, larger availability windows may generate inconveniences for drivers. Operators are therefore interested in identifying a reasonable value for τ_{D}^1 that generates sufficient profit, while causing minimal inconveniences.

The results are summarized in Table 5.4. Column "book (%)" indicates the average proportion of booked driver over all drivers. Column "failed (%)" shows the average proportion of failed bookings (i.e., drivers that were booked but not assigned to any rideshare) over all booked drivers. Column "sat. riders (%)" represents the average proportion of satisfied rider requests (i.e., riders that have been assigned to a rideshare) over all released riders. Column "cost (%)" shows the percent average trip cost relative to individual trips (i.e., the ratio of shared distances plus platform cut over the sum of individual distances).

Table 5.4. SAA-R results for different values of τ_{D}^1 on variant PEN-1 (averaged over fixed $\gamma_{\text{P}}^1 = 0.3$ with $\gamma_{\text{R}}^1 \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$; and fixed $\gamma_{\text{R}}^1 = 0.3$ with $\gamma_{\text{P}}^1 \in \{0.1, 0.3, 0.5, 0.7, 0.9, 1.1, 1.3, 1.5\}$).

τ_{D}^1	$\text{profit}_{\text{GT}}$	gap_{WS} (%)	book (%)	failed (%)	sat. riders (%)	cost (%)	T (sec)
1.3	464.40	12.55	34.33	4.24	75.55	76.36	10.80
1.4	528.58	7.47	36.29	2.77	87.95	75.85	64.29
1.5	566.73	4.48	37.79	1.59	95.98	75.61	58.77

The results indicate that, as τ_{D}^1 increases, the platform profit also increases. This is explained by the fact that the platform is able to form more rideshares: there are more booked drivers and fewer failed bookings, as drivers can adjust more easily to different rideshares. Furthermore, a direct consequence of having less failed bookings is a smaller gap to the wait-and-see solution, which has no failed bookings. Rider demand can also be better met with larger driver availability windows: almost all riders (95.98%) are provided service when $\tau_{\text{D}}^1 = 1.5$, while the trips costs for users remain stable (around 76%, representing 24% cost savings). However, using such large availability windows would not provide the best user experience for drivers. A reasonable trade-off value for τ_{D}^1 seems to be 1.4, as it improves all performance metrics and does not require a large level of driver flexibility.

5.5.2.4. PEN-1: impact of revenue share and penalty levels. The goal of this section is to identify values for the revenue share and penalties such that the ridesharing platform attains a good profit while keeping users satisfied. To this end, we explore the impact of changing the platform revenue share and penalty levels independently. Then, we study the joint impact of these parameters.

Impact of revenue share. We analyze the impact of changing the scaling parameter γ_{R}^1 for the revenue share collected by the ridesharing platform (defined in Section 5.5.1.5). We limit these experiments to variant PEN-1. We fix the penalty coefficient to $\gamma_{\text{P}}^1 = 0.3$ and vary the share of savings collected by the ridesharing platform $\gamma_{\text{R}}^1 \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ (note that results are averaged over instances with driver availability window scaling coefficients $\tau_{\text{D}}^1 \in \{1.3, 1.4, 1.5\}$).

Table 5.5. SAA-R results for different values of γ_{R}^1 on variant PEN-1 (averaged over instances with $\gamma_{\text{P}}^1 = 0.3$ and $\tau_{\text{D}}^1 \in \{1.3, 1.4, 1.5\}$).

γ_{R}^1	$\text{profit}_{\text{GT}}$	gap_{WS} (%)	book (%)	failed (%)	sat. riders (%)	cost (%)	T (sec)
0.1	130.93	10.31	34.67	1.54	85.13	64.81	43.31
0.3	408.47	6.70	36.33	2.42	87.14	72.85	48.82
0.5	690.73	5.30	38.17	4.05	88.91	80.74	45.21
0.7	976.08	4.39	39.17	5.12	89.72	88.44	39.61
0.9	1,261.43	3.89	40.33	6.42	90.30	96.16	29.67

The results are summarized in Table 5.5. As expected, when increasing the revenue share, more matching decisions become economically profitable, which leads to more booked drivers and more matched riders, resulting in a larger profit for the platform. The platform profit also does not seem to be critically affected by the increasing proportion of failed driver bookings (which are compensated an equivalent of 30% of their individual trip costs, as $\gamma_{\text{P}}^1 = 0.3$). This is explained by the fact that the total paid penalties become negligible compared to the revenues generated from the rideshares. This disparity between revenue and penalty values also explains why larger revenue shares are linked to smaller gaps to the wait-and-see solution. Unfortunately, an increasing revenue share kept by the operator also results in higher costs for the users: with $\gamma_{\text{R}}^1 = 0.9$, the participants pay on average 96% of the costs of driving alone, which are rather unattractive cost savings. The revenue share γ_{R}^1 therefore has to be kept at reasonable levels (e.g., at 0.3) to ensure that users save around 30% when compared to driving alone.

Impact of different penalty levels. We now investigate how the results change according to different penalty levels γ_{P}^1. We limit these experiments to variant PEN-1 (for availability window scaling coefficients $\tau_{\text{D}}^1 \in \{ 1.3, 1.4, 1.5 \}$), fix the share of savings collected by the ridesharing platform to $\gamma_{\text{R}}^1 = 0.3$, and vary the penalty scaling coefficients $\gamma_{\text{P}}^1 \in \{ 0.1, 0.3, 0.5, 0.7, 0.9, 1.1, 1.3, 1.5 \}$. Note that any value $\gamma_{\text{P}}^1 > 1$ implies that drivers are compensated more than their trip cost in case they are not assigned to any rideshare.

Table 5.6. SAA-R results for different values of γ_{P}^1 on variant PEN-1 (averaged over instances with $\gamma_{\text{R}}^1 = 0.3$ and $\tau_{\text{D}}^1 \in \{ 1.3, 1.4, 1.5 \}$).

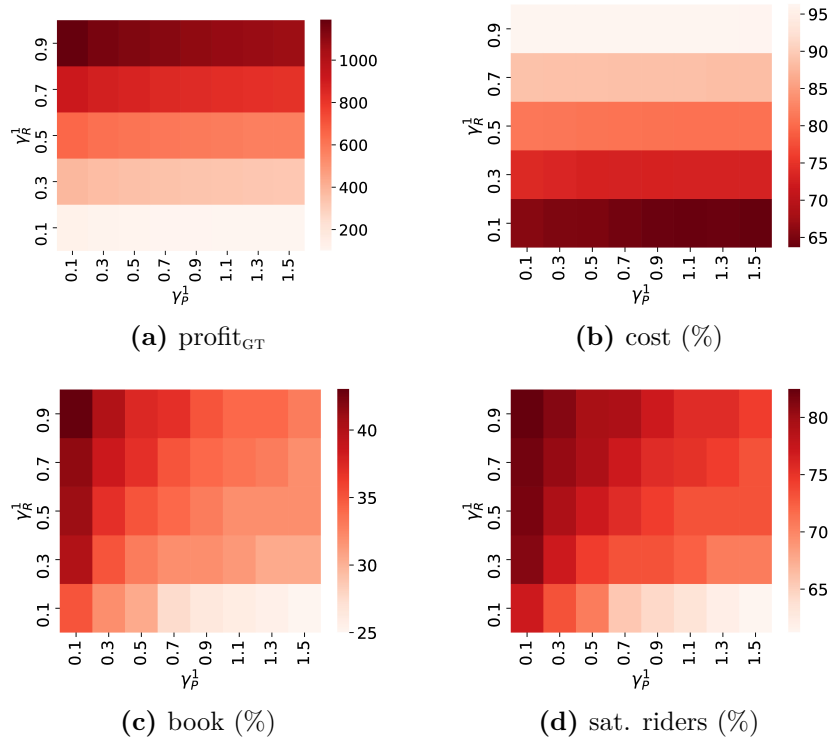
γ_{P}^1	$\text{profit}_{\text{GT}}$	gap_{WS} (%)	book (%)	failed (%)	sat. riders (%)	cost (%)	T (sec)
0.1	420.48	3.89	40.33	6.42	90.30	73.10	42.26
0.3	408.47	6.70	36.33	2.42	87.14	72.85	48.82
0.5	402.14	8.18	35.33	1.88	85.91	72.67	49.85
0.7	397.06	9.36	34.67	1.66	85.29	72.55	53.27
0.9	392.78	10.36	34.67	1.54	85.15	72.63	44.85
1.1	389.25	11.18	33.67	1.25	83.89	72.46	54.36
1.3	386.24	11.88	33.17	1.06	83.09	72.42	42.01
1.5	383.23	12.59	33.17	1.06	83.10	72.42	42.23

The results are summarized in Table 5.6. Surprisingly, larger penalties do not have a major impact on the system's profit: the proportion of failed bookings decreases quickly as the penalties increase, which avoids large profit losses. For the smallest penalty level $\gamma_{\text{P}}^1 = 0.1$, about 5 of the 81 booked drivers are paid penalties, while at the highest penalty level, this is the case for less than 1 booked driver on average. However, higher penalties result in less driver bookings, and therefore also in less rider matches. This explains the small decrease in the profit. Higher penalties may therefore be a good "marketing" strategy to attract new drivers, as they are likely to be matched; even if they are not, they can expect a large compensation. Additionally, from a user perspective, the average trip cost remains

relatively stable. Penalty values at around $\gamma_{\mathrm{P}}^1 = 0.5$ therefore seem to provide a good trade-off, given that both the profit and the user booking percentages are relatively high, while a 50% compensation is still sufficiently high to attract new drivers to the user-base. Finally, we note that larger penalties increase the gap to the wait-and-see solution. This is explained by the larger penalty costs that are paid by the stochastic solution, while the wait-and-see solution avoids all penalties due to perfect knowledge.

Joint impact of different revenue share and penalty levels. While the above explores the impact of increasing revenue share and penalty levels separately, we now investigate the impact of changing both parameters at the same time. We consider all combinations of $\gamma_{\mathrm{R}}^1 \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ and $\gamma_{\mathrm{P}}^1 \in \{0.1, 0.3, 0.5, 0.7, 0.9, 1.1, 1.3, 1.5\}$ (for a fixed driver availability window coefficient $\tau_{\mathrm{D}}^1 = 1.3$).

Fig. 5.3. Impact of different values of γ_{R}^1 and γ_{P}^1 on key performance indicators (averaged over instances with $\tau_{\mathrm{D}}^1 = 1.3$, $\gamma_{\mathrm{R}}^1 \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ and $\gamma_{\mathrm{P}}^1 \in \{0.1, 0.3, 0.5, 0.7, 0.9, 1.1, 1.3, 1.5\}$).



(a) profit$_{\mathrm{GT}}$ **(b)** cost (%)

(c) book (%) **(d)** sat. riders (%)

Figure 5.3 illustrates the impact of the different parameter value combinations on the four most important key performance indicators: the operator profit, the remaining costs for users, the percentage of booked drivers and the percentage of matched rider requests. Generally, these results confirm the previous observations: operator profits and user costs are inversely related in regards to the profit share. It is interesting to note that system profit and user costs roughly sum to 100%, which is not obvious, given that the latter is computed

as the cost savings (directly correlated to the total travel distance) that a user can obtain by sharing a ride as when opposed to driving alone. For the operator, this suggests an important trade-off, which will be mostly dictated by the importance of the user savings. For example, if the system aims at an average cost of about 70% for the riders and drivers, the operator should expect that the revenue share γ_{R}^1 has to be kept below 30% (i.e., a value of 0.3).

In contrast, the penalty level only marginally impacts the operator profit and user costs, but considerably impacts the percentages of booked drivers and matched rider requests. In order to keep users happy, it appears to be beneficial to keep the penalty value at a reasonably low value around $\gamma_{\text{P}}^1 = 0.5$. As shown in the previous study, this may slightly increase the percentage of drivers that have been booked, but not matched; however, in such rare cases, a 50% compensation of the total trip costs are likely to be sufficiently attractive to drivers.

5.5.2.5. PEN-2: larger availability windows at the cost of higher penalties. We now assess whether large availability windows are a more beneficial choice, even in the case of larger penalty values. To this end, we assess problem variant PEN-2 with type-1 driver availability windows of ($\tau_{\text{D}}^1 = 1.3$ and larger type-2 availability windows of $\tau_{\text{D}}^2 = 1.4$). Larger availability windows clearly lead to a higher matching flexibility, as more ridesharing routes become time-feasible. To properly compensate type-2 drivers for the inconvenience, we evaluate the impact of larger values of type-2 penalties. Specifically, we test $\gamma_{\text{P}}^2 \in \{ 0.5, 1, 2, 4, 8, 16 \}$, while the penalty for type-1 drivers remains at the same (smaller) value of $\gamma_{\text{P}}^1 = 0.3$. Note that all experiments assume a fixed revenue share of $\gamma_{\text{R}}^1 = \gamma_{\text{R}}^2 = 0.3$, which has been found in previous experiments to be a beneficial trade-off between operator profits and savings for users.

Table 5.7. SAA-R results for different values of γ_{P}^2 on variant PEN-2 ($\tau_{\text{D}}^1 = 1.3$, $\tau_{\text{D}}^2 = 1.4$ and $\gamma_{\text{P}}^1 = 0.3$).

γ_{P}^2	$\text{profit}_{\text{GT}}$	gap_{WS} (%)	book L1 (%)	book L2 (%)	failed L1 (%)	failed L2 (%)	sat. riders (%)	cost (%)	T (sec)
0.50	271.22	8.04	4.54	30.31	9.25	0.62	87.32	68.71	423.83
1.00	269.32	8.68	6.26	29.42	10.53	0.28	87.84	69.00	760.35
2.00	266.79	9.53	7.73	26.96	9.10	0.14	86.47	68.66	262.40
4.00	263.26	10.73	8.19	26.97	8.97	0.13	86.45	68.89	283.12
8.00	256.42	13.03	8.70	25.47	8.43	0.12	85.79	68.54	187.28
16.00	249.34	15.43	8.26	25.48	8.26	0.09	84.96	68.47	224.06

The results are summarized in Table 5.7. Columns "book L1 (%)" and "book L2 (%)" indicate the average percentages of all drivers booked at types 1 and 2, respectively. Columns "failed L1 (%)" and "failed L2 (%)" show the average proportions of booked, but unmatched type-1 and type-2 drivers, respectively. The results show that the optimal solutions book a much larger percentage with larger availability windows (i.e., type-2 drivers), even when faced with unrealistically high penalties (such as $\gamma_{\text{P}}^2 = 16$). Indeed, larger availability windows

reduce the likelihood of failing a booking (from about 10% for type-1 drivers to at most 0.62% for type-2 drivers), as drivers have more flexibility. In absolute terms, far less than 1 out of the more than 50 booked type-2 drivers remain unmatched on average. The monetary risk to the ridesharing platform is therefore marginal, even when penalties are high. Interestingly, trip costs for users remain stable. On the downside, larger penalties imply that a smaller proportion of the available drivers is booked, Nevertheless, high penalties may be an interesting temporary "marketing strategy" to attract a large driver user-base in the near-future without sacrificing much profit. Finally, as observed in previous experiments, larger penalties increase the gaps to the wait-and-see solution. This is expected, given that the latter has perfect knowledge of rider requests and can therefore avoid penalties.

On the EVP performance for variant PEN-2. Intrigued by the rather bad performance of the EVP for problem variant PEN-2 (see Table 5.2), we now explore the model performance for different penalty values γ_{P}^2 in Table 5.8. As before, these experiments involve high penalties for type-2 drivers, ranging from $\gamma_{\text{P}}^2 = 0.5$ to 16.

Table 5.8. EVP results for different values of γ_{P}^2 on variant PEN-2 ($\tau_{\text{D}}^1 = 1.3$, $\tau_{\text{D}}^2 = 1.4$ and $\gamma_{\text{P}}^1 = 0.3$).

γ_{P}^2	$\text{profit}_{\text{GT}}$	gap_{WS} (%)	book L1 (%)	book L2 (%)	failed L1 (%)	failed L2 (%)	sat. riders (%)	cost (%)	T (sec)
0.50	260.17	11.77	6.12	30.79	12.54	2.25	89.62	69.05	1.72
1.00	244.22	17.15	4.27	32.69	14.59	2.41	89.73	69.05	2.00
2.00	203.87	30.79	3.39	33.56	15.30	2.71	89.71	69.05	1.76
4.00	140.37	52.24	5.33	31.63	11.10	2.67	89.59	69.00	1.73
8.00	-30.50	109.97	6.35	30.56	9.22	2.98	89.65	69.04	1.85
16.00	-322.52	208.62	5.37	31.59	10.45	2.81	89.61	69.08	1.55

The results are quite conclusive and highlight the negative consequences of the EVP's inability to discriminate the probabilistic risk of failed driver bookings. The EVP solution books more type-2 drivers than the SAA-R solution (31.59% vs. 25.48% for $\gamma_{\text{P}}^2 = 16$; compare Table 5.7), which allows for a high flexibility to match the fractional rider requests. However, the proportion of failed type-2 bookings remains relatively high when compared to the SAA-R solutions (2.81% vs. 0.09% for $\gamma_{\text{P}}^2 = 16$), resulting in several penalties when evaluated on the ground truth. We expect these results to generalize to other problem variants if large penalty values are evaluated. This, in turn, may ultimately jeopardize the profitability of the system. In practice, this is likely to be unacceptable and makes the EVP an unnecessarily risky choice as a planning model.

5.5.2.6. FEE-2: larger availability windows at the cost of higher booking fees. We now investigate whether the conclusions from the previous section also hold in the case of the problem variant, where booking fees are paid instead of penalties. To this end, we consider problem variant FEE-2, which has two different booking types. Drivers can be

booked at type 1 for a fee of $f_i^1 = 1$, which are available for 45 minutes. Alternatively, drivers can be booked at type 2, for a larger time window of 1 hour and a higher booking fee of $f_i^2 \in \{ 1.5, 2.0, 2.5, 3.0, 3.5, 4.0 \}$.

Table 5.9. SAA-R results for different values of f_i^2 on variant FEE-2 ($f_i^1 = 1$; type-1 drivers are available for 45 minutes; type-2 drivers are available for 1 hour).

f_i^2	$\text{profit}_{\text{GT}}$	gap_{WS} (%)	opt. gap (%)	book L1 (%)	book L2 (%)	failed L1 (%)	failed L2 (%)	sat. riders (%)	cost (%)	T (sec)
1.5	148.68	10.84	0.59	1.32	16.13	12.22	12.81	77.61	71.53	3,600.00
2.0	131.88	14.06	0.85	2.76	14.15	21.21	11.57	75.24	71.39	3,600.00
2.5	117.34	16.69	1.11	4.11	12.47	25.21	10.96	73.66	71.42	3,600.00
3.0	106.00	17.99	0.61	5.04	11.07	22.53	7.73	71.28	71.45	3,600.00
3.5	94.15	20.39	0.67	5.46	10.17	27.25	7.58	69.04	71.42	3,600.00
4.0	84.14	21.90	0.44	5.93	9.34	25.91	6.62	66.95	71.53	3,600.00

The results are summarized in Table 5.9. As the models tend to exceed the CPU time limit for this problem variant, column "opt. gap (%)" shows the optimality gap of the best known solution as reported by CPLEX. Even though CPLEX did not prove optimality for any instance, the final optimality gaps are rather small. The difficulty of solving problem instances for this problem variant can be explained by the fact that drivers have no fixed destination, and are therefore more flexible. This leads to a larger number of feasible routes when compared to other problem variants (as previously shown in Table 5.1), which leads to larger formulations that require more computational resources to be solved.

Although the proportion of booked type-2 driver decreases and the proportion of booked type-1 drivers increases with larger type-2 penalties, the system tends to prioritize type-2 drivers. This highlights the benefit of larger availability windows, which is in line with previous findings for other problem variants (see Sections 5.5.2.3 and 5.5.2.5). However, in contrast to variant PEN-2, fees that are paid for every booked driver, and not only for those who remain unmatched, result in less driver bookings in general, and therefore in less matched rider requests. Larger type-2 booking fees naturally amplify this phenomena: the system starts matching more type-1 drivers, which further reduces the number of successful matches, and therefore reduces the overall system profit. The percentage of failed bookings are relatively high on both booking types, given that no additional penalty is paid if a booked driver remains unmatched. For a similar reason, such higher number of failed booking do not imply less satisfied drivers, since drivers are compensated in either case. Finally, similar to previous results, the gaps to the wait-and-see solution are relatively high, given that the latter can avoid paying booking fees to drivers that are not matched.

5.5.2.7. PEN-FEE: impact of mixed booking costs and penalties. As a final set of experiments, we now focus on variant PEN-FEE, a context where drivers can be booked either by means of penalties (booking type 1) or by a fixed booking fee without risk of future

penalties (booking type 2). Specifically, type-1 drivers can be booked with potential penalty scaling values $\gamma_{\text{P}}^1 \in \{ 0.5, 1, 1.5, 2, 2.5, 3, 8, 16 \}$. Type-2 drivers are booked without penalties, but with a constant booking fee $f_i^2 = 1, \forall i \in D$. Both booking types have the same relative availability window scaling values $\tau_{\text{D}}^1 = \tau_{\text{D}}^2 = 1.3$ and revenue share $\gamma_{\text{R}}^1 = \gamma_{\text{R}}^2 = 0.3$.

Table 5.10. SAA-R results for different values of γ_{P}^1 on variant PEN-FEE ($\tau_{\text{D}}^1 = \tau_{\text{D}}^2 = 1.3$, $\gamma_{\text{R}}^1 = \gamma_{\text{R}}^2 = 0.3$ and $f_i^2 = 1$).

γ_{P}^1	$\text{profit}_{\text{GT}}$	gap_{WS} (%)	book L1 (%)	book L2 (%)	failed L1 (%)	failed L2 (%)	sat. riders (%)	cost (%)	T (sec)
0.5	369.27	9.34	29.08	6.41	1.41	32.48	80.65	73.59	16.17
1.0	363.25	10.81	25.33	9.51	0.68	26.86	80.28	73.26	16.17
1.5	360.86	11.40	24.38	10.35	0.50	26.04	80.28	73.21	15.28
2.0	358.69	11.93	23.41	11.32	0.39	24.51	80.28	73.21	18.87
2.5	356.49	12.47	22.43	11.86	0.33	23.46	79.95	73.03	17.67
3.0	355.56	12.69	21.94	12.35	0.26	22.83	79.95	73.03	16.13
8.0	346.81	14.83	20.96	13.60	0.18	22.27	80.24	73.06	20.94
16.0	338.49	16.87	19.97	14.59	0.13	21.14	80.24	73.05	20.88

The results, summarized in Table 5.10, indicate that, for lower penalty values, the system favors type-1 drivers (those that potentially imply penalties) over type-2 drivers (those paid booking fees). While the proportion of type-1 drivers decreases as the penalties increase, even for unrealistically large penalty values (i.e., $\gamma_{\text{P}}^1 = 16$), we do not observe an inflection point at which booking fees become more favorable. Such behavior is mostly motivated by the underlying percentages of booked, but unmatched drivers: the proportion of failed type-1 drivers is negligibly small, while the proportion of failed type-2 drivers is quite high (but without economic impact). For $\gamma_{\text{P}}^1 = 16$, the average penalty for an unmatched type-1 driver is 335.85, which may pose considerable risks to the ridesharing platform, as a single penalty paid would almost be in the same orders of the entire profit. On average, however, only 0.05 type-1 drivers remain unmatched (200 drivers $\times 19.97\% \times 0.13\%$). Among all booked type-2 drivers, only about 4 to 6 remain unmatched, on average (and in this case, without any economic impact).

Similar to previous results, larger penalties only have a limited negative impact on the profit, as the system rarely fails a type-1 booking and has to pay the penalty. Further, larger penalties lead to larger gaps to the wait-and-see solution, similar to previous results. All other performance metrics remain similar, which confirms the conclusions of previous experiments, suggesting that penalties are an attractive driver compensation mechanism to ridesharing operator (as opposed to booking fees), while maintaining an overall customer satisfaction.

5.5.2.8. Summary of managerial insights and recommendations. Throughout our extensive computational experiments, a series of managerial insights have been derived. These insights concern the benefits of the different operating modes, as well as the values of

the system's parameters that ensure both system profitability and a high user satisfaction. We summarize these insights as follows:

(1) Larger availability windows allow for more matching flexibility and ultimately lead to a higher number of successful matches and hence a higher platform profit. Operators may want to carefully study the possibility and costs for having drivers committed to larger availability windows, given the higher matching probability associated to such increased flexibility;

(2) Large penalties do not necessarily jeopardize the profitability of the system, given that optimal planning solutions will adjust the driver bookings in order to keep failed bookings low, and therefore mostly avoid penalties. As a result, announcing (potentially temporary) high penalty compensation may be a valid marketing strategy to attract new drivers to the system's user-base;

(3) In a similar vein, if penalties can mostly be avoided on average, a driver compensation mechanism via penalties is likely to be preferable to system operators over fixed booking fees;

(4) Even when both compensation schemes (i.e., booking fees and penalties) are available and penalty values are unrealistically large, compensating drivers via penalties is likely to remain favorable;

(5) While a higher revenue share kept by the operator also implies more profit, it is inversely related to the savings for the users. As such, operators may want to carefully study the maximum revenue-share percentage that results in reasonable cost savings for the riders. For the settings used in our experiments, we have found that the share of the total profit retained by the company should not exceed 30% to ensure that matched riders save approximately 30% of costs when compared to driving alone.

5.6. Conclusions

We have introduced a general modeling framework for ridesharing systems, where drivers have to be selected before driver requests become available. This framework allows for the representation of many-to-one ridesharing systems, and addresses the uncertainty of rider requests while taking into consideration compensation strategies for drivers (booking fees and penalties for not providing a rideshare). The mixed-integer programming modeling framework consists of a two-stage set-packing model with stochastic nodes. We exemplify the usage of this model by means of three instances of ridesharing system operating modes and identify the conditions under which the here proposed problem also solves the more general multi-stage problem variant used in a dynamic context.

We evaluate three models that approximate the second-stage value function: the SAA model, the expected value model, and the SAA model with a relaxed second-stage problem

(SAA-R). Our computational results show that the SAA and SAA-R models generate better solutions, while the latter is solved within a a fraction of the required computing time.

Furthermore, based on our computational results, we have derived valuable managerial insights on the possible operating modes and parameter values. In summary, the experiments highlighted 1) the value of having larger availability windows; 2) that larger penalties do not necessarily negatively impact the system's profit; 3) the value of using booking penalties over booking fees, and 4) appropriate parameter settings for revenue share and penalty that result in a balanced system profit and customer satisfaction.

This research opens several promising research directions. From a practical viewpoint, it may be worthwhile investigating multi-stage problem variants that cannot be solved exactly by the here proposed two-stage stochastic model (e.g., variants where drivers can be selected at any time during the day, as well as the possibility of un- and re-matching drivers and riders). From a methodological point of view, mathematical decomposition algorithms may be employed to solve the here proposed models faster and on larger scale. In this regards, integrating route-generation via column generation may be a promising avenue to solve the SAA, while Benders decomposition may be employed to solve the SAA-R.

Acknowledgements

We would like to thank Netlift and Mathieu Lozeau for their thoughtful input, domain knowledge, and for providing us with access to data. We are also grateful to Compute Canada for providing computational resources. The work of the first author was supported by the Chaire en transformation du transport (Université de Montréal/Polytechnique Montréal), which is funded by the Ministère de l'Économie, de l'Innovation et de l'Énergie du Québec. The work of the third author was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada under grant 2017-05224 and the Fonds de Recherche Nature et Technologies (FRQNT) under grant 70739202.

Chapter 6

Strategic Planning of Carbon Capture and Storage: a Multiperiod Slope Scaling Heuristic

The previous three chapters of this thesis focused on the operational and tactical planning of ridesharing systems. While improving the efficiency of ridesharing systems may contribute to generating less greenhouse gas emissions, other mitigation strategies must be used in parallel to significantly reduce emissions. This chapter focuses on another mitigation strategies: carbon capture and storage (CCS). More specifically, this chapter focuses on the value chain optimization of a multiperiod CCS problem. This problem has the characteristics of joint network design and facility location problems, and solving it poses considerable challenges to commercial mathematical programming solvers. To address these challenges, this chapter introduces a slope scaling heuristic that can generate, on average, better solutions than a commercial mathematical programming solver, for a fraction of the computational time. These results are particularly important for analysts, which may need to conduct hundreds or thousands of sensitivity studies to assess the costs of deploying a CCS infrastructure. An extended version of this chapter will be submitted for publication in a peer-reviewed journal. The contributions of the student are:

- The introduction of a slope scaling heuristic for a multiperiod combined network design and facility location problem;
- A dynamic programming algorithm to improve the quality of the slope scaling solution;
- The implicit representation of pipeline cost functions within the slope scaling approximation;
- The introduction of a restricted model to improve the best-known solution found by slope scaling;
- Computational experiments and insights on the performance of commercial mathematical programming solvers and slope scaling, according to different instance characteristics.

Strategic Planning of Carbon Capture and Storage: a Multiperiod Slope Scaling Heuristic

Gabriel Homsi [1,2,4], Étienne Ayotte-Sauvé [2], Sanjay Dominik Jena [3,4]

[1] Department of Computer Science and Operations Research, Université de Montréal, Canada

[2] CanmetENERGY, Natural Resources Canada, 1615 Lionel-Boulet Boulevard, P.O. Box 4800, Varennes, Quebec J3X 1S6, Canada

[3] Department of Analytics, Operations and Information Technology, École des Sciences de la Gestion, Université du Québec à Montréal, Canada

[4] Centre interuniversitaire de recherche sur les reseaux d'entreprise, la logistique et le transport (CIRRELT)

Abstract

Global efforts are underway to implement various decarbonization strategies in all sectors of economic activity to meet the objectives of the Paris Agreement. One major strategy is carbon capture and storage (CCS), which involves capturing CO_2 at emitter sites, and transporting it to geological storage sites, where it is to be injected underground for long-term storage. In this work, we focus on the multiperiod strategic planning of a CCS supply chain involving pipeline CO_2 transportation. From an Operations Research standpoint, such a problem exhibits the characteristics of combined facility location and network design. To account for multiple scenarios of potentially realistic input parameters values, this problem may have to be be solved thousands of times. Thus, reaching high-quality solutions quickly is paramount. As commercial solvers struggle to provide high-quality solutions under these time constraints, we propose a slope scaling heuristic based on previous work for single-period CCS and network design. This heuristic approximates the cost of design variables, generates upper bounds via dynamic programming, uses a long-term memory search strategy, and includes a final improvement phase where a restricted model is solved. Computational experiments show that the proposed heuristic generates better solutions than CPLEX for most considered instances, at a fraction of the time.

Keywords: carbon and capture and storage; heuristics; slope scaling; dynamic programming.

6.1. Introduction

In order to meet the target of the Paris Agreement, that is, to "limit global warming to well below 2, preferably to 1.5 degrees Celsius, compared to pre-industrial levels" (Paris Agreement to the UN FCCC, 2015), global efforts are currently underway to decarbonize all sectors of economic activity. In particular, Canada has recently passed legislation reflecting its commitment to achieve net-zero carbon emissions by 2050 (Canadian Net-Zero Emissions Accountability Act, 2021). Similarly, the United States introduced in 2021 "The Long-Term Strategy of the United States, Pathways to Net-Zero Greenhouse Gas Emissions by 2050" [United States Department of State and United States Executive Office of the President, 2021]. According to the UN, as of March 2022, more than 70 countries responsible for roughly 76% of global emissions have adopted a net-zero target.

Studies around the world indicate that meeting those objectives will require a combination of various greenhouse gas (GHG) mitigating strategies [e.g, Riahi et al., 2017]. Those include improving the energy efficiency of existing industrial facilities, replacing industrial processes by less GHG-intensive alternatives, using substitutes to fossil fuels (e.g. biomass), alternative materials, electrifying the transportation and industrial sectors, as well as using hydrogen (H_2) as an alternative energy source.

In some industrial sectors, eliminating GHG emissions and short-term use of fossil fuels can be difficult. For example, roughly 60% of the CO_2 equivalent emissions in a cement plant originate from the calcination reaction itself, not from the burning of fossil fuels [Strunge et al., 2022]. Alternative pathways, such as replacing cement with alternative materials, could take decades to implement, as they impact global supply chains. This example illustrates the fact that a careful investigation is necessary to devise realistic strategies towards net-zero targets.

Of all anthropogenic GHGs, carbon dioxide (CO_2) is the most prevalent, and the analysis of global energy and economy scenarios [Lane et al., 2021, Riahi et al., 2017, IEA, 2017] indicates that net-zero carbon emissions may not be achievable without capturing CO_2 emissions from the atmosphere [Vinca et al., 2018, IEA, 2019], and either storing them underground in geological formations (geological CO_2 storage), storing them temporarily in living organisms such as plants and trees (biological CO_2 sequestration), or reusing them to synthesize materials or fuels (CO_2 utilization). The present study focuses on carbon capture and geological storage (CCS), which involves capturing CO_2 at emitter sites and transporting it to geological storage sites, where it is to be injected underground for long-term storage.

In order to be an effective path to reach net-zero carbon emissions, the current pace of CCS implementation has to be increased [Lane et al., 2021]. However, this poses a complex issue. The deployment of CCS involves planning ahead for decades (e.g. up to 2050), billions

of dollars of investment across industries and governments, as well as defining policies that can accelerate its adoption (e.g. carbon credits, taxes, markets, subsidies).

6.1.1. Mathematical optimization for CCS

Given the importance and urgency of developing efficient CCS networks, various deterministic planning models based on mathematical optimization are being developed to aid in the strategic planning of CCS value chains, namely for CO_2 pipeline networks (see Section 6.2). These models can aid in the preliminary analysis done by government stakeholders to assess the CO_2 mitigation potential and the costs of CCS, as well as the impact of policy measures (e.g., CO_2 taxes and subsidies). As such, these models can contribute to accelerate CCS adoption and implementation by providing a holistic view of an optimized supply chain.

The CCS infrastructure strategic planning problem is inherently multiperiod in nature. Capture targets and CO_2 incentives (e.g. taxes, subsidies) may change over time depending on government policy. Market fluctuations (e.g. commodities and materials) may influence infrastructure costs. Moreover, sources and sinks may only be available after (or until) a certain period. For example, emitter plants may close due to market changes and storage site availability may change as investor confidence grows. Furthermore, capture, transportation and storage operations will go through a ramping-up period, during which the volume of CO_2 captured, transported and stored will increase. As such, pipeline capacity planning can be anticipatory, or alternatively, be increased as needed. Although single-period CCS models exist, they can overestimate costs by more than 50% when compared to their multiperiod counterparts [Middleton et al., 2012b]. Therefore, the present study focuses on multiperiod CCS models.

To the best of our knowledge, for multiperiod pipeline-based CCS value chain optimization (VCO), all published works use commercial optimization solvers to tackle a single mixed-integer linear programming (MILP) model. Unfortunately, models that consider the pipeline routing explicitly may reach large optimality gaps (13%+) after one hour of calculations (see Section 6.6). At the scale of a country like the United States, decarbonization efforts may require capturing hundreds of millions of tonnes per year of CO_2 from roughly 1000 sources [Talsma et al., 2022]. Furthermore, for sensitivity studies models may need to be evaluated hundreds or thousands of times to assess trade-offs between costs, overall CO_2 capture potential and other model input parameter values [e.g., Middleton and Yaw, 2018]. In that context, for large geographic regions, providing sensitivity results with acceptable optimality gaps may be impractically long. Adding more complex model attributes such as alternative transportation modes, distinguishing between CO_2 phases and considering uncertain parameters may exacerbate this problem. To address this issue, in this study, we propose a heuristic for a multiperiod CCS planning problem. This heuristic finds better

solutions than CPLEX on average, at a fraction of the computational cost. Numerical experiments on instances having up to 50 time periods and six pipeline capacity trends show that the proposed heuristic attains relative CCS cost improvements of up to 80.88% over CPLEX.

From an Operations Research standpoint, this model exhibits the characteristics of facility location problems (sources and sink decisions) and of network design problems (pipeline decisions). Sources and sink decisions consist of when to open source nodes (emitters), which capture technologies to use and how much CO_2 to capture. Sink decisions include when to open sink nodes (geological storage), how many wells to dig and how much CO_2 to store underground. Network design decisions include which pipeline arcs to build, and when. These arcs can be selected out of a candidate network of pipelines. Network decisions also include the choice of pipeline capacity and CO_2 flow for each arc. Parallel pipelines (i.e., several pipelines on the same arc) are allowed in order to reflect the trade-off between over-sizing pipelines in advance and adding capacity incrementally as needed (i.e. building new pipelines on the same arc).

In order to instantiate such model, one must first determine a candidate pipeline network embedding all potential pipeline routes linking candidate sources and sinks. Popular approaches to generate such a directed graph, like the CostMAP open-source code [Hoover et al., 2020], do so in a stepwise manner. First, geographic features of the region of interest are rasterized, e.g. topography, waterways, rights of way, existing infrastructure, such as roads, electric grid lines, demographics, and protected areas (wildlife, first nations). Afterwards, a cost is associated to passing a pipeline segment through the center of each raster cell of the map, depending on the local geographic data and on the orientation of the pipeline segment. One can then associate a cost to any pipeline path on that map by summing the costs for passing through each raster cell (center) it intersects. The candidate network results from applying a least-cost path algorithm (e.g. Dijkstra) to each pair of source and sink nodes of interest. In practice, to be numerically efficient, to avoid near parallel arcs and impractically long paths linking nodes that are far apart, least-cost path calculations are carried out for each adjacent node pair in a Delaunay triangulation of the set of sources and sinks. Further details can be found in Middleton et al. [2012a] and Hoover et al. [2020].

Problem data includes the geographic region of interest (e.g. a state, province or country) and a time horizon (e.g. up to 2050) partitioned in discrete periods that typically span multiple years. We consider a set of source sites (i.e. emitters), each with possibly multiple alternative capture units as well as a CO_2 capture target for that region. That target corresponds to the total quantity of CO_2 to be captured per year and per time period. The CO_2 can be stored in a set of sinks (i.e. geological storage sites), each possibly containing sub-reservoirs (i.e. stacked reservoirs). Transportation from sources to sinks is done through a candidate pipeline network. Each source and sink has a capacity parameter, corresponding

to maximal capture rates for each capture unit and maximal injection rates for each reservoir subsite. Cost data consists of fixed and variable costs for each capture unit, reservoir subsite, and pipeline arc.

Given that data, one aims to minimize the total cost of the project while meeting annual capture targets for each time period. Decisions at each time period include the selection of capture units, pipeline arcs, and reservoir subsites, how much CO_2 to annually capture and inject for each capture unit and reservoir subsite (respectively), how much CO_2 to annually transport on each opened pipeline arc and which pipeline capacity to build on each arc as well how many wells to drill at each reservoir subsite.

6.1.2. Contributions and outline

The main contribution of this work is the introduction of a slope scaling heuristic for the multiperiod strategic planning of pipeline-based CCS supply chains. Such heuristic is a novel multiperiod extension of previous network design and CCS slope scaling heuristics. It has novel attributes such as the implicit representation of pipeline capacity cost functions, generation of improved upper bounds through dynamic programming, and solution refining through a restricted model. Such heuristic is flexible enough to be adapted to different CCS problems, such as problems with non-linear pipeline cost functions, problem settings with multiple modes of CO_2 transportation, and models with a CO_2 price for uncaptured emissions. Computational experiments on a wide range of CCS instances with different sizes show that the proposed slope scaling heuristic outperforms CPLEX on most instances, at a fraction of the computational time.

This paper is organized as follows. In Section 6.2, we conduct a review of the literature on strategic planning of CCS pipeline networks and related problems. In Section 6.3 and Section 6.4, we define the planning problem and the corresponding mathematical programming model (respectively). We then introduce the slope scaling heuristic in (Section 6.5). To assess the performance of the proposed heuristic, the results of extensive computational experiments are analyzed in Section 6.6. Conclusions and avenues for future research are then summarized in Section 6.7.

6.2. Literature Review

In this section, we give an overview of past work related to the present study. As such, we cover the following topics: pipeline-based CCS value chain optimization models, related Operations Research problems, computational challenges and heuristics to address such planning problems.

6.2.1. Pipeline-based CCS value chain optimization models

This work focuses on strategic planning models for CCS VCO. At the strategic level, the planning horizon typically spans decades (e.g. up to 2050, or beyond). Some of these models consider only pipelines [e.g., Jones et al., 2022], while others consider a mix of transportation modes [e.g., Han and Lee, 2011]. Pipelines play an important role in transporting large volumes of CO_2, and this study considers solely this transportation mode. Middleton and Bielicki [2009] introduced SimCCS, a single-period pipeline-based VCO model that is now widely recognized. In SimCCS, emitter sites and geological reservoirs are connected through a network of pipelines which stems from an analysis of geographical features of the region considered. Not all CCS VCO models represent pipeline routing explicitly. For example, in Diamante et al. [2013], emitter sites and geological reservoirs are connected directly, without considering pipeline routes. Often termed "source-sink" matching problems, the problem studied by the authors can be represented with a linear programming formulation, which is solvable in polynomial time. Representing a complex network of pipelines leads to a more accurate representation of CCS transportation costs, but it comes with a computational complexity disadvantage: these models extend network design problems, which are NP-hard and therefore challenging to solve. This contrasts with source-sink matching models that can be solved in polynomial time (e.g., bipartite matching or transportation linear programming models). Single-period models assume that all infrastructure is built at the beginning of the planning horizon, and therefore disregard the tradeoff between making infrastructure available early and their financing and discounting implications on the remainder of the planning horizon. In fact, single-period models can overestimate costs by more than 50% [see Middleton et al., 2012b]. To address this issue, Middleton et al. [2012b] introduced a multiperiod version of SimCCS, referred to as SimCCS-Time, which was later used and adapted in other studies [Yaw et al., 2021, Jones et al., 2022]. Given the importance of multiperiod CCS planning problems, this paper focuses on SimCCS-Time. For reviews on different CCS models, we refer the reader to Tapia et al. [2018] and Zhang et al. [2022].

6.2.2. Related problem groups

SimCCS-Time combines facility location and network design decisions and therefore shares similarities with other problems in the Operations Research field. For example, source and sink activation are classical facility location decisions [see, e.g., Laporte and Nickel, 2015]. The facility decisions in SimCCS-Time are more complex than the ones found in traditional facility location problems: there are decisions on source capture processes, reservoirs subsites and digging of wells on reservoir subsites. Multiperiod CCS planning problems also have connections with multiperiod facility location problems, and the ability to expand a CCS network over time has similarities to multiperiod facility location problems where the capacity

level of facilities can be changed over time [Jena et al., 2015]. Pipeline activation and flow decisions are present in network design problems [Crainic et al., 2021] and multiperiod network design problems [Fragkos et al., 2021]. Furthermore, the combination of these two decision groups are present in combined facility location and network design problems [e.g., Melkote and Daskin, 2001a,b, Contreras and Fernández, 2012]. Other related problems that combine facility location decisions with some form of transportation of goods are hub location problems [combined hub location and network decisions, see, e.g., Contreras and O'Kelly, 2019], and location routing problems [combined facility location and routing decisions, see, e.g., Nagy and Salhi, 2006]. Particularly, SimCCS-Time generalizes the facility location problem and the single-commodity network design problem. As such, it is NP-hard. The closest planning problem to SimCCS in the literature is, to the best of our knowledge, the combined facility location and network design problem of Melkote and Daskin [2001a]. The authors propose valid inequalities to strengthen the LP relaxation of the proposed formulation. However, in their formulation, the demand of sources and sinks is fixed, as opposed to SimCCS (and SimCCS-Time), where the demands at sources (capture rates) and sinks (injection rates) are decision variables that are linked by a global annual CO_2 capture target constraint. Furthermore, their model is single-period, and does not allow for the representation of capture processes, reservoir subsites, and wells.

6.2.3. Computational challenges: valid inequalities and model approximations

In the original version of SimCCS and SimCCS-Time, pipelines have discrete capacities, which may pose considerable computational challenges to solve the problem: the industry-standard pipeline diameters in North America (NPS) has 25 pipeline diameters ranging from 4 to 48 inches, and models with 25 pipeline sizes are unlikely to be solved within a reasonable time [Middleton, 2013]. For example, in Middleton [2013], a case study on the Gulf Coast region with 24 sources and 29 reservoirs cannot be solved by parallel CPLEX (with 12 cores) within a day. To improve model solution time, Lobo [2017] studied valid inequalities to improve the SimCCS linear relaxation bounds. Alternatively, to reduce model size and solution time, Middleton [2013] proposed a single-period model approximation where the cost of different pipeline capacities is approximated by a piecewise linear function and diameters are replaced with capacity trends, i.e., the corresponding piecewise segments. The author shows that the piecewise model is, on average, two to three orders of magnitude faster to solve than the discrete pipeline capacity model, and that the approximation error is rather small: they are considerably lower than the uncertainty ranges when estimating pipeline costs. SimCCS-Time has also been adapted to represent piecewise linear costs for pipelines [Yaw et al., 2021, Jones et al., 2022]. Given the recent spotlight given to piecewise pipeline

capacities for SimCCS-Time, the present study focuses on the latest version of SimCCS time with piecewise pipeline capacities [i.e., the version of Jones et al., 2022]. Despite the computational benefits associated with the piecewise SimCCS-Time model, Jones et al. [2022] highlight strategies to mitigate the computational effort required to solve multiperiod problems, such as limiting the number of sources and sinks considered, and by limiting the number of time periods. However, the authors expect that in many cases, single-period models will have to be used due to the sheer size of multiperiod formulations.

6.2.4. Heuristics for CCS planning problems

To generate high-quality solutions quickly, Whitman et al. [2021] proposed three heuristics for a single-period CCS problem with discrete pipeline capacities. Namely, the authors proposed a greedy shortest-path heuristic, a slope scaling heuristic with diversification and intensification features [based on the network design slope scaling heuristic of Crainic et al., 2004], and a hybrid heuristic that combines the first two heuristics. The authors' computational experiments show that the proposed heuristics execute much faster than CPLEX, but result in inferior solution quality. Specifically, costs can be up to 20.92% higher than the costs of the solutions provided by CPLEX. In d'Amore and Bezzo [2017], the authors propose a two-stage heuristic for a multiperiod and multi-modal CCS problem. The heuristic relaxes the binary transportation variables in the first stage. The CO_2 capture decisions obtained from the first stage are then used to build a reduced second-stage model with binary transportation variables. To the best of our knowledge, there is only one study that proposes a heuristic for SimCCS-Time: Jones et al. [2022] use a single-period CCS model to generate decisions for SimCCS-Time. This heuristic falls into the category of rolling horizon heuristics, which were previously used on multiperiod network design problems [e.g., see Papadimitriou and Fortz, 2015]. As pointed out Jones et al. [2022], such a heuristic cannot anticipate future demand by building more infrastructure in advance, and therefore may plan a CCS infrastructure that opens additional infrastructure as needed, which may be more costly than overprovisioning infrastructure (e.g., building pipelines with more capacity than what is immediately needed).

Our work introduces a multiperiod slope scaling heuristic for CCS VCO based on the single-period slope scaling heuristics for CCS [Whitman et al., 2021] and network design [Crainic et al., 2004], using piecewise linear costs for pipelines instead of discrete pipeline capacities [e.g., namely, the version of SimCCS-Time introduced by Jones et al., 2022]. In addition to the extension of the slope scaling heuristic to a multiperiod problem, we introduce novel heuristic components, such as the generation of improved upper bounds through dynamic programming and the implicit representation of pipeline capacity cost functions. The computational experiments show that the proposed heuristic does not only

generate solutions at a fraction of the time required by CPLEX, but that these solutions are, on average, of higher quality than those found by CPLEX.

6.3. Problem Definition

In this section we describe the multiperiod strategic planning problem for CCS pipeline networks. In a given geographical region of interest, we consider a set of CO_2 emitters (sources) from which CO_2 could be captured and a set of potential sinks (geological sites) for long term underground storage of CO_2. Each source can contain multiple CO_2 capture units and each sink can contain multiple subsites (stacked reservoirs), and each subsite can contain several injection wells. We assume that a candidate pipeline network embedding all potential pipeline arcs is given. Such a network can be obtained by constructing a rasterized cost map of the region of interest and generating corresponding least-cost paths (Middleton et al., 2012a; Hoover et al., 2019a-b).

Infrastructure for CCS will evolve over time. As such, the planning horizon usually spans decades (e.g. up to 2050, or beyond). It is discretized in a finite set of time periods, possibly of varying duration, each typically lasting at least one year. For each period, an annual CO_2 capture target must be met for the whole region of interest (i.e. cumulative over all sources). These targets can reflect increasing CO_2 capture incentives to meet a net-zero CO_2 emissions objective. Many factors can influence the number and duration of time periods. These include the dynamics of carbon price and capture target forecasts, as well as construction cycles and infrastructure build times. Moreover, the duration of time periods also depends on the degree of confidence in economic and market forecasts. Longer periods are typically favored when uncertainty levels increase. As in similar studies [Middleton et al., 2012b, Yaw et al., 2021, Jones et al., 2022], we neglect infrastructure build times. In other terms, infrastructure can be operated immediately when it is introduced.

One of the main decisions to be determined is which sources, sinks and pipeline arcs should be activated (and when) in order to meet CO_2 targets. In practical terms, source activation corresponds to retrofitting a CO_2-emitting industrial process with a capture technology. We use the term capture unit to represent a CO_2 emitting process at a given source retrofitted with a capture technology, including separation, purification, compression and drying to adjust the CO_2 temperature, pressure and composition. Pipeline activation corresponds to choosing a pipeline capacity to be built on an arc in the candidate network. Recall that we allow for parallel pipelines to be built on a given arc, so as to reflect the trade-off between building an oversized pipeline as soon as possible and adding capacity incrementally over time. As in similar studies, we do not consider the repurposing of existing oil and gas pipelines to transport CO_2 in the gas phase [Cauchois et al., 2021]. We solely consider building new CO_2 pipelines (supercritical phase) and therefore do not distinguish between CO_2 phases. Sink

activation corresponds to doing prospective studies of a given geological formation, digging wells and installing injection equipment. Each activation decision has a corresponding fixed cost, corresponding to fixed O&M (operation and maintenance) and capital costs.

Once infrastructure is built, one must determine for each time period the capture rate of capture units at sources, the flow of CO_2 in pipelines and the injection rate at storage sites (i.e. sinks). Each of these decisions has a corresponding variable cost, corresponding to variable O&M and capital costs.

Each aforementioned decision occurs at each year within each time period. A common simplifying assumption, which we also make in this study, is to assume that for each year within a time period, all decisions are identical [e.g., see Middleton et al., 2012b]. This improves computational tractability.

The objective of the decision problem is to determine source, sink and pipeline decisions over the whole time horizon in order to define a CCS infrastructure that satisfies the CO_2 capture targets at each time period while minimizing the fixed and variable costs for capture, storage and pipeline transportation. In the following section, we introduce a mixed-integer programming model for the multiperiod strategic planning of CCS infrastructure.

6.3.1. Application context

In a CCS supply chain, CO_2 is captured from emitting processes at industrial sites in various sectors, such as steel, cement, pulp and paper, bioenergy, fertilizer and chemicals, combustion-based power, oil and gas, as well as fossil fuel-based hydrogen generation. Capture technologies fall into three broad categories: post-combustion, pre-combustion, and oxyfuel combustion. The industrial combustion of fossil fuels results in CO_2-rich gases, called flue gases. As its name indicates, post-combustion capture separates CO_2 from flue gases after the combustion process. Alternatively, pre-combustion capture takes place before combustion. It consists in mixing fuel with air (or oxygen) and using steam to produce H_2 and CO_2, which are subsequently separated. Oxyfuel combustion replaces the air necessary for combustion by air with a high oxygen concentration ($\sim 95\%$). This results in less exhaust gas and a high CO_2 concentration ($\sim 80 + \%$). For each of these broad categories, several technologies can be considered. The specific choice of technology depends on the nature of the emitting process, namely on gas composition, as well as on economic and technological considerations, all of which fall outside the scope of this work [Bui et al., 2018, Baylin-Stern and Berghout, 2021].

After being captured, CO_2 can be transported via a mix of pipelines, ships, trains and trucks to eventual storage locations. The choice of transport mode depends on costs, on the volumes to be transported, and on geographic considerations. In the beginning stages of CCS projects, transport modes need to be flexible and scalable. Vehicle transportation can offer

that flexibility and adapt to the initial ramp-up of transported volumes. Pipelines offer the possibility of transporting large volumes of pressurized CO_2 to points linked to hubs and trunklines, but infrastructure takes a longer time to build. Trains can be an economically sound alternative to pipelines when smaller volumes need to be transported to remote regions, i.e. far from pipeline trunk lines. Ships and offshore pipelines (on the seabed floor) are the main transportation options available when storing CO_2 in offshore geological formations. Economic analysis shows that above a certain distance threshold, ranging from 500 to 1000 km depending on vessel tank pressure, offshore pipelines become more costly than ships [Orchard et al., 2021]. Trucks can offer a backup solution to increase inland network robustness, namely when pipeline links become temporarily unavailable due to failures (e.g. corrosion). In order to eventually capture, transport and sequester the annual gigatons of CO_2 necessary to meet net-zero targets, pipeline infrastructure may need to grow in many countries around the world, including in the US [Middleton et al., 2021].

The state in which CO_2 is transported depends on the choice of transportation mode. If repurposed oil and gas pipelines are considered, a recent survey indicates that pipeline operators favor transporting CO_2 in gaseous phase due to engineering and safety considerations [Cauchois et al., 2021]. On the other hand, when building new pipelines for CO_2 transportation, the supercritical phase is favored, in part due to its advantageous viscosity, density and low rate of corrosion [IPCC, 2005]. Transportation in railcars, ships and trucks is done in cryogenic tanks containing liquid CO_2. In this study, we focus on transporting large volumes of CO_2 through pipelines.

After being transported to its destination, CO_2 is injected underground to be stored inside geological formations for 10,000+ years [Alcalde et al., 2018]. Both depleted oil fields and saline aquifers will play a role in geological CO_2 storage. The former offers the advantage of having well-characterized geology coming from years of operations. On the other hand, saline aquifers offer the advantage of having no existing wells that need to be cemented (for pressure control and leak prevention). In North America, saline aquifers may constitute up to 95% of available CO_2 storage capacity [Middleton et al., 2020].

6.4. Mathematical formulation

We now describe the mathematical formulation for the multiperiod strategic planning problem for CCS pipeline networks. This model is a variant of the multiperiod combined facility location and network design model for CCS planning of Jones et al. [2022] obtained by adding capture units [Middleton et al., 2012b], reservoir subsites [Ellett et al., 2017] and wells [Middleton et al., 2012b] together with their corresponding binary or integer variables. CCS infrastructure can be built on a planning horizon (e.g., up to 2050) that is discretized into n time periods $H = \{1, \ldots, n\}$. Each period h has a duration of O_h years. As such,

the total number of years of the planning horizon is $\sum_{h \in H} O_h$. For each period $h \in H$, there is an annual CO_2 capture target CapCO2_h (in tonnes/year) to be satisfied. During that planning horizon, there are three groups of decisions to be made: pipeline-related decision, source-related decisions, and reservoir-related decisions, as described next.

Pipeline-related decisions. As initially proposed by Middleton [2013] and later used by Jones et al. [2022], pipeline costs are approximated with piecewise linear functions, and it is assumed that there are no variable costs for transporting CO_2 [as in Middleton, 2013, Jones et al., 2022]. Let (I, K) be a directed graph where $I = S \cup R \cup I_0$ is the union set of sources S, reservoirs R and junction nodes I_0, and K is a set of arcs. An arc $(i, j) \in K$ has origin $i \in I$ and destination $j \in I$. For each node $i \in I$, let $\delta^-(i) \subseteq K$ be the set of incoming arcs for i, and $\delta^+(i) \subseteq K$ the set of outgoing arcs for i. Let C be a set of pipeline capacity trends and $K_C = \{ (i, j, c), \forall (i, j) \in K, c \in C \}$. The construction of a pipeline on arc k with capacity trend c at period h is represented by a binary variable y^{P}_{kch}. A continuous variable p_{kch} represents the capacity chosen for this arc (in tonnes/year), which is bounded by Q^{max}_{kc} (from above). If this arc is activated during period $h \in H$ with capacity p_{kch}, then a cost of $\alpha_{kch} p_{kch} + \beta_{kch}$ is paid (in dollars). A continuous variable x_{kh} represents the flow of CO_2 (in tonnes/year) on arc k during time period h.

Source-related decisions. We consider a set of sources S that emit CO_2. These sources have one or more CO_2 capture processes that can be activated. Let G denote the set of these source capture processes, where a tuple $(i,g) \in G$ denotes source i with capture process g. If activated, this capture process has a maximum yearly CO_2 capture rate (capacity) of Q^{S}_{ig} (in tonnes/year). For each period $h \in H$, a binary variable y^{S}_{igh} is associated with the activation of the capture process g at source i during the time period h. Activating capture process (i, g) at time period h (i.e., $y^{\text{S}}_{igh} = 1$) incurs a fixed cost F^{S}_{igh} (in dollars), making the capture process available for CO_2 capture from period h onwards. The activation of a capture process involves the retrofitting of a CO_2 emitting process such that CO_2 can be captured from that process. A non-negative variable a_{igh} specifies the amount of CO_2 captured (in tonnes/year) at this capture process during period h, and it is associated with a variable cost V^{S}_{igh} (in dollars/tonne).

Reservoir-related decisions. Let R be a set of reservoirs and U be a set of subsites at these reservoirs. For each $(j,u) \in U$ and period $h \in H$, let y^{R}_{juh} be a binary variable associated with the activation of the subsite u at reservoir j during the time period h. Activating a reservoir subsite (j, u) at time period h (i.e., $y^{\text{R}}_{juh} = 1$) incurs a fixed cost F^{R}_{juh} (in dollars), making the subsite available for digging CO_2 injection wells from period h onwards. Let w_{juh} be an integer variable associated with the number of injection wells dug into reservoir subsite (j, u) at period h. Digging an injection well at reservoir subsite (j, u) at time period h incurs a fixed cost F^{W}_{juh} (in dollars), making the well available for injection from period h onwards. A non-negative variable b_{juh} specifies the amount of CO_2 injected at reservoir subsite (j, u) at

time period h, and it is associated with a variable cost V^{R}_{juh} (in dollars/tonne). This reservoir subsite has a maximum yearly CO_2 injection rate of Q^{R}_{ju} (in tonnes/year), and a lifetime capacity of Q^{RES}_{ju} tonnes (that is, the total CO_2 storage capacity over the whole planning horizon). The maximum number of wells that can be dug at this reservoir is P^{W}_{ju}, and each well has a yearly CO_2 injection rate of Q^{W}_{ju} (in tonnes/year). Such multiperiod combined network design and facility location problem for CCS can be formulated as follows:

$$\min \sum_{h\in H,(i,g)\in G}(F^{\mathrm{S}}_{igh}y^{\mathrm{S}}_{igh} + V^{\mathrm{S}}_{igh}a_{igh}) + \sum_{h\in H,(j,u)\in U}(F^{\mathrm{R}}_{juh}y^{\mathrm{R}}_{juh} + F^{\mathrm{W}}_{juh}w_{juh} + V^{\mathrm{W}}_{juh}b_{juh}) + \sum_{h\in H,(k,c)\in K_C}(\alpha_{kch}p_{kch} + \beta_{kch}y^{\mathrm{P}}_{kch}) \tag{6.4.1}$$

$$\sum_{h\in H} y^{\mathrm{S}}_{igh} \leq 1 \qquad \forall (i,g) \in G \tag{6.4.2}$$

$$a_{igh} \leq Q^{\mathrm{S}}_{ig} \sum_{\tau=1}^{h} y^{\mathrm{S}}_{ig\tau} \qquad \forall h \in H, (i,g) \in G \tag{6.4.3}$$

$$\sum_{(i,g)\in G} a_{igh} = \mathrm{CapCO2}_h \qquad \forall h \in H \tag{6.4.4}$$

$$\sum_{h\in H} y^{\mathrm{R}}_{juh} \leq 1 \qquad \forall (j,u) \in U \tag{6.4.5}$$

$$b_{juh} \leq Q^{\mathrm{R}}_{ju} \sum_{\tau=1}^{h} y_{ju\tau} \qquad \forall h \in H, (j,u) \in U \tag{6.4.6}$$

$$b_{juh} \leq Q^{\mathrm{W}}_{ju} \sum_{\tau=1}^{h} w_{ju\tau} \qquad \forall h \in H, (j,u) \in U \tag{6.4.7}$$

$$\sum_{\tau=1}^{h} w_{ju\tau} \leq P^{\mathrm{W}}_{ju} \sum_{\tau=1}^{h} y^{\mathrm{R}}_{ju\tau} \qquad \forall h \in H, (j,u) \in U \tag{6.4.8}$$

$$\sum_{h\in H} O_h b_{juh} \leq Q^{\mathrm{RES}}_{ju} \sum_{h\in H} y^{r}_{juh} \qquad \forall (j,u) \in U \tag{6.4.9}$$

$$p_{kch} \leq Q^{max}_{kc} y^{\mathrm{P}}_{kch} \qquad \forall h \in H, (k,c) \in K_C \tag{6.4.10}$$

$$x_{kh} \leq \sum_{\tau=1}^{h} \sum_{c\in C} p_{kc\tau} \qquad \forall h \in H, k \in K \tag{6.4.11}$$

$$\sum_{k\in\delta^-(i)} x_{kh} - \sum_{k\in\delta^+(i)} x_{kh} = \begin{cases} \sum_{(i,g)\in G} a_{igh} \text{ if } i \in S, \\ -\sum_{(j,u)\in U} b_{juh} \text{ if } i \in R, \\ 0 \text{ otherwise.} \end{cases} \qquad \forall h \in H, i \in I \tag{6.4.12}$$

$$y^{\mathrm{S}}_{igh} \in \{0,1\}, a_{igh} \in \mathbb{R}_{\geq 0} \qquad \forall h \in H, (i,g) \in G \tag{6.4.13}$$

$$y^{\mathrm{R}}_{juh} \in \{0,1\}, w_{juh} \in \mathbb{Z}_{\geq 0}, b_{juh} \in \mathbb{R}_{\geq 0} \qquad \forall h \in H, (j,u) \in U \tag{6.4.14}$$

$$y^{\mathrm{P}}_{kch} \in \{0,1\}, p_{kch} \in \mathbb{R}_{\geq 0}, x_{kh} \in \mathbb{R}_{\geq 0} \qquad \forall h \in H, (k,c) \in K_C \tag{6.4.15}$$

Objective (6.4.1) minimizes, for the whole planning horizon, the fixed source activation costs, the variable source CO_2 capture costs, the fixed reservoir activation costs, the fixed reservoir well digging costs, the variable reservoir CO_2 injection costs, and the fixed pipeline activation costs. Costs are discounted to present value and amortized. This is further explained in Section 6.6, and it implies that all costs are non-increasing over time (e.g., $F^{\mathrm{S}}_{ig1} \geq F^{\mathrm{S}}_{ig2}$). Opening all infrastructure at the beginning of the planning horizon is therefore unlikely to be optimal.

Source-related constraints. Constraints (6.4.2) ensure that a source can only be activated once during the planning horizon. Constraints (6.4.3) are the capacity constraints for the capture rate of CO_2 at the sources. Constraints (6.4.4) impose the annual CO_2 capture target for each time period.

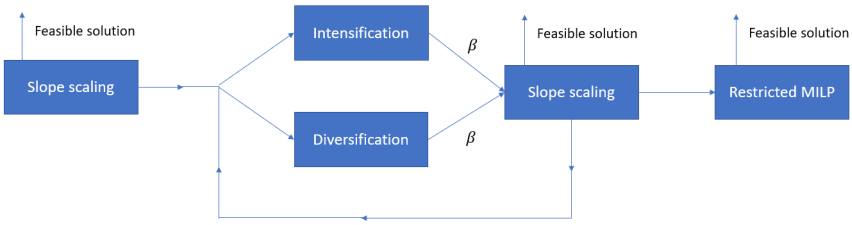
Reservoir-related constraints. Constraints (6.4.5) ensure that a reservoir subsite can only be activated once during the planning horizon. Constraints (6.4.6) and (6.4.7) are capacity constraints for the injection rate of CO_2 at the reservoirs. Constraints (6.4.8) are capacity constraints for the maximum number of wells that can be dug at reservoirs. Constraints (6.4.9) are the reservoir lifetime capacity constraints. Multiplying the right hand side of these constraints by the reservoir activation variables is not necessary to obtain a valid formulation, but it strengthens the bound of the linear programming relaxation.
Pipeline-related constraints. Constraints (6.4.10) set the upper pipeline capacity for each linear trend $c \in C$. Constraints (6.4.11) limit the flow of pipelines to their maximum capacities. Constraints (6.4.12) are flow conservation constraints for sources, reservoirs and junction nodes.

To improve the linear programming relaxation bound (without removing any optimal integer solution), the following coefficients are tightened based on the largest annual CO_2 capture target: $Q^{\mathrm{S}}_{ig}, \forall (i,g) \in G, Q^{\mathrm{R}}_{ju}, \forall (j,u) \in U, Q^{\mathrm{W}}_{ju}, \forall (j,u) \in U$, and $Q^{max}_{kc}, \forall (k,c) \in K_C$. The model above is CO_2 phase agnostic (liquid, gas, supercritical) and it is assumed that all pipeline arcs are new. Moreover, infrastructure build times are not considered and new infrastructure (pipelines, capture units, reservoir subsites) is assumed to be immediately available at the beginning of the time period it is opened. In order to reduce the number of variables and equations, for each time period, each year is assumed to be identical. Hence, the model treats CO_2 flows, capture rates and injection rates as annualized and indexes them by time period (as opposed to year).

6.5. A Slope Scaling Heuristic

We now describe a heuristic based on slope scaling (SS). Slope scaling is an iterative heuristic that solves several approximations of an optimization problem, and was first presented by Yaged [1971] to solve network optimization problems. At each slope scaling iteration, feasible solutions to the original problem are generated based on the approximate solution. We propose a novel way to generate feasible solutions within slope scaling by using dynamic programming (see Section 6.5.4). The heuristic framework is outlined in Figure 6.1. The heuristic runs slope scaling several times, alternating between intensification and diversification phases (described in Section 6.5.5), as proposed by Crainic et al. [2004] for a network design problem, and later used by Whitman et al. [2021] for a single-period CCS planning problem. Furthermore, a final refining phase solves a restricted MILP to improve the best-known solution found by slope scaling, described in Section 6.5.6.

Fig. 6.1. Outline of the heuristic framework.



Being a heuristic, slope scaling has no optimality guarantees, and it may converge to a local optimum instead of a global optimum. To improve the chances of slope scaling finding better local optima or even global optima, Crainic et al. [2004] proposed Lagrangian perturbation and intensification and diversification techniques for slope scaling. Slope scaling has also been used for combined facility location and network design problems: Whitman et al. [2021] proposed a slope scaling heuristic for a single-period CCS problem with discrete pipeline capacities. Their heuristic has intensification and diversification phases as in Crainic et al. [2004]. Slope scaling heuristics have also been used to generate feasible solutions for exact methods, such as Lagrangian decomposition [e.g., Kadri et al., 2022], or in matheuristics such as the combined column generation and slope scaling of Zetina et al. [2019]. The general framework of a slope scaling heuristic is outlined in Algorithm 6.1:

Algorithm 6.1: General framework of a slope scaling heuristic.

```
1 initialize coefficients;
2 while stopping criteria not met do
3     solve approximation of original problem;
4     update approximation coefficients;
5     compute feasible solution;
6 return best feasible solution found;
```

At each slope scaling iteration, an approximation of the original optimization problem is solved. In this approximation, design variables are removed and their approximate activation costs are added to the coefficients of the continuous variables that are linked to these design variables (i.e., pipeline capacity). After solving the simplified problem, the objective function coefficients of these approximations are adjusted to reflect the true cost of the obtained solution. If the next iteration generates exactly the same solution, the approximation costs equate to the original design costs of the solution. Furthermore, a feasible solution is computed within each iteration based on the approximate solution. The procedure stops when the objective function value of the approximation repeats in two consecutive iterations.

6.5.1. Selection of design variables

When developing slope scaling heuristics, usually all design variables are removed from the formulation such that the resulting model is a linear program. This allows for a quick reoptimization of the formulation when the approximate objective function coefficients change with the slope scaling iterations. For example, in Crainic et al. [2004], the authors approximate the binary arc activation variables, and the resulting formulation is a linear multi-commodity minimum cost network flow problem. On the slope scaling for single-period CCS of Whitman et al. [2021], besides approximating pipeline design variables, the authors approximate source and sink design variables, which also leads to a linear program. However, our preliminary experiments have shown that the pipeline activation decisions are the main decisions related to the hardness of the problem. We therefore decided to develop a slope scaling heuristic that only approximates pipeline decisions (i.e., variables y^{P}_{kch} are removed), while keeping the integrality constraint for source and sink design variables. Therefore, our slope scaling approximation is still a MILP problem. Such formulation leads to stronger approximations (i.e., better feasible solutions), but implies longer solution times. Our preliminary experiments have shown, however, that for the same amount of CPU time, a MILP approximation with early stopping leads to better overall solutions than a LP approximation. We describe later in this section which stopping criteria are used for the MILP approximation.

To reduce the SS model size, we remove the pipeline capacity trend decisions from the SS formulation and compute them a posteriori, based on the SS solution capacity. Thus, the SS pipeline capacity variables p_{kch} are not indexed by the capacity trend (i.e., p_{kh}). This leads to smaller models that may be faster to solve. Furthermore, our preliminary experiments have shown that the slope scaling with implicit pipeline capacities implies better overall solutions. This contrasts with the existing literature: the slope scaling heuristic proposed by Whitman et al. [2021] represents different pipeline capacities explicitly within their SS formulation. We expect that similar improvements would be observed if such discrete pipeline capacities are represented implicitly. A limitation of representing the pipeline capacity (or capacity trend) implicitly is that it does not allow for the activation of two pipelines with different capacities at the same time period (as opposed to model (6.4.1)–(6.4.15)). We assume, however, that the capacity of the largest pipeline trend is arbitrarily large, as in Jones et al. [2022]. Furthermore, we assume that building a larger pipeline at a certain period is no more expensive than building two smaller pipelines with the same capacity at the same period. Thus, under these assumptions, the implicit representation of pipeline trends does not remove the optimal solution to the original problem from the slope scaling search space.

The slope scaling model is defined below:

$$SS(\tilde{\beta}) = \min \sum_{h\in H,(i,g)\in G}(F^{\mathrm{S}}_{igh}y^{\mathrm{S}}_{igh} + V^{\mathrm{S}}_{igh}a_{igh}) + \sum_{h\in H,(j,u)\in U}(F^{\mathrm{R}}_{juh}y^{\mathrm{R}}_{juh} + F^{\mathrm{W}}_{juh}w_{juh} + V^{\mathrm{W}}_{juh}b_{juh}) + \sum_{h\in H,k\in K}\tilde{\beta}_{kh}p_{kh} \tag{6.5.1}$$

$$\sum_{h\in H} y^{\mathrm{S}}_{igh} \le 1 \qquad \forall (i,g)\in G \tag{6.5.2}$$

$$a_{igh} \le Q^{\mathrm{S}}_{ig}\sum_{\tau=1}^{h} y^{\mathrm{S}}_{ig\tau} \qquad \forall h\in H,(i,g)\in G \tag{6.5.3}$$

$$\sum_{(i,g)\in G} a_{igh} = \mathrm{CapCO2}_h \qquad \forall h\in H \tag{6.5.4}$$

$$\sum_{h\in H} y^{\mathrm{R}}_{juh} \le 1 \qquad \forall (j,u)\in U \tag{6.5.5}$$

$$b_{juh} \le Q^{\mathrm{R}}_{ju}\sum_{\tau=1}^{h} y_{ju\tau} \qquad \forall h\in H,(j,u)\in U \tag{6.5.6}$$

$$b_{juh} \le Q^{\mathrm{W}}_{ju}\sum_{\tau=1}^{h} w_{ju\tau} \qquad \forall h\in H,(j,u)\in U \tag{6.5.7}$$

$$\sum_{\tau=1}^{h} w_{ju\tau} \le P^{\mathrm{W}}_{ju}\sum_{\tau=1}^{h} y^{\mathrm{R}}_{ju\tau} \qquad \forall h\in H,(j,u)\in U \tag{6.5.8}$$

$$\sum_{h\in H} O_h b_{juh} \le Q^{\mathrm{RES}}_{ju}\sum_{h\in H} y^{r}_{juh} \qquad \forall (j,u)\in U \tag{6.5.9}$$

$$p_{kh} \le \max_{c\in C} Q^{max}_{kc} \qquad \forall h\in H,k\in K \tag{6.5.10}$$

$$x_{kh} \le \sum_{\tau=1}^{h} p_{k\tau} \qquad \forall h\in H,k\in K \tag{6.5.11}$$

$$\sum_{k\in\delta^-(i)} x_{kh} - \sum_{k\in\delta^+(i)} x_{kh} = \begin{cases} \sum_{(i,g)\in G} a_{igh} \text{ if } i\in S, \\ -\sum_{(j,u)\in U} b_{juh} \text{ if } i\in R, \\ 0 \text{ otherwise.} \end{cases} \qquad \forall h\in H, i\in I \tag{6.5.12}$$

$$y^{\mathrm{S}}_{igh} \in \{0,1\}, a_{igh} \in \mathbb{R}_{\ge 0} \qquad \forall h\in H,(i,g)\in G \tag{6.5.13}$$

$$y^{\mathrm{R}}_{juh} \in \{0,1\}, w_{juh} \in \mathbb{Z}_{\ge 0}, b_{juh} \in \mathbb{R}_{\ge 0} \qquad \forall h\in H,(j,u)\in U \tag{6.5.14}$$

$$p_{kh} \in \mathbb{R}_{\ge 0}, x_{kh} \in \mathbb{R}_{\ge 0} \qquad \forall h\in H,k\in K \tag{6.5.15}$$

Here, $\tilde{\beta}_{kh}$ approximates the activation costs of pipelines through the continuous pipeline capacity variables p. The resulting model is a mixed-integer linear programming problem with a minimum-cost flow structure for the pipeline decisions. While such model is not a linear program and therefore cannot be solved in polynomial time, our preliminary computational experiments have shown that under the same CPU time, the overall SS heuristic with a MILP

approximation leads to better solutions than SS with a linear programming approximation. This is due to the stronger approximations provided by the MILP, which generate better feasible solutions.

As each iteration, SS solves a MILP, which may have an unpredictable solution time. As each iteration of the SS is an approximation to the original problem with very specific approximation coefficients, it is more favorable to quickly generate solutions rather than solving this MILP to optimality. Hence, we use two stopping criterias for that MILP: a maximum number of feasible solutions found, and an optimality gap threshold. Particularly, using the number of feasible solutions as a stopping criteria allows the SS iterations to be flexible enough to stop early even if finding small optimality gaps for some instances is challenging, which is the case for many of our instances.

6.5.2. A note on discrete pipeline sizes.

We further note that the pipeline capacity trends are implicitly represented through the SS coefficients. Thus, the size of the SS formulation does not depend on the number of capacity trends (i.e., $|C|$). If discrete pipeline capacities were used on the original MILP formulation, the model size would increase, as well as the computational effort necessary to solve such model [e.g., see Middleton, 2013]. On the other hand, the implicit representation of pipeline functions makes the SS flexible enough to represent discrete pipeline sizes or other pipeline cost functions without increasing its model size.

6.5.3. Update of slope scaling coefficients

Slope scaling implementations initialize their coefficients to reflect the costs of the LP relaxation of the original problem [e.g., see Crainic et al., 2004, Gendron et al., 2018]. This usually equates to the activation cost of the design variables divided by the maximum capacity associated with the infrastructure in question. In our case, as the capacity variables do not represent any specific capacity trend, we use the mean ratio to initialize the slope scaling coefficients. Let $\tilde{\beta}^t_{kh}$ be the value of the SS coefficients at iteration t for activating arc k at time period h. For the first iteration, we initialize $\tilde{\beta}^1_{kh}$ as follows:

$$\tilde{\beta}^1_{kh} = \frac{1}{|C|} \sum_{c \in C} \left(\frac{\beta_{kch}}{Q^{max}_{kc}} + \alpha_{kch}\right), \forall k \in K, h \in H.$$

We now describe the update of these coefficients. Let $(\bar{y}^{\mathrm{S}}, \bar{a}, \bar{y}^{\mathrm{R}}, \bar{w}, \bar{b}, \bar{p}, \bar{x})$ be a solution to $SS(\tilde{\beta}^t)$. We want to set the values of the coefficients $\tilde{\beta}^{t+1}_{kh}$ such that if the same solution is found again, then the objective value of the slope scaling model reflects the true value of the corresponding integer solution on the original problem with binary design variables. To this

end, we update the coefficients for the next SS iteration as below:

$$\tilde{\beta}_{kh}^{t+1} = \begin{cases} \beta_{kc^\star h}/\bar{p}_{kh}^{t} + \alpha_{kc^\star h}, \text{ if } \bar{p}_{kh}^{t} > 0, \\ \tilde{\beta}_{kh}^{t}, \text{ otherwise,} \end{cases} \tag{6.5.16}$$

where:

$$c^\star = min\{c \in C : \bar{p}_{kh}^{t} \leq Q_{kc}^{max}\}. \tag{6.5.17}$$

Therefore, if $\bar{p}_{kh}^{t+1} = \bar{p}_{kh}^{t}$ (i.e., if the same pipeline capacity decisions are taken in the next SS iteration), then:

$$\tilde{\beta}_{kh}^{t+1}\bar{p}_{kh}^{t+1} = \begin{cases} \beta_{kc^\star h} + \alpha_{kc^\star h}\bar{p}_{kh}^{t+1}, \text{ if } \bar{p}_{kh}^{t} > 0, \\ 0, \text{ otherwise,} \end{cases} \tag{6.5.18}$$

which reflects the pipeline activation costs for the original formulation that SS approximates.

6.5.4. Generation of feasible solutions

Based on the solution found by the SS, a solution to the original problem is generated in the following manner: we retain the solution values for both sources and sinks $(\bar{y}^{\mathrm{S}}, \bar{a}, \bar{y}^{\mathrm{R}}, \bar{w}, \bar{b})$, as they are feasible in the original formulation. We also retain the flow values $\bar{x}$, as they are also feasible in the original formulation. Finally, what is left to decide is the value of the pipeline activation variables y_{kch}^{P} and the capacity $\bar{p}$. Usually, slope scaling implementations [e.g., Whitman et al., 2021] keep the same pipeline capacities $\bar{p}$ found by SS and directly derive binary arc decisions based on the pipeline capacities $\bar{p}$ (or the flow $\bar{x}$), as below:

$$y_{kch}^{\mathrm{P}} = \begin{cases} 1 \text{ if } \bar{p}_{kch} > 0, \\ 0 \text{ otherwise.} \end{cases}$$

This assumes, however, that the pipeline capacity variables are indexed by the capacity trend (i.e. $\bar{p}_{kch}$), which requires a larger model. Even if such model is used, the SS approximation coefficients may generate a solution where suboptimal capacity trends are selected. If pipeline capacities are not indexed by capacity trend (i.e. $\bar{p}_{kh}$, which is the case for the here proposed SS), the generation of feasible solutions can be improved by inspecting, for each arc, the cost of each capacity trend that allows for a pipeline with capacity $\bar{p}_{kh}$, and opening the pipeline at the trend with lowest cost. Not only may the pipeline capacity trend be suboptimal, but also the timing decisions of when and how much capacity is opened. This is due to the lack of binary pipeline activation variables on SS. The network costs therefore may not reflect the true costs of opening pipelines over the planning horizon. To address this issue, we generate improved feasible solutions by optimizing for the pipeline capacity, its capacity trend, and its activation time period (i.e., y_{kch}^{P} and p_{kch}) while satisfying the fixed flow $\bar{x}$. This generates a

network of pipelines that is feasible for the source, sink and flow decisions previously selected. We refer to this as the capacity assignment problem.

The capacity assignment problem. For each arc $k \in K$, the pipeline decisions associated with this arc are independent of other arcs. Thus, the capacity assignment problem is decomposable by arc. Thus, for each $k \in K$, the capacity assignment problem $[CA(\bar{x})]$ is defined as:

$$[CA_k(\bar{x})] \;\; f_k^{\mathrm{CA}}(\bar{x}) = \min \sum_{c \in C} \sum_{h \in H} (\alpha_{kch} p_{kch} + \beta_{kch} y_{kch}^{\mathrm{P}}) \tag{6.5.19}$$

$$p_{kch} \leq Q_{kc}^{max} y_{kch}^{\mathrm{P}} \qquad \forall h \in H, c \in C \tag{6.5.20}$$

$$\sum_{\tau=1}^{h} \sum_{c \in C} p_{kc\tau} \geq \bar{x}_{kh} \qquad \forall h \in H \tag{6.5.21}$$

$$y_{kch}^{\mathrm{P}} \in \{0, 1\}, p_{kch} \in \mathbb{R}_{\geq 0} \qquad \forall h \in H, c \in C. \tag{6.5.22}$$

By splitting decisions into two phases (flow decisions and capacity assignment decisions), our generation of upper bounds has some similarities to select-and-time heuristics for multi-period problems [e.g., Fragkos et al., 2021], where decisions are first *selected* regardless of the time period (e.g., how much pipeline capacity to open in total), and then *timed* (e.g., when to open the pipeline capacity). In our case, we disregard the initial pipeline capacity timing decisions, and find new timing decisions based on the selected flow of CO_2 for each time period. While our selecting decisions (x_{kh}) do include the timing of flow decisions (and therefore they are not identical to the decisions in traditional select-and-time heuristics), there is an additional layer of timing decisions concerning when pipelines are built (y_{kch}^{P} and p_{kch}).

Solving $CA_k(\bar{x})$ for all pipelines $k \in K$ can be time intensive, as it is a multiperiod problem. This can be sped up as follows. First, such problem has a trivial optimal solution of 0 if no flow is ever passed on an arc. Thus, $CA_k(\bar{x})$ only has to be optimized for arcs that pass flow. This may still be time intensive, as it has to be solved for all SS iterations. Several different heuristic decisions can be applied to find feasible solutions to $CA_k(\bar{x})$: it may be advantageous to build all pipeline capacity in advance, or to greedily build pipeline capacity over time, as needed. The latter heuristic would require iterating over the first time period until the last one, and would only build additional pipeline capacity when necessary. In other words, spare capacity would never be built in advance. Both strategies have shortcomings: building all capacity in advance may not take advantage of discounting (i.e., building later is cheaper). On the other hand, greedily building pipelines may risk building more pipelines than necessary, which would incur high activation costs.

To address the shortcomings of these heuristics, we propose a depth-first search dynamic programming algorithm for the capacity assignment problem. Let $z(h, Q)$ be the cost for

opening pipelines from period h to the last time period. $z(h, Q)$ assumes that there is a total pipeline capacity of Q already available that was open before h. The depth-first search algorithm is defined by the recursive equation for $z(h, Q)$ below:

$$z(h,Q) = \begin{cases} 0, \text{ if } h > \max\{H\}, \\ z(h+1,Q), \text{ if } \bar{x}_{kh} = 0, \\ \min_{\Delta \in L(h,Q)} \{ f(k,h,\Delta) + z(h+1, Q+\Delta) \}, \text{ otherwise.} \end{cases} \tag{6.5.23}$$

Here,

$$L(h,Q) = \{ \bar{x}_{k\tau} - Q \mid \forall \tau \in H, \tau \geq h, \bar{x}_{k\tau} \geq Q, \bar{x}_{k\tau} \geq \bar{x}_{kh} \}, \tag{6.5.24}$$

and

$$f(k,h,\Delta) = \begin{cases} 0 \text{ if } \Delta = 0, \\ \min\limits_{c \in C, \Delta \leq Q_{kc}^{max}} \{ \beta_{kch} + \alpha_{kch}\Delta \} \text{ otherwise.} \end{cases} \tag{6.5.25}$$

At a given node (h, Q) of the search tree, three groups of decisions are possible:

(1) if the current time period exceeds that last period of the planning horizon ($h > \max\{H\}$), then no more pipelines can be built;

(2) if the flow at the current time period is zero ($\bar{x}_{kh} = 0$), then building a pipeline at this period is never optimal (assuming that pipeline costs decrease with time due to discounting);

(3) if the flow at the current time period is not zero, then adding extra pipeline capacity that satisfy this flow (at least) is necessary.

The best capacity assignment cost for all periods is given by $z(1, 0)$, and the associated solution can be obtained in $O(|H|)$ time by inspecting the costs of the dynamic programming recursion tree.

Being an enumerative procedure for each pipeline, the dynamic programming may require a non-negligible portion of the CPU time available for the entire heuristic procedure. To reduce the CPU time footprint of the dynamic programming, we employ memoization of previously-found solutions on the dynamic programming recursion tree. We also propose a pruning scheme: we keep the best upper bound for the capacity assignment cost. Being a depth-first search algorithm, this equates to the minimum cost at the leaf nodes of our search. During the depth first search, we keep the accumulated cost until the node we are currently evaluating. As the costs during the recursion are non-decreasing, we can prune the search tree if, at a certain node, the current accumulated cost is larger than the current best upper bound.

6.5.5. Long-term memory

In this section, we describe an intensification and diversification procedure for SS based on information from previous SS iterations.

Slope scaling limitations. As described by Crainic et al. [2004], slope scaling is capable of finding good solutions quickly, but might converge far from globally optimal solutions. To improve the quality of solutions found by slope scaling, the authors proposed an intensification and diversification procedure based on long-term memory. We use this procedure in our heuristic, adapted to a multiperiod problem. Based on information from previously-found solutions, the SS alternates between intensification and diversification phases. In the intensification phase, SS will favor building pipelines that were used often. In the diversification phase, SS will favor building pipelines that were rarely used, and therefore direct the search towards new regions of the search space.

The intensification and diversification is controlled by a long-term memory that tracks information from previous SS iterations. To this end, we keep track of the following information updated for each $k \in K, h \in H$:

- the number of iterations that $\bar{p}_{kh} > 0$, denoted as $\bar{p}^{\text{NZ}}_{kh}(t)$;
- the average capacity of pipeline k at period h for all iterations until t, defined as $\bar{p}^{\text{AVG}}_{kh}(t) = \frac{1}{t}\sum_{\tau=1}^{t} \bar{p}_{kh}(\tau)$.
- the maximum capacity of pipeline k at period h for all iterations until t, defined as $\bar{p}^{\text{MAX}}_{kh}(t) = \max_{\tau \in 1,\ldots,t} \{ \bar{p}_{kh}(\tau) \}$.

Let $\bar{\beta}^t$ be the values of the SS coefficients on iteration t and

$$\bar{p}^{\text{RAT}}_{kh}(t) = \begin{cases} \bar{p}^{\text{AVG}}_{kh}(t)/\bar{p}^{\text{MAX}}_{kh}(t), \text{ if } \bar{p}^{\text{MAX}}_{kh}(t) > 0 \\ 0, \text{ otherwise.} \end{cases} \tag{6.5.26}$$

If intensification is performed within this iteration, then

- $\beta^t_{kh} = \beta^t_{kh}(1 - \bar{p}^{\text{RAT}}_{kh}(t))$ if $\bar{p}^{\text{NZ}}_{kh}(t) \geq \mu^t + \sigma^t$
- $\beta^t_{kh} = \beta^t_{kh}(2 - \bar{p}^{\text{RAT}}_{kh}(t))$ if $\bar{p}^{\text{NZ}}_{kh}(t) < \mu^t$

If diversification is performed within this iteration, then

- $\beta^t_{kh} = \beta^t_{kh}(1 + \bar{p}^{\text{RAT}}_{kh}(t))$ if $\bar{p}^{\text{NZ}}_{kh}(t) \geq \mu^t + \sigma^t$
- $\beta^t_{kh} = \beta^t_{kh}\bar{p}^{\text{RAT}}_{kh}(t)$ if $\bar{p}^{\text{NZ}}_{kh}(t) < \mu^t$,

where μ^t and σ^t are the mean and standard deviation of $\{ \bar{p}^{\text{NZ}}_{kh}(t), \forall k \in K, h \in H \}$.

Algorithm outline. The slope scaling procedure with intensification and diversification is outlined below (Algorithm 6.2):

Algorithm 6.2: SS with intensification and diversification

Result: best known upper bound found by slope scaling

initialize coefficients $\tilde{\beta}^1$;
solve $SS(\tilde{\beta}^1)$;
$phase \leftarrow 1$;
$t \leftarrow 2$;
while *time limit is not exceeded* **do**
    $\tilde{\beta}^t \leftarrow \tilde{\beta}^1$;
    if $phase = 1$ **then**
        $\tilde{\beta}^t \leftarrow$ INTENSIFY$(\tilde{\beta}^t)$;
    else
        $\tilde{\beta}^t \leftarrow$ DIVERSIFY$(\tilde{\beta}^t)$;
    solve $SS(\tilde{\beta}^t)$;
    if $phase = 1$ *and best-known solution was improved* **then**
        $phase \leftarrow 1$ (continue intensification);
    else
        $phase \leftarrow phase * (-1)$;
    $t \leftarrow t + 1$;

6.5.6. Restricted MILP

In this section, we introduce a final refinement step to further improve the best-known feasible solution found by the SS heuristic. As usual for heuristics, SS may not be able to find an optimal solution, and cannot certify the optimality of a solution. For example, the approximation of pipeline costs may lead to solutions with a pipeline network different than an optimal network. Furthermore, as the MILP used within a SS iteration is not necessarily solved to optimality, the source and sink decisions may not be optimal for the associated pipeline approximation decisions. To further improve the odds of SS finding improved solutions (or even an optimal solution), we introduce a heuristic refinement phase, employed after the execution of SS, where we solve a restricted MILP model. This model consists of the original CCS formulation, but only allows for the use of a limited set of arcs. As such, this may allow for a quick improvement of the SS solution. Let $A \subseteq K$ be such set of allowable arcs. To build the restricted model, we introduce the constraint below to the original formulation (see Section 6.4):

$$y^{\mathrm{P}}_{kch} = 0 \qquad \forall k \in K \setminus A, c \in C, h \in H. \tag{6.5.27}$$

To prioritize quickly finding improved feasible solutions, we limit the size of the restricted MILP formulation by allowing only for the arcs belonging to the best known solution found by the SS. Furthermore, we provide the best known solution found by SS (with the improved dynamic programming capacity assignment decisions) as an incumbent to the restricted model. We let CPLEX solve this restricted MILP for a specified amount of time.

6.6. Computational Experiments

In this section, we compare the performance of the SS heuristic against CPLEX. Our objective is to understand if CCS stakeholders are able to use SS to generate solutions that are comparable or better than those generated by CPLEX. We first describe the benchmark set of instances that we use to evaluate SS and CPLEX on in Section 6.6.1. Such benchmark set covers different geographical regions, and has instances of small, medium and large scale. Then, we report the results of computational experiments and discuss the performance of SS and CPLEX according to different problem attributes in Section 6.6.2.

6.6.1. Benchmark instances

To provide a large testbed for evaluating the SS heuristic and CPLEX, we adapt six single-period case studies from the United States Department of Energy to a multiperiod context. These case studies cover different regions in the United States, and one region in China. A characterization of each case study is present in Table 6.1. The costs in these multiperiod datasets do not take into account commodity forecasts. Instead, they simply use the costs as in the single-period case studies [e.g., Middleton, 2013] with fixed discount and interest rates, and zero inflation [as in Middleton et al., 2012b]. A more detailed multiperiod cost structure, including the aforementioned forecasts, is outside the scope of the present study. The here considered model is, however, flexible enough to accommodate for that without any changes to the model structure. We assumed a discount rate of 7% and an interest rate of 9%, as in Middleton et al. [2012b], and refer the reader to that paper for the discounting and amortization formulas that we used.

To represent small, medium and large-scale instances, we run experiments with four different time periods. For case studies with a planning horizon of 30 years, we split the planning horizon into 5, 10, 15 and 30 time periods of equal length. For case studies with a planning horizon of 50 years, we split the planning horizon into 5, 10, 25 and 50 time periods of equal length. Furthermore, we consider four different annual CO_2 capture target profiles, where the annual target for the last time period amounts to 25%, 50%, 75% and 100% of the maximum annually capturable/injectable CO_2, and the rates for the previous time periods are linearly interpolated based on that value. Finally, as the accuracy level of the piecewise approximation of pipeline costs may change according to the needs of analysts,

we run experiments with two and six piecewise linear trends for pipeline capacities. In total, this accounts to a set of 192 different experimental settings with different network topologies, number of time periods, target capture rates and number of linear pieces. While previous models and heuristics for CCS planning problems usually focus on one case study, the use of several case studies with different characteristics will allow us to understand how well the proposed slope scaling heuristic and CPLEX would perform under diverse, yet realistic settings.

Table 6.1. Instance information for the six case studies considered.

inst	planning horizon (years)	$\|S\|$	$\|R\|$	$\|K\|$
China Ordos Basin	30	38	53	1040
Illinois	30	1	15	136
Southeast US	30	20	7	366
Texas	30	9	3	92
Gulf Coast	50	19	29	656
Midwest	50	9	15	236

6.6.2. Computational results

In this section, we report on the performance of the SS heuristic and CPLEX and discuss under which circumstances one approach may be more favorable than the other. We implemented all of our models and algorithms in Python. We allocate 5 minutes for SS, with an additional 30 seconds for the restricted MILP. We allocate one hour for CPLEX, which is therefore much more generous. Both the SS and CPLEX are limited to one thread. Experiments were conducted on a machine with an AMD 3970X processor and 256 GB of memory, using CPLEX version 22.1.

To understand the impact of instance attributes on the performance of SS and CPLEX, we group results based on the following attributes: the case study, the number of time periods (and the length of the planning horizon), the number of capacity trends, and the annual CO_2 capture target. This will allow us to derive attribute-specific insights into the performance of SS and CPLEX.

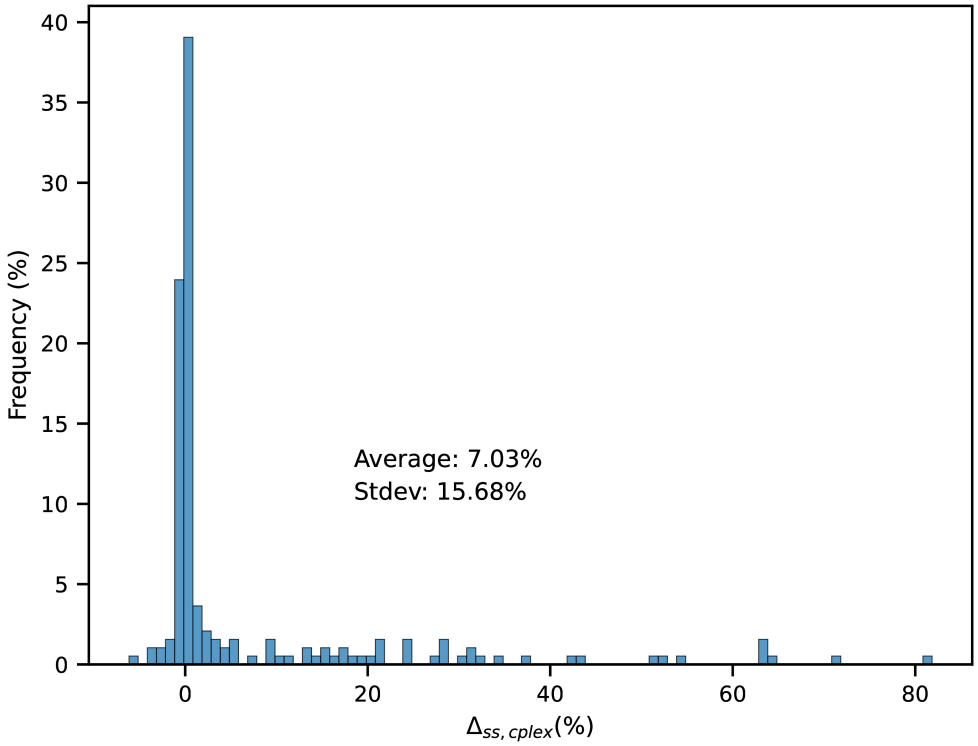
6.6.2.1. Average results. Average results are shown in Table 6.2. Columns $\Delta^{\min}_{\text{SS,CPLEX}}$, $\Delta_{\text{SS,CPLEX}}$ and $\Delta^{\max}_{\text{SS,CPLEX}}$ show the minimum, average and maximum relative objective improvement (%) between SS and CPLEX. Column $\text{freq}_{\leq}$ shows the proportion of experiments where the objective of the solution found by SS is less than or equal to the objective of the solution found by CPLEX. Column "T (sec)" shows the average CPU time taken by SS. The first row ("all") summarizes average results for all experiments. The second row ("SS $\leq$ CPLEX") shows average results for the cases where the solution found by SS is the same or better than CPLEX. The third row ("SS $>$ CPLEX") presents average results for the cases where the

solution found by SS is worse than CPLEX. When SS outperforms CPLEX, results are shown in boldface. Furthermore, Figure 6.2, illustrates the distribution of $\Delta_{\text{SS,CPLEX}}$.

Table 6.2. Average results for all instances.

experiments	$\Delta^{\min}_{\text{SS,CPLEX}}$	$\Delta_{\text{SS,CPLEX}}$	$\Delta^{\max}_{\text{SS,CPLEX}}$	$\text{freq}_{\leq}$	T (sec)
all	-6.16	**7.03**	**80.88**	**0.59**	**325.94**
SS $\leq$ CPLEX	**0.00**	**12.37**	**80.88**	**1.00**	**329.11**
SS $>$ CPLEX	-6.16	-0.60	-0.001	0.00	**321.41**

Fig. 6.2. Histogram of $\Delta_{\text{SS,CPLEX}}$ for all cases.



The results show that the average SS improvement over CPLEX is 7.03%, and SS is better than CPLEX for 59% of the instances. For those instances, the average SS improvement is 12.37% (min. 0%, max. 80.88%). SS is worse than CPLEX for 41% of the instances. For those instances, SS is on average 0.60% worse than CPLEX (min. 0.001%, max. 6.16%). The histogram also shows that most SS solutions have a similar cost to those found by CPLEX. When SS is worse than CPLEX, most solutions are at most 1% worse than CPLEX. When SS is better than CPLEX, most solutions have relative improvements between 0% and 10%, and a sizeable portion of solutions have improvements up to 30%.

6.6.2.2. Results by case study. We now show average results grouped by the six case studies. We aim to understand why SS performs better or worse according to the case study.

Attributes that change with the case study are the number and location of sources and sinks, their capacities, and the candidate network topology. Results are shown in Table 6.3. Columns $\%\text{gap}_{\text{SS}}^{\text{avg}}$ and $\%\text{gap}_{\text{SS}}^{\text{std}}$ ($\%\text{gap}_{\text{CPLEX}}^{\text{avg}}$ and $\%\text{gap}_{\text{CPLEX}}^{\text{std}}$) show the average and standard deviation of the optimality gaps of the solutions found by SS (CPLEX) when compared to the lower bounds found by CPLEX.

Table 6.3. Results by case study.

case study	$\Delta_{\text{SS,CPLEX}}$	$\text{freq}_{\leq}$	T (sec)	$\%\text{gap}_{\text{SS}}^{\text{avg}}$	$\%\text{gap}_{\text{SS}}^{\text{std}}$	$\%\text{gap}_{\text{CPLEX}}^{\text{avg}}$	$\%\text{gap}_{\text{CPLEX}}^{\text{std}}$
China Ordos Basin	**8.35**	**0.78**	335.97	**6.08**	**3.41**	14.00	8.98
Gulf Coast	**27.71**	**0.94**	335.17	**50.26**	**20.92**	60.04	25.73
Illinois	-0.15	0.25	326.21	0.50	**0.27**	**0.34**	0.33
Midwest	**6.46**	**0.53**	331.91	**7.53**	**5.39**	13.10	16.75
Southeast	**0.13**	**0.72**	319.63	**1.46**	**0.90**	1.58	1.07
Texas	-0.29	0.31	306.78	1.14	1.24	**0.86**	**1.04**

Illinois and Texas are the only case studies where the average improvement is negative. That is, for these case studies, on average, SS is worse than CPLEX. The proportion of experiments that SS is better than CPLEX is also rather small: 0.25 and 0.28, respectively. However, the values of $\Delta_{\text{SS,CPLEX}}$ are small (-0.15 and -0.29, respectively), which suggests that SS is finding solutions similar (albeit slightly worse) to CPLEX. These instances are also the smallest instances of the benchmark set (by number of sources, sinks, and pipelines), are commercial solvers are known to perform better on small instances. Still, the CPU time required by CPLEX to find these solutions is about 10 times larger than SS, for a marginal improvement. For all other case studies, SS improves over CPLEX on average and is better than CPLEX on most instances. The optimality gaps of SS and CPLEX show that, for all case studies except Gulf Coast, the solutions found by SS and CPLEX have average optimality gaps of at most 7.53% (for SS) and 14.00% (for CPLEX). This indicates that not only SS provides solutions of higher quality than CPLEX, but also that these solutions may have an acceptable quality for practical use. For the Gulf Coast case study, preliminary experiments indicate that the large optimality gaps may be associated with the difficulty of improving the lower bounds for that case study.

6.6.2.3. Results by time period. In this section, we analyze how the performance of the SS and CPLEX compare when the number of time periods is increased. Results for case studies with 30-years planning horizon are shown in Table 6.4 and results for case studies with 50-years planning horizon are shown in Table 6.5.

Table 6.4. Results for instances with a 30-year planning horizon

periods	$\Delta_{\text{SS,CPLEX}}$	$\text{freq}_{\leq}$	T (sec)
5	**0.92**	0.47	315.90
10	**1.57**	0.38	319.66
15	**2.03**	**0.50**	323.69
30	**3.51**	**0.72**	329.33

Table 6.5. Results for instances with a 50-year planning horizon (that is, Gulf Coast and Midwest).

periods	$\Delta_{\text{SS,CPLEX}}$	$\text{freq}_{\leq}$	T (sec)
5	**2.31**	**0.50**	325.00
10	**11.61**	**0.75**	332.00
25	**23.05**	**0.75**	334.67
50	**31.37**	**0.94**	342.47

The results for a planning horizon of 30 years show that both $\Delta_{\text{SS,CPLEX}}$ and $\text{freq}_{\leq}$ increase with the number of periods. SS only starts outperforming CPLEX on the majority of instances with 15 time periods or more. The average improvement over CPLEX is positive even for time periods smaller than 15 (starting at 0.92%). For a planning horizon of 50 years, the results also indicate that $\Delta_{\text{SS,CPLEX}}$ and $\text{freq}_{\leq}$ increase with the number of periods. Differently than 30-year instances, SS outperforms CPLEX on the majority of instances for any amount of time periods (i.e., $\text{freq}_{\leq} \geq 0.5$ and $\Delta_{\text{SS,CPLEX}} > 0$). The average improvement over CPLEX quickly increases with the number of time periods, and for 50 periods, this improvement is 31.37%.

6.6.2.4. Results by capacity trend. We now analyze how the SS and CPLEX results change based on the number of pipeline capacity trends. An analyst may want to increase the number of trends to represent pipeline costs more accurately. However, as increasing the number of trends leads to more pipeline-related variables, we expect that the instances with more capacity trends will be harder for CPLEX to solve. On the other hand, we expect that the performance of SS will not degrade the same way, as SS represents the capacity trends implicitly (see Section 6.5). Results are shown in Table 6.6.

Table 6.6. Results for different pipeline capacity trends.

trends	$\Delta_{\text{SS,CPLEX}}$	$\text{freq}_{\leq}$	T (sec)
2	**5.27**	**0.50**	325.34
6	**8.80**	**0.68**	326.55

The results show that both $\Delta_{\text{SS,CPLEX}}$ and $\text{freq}_{\leq}$ increase with the number of piecewise linear trends, which validates that SS has an advantage over CPLEX on larger formulations.

6.6.2.5. Results by the annual CO_2 capture target. Larger CO_2 capture targets do not change the model size but likely lead to solutions with more activated sources and sinks and larger pipeline networks. Thus, we expect instances to be more difficult to solve when the CO_2 capture targets are larger. Results are shown in Table 6.7.

Table 6.7. SS compared to CPLEX.

CapCO2	$\Delta_{\text{SS,CPLEX}}$	$\text{freq}_{\leq}$	T (sec)
25%	**1.30**	**0.56**	320.12
50%	**6.50**	**0.60**	324.01
75%	**8.64**	**0.54**	328.11
100%	**11.69**	**0.65**	331.54

The results show that the performance of SS over CPLEX increases with the CO_2 capture target. While $\Delta_{\text{SS,CPLEX}}$ improves consistently, $\text{freq}_{\leq}$ does not seem to follow a trend. The results confirm, however, that the model size is not the only factor influencing the performance of SS and CPLEX, but the number of activated variables also seem to play a role.

6.7. Conclusions and Future Work

We have introduced a new slope scaling heuristic for a strategic multiperiod CCS planning problem. We adapted elements from previous slope scaling heuristics for single-period CCS and network design, and also proposed novel attributes to slope scaling, such as the implicit representation of pipeline capacity cost functions and the generation of improved upper bounds through dynamic programming. Computational experiments have shown that the proposed heuristic outperforms CPLEX on the majority of experiments (relative improvements of up to 80.88%), at a fraction of the computational time. This was not the case for previous CCS slope scaling heuristics, where the heuristic runs for a fraction of the time, but the solutions are worse than those found by CPLEX. Furthermore, computational experiments were conducted on six different geographical regions, while previous papers on models and algorithms for CCS limit themselves to one geographical region. Thus, this work provides valuable algorithmic insights for the CCS community, and the proposed slope scaling heuristic allows for the representation and resolution of large multiperiod CCS problems, and may be adapted for more complex problem settings (e.g., models with uncertainty). As the deployment of CCS involves billions of dollars of investment across industries and governments, the superior performance of the slope scaling can lead to significant cost savings when deploying CCS, and therefore accelerate the pathway towards net-zero emissions.

Future work. A common use case in CCS VCO is to conduct sensitivity analyses such that stakeholders can understand the impact, for example, of government policies on the adoption of CO_2. Conducting sensitivity analyses means that several experiments must be run within a reasonable timeframe such that stakeholders can quickly have a good overview of the impact

of policy changes. One may explore the reoptimization capabilities of the proposed slope scaling heuristic. For example, for a specific case study, there may be different realizations of an uncertain input parameter. A slope scaling reoptimization procedure could iteratively solve these different input instances, reusing solution information from the previous iteration.

Additionally, the development of exact solution methods and bounding mechanisms may be a promising research direction, given the magnitude of strategic planning decisions and the difficulty of generating good lower bounds for some case studies.

Acknowledgements

We would like to thank the SimCCS Gateway team, in particular Ryan Kammer at Indiana University, for sharing data on SimCCS Gateway case studies with the CanmetENERGY team. The work of the first author was supported by CanmetENERGY Varennes (Natural Resources Canada, Government of Canada) and by the Chaire en transformation du transport (Université de Montréal/Polytechnique Montréal), which is funded by the Ministère de l'Économie, de l'Innovation et de l'Énergie du Québec. The work of the second author was supported by CanmetENERGY Varennes (Natural Resources Canada, Government of Canada). The work of the third author was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada under grant 2017-05224.

Chapter 7

Conclusions

In this thesis, two groups of problems with the potential to reduce greenhouse gas emissions were studied: ridesharing and carbon capture and storage (CCS). Three articles studied ridesharing problems, and one article was dedicated to studying a CCS problem.

The three articles on ridesharing consider problem attributes such as rematching flexibility, dynamic decisions, rider and driver stochasticity, driver booking, booking fees, and penalties for failing to provide a rideshare for booked drivers. The first article introduces a one-to-one dynamic and stochastic matching problem for ridesharing, and model reductions are given to reduce the model size and solution time. The second article evaluates stochastic models for that same problem on a rolling horizon framework and provides valuable managerial insights on model parameters that reflect different ridesharing platform settings. The third ridesharing article introduces a general modeling framework capable of representing a wide range of ridesharing systems. Stochastic model approximations are proposed, and the trade-off between each model is discussed based on the results of extensive computational experiments. Recommendations for platform configuration are also given based on the results of these computational experiments. Overall, these three articles on ridesharing may help practitioners improve the efficiency of their ridesharing platforms, attract a larger user base, assess the benefits of different platform configurations, and consequently reduce emissions.

The fourth article of this thesis concerns a multiperiod CCS problem. Deploying a CCS infrastructure would require investments in the order of billions of dollars, and efficiently estimating CCS costs is paramount to a successful deployment of CCS. Due to the computational complexity associated with multiperiod CCS value chain optimization problems, this article proposes a slope scaling heuristic. Computational experiments show that the proposed heuristic outperforms a state-of-the-art general-purpose mathematical programming solver, both in solution quality and CPU time. This provides a valuable tool for CCS analysts, that may want to run hundreds or thousands of sensitivity experiments to assess the costs

of deploying CCS. Overall, the articles in this thesis provide a wide range of strategies and tools to help reduce emissions.

7.1. Future Work

The topics studied in this thesis open several opportunities for future research work. Some directions for ridesharing and CCS are provided below.

7.1.1. Ridesharing

A potential extension of the article in Chapter 3 is the introduction of a multi-stage stochastic programming formulation for the one-to-one stochastic matching and unmatching problem for ridesharing. Such formulation may pose computational challenges, and approximations to the multi-stage problem that go beyond two-stage approximations could be proposed. In this regard, an interesting research question is to understand the economic gains of using multi-stage models compared to two-stage models, and how difficult it is to solve these models. For the two-stage set-packing problem introduced in Chapter 5, a decomposition algorithm could be developed to improve the model solution time. As the set-packing formulation makes no assumptions on route size, it can be easily used to represent vanpooling systems, where different riders share a van trip. The set-packing model can also be extended to allow for the multimodal transportation of passengers, through, for example, ridesharing and mass transit, park-and-ride, or ridesharing and vanpooling. For these problems, larger routes are expected, and the associated instances may pose additional computational challenges, which may require the development of faster enumeration algorithms and model solution methodologies, such as decomposition algorithms. Another possible research direction is to allow for rematching in a one-to-many ridesharing system such as the one introduced in Chapter 5. This would require combining the problem characteristics of Chapter 3 and Chapter 5. A possible contribution related to the generation of ridesharing instances is the improvement of the prediction of travel times. The linear regression model introduced in Chapter 3 and Chapter 5 may be extended to better represent real-life routing attributes such as speed limits, bridges, highways, road closures, and tunnels. Furthermore, although the three ridesharing articles in this thesis considered the uncertainty associated with the release of drivers and riders, another important source of uncertainty in ridesharing is the travel time. The set-packing formulation of Chapter 5 can be extended to allow for stochastic travel times.

7.1.2. Carbon capture and storage

For CCS, the slope scaling heuristic proposed in Chapter 6 could be extended for multi-modal problem settings, or problem settings with uncertainty (e.g., when reservoir storage

capacity is uncertain). Given that the slope scaling subproblem is a mathematical programming formulation that is fed to a solver, it allows for a high degree of flexibility to include additional problem features, which highlights the potential of this research direction. The performance of the slope scaling heuristic can be further investigated as well: the heuristic introduced in Chapter 6 performed a partial approximation of the original formulation. In that context, one potential research direction is to investigate which variables of a formulation should be approximated through slope scaling such that high-quality feasible solutions are generated on each slope scaling iteration at a reasonable computational time. This research direction can also be evaluated on traditional network design problems, where slope scaling has been used before. Finally, due to difficulty of solving CCS planning problems with mathematical programming solvers, one potential research direction is to improve the linear relaxation bounds obtained with the multiperiod formulation of Chapter 6. This could be achieved by developing, for example, valid inequalities and Lagrangian decompositions.

References

Carpooling and congestion pricing: HOV and HOT lanes. *Regional Science and Urban Economics*, 40(4):173–186, 2010.

N. Agatz, A. Erera, M. Savelsbergh, and X. Wang. Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, 223(2):295–303, 2012.

N. A. Agatz, A. L. Erera, M. W. Savelsbergh, and X. Wang. Dynamic ride-sharing: A simulation study in metro Atlanta. *Transportation Research Part B: Methodological*, 45 ‘ (9):1450–1464, 2011.

S. Ahmed, A. J. King, and G. Parija. A multi-stage stochastic integer programming approach for capacity expansion under uncertainty. *Journal of Global Optimization*, 26(1):3–24, 2003.

J. Alcalde, S. Flude, M. Wilkinson, G. Johnson, K. Edlmann, C. E. Bond, V. Scott, S. M. Gilfillan, X. Ogaya, and R. S. Haszeldine. Estimating geological CO2 storage security to deliver on climate mitigation. *Nature Communications*, 9, 2018.

M. Aresta and A. Dibenedetto. 14 - industrial utilization of carbon dioxide (CO2). In M. M. Maroto-Valer, editor, *Developments and Innovation in Carbon Dioxide (CO2) Capture and Storage Technology*, volume 2 of *Woodhead Publishing Series in Energy*, pages 377–410. Woodhead Publishing, 2010.

R. Baldacci, E. Hadjiconstantinou, V. Maniezzo, and A. Mingozzi. A new method for solving capacitated location problems based on a set partitioning approach. *Computers and Operations Research*, 29(4):365–386, 2002.

R. Baldacci, V. Maniezzo, and A. Mingozzi. An Exact Method for the Car Pooling Problem Based on Lagrangean Column Generation. *Operations Research*, 52(3):422–439, 2004.

A. Baylin-Stern and N. Berghout. Is carbon capture too expensive? *IEA: International Energy Agency*, 2021.

D. J. Beerling, E. P. Kantzas, M. R. Lomas, P. Wade, R. M. Eufrasio, P. Renforth, B. Sarkar, M. G. Andrews, R. H. James, C. R. Pearce, J.-F. Mercure, H. Pollitt, P. B. Holden, N. R. Edwards, M. Khanna, L. Koh, S. Quegan, N. F. Pidgeon, I. A. Janssens, J. Hansen, and S. A. Banwart. Potential for large-scale CO2 removal via enhanced rock weathering with croplands. *Nature*, 583(7815):242–248, 2020.

T. Bektaş, J. F. Ehmke, H. N. Psaraftis, and J. Puchinger. The role of operational research in green freight transportation. *European Journal of Operational Research*, 274(3):807–823, 2019.

D. Bertsimas, P. Jaillet, and S. Martin. Online Vehicle Routing: The Edge of Optimization in Large-Scale Applications. *Operations Research*, 67(1):143–162, 2019.

J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer New York, 2nd edition, 2011.

R. Borndörfer. *Aspects of set packing, partitioning, and covering*. PhD thesis, 1998.

D. Brownstone and T. F. Golob. The effectiveness of ridesharing incentives. Discrete-choice models of commuting in Southern California. *Regional Science and Urban Economics*, 22 (1):5–24, 1992.

M. Bui, C. S. Adjiman, A. Bardow, E. J. Anthony, A. Boston, S. Brown, P. S. Fennell, S. Fuss, A. Galindo, L. A. Hackett, J. P. Hallett, H. J. Herzog, G. Jackson, J. Kemper, S. Krevor, G. C. Maitland, M. Matuszewski, I. S. Metcalfe, C. Petit, G. Puxty, J. Reimer, D. M. Reiner, E. S. Rubin, S. A. Scott, N. Shah, B. Smit, J. P. Trusler, P. Webley, J. Wilcox, and N. M. Dowell. Carbon capture and storage (CCS): The way forward, 2018.

G. Cauchois, C. Renaud-Bezot, M. Simon, V. Gentile, M. Carpenter, and L. E. Torbergsen. Re-stream – study on the reuse of oil and gas infrastructure for hydrogen and CCS in Europe. *DNV, Carbon Limits, Tech. Rep*, 2021.

N. D. Chan and S. A. Shaheen. Ridesharing in North America: Past, Present, and Future. *Transport Reviews*, 32(1):93–112, 2012.

N. Chen, N. Immorlica, A. R. Karlin, M. Mahdian, and A. Rudra. Approximating matches made in heaven. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5555 LNCS, pages 266–278, 2009.

J. Comendador, A. Monzón, and M. E. López-Lambas. A general framework to testing the effect of transport policy measures to achieve a modal shift: A sequential hybrid model. *Procedia - Social and Behavioral Sciences*, 162:243–252, 2014.

B. Comer and B. Sathiamoorthy. How updating IMO regulations can promote lower greenhouse gas emissions from ships. *International Council on Clean Transportation*, 2022.

I. Contreras and E. Fernández. General network design: A unified view of combined location and network design problems. *European Journal of Operational Research*, 219:680–697, 2012.

I. Contreras and M. O'Kelly. *Hub Location Problems*, pages 327–363. Springer International Publishing, Cham, 2019.

L. Costa, C. Contardo, and G. Desaulniers. *Exact branch-price-and-cut algorithms for vehicle routing*, volume 53. 2019.

N. Coulombel, V. Boutueil, L. Liu, V. Viguié, and B. Yin. Substantial rebound effects in urban ridesharing: Simulating travel decisions in paris, france. *Transportation Research Part D: Transport and Environment*, 71:110–126, 2019.

T. G. Crainic, B. Gendron, and G. Hernu. A slope scaling/lagrangean perturbation heuristic with long-term memory for multicommodity capacitated fixed-charge network design. *Journal of Heuristics*, 10(5):525–545, 2004.

T. G. Crainic, M. Gendreau, and B. Gendron, editors. *Network Design with Applications to Transportation and Logistics*. Springer International Publishing, 2021.

D. J. Dailey, D. Loseff, and D. Meyers. Seattle smart traveler: Dynamic ridematching on the World Wide Web. *Transportation Research Part C: Emerging Technologies*, 7(1):17–32, 1999.

F. d'Amore and F. Bezzo. Economic optimisation of european supply chains for CO2 capture, transport and sequestration. *International Journal of Greenhouse Gas Control*, 65:99–116, 2017.

R. Dekker, J. Bloemhof, and I. Mallidis. Operations research for green logistics – an overview of aspects, issues, contributions and challenges. *European Journal of Operational Research*, 219(3):671–679, 2012.

S. J. Del Grosso and D. W. Grant. Reducing agricultural GHG emissions: role of biotechnology, organic systems and consumer behavior, 2011.

K. Dennis. Environmentally Beneficial Electrification: Electricity as the End-Use Option. *Electricity Journal*, 28(9):100–112, 2015.

J. A. R. Diamante, R. R. Tan, D. C. Foo, D. K. Ng, K. B. Aviso, and S. Bandyopadhyay. A graphical approach for pinch-based source-sink matching and sensitivity analysis in carbon capture and storage systems. *Industrial and Engineering Chemistry Research*, 52: 7211–7222, 2013.

J. Edmonds and E. L. Johnson. *Matching: A Well-Solved Class of Integer Linear Programs*, pages 27–30. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

K. M. Ellett, R. S. Middleton, P. H. Stauffer, and J. A. Rupp. Facilitating CCS business planning by extending the functionality of the simCCS integrated system model. *Energy Procedia*, 114:6526–6535, 2017.

L. F. Escudero, M. Landete, and A. M. Rodríguez-Chía. Stochastic set packing problem. *European Journal of Operational Research*, 211(2):232–240, 2011.

I. Fragkos, J. F. Cordeau, and R. Jans. Decomposition methods for large-scale network expansion problems. *Transportation Research Part B: Methodological*, 144:60–80, 2021.

M. Fridahl and M. Lehtveer. Bioenergy with carbon capture and storage (BECCS): Global potential, investment preferences, and deployment barriers. *Energy Research & Social Science*, 42:155–165, 2018.

M. Furuhata, M. Dessouky, F. Ordóñez, M. E. Brunet, X. Wang, and S. Koenig. Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 57:28–46, 2013.

A. Gambhir and M. Tavoni. Direct air carbon capture and sequestration: how it works and how it could contribute to climate-change mitigation. *One Earth*, 1(4):405–409, 2019.

E. Gargiulo, R. Giannantonio, E. Guercio, C. Borean, and G. Zenezini. Dynamic Ride Sharing Service: Are Users Ready to Adopt it? *Procedia Manufacturing*, 3(Ahfe):777–784, 2015.

B. Gendron, S. Hanafi, and R. Todosijević. Matheuristics based on iterative linear programming and slope scaling for multicommodity capacitated fixed charge network design. *European Journal of Operational Research*, 268:70–81, 2018.

B. Girod, D. P. van Vuuren, and E. G. Hertwich. Climate policy through changing consumption choices: Options and obstacles for reducing greenhouse gas emissions. *Global Environmental Change*, 25:5–15, 2014.

G. Giuliano, D. W. Levine, and R. F. Teal. Impact of high occupancy vehicle lanes on carpooling behavior. *Transportation*, 17(2):159–177, 1990.

M. Granovskii, I. Dincer, and M. A. Rosen. Greenhouse gas emissions reduction by use of wind and solar energies for hydrogen and electricity production: Economic factors. *International Journal of Hydrogen Energy*, 32(8):927–931, 2007.

J.-H. Han and I.-B. Lee. Development of a scalable and comprehensive infrastructure model for carbon dioxide utilization and disposal. *Industrial & Engineering Chemistry Research*, 50(10):6297–6315, 2011.

G. Homsi, B. Gendron, and S. D. Jena. Dynamic and stochastic rematching for ridesharing systems: Formulations and reductions. In M. Baïou, B. Gendron, O. Günlük, and A. R. Mahjoub, editors, *Combinatorial Optimization*, pages 193–201. Springer, 2020.

G. Homsi, B. Gendron, and S. D. Jena. Rolling horizon strategies for a dynamic and stochastic ridesharing problem with rematches. Technical report, CIRRELT, 2021.

B. Hoover, S. Yaw, and R. Middleton. CostMAP: an open-source software package for developing cost surfaces using a multi-scale search kernel. *International Journal of Geographical Information Science*, 34(3):520–538, 2020.

Y. Huang, S. Rebennack, and Q. P. Zheng. Techno-economic analysis and optimization models for carbon capture and storage: A survey. *Energy Systems*, 4(4):315–353, 2013.

IEA. *Energy technology perspectives 2017: Catalysing energy technology transformations*. 2017.

IEA. *Exploring Clean Energy Pathways: The Role of CO2 Storage*. 2019.

IPCC. *IPCC special report on carbon dioxide capture and storage*. Cambridge University Press, 2005.

S. D. Jena, J. F. Cordeau, and B. Gendron. Dynamic facility location with generalized modular capacities. *Transportation Science*, 49:484–499, 2015.

E. C. Jones, S. Yaw, J. A. Bennett, J. D. Ogland-Hand, C. Strahan, and R. S. Middleton. Designing multi-phased CO2 capture and storage infrastructure deployments. *Renewable and Sustainable Energy Transition*, 2:100023, 2022.

J. Jung and Y. Koo. Analyzing the effects of car sharing services on the reduction of greenhouse gas (GHG) emissions. *Sustainability*, 10(2):539, 2018.

A. Kadri, O. Koné, and B. Gendron. A Lagrangian heuristic for the multicommodity capacitated location problem with balancing requirements. *Computers and Operations Research*, 142:105720, 2022.

G. N. Keating, R. S. Middleton, P. H. Stauffer, H. S. Viswanathan, B. C. Letellier, D. Pasqualini, R. J. Pawar, and A. V. Wolfsberg. Mesoscale carbon sequestration site screening and CCS infrastructure analysis. *Environmental Science & Technology*, 45(1): 215–222, 2011.

Z. Kou, X. Wang, S. F. A. Chiu, and H. Cai. Quantifying greenhouse gas emissions reduction from bike share systems: a model considering real-world trips and transportation mode choice patterns. *Resources, Conservation and Recycling*, 153:104534, 2020.

R. Lal. Soil carbon sequestration to mitigate climate change. *Geoderma*, 123(1):1–22, 2004.

R. Lampitt, E. Achterberg, T. Anderson, J. Hughes, M. Iglesias-Rodriguez, B. Kelly-Gerreyn, M. Lucas, E. Popova, R. Sanders, J. Shepherd, D. Smythe-Wright, and A. Yool. Ocean fertilization: a potential means of geoengineering? *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1882):3919–3945, 2008.

J. Lane, C. Greig, and A. Garnett. Uncertain storage prospects create a conundrum for carbon capture and storage ambitions. *Nature Climate Change*, 11:925–936, 2021.

G. Laporte and S. Nickel. *Location Science*. 2015.

A. Lee and M. Savelsbergh. Dynamic ridesharing: Is there a role for dedicated drivers? *Transportation Research Part B: Methodological*, 81:483–497, 2015.

J. Lehmann, A. Cowie, C. A. Masiello, C. Kammann, D. Woolf, J. E. Amonette, M. L. Cayuela, M. Camps-Arbestain, and T. Whitman. Biochar in climate change mitigation. *Nature Geoscience*, 14(12):883–892, 2021.

J. K. Lenstra and A. R. Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981.

C. Levinger, N. Hazon, and A. Azaria. Computing the shapley value for ride-sharing and routing games. *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, 2020-May:1895–1897, 2020.

Y. Liu and Y. Li. Pricing scheme design of ridesharing program in morning commute problem. *Transportation Research Part C: Emerging Technologies*, 79:156–177, 2017.

L. J. Lobo. An improved mathematical formulation for the carbon capture and storage (CCS) problem. Master's thesis, Arizona State University, 2017.

D. Luxen and C. Vetter. Real-time routing with openstreetmap data. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '11, pages 513–516, New York, NY, USA, 2011. ACM.

A. Madansky. Inequalities for Stochastic Linear Programming Problems. *Management Science*, 6(2):197–204, 1960.

L. d. C. Martins, R. de la Torre, C. G. Corlu, A. A. Juan, and M. A. Masmoudi. Optimizing ride-sharing operations in smart sustainable cities: Challenges and the need for agile algorithms. *Computers and Industrial Engineering*, 153:107080, 2021.

N. Masoud and R. Jayakrishnan. A decomposition algorithm to solve the multi-hop Peer-to-Peer ride-matching problem. *Transportation Research Part B: Methodological*, 99:1–29, 2017.

C. Matasci, M. Gauch, H. Böni, and P. Wäger. The influence of consumer behavior on climate change: The case of switzerland. *Sustainability*, 13(5):2966, 2021.

N. McGuckin and A. Fucci. Summary of travel trends: 2017 national household travel survey. *Washington, DC: Federal Highway Administration, US Department of Transportation*, 2018.

A. Mehta. AdWords and Generalized On-line Matching. pages 1–19, 2005.

S. Melkote and M. S. Daskin. Capacitated facility location/network design problems. *European Journal of Operational Research*, 129(3):481–495, 2001a.

S. Melkote and M. S. Daskin. An integrated model of facility location and transportation network design. *Transportation Research Part A: Policy and Practice*, 35(6):515–538, 2001b.

R. S. Middleton. A new optimization approach to energy network modeling: Anthropogenic CO2 capture coupled with enhanced oil recovery. *International Journal of Energy Research*, 37, 2013.

R. S. Middleton and J. M. Bielicki. A scalable infrastructure model for carbon capture and storage: SimCCS. *Energy Policy*, 37(3):1052–1060, 2009.

R. S. Middleton and A. R. Brandt. Using infrastructure optimization to reduce greenhouse gas emissions from oil sands extraction and processing. *Environmental Science and Technology*, 47(3):1735–1744, 2013.

R. S. Middleton and S. Yaw. The cost of getting CCS wrong: Uncertainty, infrastructure design, and stranded CO2. *International Journal of Greenhouse Gas Control*, 70:1–11, 2018.

R. S. Middleton, M. J. Kuby, and J. M. Bielicki. Generating candidate networks for optimization: The CO2 capture and storage optimization problem. *Computers, Environment and Urban Systems*, 36(1):18–29, 2012a.

R. S. Middleton, M. J. Kuby, R. Wei, G. N. Keating, and R. J. Pawar. A dynamic model for optimally phasing in CO2 capture and storage infrastructure. *Environmental Modelling and Software*, 37:193–205, 2012b.

R. S. Middleton, B. Chen, D. R. Harp, R. M. Kammer, J. D. Ogland-Hand, J. M. Bielicki, A. F. Clarens, R. P. Currier, K. M. Ellett, B. A. Hoover, D. N. McFarlane, R. J. Pawar, P. H. Stauffer, H. S. Viswanathan, and S. P. Yaw. Great SCO2T! rapid tool for carbon sequestration science, engineering, and economics. *Applied Computing and Geosciences*, 7, 2020.

R. S. Middleton, J. M. Bielicki, K. M. Ellett, B. A. Hoover, R. M. Kammer, and S. P. Yaw. Gigatonne one: A CCS roadmap for the united states and beyond, 2021.

S. Mitrović-Minić, R. Krishnamurti, and G. Laporte. Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 38(8):669–685, 2004.

A. Mourad, J. Puchinger, and C. Chu. A survey of models and algorithms for optimizing shared mobility. *Transportation Research Part B: Methodological*, 123:323–346, 2019.

G. Nagy and S. Salhi. Location-routing: Issues, models and methods. *European Journal of Operational Research*, 177(2):649–672, 2006.

J. G. Neoh, M. Chipulu, and A. Marshall. What encourages people to carpool? An evaluation of factors with meta-analysis. *Transportation*, 44(2):423–447, 2017.

K. Orchard, M. Hay, I. Ombudstvedt, R. Skagestad, M. Joos, G. Nysæter, C. Sjøbris, A. G. Jarøy, E. Durusut, and J. Craig. The status and challenges of CO2 shipping infrastructures. *15th International Conference on Greenhouse Gas Control Technologies, GHGT-15*, 2021.

D. Papadimitriou and B. Fortz. A rolling horizon heuristic for the multiperiod network design and routing problem. *Networks*, 66:364–379, 2015.

S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems. *Journal fur Betriebswirtschaft*, 58(1):21–51, 2008a.

S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems: Part II: Transportation between pickup and delivery locations. *Journal fur Betriebswirtschaft*, 58(2):81–117, 2008b.

Z. Peng, W. Shan, X. Zhu, and B. Yu. Many-to-one stable matching for taxi-sharing service with selfish players. *Transportation Research Part A*, 160:255–279, 2022.

J. Popp, S. Kovács, J. Oláh, Z. Divéki, and E. Balázs. Bioeconomy: Biomass and biomass-based energy supply and demand. *New Biotechnology*, 60(October 2020):76–84, 2021.

M. Ranjan and H. J. Herzog. Feasibility of air capture. *Energy Procedia*, 4(2010):2869–2876, 2011.

F. Razi and I. Dincer. Challenges, opportunities and future directions in hydrogen sector development in canada. *International Journal of Hydrogen Energy*, 47(15):9083–9102, 2022.

K. Riahi, D. P. van Vuuren, E. Kriegler, J. Edmonds, B. C. O'Neill, S. Fujimori, N. Bauer, K. Calvin, R. Dellink, O. Fricko, W. Lutz, A. Popp, J. C. Cuaresma, S. KC, M. Leimbach, L. Jiang, T. Kram, S. Rao, J. Emmerling, K. Ebi, T. Hasegawa, P. Havlik, F. Humpenöder, L. A. D. Silva, S. Smith, E. Stehfest, V. Bosetti, J. Eom, D. Gernaat, T. Masui, J. Rogelj,

J. Strefler, L. Drouet, V. Krey, G. Luderer, M. Harmsen, K. Takahashi, L. Baumstark, J. C. Doelman, M. Kainuma, Z. Klimont, G. Marangoni, H. Lotze-Campen, M. Obersteiner, A. Tabeau, and M. Tavoni. The shared socioeconomic pathways and their energy, land use, and greenhouse gas emissions implications: An overview. *Global Environmental Change*, 42:153–168, 2017.

C. Riley, A. Legrain, and P. Van Hentenryck. Column generation for real-time ride-sharing operations. In L.-M. Rousseau and K. Stergiou, editors, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 472–487, Cham, 2019. Springer International Publishing. ISBN 978-3-030-19212-9.

R. Saidur, E. A. Abdelaziz, A. Demirbas, M. S. Hossain, and S. Mekhilef. A review on biomass as a fuel for boilers. *Renewable and Sustainable Energy Reviews*, 15(5):2262–2289, 2011.

E. Scolaro, M. Beligoj, M. P. Estevez, L. Alberti, M. Renzi, and M. Mattetti. Electrification of agricultural machinery: A review. *IEEE Access*, 9:164520–164541, 2021.

S. Shaheen and N. Chan. Mobility and the sharing economy: Potential to facilitate the first- and last-mile public transit connections. *Built Environment*, 42(4):573–588, 2016.

S. Shaheen, A. Cohen, A. Bayen, et al. The benefits of carpooling. *UC Berkeley: Transportation Sustainability Research Center*, 2018.

H. Si, Y. Su, G. Wu, W. Li, and L. Cheng. Can government regulation , carbon-emission reduction certification and information publicity promote carpooling behavior ? *Transportation Research Part D*, 109:103384, 2022.

B. Smit, J. A. Reimer, C. M. Oldenburg, and I. C. Bourg. *Introduction to carbon capture and sequestration*, volume 1. World Scientific, 2014.

F. Sprei. Online and App-Based Carpooling in France: Analyzing Users and Practices—A Study of BlaBlaCar. *Energy Research and Social Science*, 37:238–242, 2018.

M. Stiglic, N. Agatz, M. Savelsbergh, and M. Gradisar. The benefits of meeting points in ride-sharing systems. *Transportation Research Part B: Methodological*, 82:36–53, 2015.

M. Stiglic, N. Agatz, M. Savelsbergh, and M. Gradisar. Making dynamic ride-sharing work: The impact of driver and rider flexibility. *Transportation Research Part E: Logistics and Transportation Review*, 91:190–207, 2016.

M. Stiglic, N. Agatz, M. Savelsbergh, and M. Gradisar. Enhancing urban mobility: Integrating ride-sharing and public transit. *Computers and Operations Research*, 90:12–21, 2018.

T. Strunge, L. Küng, P. Renforth, and M. Van Der Spek. Marginal cost curves for decarbonizing the European cement industry. *Proceedings of the 16th Greenhouse Gas Control Technologies Conference (GHGT-16)*, 2022.

R. Svensson, M. Odenberger, F. Johnsson, and L. Strömberg. - transportation infrastructure for CCS —experiences and expected development. In E. Rubin, D. Keith, C. Gilboy, M. Wilson, T. Morris, J. Gale, and K. Thambimuthu, editors, *Greenhouse Gas Control*

Technologies 7, pages 2531–2534. Elsevier Science Ltd, Oxford, 2005.

C. Talsma, E. Middleton, and R. Middleton. Costmap^{PRO}: Addressing the massive-scale CO_2 pipeline challenge. *Proceedings of the 16th Greenhouse Gas Control Technologies Conference (GHGT-16)*, 2022.

J. F. D. Tapia, J. Y. Lee, R. E. Ooi, D. C. Foo, and R. R. Tan. A review of optimization and decision-making models for the planning of CO2 capture, utilization and storage (ccus) systems. *Sustainable Production and Consumption*, 13:1–15, 2018.

R. F. Teal. Carpooling: Who, how and why. *Transportation Research Part A: General*, 21 (3):203–214, 1987.

T. Terlouw, C. Bauer, L. Rosa, and M. Mazzotti. Life cycle assessment of carbon dioxide removal technologies: a critical review. *Energy & Environmental Science*, 14(4):1701–1721, 2021.

A. Trabucco, R. J. Zomer, D. A. Bossio, O. van Straaten, and L. V. Verchot. Climate change mitigation through afforestation/reforestation: A global analysis of hydrologic impacts with four case studies. *Agriculture, Ecosystems & Environment*, 126(1):81–97, 2008. International Agricultural Research and Climate Change: A Focus on Tropical Systems.

UN. Revision of world urbanization prospects. *United Nations: New York, NY, USA*, 799, 2018.

United States Department of State and United States Executive Office of the President. The long-term strategy of the united states: pathways to net-zero greenhouse gas emissions by 2050, 2021.

A. Vinca, M. Rottoli, G. Marangoni, and M. Tavoni. The role of carbon capture and storage electricity in attaining 1.5 and 2°C. *International Journal of Greenhouse Gas Control*, 78: 148–159, 2018.

H. Wang and H. Yang. Ridesourcing systems : A framework and review. 129:122–155, 2019.

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

C. Whitman, S. Yaw, B. Hoover, and R. Middleton. Scalable algorithms for designing CO2 capture and storage infrastructure. *Optimization and Engineering*, 2021.

M. Xylia, S. Leduc, A. B. Laurent, P. Patrizio, Y. van der Meer, F. Kraxner, and S. Silveira. Impact of bus electrification on carbon emissions: The case of Stockholm. *Journal of Cleaner Production*, 209:74–87, 2019.

B. Yaged. Minimum cost routing for static network models. *Networks*, 1(2):139–172, 1971.

L. Yan, X. Luo, R. Zhu, P. Santi, H. Wang, D. Wang, S. Zhang, and C. Ratti. Quantifying and analyzing traffic emission reductions from ridesharing: A case study of Shanghai. *Transportation Research Part D: Transport and Environment*, 89:102629, 2020.

S. Yaw, R. Middleton, B. Hoover, K. Ellett, and J. Bielicki. Keeping up with the times: Modelling temporally phased CO2 capture and storage infrastructure. *SSRN Electronic Journal*, 2021.

C. A. Zetina, I. Contreras, and J. F. Cordeau. Profit-oriented fixed-charge network design with elastic demand. *Transportation Research Part B: Methodological*, 127:1–19, 2019.

X. Zhang, K. Li, N. Wei, Z. Li, and J. L. Fan. Advances, challenges, and perspectives for CCUS source-sink matching models under carbon neutrality target. *Carbon Neutrality*, 1 (1):12, 2022.

Appendix A

Appendix to the third article

This is an appendix to the article in Chapter 5.

A.1. A note on the multi-stage planning problem

The above presented planning problem assumes that the driver-selection is carried out independently for a predefined planning period, for which all driver requests are known, while rider requests are yet unknown. This is a realistic setting in a variety of contexts, for example, for the planning of work-related trips, where the planning takes place twice a day: morning trips from home to work, and afternoon return trips from work to home. We now consider a more dynamic context, where requests may be revealed gradually over time and driver-selection takes place several times throughout the day. Specifically, consider a k-stage planning problem, where drivers can be selected and riders can be matched at any of the k stages, and decisions at each stage are made for the upcoming planning period (which are therefore disjoint). We now elaborate on the suitability of the 2-stage approach presented by the 2S-SDMP for specific assumptions on this k-stage problem.

Proposition 1. If driver-selection can only be made in the first stage and rider matches are decided in the following stages as rider requests occur, this multi-stage problem variant reduces to the 2S-SDMP.

While the 2S-SDMP originally assumes that all rider requests are uncertain at the time of driver-selection, it can easily account for the case of gradual rider request release. Rider requests that have not yet been observed in the first stage are all considered uncertain, no matter when in the future they will be released. Rider requests that have already occurred can still be accounted for as an uncertain rider request, with an occurrence probability of 100%. As a result, the problem structure remains unchanged, as drivers still have to be selected in the first stage, while riders are uncertain.

In contrast, a gradual release of the drivers in a multi-stage context or the possibility of selecting drivers at later stages may impact the suitability of the 2S-SDMP problem setting,

depending on the assumptions made on when those drivers can be selected. In many contexts, drivers will have to be selected at specific times, h hours before, for example, the beginning of the planning period. In the case of the 2S-SDMP where the planning may span a time period from 6am to 10am, the driver-selection planning may be required to be carried out at 10pm the day before, i.e., 8 hours before the beginning of the planning period (in order to give drivers a response sufficiently early). Even in a dynamic context, this is a realistic assumption: while driver and rider requests arrive gradually over the day, the planning is performed several times per day, h hours (e.g., 4 hours) before the beginning of the actual planning period.

Proposition 2. If drivers with itineraries within a given time-window have to be selected a specific time before the start of that time-window, and if each rider request appears in only one decision stage, then this multi-stage problem variant can be solved exactly by multiple executions of the 2S-SDMP.

This proposition follows from the fact that the planning over the entire planning horizon separates into independent planning problems, one for each planning period given by the various stages. Each of those planning problems then corresponds to an independent instance of the 2S-SDMP. Note that the assumption on the rider requests is quite realistic, given that it is unlikely that the combination of earliest departure time and latest arrival time allows for feasible driver matches in two consecutive planning periods. Finally, also note that this proposition assumes that rider-request uncertainty is time-independent, i.e., there is uncertainty whether a specific request occurs (such as in the case of morning/afternoon commuting), but no uncertainty regarding the time-window of the rider. This also appears to be a realistic assumption in several contexts, such as work-related trips, which typically occur around the same time.

If, however, a driver can be booked at any decision stage before the start of the time-window of her itinerary, or rider request uncertainty is time-dependent (i.e., there is additional uncertainty on the time for which the ride is requested), theoretically, a multi-stage problem is required. The 2-stage stochastic problem can be extended to approximate this multi-stage problem by using multiple time-periods in the second stage, in which drivers can be selected at a later moment in time and riders can be assigned to the selected drivers. An example of such a model is given by Homsi et al. [2020], which additionally allows for rematching participants in later time periods.

A.2. Pseudo-code for the depth-first search route enumeration algorithm

We now provide the pseudo-code for the depth-first search algorithm used to enumerate feasible rideshares. Let $feasible(\sigma) \rightarrow \{true, false\}$ be a Boolean function, indicating

whether (partial) route σ satisfies all feasibility criteria (e.g., time-feasibility). The algorithm can be summarized as follows (Algorithm A.1):

Algorithm A.1: Outline of the depth-first search algorithm used to enumerate ridesharing routes.

Result: set of feasible routes Ω

initialize set of feasible routes $\Omega \leftarrow \emptyset$;
initialize set of allowable rider pickups $A_p \leftarrow R$;
initialize set of allowable rider deliveries $A_d \leftarrow \emptyset$;
initialize an empty stack S;
for *each driver* $i \in D$ **do**
    $\sigma \leftarrow (i)$ // partial route with driver departure;
    add (A_p, A_d, σ) to the top of S;
while S *is not empty* **do**
    extract (A_p, A_d, σ) from the top of S;
    if A_p *is empty and* A_d *is empty* **then**
        $\sigma' \leftarrow \sigma + \sigma_1$ // route σ extended with arrival of driver;
        if $feasible(\sigma')$ **then**
            add (A'_p, A'_d, σ') to Ω;
    else
        for *each pickup* $i \in A_p$ **do**
            $\sigma' \leftarrow \sigma + i$ // route σ extended with pickup of rider i;
            if $feasible(\sigma')$ **then**
                $A'_p \leftarrow A_p \setminus \{ i \}$;
                $A'_d \leftarrow A_d \cup \{ i \}$;
                add (A'_p, A'_d, σ') to the top of S;
        for *each delivery* $i \in A_d$ **do**
            $\sigma' \leftarrow \sigma + i$ // route σ extended with delivery of rider i;
            if $feasible(\sigma')$ **then**
                $A'_p \leftarrow \emptyset$ // pickups after deliveries are not allowed;
                $A'_d \leftarrow A_d \setminus \{ i \}$;
                add (A'_p, A'_d, σ') to the top of S;