CIRRELT
1971 – 2021

CIRRELT-2021-20

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

Gabriel Homsi
Bernard Gendron
Sanjay Dominik Jena

May 2021

Bureau de Montréal
Université de Montréal
C.P. 6128, succ. Centre-Ville
Montréal (Québec) H3C 3J7
Tél : 1-514-343-7575
Télécopie : 1-514-343-7121

Bureau de Québec
Université Laval,
2325, rue de la Terrasse
Pavillon Palasis-Prince, local 2415
Québec (Québec) G1V 0A6
Tél : 1-418-656-2073
Télécopie : 1-418-656-2624

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

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

1. Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)
2. Department of Computer Science and Operations Research, Université de Montréal
3. Department of Analytics, Operations and Information Technology, School of Management, ESG UQAM

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 an 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.

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 of Canada (NSERC) 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.

---

* Corresponding author: gabrielhomsi@gmail.com

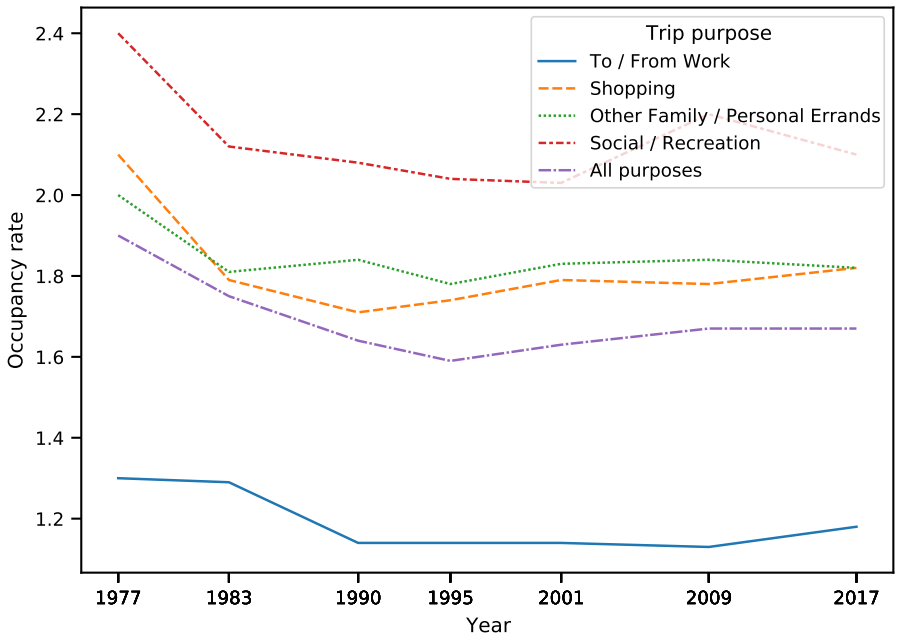
Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2021

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 [1]. 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 [2]. Additionally, the occupancy rate of vehicles used for work-related trips is even smaller (see Figure 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 [3]. We therefore use these terms interchangeably.

Figure 1: Vehicle occupancy rate by trip purpose, source: [2].



In this work, we revisit a matching and rematching problem with applications in ridesharing systems, previously defined in [4]. 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 afterward (for a certain penalty cost) such that they can be reassigned to better rideshares.

In [4], 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 have been proposed. In this work, we additionally consider the expected value problem, which optimizes over the average demand of the considered scenarios. We then evaluate the performance of all models in extensive computational experiments on 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.

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 [5], [6] and [7]. In [5] components and challenges of ride-matching optimization for dynamic ridesharing systems are discussed. Shortly after, [6] 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 [7], 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.

2.1. Operational planning studies

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

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 [8]. The authors solve a carpooling problem with a common workplace destination. The authors propose exact and heuristic methods based on Lagrangean column generation. In [9], the authors study the impact of

dedicated drivers on ridesharing systems. A ridesharing problem with transfers is studied in [10]. In [11], 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.

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. [12] 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 [13], participant time flexibility [14], integration with mass transit systems [15], and matching stability [16]. Recently, [4] first defined a rematching problem with applications in ridesharing. In this work, we further investigate this problem.

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 [17], the author studies the generalized online matching problem and its applications in search engine advertisement. In [18], 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.

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 [19], 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 [20]. Later, [21] 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, [10] 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.

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.

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.

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 [12, 16]. 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.

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. 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:

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

$$\text{s.t.} \sum_{(ij)\in\delta_t(v)} (x_{ij}^t - y_{ij}^t) \leq q'_v - \sum_{(ij)\in\delta_t(v)} z_{ij}^t \qquad \forall t = 1, \ldots, h, v \in V \tag{2}$$

$$y_{ij}^t \leq z_{ij}^t \qquad \forall t = 1, \ldots, h, (ij) \in E_t \tag{3}$$

$$z_{ij}^{t+1} = z_{ij}^t + \begin{cases} x_{ij}^t - y_{ij}^t \text{ if } (ij) \in E_t \\ 0 \text{ otherwise} \end{cases} \qquad \forall t = 1, \ldots, h, (ij) \in E \tag{4}$$

$$x_{ij}^t, y_{ij}^t \in \{\, 0, 1 \,\} \qquad \forall t = 1, \ldots, h, (ij) \in E_t \tag{5}$$

$$z_{ij}^{t+1} \in \{\, 0, 1 \,\} \qquad \forall t = 1, \ldots, h, (ij) \in E. \tag{6}$$

Objective (1) maximizes the net profit over the full planning horizon. Constraints (2) ensure that requests can only be matched if they are released and inactive. Constraints (3) ensure that requests can only be unmatched if they are active. Constraints (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. To tackle the dynamic arrival of requests, we next introduce a rolling horizon simulation framework to evaluate the myopic and stochastic decision strategies.

4. The Rolling Horizon Framework

We now describe the rolling horizon framework. Such frameworks are common, for example, in dynamic vehicle routing problems [22]. 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 1.

Algorithm 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 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 $\mathbf{q}$.

4.1. 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:

$$f_{\text{MYO}}(k, z^k) \coloneqq \max \sum_{(ij)\in E_k} (p_{ij}^k x_{ij}^k - c_{ij}^k y_{ij}^k) \tag{7}$$

$$\text{s.t.} \sum_{(ij)\in\delta_k(v)} (x_{ij}^k - y_{ij}^k) \leq q'_v - \sum_{(ij)\in\delta_k(v)} z_{ij}^k \qquad \forall v \in V, r_v \leq k \tag{8}$$

$$y_{ij}^k \leq z_{ij}^k \qquad \forall (ij) \in E_k \tag{9}$$

$$z_{ij}^{k+1} = z_{ij}^k + \begin{cases} x_{ij}^k - y_{ij}^k \text{ if } (ij) \in E_k \\ 0 \text{ otherwise} \end{cases} \qquad \forall (ij) \in E \tag{10}$$

$$x_{ij}^k, y_{ij}^k \in \{0, 1\} \qquad \forall (ij) \in E_k \tag{11}$$

$$z_{ij}^{k+1} \in \{0, 1\} \qquad \forall (ij) \in E. \tag{12}$$

The objective (7) maximizes the net profit of matching and unmatching requests at time period k. The constraints (8) ensure that only inactive requests can be matched. The constraints (9) ensure that only active requests can be unmatched. The constraints (10) 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 ($r_v \leq k$).

4.2. 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 [23].

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.

$$f_{\text{STO}}(k, z^k) := \max \sum_{(ij)\in E_k} (p_{ij}^k x_{ij}^k - c_{ij}^k y_{ij}^k) + \mathscr{Q}(k+1, z^{k+1}) \tag{13}$$

$$\text{s.t.} \sum_{(ij)\in\delta_k(v)} (x_{ij}^k - y_{ij}^k) \leq q_v' - \sum_{(ij)\in\delta_k(v)} z_{ij}^k \quad \forall v \in V, r_v \leq k \tag{14}$$

$$y_{ij}^k \leq z_{ij}^k \quad \forall (ij)\in E_k \tag{15}$$

$$z_{ij}^{k+1} = z_{ij}^k + \begin{cases} x_{ij}^k - y_{ij}^k \text{ if } (ij) \in E_k \\ 0 \text{ otherwise} \end{cases} \quad \forall (ij)\in E \tag{16}$$

$$x_{ij}^k, y_{ij}^k \in \{0, 1\} \quad \forall (ij)\in E_k \tag{17}$$

$$z_{ij}^{k+1} \in \{0, 1\} \quad \forall (ij)\in E, \tag{18}$$

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 (13) maximizes the first-stage net profit (for period k) plus the expected second-stage net profit (for periods $k+1, \ldots, h$). The constraints (14) and (15) match and unmatch requests for period k. The constraints (16) 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 (7)–(11). 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.

$$Q(k, z^k, q) := \max \sum_{t=k}^{h} \sum_{(ij)\in E_t} (p_{ij}^t x_{ij}^t - c_{ij}^t y_{ij}^t) \tag{19}$$

$$\text{s.t.} \sum_{(ij)\in\delta_t(v)} (x_{ij}^t - y_{ij}^t) \leq q_v' - \sum_{(ij)\in\delta_t(v)} z_{ij}^k \quad \forall t = k, \ldots, h, v\in V, r_v < k \tag{20}$$

$$\sum_{(ij)\in\delta_t(v)} (x_{ij}^t - y_{ij}^t) \leq q_v - \sum_{(ij)\in\delta_t(v)} z_{ij}^k \quad \forall t = k, \ldots, h, v\in V, r_v \geq k \tag{21}$$

$$y_{ij}^t \leq z_{ij}^t \quad \forall t = k, \ldots, h, (ij)\in E_t \tag{22}$$

$$z_{ij}^{t+1} = z_{ij}^t + \begin{cases} x_{ij}^t - y_{ij}^t \text{ if } (ij) \in E_t \\ 0 \text{ otherwise} \end{cases} \quad \forall t = k, \ldots, h, (ij)\in E \tag{23}$$

$$x_{ij}^t, y_{ij}^t \in \{0, 1\} \quad \forall t = k, \ldots, h, (ij)\in E_t \tag{24}$$

$$z_{ij}^{t+1} \in \{0, 1\} \quad \forall t = k, \ldots, h, (ij)\in E, \tag{25}$$

and has a structure similar to the one of the static problem (1)–(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.

4.2.1. 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.

$$f_{\text{EVP}}(k, z^k) := \max \sum_{(ij) \in E_k} (p_{ij}^k x_{ij}^k - c_{ij}^k y_{ij}^k) + \bar{\mathscr{Q}}(k+1, z^{k+1}) \tag{26}$$

$$\text{s.t. } (14)\text{–}(18). \tag{27}$$

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.

4.2.2. 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 $\mathbf{q}$ together 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 $\mathbf{q}$. With these realizations, we replace $\mathscr{Q}(k+1, z^{k+1})$ with

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

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

$$f_{\text{SAA}}(k, z^k) := \max \sum_{(ij)\in E_k} (p_{ij}^k x_{ij}^k - c_{ij}^k y_{ij}^k) + \mathscr{Q}_N(k+1, z^{k+1}) \tag{28}$$

$$\text{s.t. } (14)\text{–}(18). \tag{29}$$

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.

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.

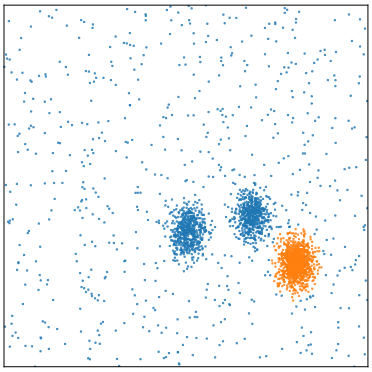
5.1.1. Patterns of origins and destinations

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}. Additionally, to represent requests outside common demand centers, all patterns have some

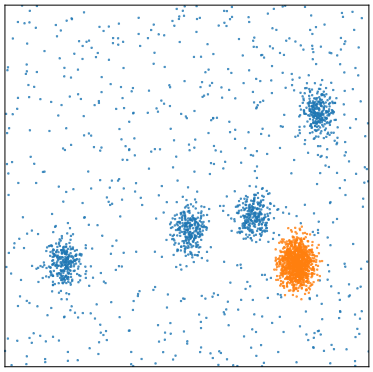
points that are generated uniformly over the greater Montreal region bounding box (described in the next section). The different patterns are illustrated in Figure 2, with the downtown region highlighted in orange.

Figure 2: The distribution of points for different patterns. Orange points refer to downtown.

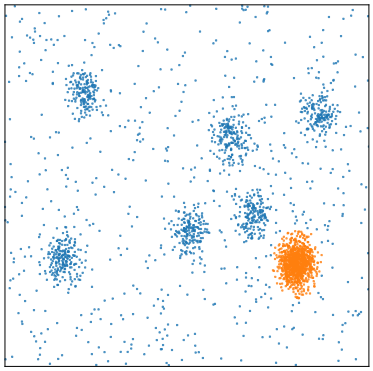
(a) Pattern 3g.



(b) Pattern 5g.



(c) Pattern 7g.

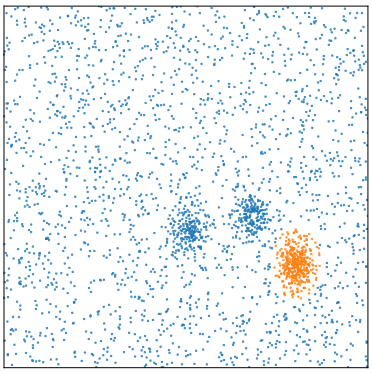


5.1.2. Request groups

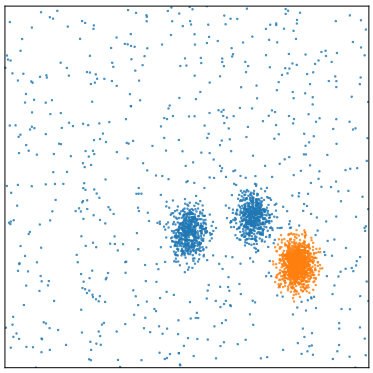
Requests belong to one of the following groups: *central* or *random*. The central requests have a downtown origin and a non-downtown destination, or a non-downtown origin and a downtown destination. These origins and destinations are generated from the Gaussian distributions of each pattern. The random 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 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.

Figure 3: Distribution of points for different values of ω_c.

(a) $\omega_c = 25\%$.



(b) $\omega_c = 75\%$.



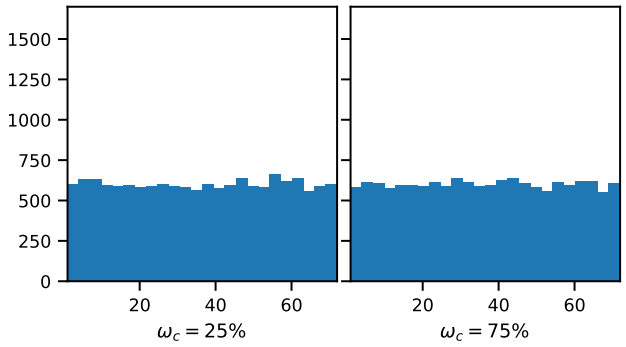
5.1.3. Release times

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.

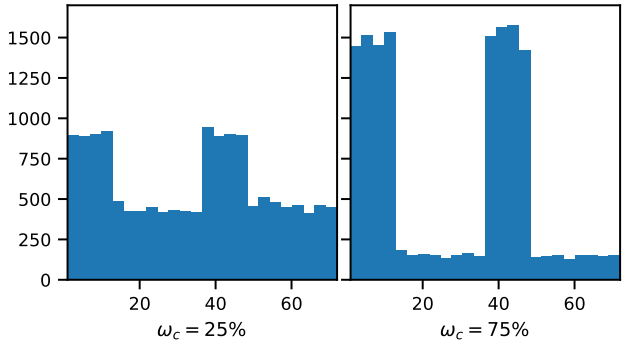
Figure 4 shows the distribution of uniform and clustered release times for different values of ω_c. For uniform release times, the value of ω_c does not affect the shape of the distribution. However, for clustered release times, smaller values of ω_c lead to more evenly distributed request releases.

Figure 4: Distribution of release times for different values of ω_c.

(a) Uniform release times.



(b) Clustered release times.



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 $\omega_r \in \{\, 5\%, 10\% \,\}$.

5.1.5. Travel distance and time

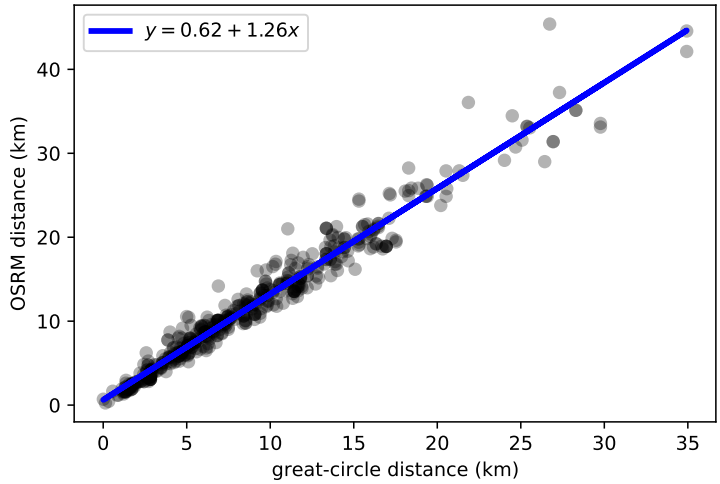
To evaluate the profitability and the time-window feasibility of matches, it is necessary to define the distances between origins and destinations. One option is to use a dedicated routing software such as OSRM [24] to find real route distances. Another more suitable option for preliminary analyses is 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 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., $t(a, b) = d(a, b)/40$.

Figure 5: The regression model.



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 $a_v + t_v(o_v, d_v)$.

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:

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

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

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 5.2.8.

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 [4] to reduce the time needed to build and solve the models.

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 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 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. Finally, the SAA always outperforms the EVP strategy but requires more CPU time. 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.

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 2.

Table 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.

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 3.

Table 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.

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.

Table 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.

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 5.

Table 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.

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 [4], 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 6.

Table 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.

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 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 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 [4], 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.

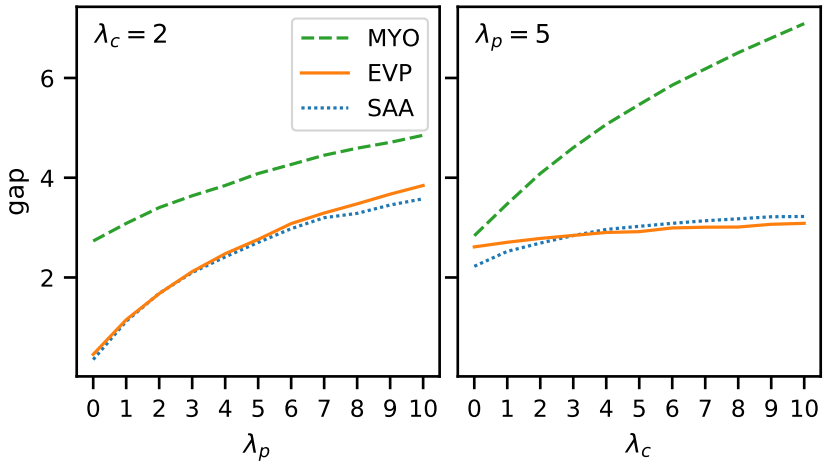
Matching delay. The myopic strategy has a matching delay similar to the one found in the static model (3.16 and 3.29 periods, i.e., about 60 minutes). The stochastic strategies have a longer delay (about 4.5 periods, i.e., about 1h30min), and increasing the number of scenarios slightly increases this 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. The SAA with 40 scenarios requires the longest computing time: about one minute at most. Considering that each period corresponds to an interval of 20 minutes, the SAA with up to 40 scenarios is solved sufficiently fast for real-time usage.

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 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.

Figure 6: Gap for different values of λ_p and λ_c.



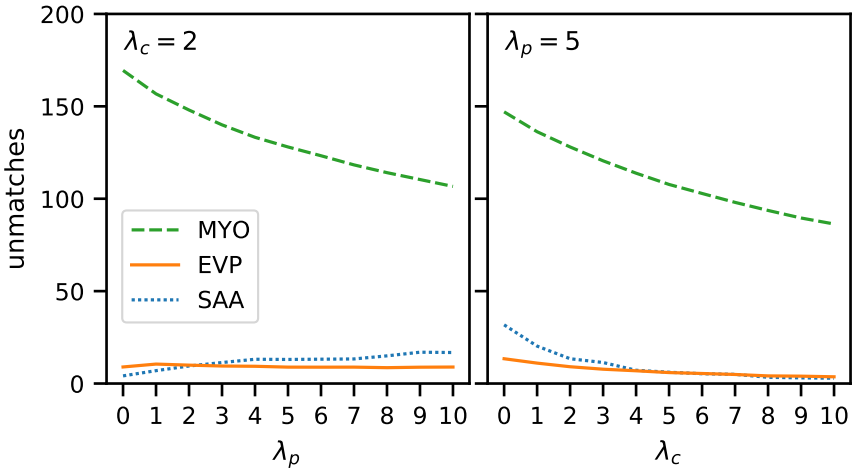
The results indicate that the stochastic strategies always outperform the myopic strategy. Below we discuss the results for interesting values for λ_p and λ_c.

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 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 7.

Figure 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.

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.

References

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

[2] N. McGuckin, A. Fucci, Summary of travel trends: 2017 national household travel survey, Washington, DC: Federal Highway Administration, US Department of Transportation.

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

[4] G. Homsi, B. Gendron, S. D. Jena, Dynamic and stochastic rematching for ridesharing systems: Formulations and reductions, in: M. Baïou, B. Gendron, O. Günlük, A. R. Mahjoub (Eds.), Combinatorial Optimization, Springer, 2020, pp. 193–201.

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

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

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

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

[9] A. Lee, M. Savelsbergh, Dynamic ridesharing: Is there a role for dedicated drivers?, Transportation Research Part B: Methodological 81 (2015) 483–497.

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

[11] C. Riley, A. Legrain, P. Van Hentenryck, Column generation for real-time ride-sharing operations, in: L.-M. Rousseau, K. Stergiou (Eds.), Integration of Constraint Programming, Artificial Intelligence, and Operations Research, Springer International Publishing, Cham, 2019, pp. 472–487.

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

[13] M. Stiglic, N. Agatz, M. Savelsbergh, M. Gradisar, The benefits of meeting points in ride-sharing systems, Transportation Research Part B: Methodological 82 (2015) 36–53.

[14] M. Stiglic, N. Agatz, M. Savelsbergh, M. Gradisar, Making dynamic ride-sharing work: The impact of driver and rider flexibility, Transportation Research Part E: Logistics and Transportation Review 91 (2016) 190–207.

[15] M. Stiglic, N. Agatz, M. Savelsbergh, M. Gradisar, Enhancing urban mobility: Integrating ride-sharing and public transit, Computers and Operations Research 90 (2018) 12–21.

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

[17] A. Mehta, AdWords and Generalized On-line Matching (2005) 1–19.

[18] N. Chen, N. Immorlica, A. R. Karlin, M. Mahdian, 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), Vol. 5555 LNCS, 2009, pp. 266–278.

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

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

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

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

[23] J. R. Birge, F. Louveaux, Introduction to Stochastic Programming, Springer New York, 2011.

[24] D. Luxen, 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, ACM, New York, NY, USA, 2011, pp. 513–516.