On-demand Ride-pooling with Walking Legs: Decomposition Approach for Dynamic Matching and Virtual Stops Selection

Navjyoth Sarma J S [1,2]
Email: nsarma.js@uci.edu ORCiD: 0000-0002-1304-0162

Krishna Murthy Gurumurthy [3]
Email: kgurumurthy@anl.gov ORCiD: 0000-0001-6791-4948

Michael F. Hyland [1,2] (Corresponding Author)
Email: hylandm@uci.edu ORCiD: 0000-0001-8394-8064

Younghun Bahk [1,2]
Email: ybahk@uci.edu ORCiD: 0000-0001-5233-1563

Felipe de Souza [3]
Email: fdesouza@anl.gov ORCiD: 0000-0002-4858-141X

Zifan Wang [4]
Email: zwang.geog@gmail.com ORCiD: 0000-0002-3458-3748

[1]University of California, Irvine Institute of Transportation Studies
4000 Anteater Instruction & Research Building, Irvine, CA 92697

[2]University of California, Irvine Civil and Environmental Engineering
4130 Engineering Gateway, Irvine, CA 92697

[3]Argonne National Laboratory, Transportation and Power Systems Division,
9700 S Cass Ave, Lemont, IL 60439, United States

[4]Inspur Systems Inc.
1501 McCarthy Blvd, Milpitas, CA 95035

Submitted to: Transportation Research Part C: Emerging Technologies
Submission Date: March 22, 2023

Declarations of interest: none

Abstract

Door-to-door (D2D) ride-hailing services currently dominate the mobility-on-demand (MOD) market, but several alternative MOD service types offer operational and societal benefits. Three such MOD services include D2D ride-pooling, corner-to-corner (C2C) ride-hailing, and C2C ride-pooling. Ride-pooling involves travelers sharing rides with strangers, and C2C service requires travelers to walk a short distance to/from a pickup/drop-off location. The goals of this study are two-fold. First, we aim to compare these four MOD services in terms of operator costs (e.g., vehicle kilometers per request served) and user costs (e.g., assignment time, wait time, walk time, and in-vehicle time). Second, we aim to develop an effective and scalable decision policy and solution algorithm for operating a C2C ride-pooling service. Underlying the C2C ride-pooling service is a highly dynamic sequential decision problem with an extremely large decision space. At each decision epoch, the operator must dynamically assign vehicles to requests, route and schedule vehicles, and assign travelers to pickup and drop-off (PUDO) locations. To address this problem, in a dynamic stochastic agent-based transportation network simulation environment, we propose decomposing the sequential decision problem into a matching, routing, and scheduling subproblem, and a PUDO locations selection subproblem. We use geographic, network, vehicle, and passenger information, as well as optimization techniques to solve the two subproblems. The computational experiments confirm a clear trade-off across the four services in terms of operator costs and user costs. With D2D ride-hailing as the baseline, (i) ride-pooling significantly reduces operator costs, while slightly increasing user costs; (ii) C2C slightly reduces operator costs while significantly increasing user costs; (iii) combining ride-pooling and C2C appears to provide additive benefits in terms of operator costs. Moreover, the computational results indicate that the proposed decision policy for operating the C2C ride-pooling is highly scalable and operationally effective.

Keywords: Shared Mobility; Service Design; Optimization; Algorithms; First-and-last-mile; Micromobility

1 Introduction

1.1 Background and Motivation

Mobility-on-demand (MOD) services, enabled by smartphones and their applications, and offered by Transportation Network Companies (TNCs) such as Uber, Lyft, and Didi, emerged over a decade ago. The dominant MOD service option offered by TNCs is ride-hailing (also known as ride-sourcing or e-hailing without shared rides), where vehicles can only carry one traveler request at a time. Several MOD service variants have emerged in recent years that aim to increase sharing and vehicle occupancies relative to ride-hailing in order to decrease negative externalities from these systems.

The second most common variant is the shared-ride or ride-pooling MOD service, such as Uber Pool and Lyft Line (now Lyft Shared) offered by Uber and Lyft. Ride-pooling MOD services involve pooling together sets of traveler requests that have similar, but not necessarily the same, origin and destination locations and request times into one vehicle. The envisioned benefits of ride-pooling services include decreases in required fleet sizes (i.e., decreases in required drivers active on Uber and Lyft platforms at a given instant) and operational miles, potentially lowering the prices service providers can offer customers. Service providers also often market ride-pooling as a service option that can reduce congestion, energy consumption, greenhouses gases, and local pollutants.

In conventional ride-hailing and ride-pooling MOD services, service providers offer and operate door-to-door (D2D) services, meaning that vehicles pick up and drop off travelers at their requested origin and destination locations, respectively. However, the need to pick up and drop off every traveler at their preferred origin and destination locations may significantly hamper MOD operational efficiency.

To address the operational inefficiencies associated with D2D MOD services, service providers now operate flexible ride-pooling MOD services wherein travelers must walk to pickup (PU) locations from their trip origins, and from drop-off (DO) locations to their trip destinations. In this paper, we refer to this MOD service as a corner-to-corner (C2C) ride-pooling service. C2C ride-pooling is quite similar to historical variants of dial-a-ride and flexible transit services, and several emerging on-demand microtransit services that require users to walk to/from their PU/DO locations instead of being served at their doorstep.

The premise behind C2C MOD services is that having travelers walk to nearby PUDO locations that are convenient for non-idle vehicles can increase MOD service fleet productivity measured in terms of travelers served per time unit, while also reducing fleet distance per request served and increasing vehicle occupancies. However, these operational improvements are likely to come at the cost of inconvenience for travelers, in terms of increased walk distances and increased request-to-destination travel times.

1.2 Research Goals, Problem Overview

Given this background on MOD services, this paper's goals are two-fold. First, we aim to systematically analyze the trade-offs between operator costs and user costs across four MOD services—D2D ride-hailing (D2D-RH), C2C ride-hailing (C2C-RH) D2D ride-pooling (D2D-RP), and C2C ride-pooling (C2C-RP), where every service responds to on-demand requests. Analyzing these four options requires decision policies and algorithmic frameworks for operating each MOD service, as well as a simulation environment. While the academic literature includes significant research on decision policies and solution algorithms for D2D-RH and D2D-RP services, the same is not true for C2C-RH and C2C-RP. Hence, the second goal of this paper is to develop a scalable and effective decision policy and algorithmic approach for dynamically operating C2C MOD services, particularly C2C-RP.

Developing a scalable and effective approach for C2C-RP services is a significant modeling and algorithmic challenge. The challenge stems from the size of the decision vector/space. A C2C-RP service provider needs to determine PUDO virtual stops for each traveler, while also matching travelers to vehicles, sequencing and scheduling PUDOs for each vehicle, and assigning vehicles to transportation network paths. Moreover, the fleet operator needs to make these decisions repeatedly in real-time, as new information (i.e., new requests and changes in network travel times) enters the system. To address the problem, we propose a decision policy that decomposes the full decision problem at each decision epoch into two subproblems that we solve sequentially and iteratively. The two subproblems are the PUDO locations selection subproblem and the vehicle-traveler matching subproblem, wherein the second subproblem embeds the vehicle sequencing and scheduling of traveler PUDOs. Given the decomposed sub-problems, we test several decision policies for a C2C-RP service. Specifically, we compare the case where the fleet operator assigns travelers to PUDO locations first for each candidate vehicle, then matches travelers to vehicles vs. a policy where the operator matches travelers to vehicles first, then assigns PUDO locations for the traveler based on the matched vehicle.

1.3 Terminology and Abbreviations

Before going any farther, we want to clarify some key terminology and abbreviations that we will use throughout the rest of the paper. We use the abbreviation PUDO when referring to pickup *and* drop-off, whereas we will use the abbreviation PU/DO when referring to pickup *or* drop-off. Additionally, if we are only referring to a pickup, we will use the abbreviation PU, and if we are only referring to a drop-off, we will use the abbreviation DO. A request is associated with two PUDO locations (or links)—a pickup location (or link) and a drop-off location (or link).

Moreover, we want to differentiate between *locations* and *links* where PUDOs occurs. We use the term PUDO *location* when discussing C2C-RP and C2C-RH in general, as in practice vehicles can pick up and drop off travelers anywhere (i.e., at nodes, or along links, or in designated PUDO spots). However, we use the term PUDO *link* to be precise in regard to the simulation model functionality in this study, as the simulation model allows PUDOs along network links.

1.4 Paper Outline

The remainder of this paper is structured as follows. Section 2 reviews the literature related to C2C MOD services and delineates the contributions of the current study. Section 3 describes the C2C-RP operational problem. Section 4 presents the decision policy and solution algorithm for the C2C-RP operational problem. Section 5 describes the simulation environment in which we test the proposed C2C-RP decision policy and compare the four MOD service options. Section 5 also describes the computational experiments, including the performance metrics and scenarios to assess the operator and user costs associated with the four MOD service options. Section 6 presents and discusses the results of the computational experiments. Section 7 concludes the paper with a summary as well as a discussions of future research directions.

2 Background and Literature Review

In this section, we present an overview of the existing literature related to C2C-RP. We focus our literature review on C2C-RP, as this is the focus of the current paper. We refrain from providing a detailed review of D2D-RH and D2D-RP, as there is extensive literature in this area. However, for readers interested in D2D-RH and D2D-RP, there are several recent review articles worth mentioning.

Zardini et al. (2022) present a recent extensive review of automated MOD (AMOD) that covers operational-level and planning-level problems related to MOD services with automated vehicles. Many of the insights from this review article relate to MOD services without automated vehicles. In addition to characterizing different elements of AMOD modeling studies, the paper reviews solutions for matching, routing, and rebalancing for D2D-RH, and these three subproblems plus ride-pooling for D2D-RP. Narayanan et al. (2020) provide a review of research on shared autonomous vehicles, which is another name for AMOD. Their review is quite broad, covering models of demand, parking, fleet sizing, vehicular traffic, matching, repositioning, pricing, and charging for electric vehicles. Mourad et al. (2019) survey shared mobility studies and focus on the operational problem. They consider travelers sharing rides with other passengers, and with parcels.

The remainder of this background and literature review section covers mobility services that include walking trips in general (Section 2.1), and mobility services with walking legs wherein the fleet operator selects PUDO locations for travelers in real-time (Section 2.2). We conclude by delineating the contributions of our study (Section 2.3).

2.1 PUDO Locations Selection

One of the critical decisions in operating C2C MOD services is where to pick up and drop off travelers. PUDO locations selection impacts (i) the distance/time/cost required for a vehicle to serve each request; and (ii) each traveler's walking distance. Thus, the PUDO locations selection problem significantly impacts both operator and user costs.

Several studies develop strategies for selecting PUDO locations in C2C services. Wang et al. (2022) categorize these strategies into three approaches: 1) strict meeting points, 2) relaxed meeting points, and 3) no meeting points. As the simplest approach, strict meeting points determines a single set of PUDO points for each vehicle (Wang et al., 2022). All riders sharing the same vehicle walk from their respective origins to one common PU location and from one common DO location to their respective destinations (Aissat and Oulamara, 2014; Czioska et al., 2017; Stiglic et al., 2015). This approach entails few stops per vehicle trip and thus is convenient for the drivers. In fact, the strict meeting points approach is more common in conventional carpooling and ridesharing where the driver has their own trip origin and destination, rather than in MOD services with a dedicated driver. While being convenient for drivers, the strict meeting points approach limits the potential to match drivers and riders in a region.

To address the shortcomings associated with the strict meeting points approach, several studies enable multiple intermediate PUDO locations on a vehicle's route, which is the relaxed meeting points approach (Wang et al. 2022). The relaxed meeting points approach clusters riders before assigning them a common location for PU/DO (Czioska et al., 2019; Martínez et al., 2015) or limits the candidate PUDO locations to a small subset of road nodes (Araldo et al., 2019; Gurumurthy and Kockelman, 2022). Studies using the relaxed meeting points approach, unlike the fixed meeting points approach, rarely assume that drivers have their own destinations (a special case is Miklas-Kalczynska and Kalczynski, 2020), rather, most studies focus on C2C-RP MOD services. A recent study by Lotze et al. (2022) is a special case of the relaxed meeting points approach, as they attempt to dynamically assign each new request to an existing planned vehicle stop. However, if the algorithm does not find a feasible existing PU and/or DO stop for a new request, then the algorithm has a vehicle pick up and/or drop off the traveler at the traveler's origin and/or destination.

Finally, the no meeting points approach does not restrict riders to share PU or DO points with other travelers (Wang et al., 2022). Some studies using the no meeting points approach (Balardino and Santos, 2015; Zheng et al., 2019) propose variants of the shortest covering path problem (SCPP) or generalized traveling salesman problem (GTSP) formulations for the carpooling or flex-route transit applications, where the selection of PUDO locations is a deterministic problem. Other studies dynamically select the PUDO locations for each stochastic request (Fielbaum et al., 2021; Li et al., 2020; Lyu et al., 2019). Given that the approach in our study involves no meeting points and dynamic selection of PUDO locations, we focus the remainder of our review on studies that dynamically select PUDO locations without explicit meeting points.

2.2 Dynamic PUDO Locations Selection

Dynamic selection of PUDO locations without explicit meeting points theoretically permits better solutions in terms of service quality and operational efficiency (Wang et al., 2022) than approaches with predetermined or limited meeting points. Table 1 compares C2C MOD studies with dynamic PUDO locations selection.

Most C2C MOD studies focus on ride-pooling since the purpose of incorporating walking access and egress trip legs is often the same as incorporating pooled rides—to reduce fleet miles, operational costs, congestion, and emissions (Fielbaum et al., 2021; Li et al., 2020; Lyu et al., 2019). However, Martin et al. (2021) consider C2C in a ride-hailing service with the objective of minimizing the riders' travel cost. With a fixed vehicle speed, Martin et al. (2021) test the performance of C2C-RH with a 400-meter walk range in the Manhattan road network. They also mainly compare static PUDO assignment with dynamic PUDO locations selection for C2C-RH and find that the impact on operational efficiency is relatively minor.

Other studies simulate C2C-RP services on various real-world city networks including Manhattan, Shanghai, and Chengdu (Fielbaum et al., 2021; Li et al., 2020; Lyu et al., 2019). These studies all assume deterministic link travel times or fixed vehicle speeds (i.e., the vehicles travel the same speed on all links). Notably, two studies that do incorporate stochastic travel times in large scale networks—Gurumurthy and Kockelman (2022) and Zwick et al. (2021) in POLARIS and MATSim, respectively—use fixed or relaxed meeting points approaches. Hence, ours is the first study to incorporate stochastic link travel times (and congestible links) in a simulation, where the C2C service does not have fixed meeting points.

The number of MOD requests for the studies in Table 1 range from 9,000 to 10.7 million. However, the hourly average request rates only range from 9,000 to 22,000 requests per hour. Fleet sizes in the literature range from 40-13,181. Our study has the longest analysis period of 24 hours, and the number of requests and fleet size are consistent with prior research.

Vehicle capacity ranges from 1-20. As our study is the only study to consider C2C-RP and C2C-RH, it is also the only one to consider vehicle capacities of one and greater than one in the same study.

For maximum walking distance, parameter values range between 300 and 1000 meters. Similar to other studies, we vary the parameter between 250 and 1000 meters, to understand its impact on key performance metrics.

For walking speed, every study assumes 5 or 5.04 km per hour. In this study, we consider a walking speed of 5 km per hour, but also a 'walking' speed of 20 km per hour that is more akin to an electrified bike or scooter. We include this parameter to evaluate the benefits of electrified bikes and scooters on system performance metrics for C2C-RH and C2C-RP.

The third to last column in Table 1 refers to whether vehicles wait for travelers at PU locations. There are two interrelated aspects here, related to the simulation environment and the fleet's operational strategy. If the simulation does not include stochastic link travel times, then the service provider fully determines whether vehicles wait for travelers at PU locations (the second aspect of this service dimension). However, if the simulation does include stochasticity, then vehicle waiting is not fully in the operator's control. As ours is the only study with stochastic travel times, it is also the only study where the arrival time of vehicles at PU locations is uncertain; hence, vehicles may arrive before requests and have to wait. Of course, this is consistent with a real-world MOD service, where travel time is uncertain. Interestingly, Lyu et al. (2019) also consider the case where the vehicles wait at PU locations. However, since their model is deterministic in link travel times, they intentionally have vehicles wait for travelers at PU locations. Their model explicitly captures the waiting cost for onboard passengers. The other studies do not specifically mention the possibility of vehicles waiting for travelers at PU locations (Fielbaum et al., 2021; Li et al., 2020).

Table 1: Summary of Existing Studies Analyzing C2C MOD Services (with dynamic PUDO locations selection)

Paper	**MOD Services in Study**	**Road Network & Travel Time**	**Avg. Request Rate (req./hr.) and Simulation Period**	**Fleet Size (veh.) and Vehicle Capacity (seats)**	**Walk Range (m) and Walk Speed (km/h)**	**Vehicle Waiting for Request?**	**Procedure for Identifying PUDO Locations Candidates**	**Selecting PUDO Locations and Vehicle-Request (R-V) Matching**
Fielbaum et al. (2021)	C2C RP D2D RP	Manhattan, NY with fixed link travel times	9,970 over 1 hour	2,000-3,000 and 6-9	417, 1000 and 5	No	Recursive search of neighboring PUDO locations within walk range that reduce vehicle travel cost	Joint optimization of PUDO locations selection and R-V matching
Li et al. (2020)	C2C RP	Shanghai, China with fixed vehicle speeds	10,000 over 2 hours	1,200 and 20	100, 200, 300, 400, 500 and 5.04	No	All PUDO locations within walk range	R-V matching first, PUDO locations selection second
Lyu et al. (2019)	C2C RP	Chengdu, China with fixed vehicle speeds	22,500 over 28 days and 17 hours per day	12,725-13,181 and 3	300 and N/A	Yes, planned early vehicle arrivals	All PUDO locations within walk range	PUDO locations selection for pooled requests first, R-V matching second
Martin et al. (2021)	C2C RH	Manhattan, NY with fixed vehicle speeds	40–160 total requests	40–400 and 1	400 and 5.04	No	All nodes from request's origin within walk range. DO location unchanged.	**Dynamic Case:** PU location selection for candidate R-V pairs first, R-V matching second. **Static Case:** PU location selection independent of vehicle
This study	C2C RH C2C RP D2D RP D2D RH	Bloomington, IL with dynamic stochastic congestion-dependent link travel times	9,200 over 24 hours	1,000–10,000 and 1 (RH), 4 (RP)	250, 500, 750, 1,000 and 5, 20	Yes, stemming from vehicle arrival time uncertainty	PUDO links selected based on walk range and vehicle travel direction	**Option 1:** R-V matching first, PUDO locations selection for matched R-V pairs second. **Option 2:** PUDO locations selection for candidate R-V pairs first, R-V matching second

Finally, we want to describe the different decision policies and algorithmic approaches for C2C-RP in the literature. Lyu et al. (2019) maintain a waiting queue for unassigned requests. When new requests arrive, their approach attempts to sequentially match each new request with unassigned requests already in the waiting queue to form companion candidate pairs. Then, they determine the 'maximum sharing satisfaction utility' set of companion candidate pairs where the PUDO locations of candidate pairs can be pooled together in one vehicle. Alternative PUDO locations are determined by finding walkable nodes from each request's origin and destination in the companion pair, and then choosing the best set of PU and DO locations that minimize total tour distance. Next, the approach aims to insert additional requests into the previously identified best companion pairs. The seat capacity is four in their study, so only two more requests can be inserted into the original companion pair. Finally, their approach assigns the closest idle vehicle to the first stop in the stop sequence corresponding to the set of least cost requests. A main difference between our approach and that in Lyu et al. (2019) is that they do not allow the insertion of new requests into a vehicle's route plan, until the vehicle completes all its current PU and DO tasks. This almost certainly limits the operational effectiveness of the proposed decision policy and solution algorithm.

Li et al. (2020) select PUDO locations for each new request after matching each request to a vehicle. Li et al. (2020) sequentially assigns each new request to a vehicle, wherein the algorithm considers the cost of inserting the new request's origin and destination locations into the planned routes of each vehicle. After request-vehicle matching, the algorithm adjusts the traveler's PUDO locations to minimize the matched vehicle's detour. The approach evaluates all walkable PUDO locations near the request's origin and destination, respectively. Additionally, the approach includes a post-processing stage in which PU and/or DO locations of two requests are merged if they are close to each other. These requests are assigned to the same vehicle subject to time window constraints and vehicle not waiting for request constraint.

There are three key differences between our study's decision policy and solution algorithm compared to Li et al. (2020). First, we perform request-vehicle matching using bi-partite matching after batching requests together over a time interval. Second, our methodology allows for the flexibility to adjust PUDO locations either before or after request-vehicle matching. Third, we shortlist PU/DO candidates for a request-vehicle pair based on the vehicle's planned travel direction, in addition to walk distance restrictions. Experiments suggest that this added condition improves operational efficiency and computational run time.

Fielbaum et al. (2021) batch new requests received between decision epochs and solve an integer-linear program (ILP) assignment problem that may assign multiple requests to the same vehicle in a single decision epoch. They extend the request-vehicle ILP matching algorithm in Alonso-Mora et al. (2017) to incorporate the selection of PUDO locations for each request-vehicle match. The extended model and algorithm shortlists PUDO candidates for a request-vehicle match by evaluating neighboring network nodes around the request's origin and destination—if a neighboring node can reduce the matching cost for the request-vehicle pair it is put on the shortlist. This procedure continues by evaluating the neighbors' neighbors until no remaining PUDO location candidates reduce the total cost of matching. The solution approach then jointly optimizes request-vehicle assignment and selection of PUDO locations for each request. The joint optimization of an ILP request-vehicle matching with PUDO locations selection is computationally intensive, limiting its scalability. Fielbaum et al. (2021) mention that incorporating PUDO locations selection increases request-vehicle matching by approximately 10 times.

Our approach differs from Fielbaum et al. (2021) in several ways. First, we decompose PUDO locations selection and request-vehicle matching to decrease computational complexity and increase scalability. Our proposed approach also considers the directionality of non-idle vehicles when shortlisting PU and DO locations for each request—this is not the case in Fielbaum et al. (2021). Finally, unlike Fielbaum et al. (2021), our approach considers the resequencing of all planned PUDO stops assigned to each vehicle, after inserting each new request into a vehicle's planned route, and after choosing each new request's PUDO locations.

2.3 Contributions

This paper makes several contributions to the academic literature. The first set of contributions relate to the evaluation of C2C-RP against alternative MOD services and the design of C2C-RP services. The second set of contributions relates to the decision policy and solution algorithm we develop to solve the C2C-RP problem.

This study compares four MOD services (C2C-RH, C2C-RP, D2D-RH, D2D-RP) in terms of operator and user costs. Prior research analyzes at most one pair of these MOD services. Conversely, we systematically compare all four services in terms of operator and user costs, under a unified modeling and algorithmic framework, and a common simulation environment. Assuming D2D-RH is the baseline service, we can isolate the operational benefits (and user costs) stemming from (i) pooling rides and (ii) incorporating walking legs. We can also evaluate the synergistic benefits (and costs) between (i) and (ii).

Moreover, we perform the comparison in a state-of-the-art agent-based transportation system simulation model—POLARIS—that captures road network congestion dynamics. This is another contribution of the study. The simulation permits analysis of a severely overlooked problem related to operating C2C MOD services, namely, vehicles needing to wait for travelers at PU locations (Wang et al., 2022). In the real world, and in ABM simulations that capture link travel time stochasticity, it is frequently the case that vehicles arrive at PU locations before travelers. This negative outcome significantly impacts the efficiency and productivity of vehicles in the fleet. Capturing link travel time uncertainty in the simulation environment permits a much more realistic assessment of the benefits and costs of C2C service relative to D2D service.

Our study also evaluates the impacts of maximum walking distance on fleet performance for a dynamic C2C-RP service without meeting points. This is an important service design parameter for C2C-RP that impacts customer and operator costs. Our study is also the first to consider a 'walking' speed in a C2C MOD service that is reflective of travelers having access to an electrified scooter or bike.

Additionally, we propose novel decision policies and algorithmic strategies for C2C-RP that are scalable and operationally effective. Ultimately, as described above, the problem of dynamically operating a C2C-RP service is a highly complex decision problem, where the enormous size of the decision space for even small problem instances essentially precludes elegant models, algorithms, and decision policies, which appears to have stymied research related to C2C-RP. This is particularly discouraging given the potential of a C2C-RP to meet societal and transportation system goals, such as reducing vehicle miles traveled, traffic congestion, energy consumption, and harmful emissions, while providing lower cost (compared to D2D-RH) and high-quality mobility. This paper tackles the C2C-RP service problem as the complex set of engineering problems that it entails. The proposed solution framework considers the trade-off between computational run time and solution quality within a particular decision epoch. We decompose the decision problem in each epoch into a subproblem that matches vehicles to travelers, sequences and schedules PUDOs for each vehicle, and a second subproblem that assigns travelers to PUDO locations. This decomposition permits the use of optimization techniques for the first subproblem, and a flexible solution algorithm that explores the solution space for the second subproblem. Moreover, we propose an algorithm that efficiently solves these two subproblems sequentially and iteratively.

Our proposed solution approach includes several important algorithmic contributions. We only allow PUDOs on links that are in the planned direction of travel of non-idle vehicles. Links in conflicting directions will either never be chosen in the PUDO selection step, or if they are chosen, will likely harm the operational efficiency of the fleet. Moreover, like several other studies, our approach decomposes the decision problem into the request-vehicle matching subproblem and the PUDO selection problem. However, unlike other studies in the literature, our approach allows PUDO selection before or after request-vehicle matching. This permits the service provider to explicitly trade-off between computational run time and solution quality, as selecting PUDOs for candidate request-vehicle matches before the matching step improves solution quality but it increases computational run time. Importantly, we also batch requests over a given time interval and solve the batched request-vehicle matching problem using the bi-partite assignment problem as the engine. Our solution approach leverages the benefits of bi-partite matching (i.e., it is highly scalable) while addressing its major limitation for ride-pooling (i.e., it can only assign one request per vehicle) by iteratively matching requests to vehicles within one decision epoch, such that multiple requests can ultimately be assigned to one vehicle in each decision epoch.

3 Problem Description

3.1 Nomenclature

R:	Set of all requests, indexed by $r \in R$
V:	Set of all vehicles, indexed by $v \in V$
L:	Set of all links in the road network, indexed by $l \in L$
T:	Set of time steps in simulation, indexed by $\tau \in T$
Δ:	Time between decision epochs
(r, v):	Request r, Vehicle v pair (match or match candidate)
o_r:	Original PU link of request r [(x,y) coordinate representing average location along the link]
d_r:	Original DO link of request r [(x,y) coordinate]
o_r':	Adjusted PU link of request r [(x,y) coordinate]
d_r':	Adjusted DO link of request r [(x,y) coordinate]
L_{o_r}:	Set of PU links that are within walk range from o_r, indexed by $o_r' \in L_{o_r}$
L_{d_r}:	Set of DO links that are within walk of d_r, indexed by $d_r^{'} \in L_{d_r}$
τ_r:	Request initiation time of request r [Simulation time (sec.)]
V_r:	Set of feasible candidate vehicles for unassigned request r
R_v:	Set of requests in v's PUDO sequence (Empty for idle vehicles)
c_{veh}:	Vehicle capacity [Seats]
k_{veh}:	Maximum number of candidate vehicles for each unassigned request
k_{veh}^{idle}:	Minimum number of candidate *idle* vehicles for each unassigned request
θ_{max}:	Maximum angle between a request's Euclidean path vector and a vehicle's average future path vector (see Figure 2)
D_{dir}^{max}:	Maximum remaining distance for a candidate vehicle to complete its current tour so as to be exempt from directionality and detour compatibility constraints [Meters]
$D_{rev}(r, v)$:	Euclidean distance between vehicle v's current link and the projection of o_r onto the average future path of vehicle v [Meters]
D_{rev}^{max}:	Maximum distance a vehicle can travel in a direction opposite to vehicle's average future path to pick up a new request (see Figure 3) [Meters]
$D_{detour}(r, v)$:	Euclidean detour distance from vehicle v's current average path to o_r (see Figure 3) [Meters]
D_{detour}^{max}:	Maximum Euclidean detour distance from vehicle v's current average path to o_r (see Figure 3) [Meters]
$\tau_{r,v}^{PU}$:	Time at which request r gets picked up by vehicle v at the finalized PU link [Simulation time (sec.)]
$\tau_{r,v}^{DO}$:	Time at which request r gets dropped off by vehicle v at the finalized DO link [Simulation time (sec.)]
$t_{r,v}^{w}$:	Total wait time for request r to be picked up by vehicle v [Duration (sec.)]
t_{max}^{w}:	Maximum PU wait time for a request [Duration (sec.)]
$t(o_r, d_r)$:	Approximate direct travel time from o_r to d_r, calculated based on Euclidean distance and average hourly speeds in the zones associated with o_r and d_r during time of matching [Duration (sec.)]
$t_{r,v}^{iv}$:	In-vehicle travel time (IVTT) for request r in vehicle v from the finalized PU link location to finalized DO link location [Duration (sec.)]
$t_{r,v}^{iv_{max}}$:	Maximum IVTT for request r in vehicle v from the finalized PU link location to finalized DO link location (initially o_r and d_r, respectively) [Duration (sec.)]
$t_{max_{abs}}^{iv}$:	Maximum allowable increase in IVTT for any request [Duration (sec.)]
$t_{max_{rel}}^{iv}$:	Maximum allowable increase in IVTT relative to a request's direct travel time [%]
$t_{S_{r,v}}$:	Time it takes for vehicle v to complete PUDO sequence $S_{r,v}$ after insertion of request r [Duration (sec.)]
$C_{v,r_{(o_r,d_r)}}$:	Cost of inserting request r into vehicle v with PU at o_r and DO at d_r [Duration (sec.)]
w_{wt}:	Weight parameter for wait time in insertion cost function
w_{ivtt}:	Weight parameter for IVTT in insertion cost function
D_{walk}^{PU}, D_{walk}^{DO}:	Maximum effective walk ranges from o_r to o_r' and d_r' to d_r, respectively [Meters]
D_{walk}^{max}:	Maximum walk range for PU leg, and Maximum walk range for DO leg [Meters]
s_w:	Average walk speed [Kilometers per hour]
θ_{buf}:	The buffer angle around coordinate-axes to shortlist candidate PUDO links based on their link bearings and the vehicle's travel direction (Section 4.3.2) [Degrees]
k_{PUDO}:	Maximum number of candidate feasible PU/DO links to be chosen for each new (r, v) match or match candidate for PUDO links adjustment
γ_{C2C}:	Boolean parameter to set sequence of PUDO links adjustment and request-vehicle matching. TRUE indicates PUDO links are adjusted before matching stage. FALSE indicates PUDO links are adjusted after matching stage

3.2 Problem Statement

In this section we describe the C2C-RP problem. We model a central controller operating a fleet of homogeneous vehicles $V = \{1,2,..,v,...|V|\}$. These vehicles provide service to a set of requests $R = \{1,2,..,r,...|R|\}$ over the analysis/simulation period $T = \{1,2,..,\tau,...|T|\}$, where each request $r \in R$ has an origin o_r and a destination d_r within a geographical service region or network. All requests have a walking speed of s_w. Each request, r, also has a request initiation time τ_r, as well as set of feasible PU locations L_{o_r} and feasible DO locations L_{d_r}. Most importantly, the τ_r values for each request r are unknown to the fleet controller before τ_r. Similarly, the fleet controller only knows L_{o_r} and L_{d_r} for each request r at τ_r. Hence, the fleet controller faces a stochastic dynamic decision problem.

The fleet controller's problem is multi-objective; the objectives include maximizing requests served, minimizing request in-system time, and minimizing total fleet miles. To meet its objectives, the fleet controller must dynamically assign or match requests to vehicles, where (r,v) denotes a request-vehicle match or a candidate request-vehicle match, and $X_{v,r}$ is the binary decision variable equal to one if (r,v) is a match. In this study, we batch requests that arrive every Δ seconds, which is the time between decision epochs. After matching a request r to a vehicle v, the fleet controller cannot later reject the request r or serve it with another vehicle. Embedded inside the request-vehicle matching subproblem is the sequencing and scheduling of traveller PUs and DOs. Moreover, the fleet controller must determine PUDO locations for requests, where $o'_r \in L_{o_r}$ and $d'_r \in L_{d_r}$ denote the selected PU and DO locations of request r, respectively. In this study, PUDO locations are at the upstream node of each link. Therefore, the fleet controller must determine PUDO *links*, rather than a generic PUDO *location*.

The fleet controller also faces several hard constraints. The first is the vehicle capacity constraint, where c_{veh} is the parameter representing vehicle capacity. Other constraints and their associated parameters include maximum PU wait time for requests (t^{w}_{max}), maximum in-vehicle travel time detour ($t^{iv}_{max_{abs}}$) and maximum percent increase in in-vehicle travel time relative to the request's shortest path travel time ($t^{iv}_{max_{rel}}$). We also enforce a maximum walk distance for both the access and the egress walking legs (D^{max}_{walk}). We also enforce a constraint on non-idle vehicles that they can only travel in the opposite direction of their planned route for a given maximum distance (D^{max}_{rev}).

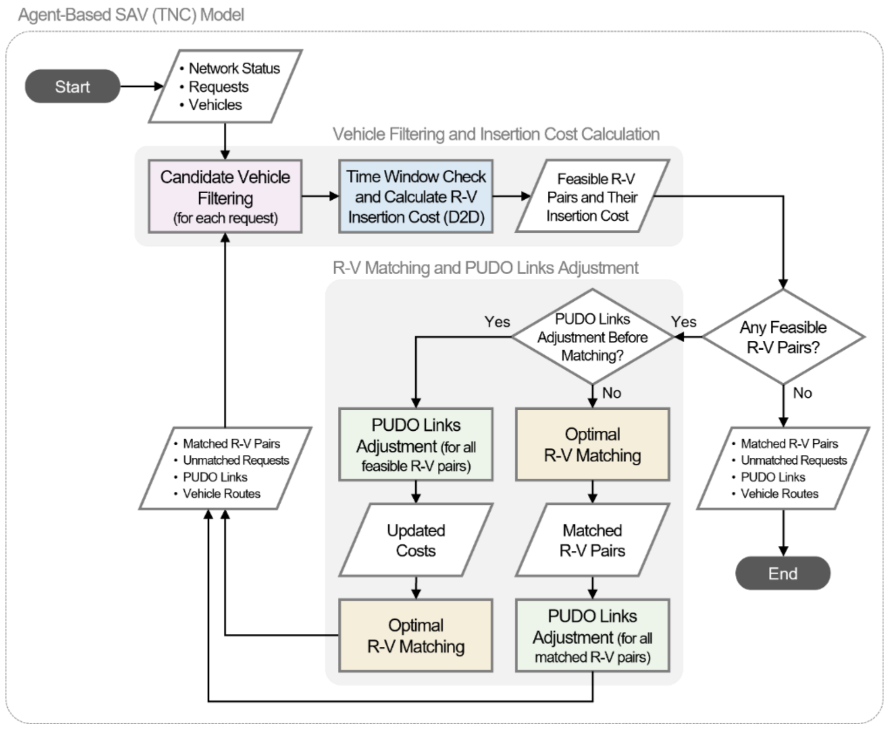
4 Solution Methodology

4.1 Overall Flow and Sequence

Figure 1 shows the flowchart of the decision policy and iterative solution algorithm at each decision epoch for the C2C-RP problem. The time between decision epochs is Δ, a fixed input parameter. The inputs at each decision epoch are the current status of requests, vehicles, and the network inclusive of link, node, and zonal information. The outputs at each decision epoch are the new assignments of requests to vehicles, the unmatched requests, the updated PUDO links for every matched request, and the updated vehicle routes and vehicle schedules. Each iteration of each decision epoch involves 3 main stages: (i) finding feasible (r,v) match candidates, (ii) optimizing request-vehicle matches, and (iii) adjusting PUDO links. The following three subsections describe these three main stages in detail.

However, we first want to motivate and describe the iterative nature of our decision policy and solution algorithm. The algorithm is iterative because we use bi-partite matching as the engine of the optimal request-vehicle matching module, similar to Hyland and Mahmassani (2020), Sarma et al. (2020), and Simonetto et al. (2019). Bi-partite matching is highly efficient because dropping the integrality constraint in the math program and using an exact solution method still returns binary solutions. Unfortunately, this property, stemming from the constraint matrix being totally unimodular, comes at a cost—a vehicle can only be matched to at most one request per call to optimal R-V matching module. To partially address the shortcoming of the bi-partite matching approach, we call the bi-partite matching module multiple times in each decision epoch. After solving one instance of the bi-partite matching problem, we then (i) insert the PU and DO links of the newly assigned requests into their matched vehicles and (ii) find a new set of feasible match candidates composed of remaining unassigned requests and vehicles with updated planned routes and schedule. Next, we call the bi-partite matching algorithm again. This iterative process terminates at each decision epoch when there are no feasible request-vehicle matches remaining.

We also want to note that we test two different decision policies, wherein the only difference is whether we adjust PUDO links for *candidate* request-vehicle matches *before* optimal request-vehicle matching, or we adjust PUDO links for *finalized* request-vehicle matches *after* optimal matching. If the γ_{C2C} parameter is equal to one, the algorithm adjusts PUDO links before matching.



Agent-Based SAV (TNC) Model
Start
• Network Status
• Requests
• Vehicles
Vehicle Filtering and Insertion Cost Calculation
Candidate Vehicle Filtering (for each request)
Time Window Check and Calculate R-V Insertion Cost (D2D)
Feasible R-V Pairs and Their Insertion Cost
R-V Matching and PUDO Links Adjustment
Any Feasible R-V Pairs?
Yes
No
PUDO Links Adjustment Before Matching?
Yes
No
PUDO Links Adjustment (for all feasible R-V pairs)
Updated Costs
Optimal R-V Matching
Optimal R-V Matching
Matched R-V Pairs
PUDO Links Adjustment (for all matched R-V pairs)
• Matched R-V Pairs
• Unmatched Requests
• PUDO Links
• Vehicle Routes
• Matched R-V Pairs
• Unmatched Requests
• PUDO Links
• Vehicle Routes
End

Figure 1. Overview of C2C-RP decision policy and solution algorithm (R-V = Request-Vehicle)

4.2 Finding Feasible Request-Vehicle Match Candidates

Ultimately, the C2C-RP sequential decision problem requires matching requests to vehicles and choosing PUDO links for each request. As a preliminary step, we first determine feasible vehicle candidates V_r for each unassigned request r at the beginning of each iteration of the solution algorithm.

We want to add a vehicle v to a request r's feasible candidate vehicle set V_r if it passes all directionality related checks and if inserting request r into the PUDO sequence of vehicle v does not violate time window constraints of r or any other requests that are already present in v's PUDO sequence. We also want to have at most k_{veh} feasible candidate vehicles for each request, out of which we also want k_{veh}^{idle} nearby idle vehicles. We include both idle and non-idle vehicles in each request's candidate vehicle set. V_r, to improve fleet utilization.

Algorithm 1 describes the overall procedure for finding feasible vehicle candidates, for a single unassigned request. The solution approach repeats Algorithm 1 for each unassigned request at the beginning of each iteration of the iterative optimal matching procedure. Importantly, as we determine candidate vehicles for each request independently, we can easily parallelize across requests.

The following subsections detail the requirements each candidate vehicle must meet to be a feasible match candidate for an unassigned request. These requirements relate to the directionality of vehicle and request travel, the maximum detour for each new request and the requests inside each non-idle vehicle, and time-window constraints. Moreover, in order to properly determine whether a request-vehicle pair meets the maximum detour and time window requirements, it is necessary to determine the optimal sequence and schedule of traveler PUDOs for a given vehicle (considering each new potential request).

Algorithm 1 – Finding feasible vehicle candidates for unassigned request r

```
Input: Set of Vehicles $V$, Unassigned request $r$
Output: Set of feasible vehicles for request $r$, $\boldsymbol{V_r}$;
Optimal PUDO sequence for all requests in $v$, including new request $r$, for all vehicles $v \in V_r$, $\boldsymbol{S^*_{r,v}}$;
Cost of inserting $r$ into each vehicle $v \in V_r$, $\boldsymbol{C_{v,r_{(o_r,d_r)}}}$
Procedure:
$V_r = \emptyset$
$k = 0$
Find $k^{idle}_{veh}$ idle vehicles near request $r$ and store in set $V^{idle}_r$
$V_r = V_r \cup V^{idle}_r$
$k = n(V_r)$
Let $V^{other}_r$ be the set of all other available vehicles (idle/non-idle with capacity available) near request $r$ within the $t^w_{max}$ time range.
for each $v \in V^{other}_r$ do
    if $k == k_{veh}$ then
        break
    end if
    if $v$ is idle and $v \notin V_r$ then
        $V_r = V_r \cup v$
        $k = k + 1$
        continue
    end if
    else
        Check direction and detour compatibility for $r$ with non-idle vehicle $v$ (Algorithm 2)
        if TRUE then
            $S^*_{r,v}$ = Find optimal PUDO sequence after inserting $r$ into $v$'s PUDO sequence (Algorithm 3)
            Evaluate time-window constraints based on $S^*_{r,v}$ (Section 4.2.3)
            if TRUE then
                Calculate insertion cost based on $S^*_{r,v}$ (Section 4.2.4)
                $V_r = V_r \cup v$
                $k = k + 1$
            end if
        end if
    end else
end for
return $V_r$; $S^*_{r,v}$ and $C_{v,r_{(o_r,d_r)}}$ $\forall v \in V_r$
```

4.2.1 *Direction and Vehicle Detour Checks*

This step involves checking the direction compatibility and vehicle path detour constraints for matching a new request r with a non-idle vehicle v that has capacity available to serve the new request. A vehicle is considered non-idle if it has a non-empty PUDO sequence due to unserved requests matched to it either from a previous decision epoch or from a previous iteration of the optimal matching procedure in the same decision epoch. The procedure to evaluate direction and detour compatibility of a candidate (r, v) pair is described in Algorithm 2. Figure 2 and Figure 3 provide illustrations of the procedure. Direction and vehicle detour constraints are not evaluated for a non-idle vehicle if it is close to completing its current PUDO sequence (Based on the D_{dir}^{max} parameter). Further description of this step is added in Appendix A.

Algorithm 2 – Evaluating direction and detour compatibility of request r with non-idle vehicle v

```
Input: Unassigned request $r$, Candidate non-idle vehicle $v$
Output: Direction Compatibility TRUE/FALSE
Procedure:
Calculate $D_v$ – The remaining distance in vehicle $v$'s current tour (PUDO sequence)
if $D_v \leq D_{dir}^{max}$ then
    return TRUE
end if
Calculate angle θ between vectors representing average future path of vehicle $v$ and Euclidean path between $o_r$ and $d_r$ (See Figure 2 and Appendix A for description)
if $\theta > \theta_{max}$ then
    return FALSE
end if
Calculate $u$ parameter value (Eqn. $A2$, Figure 3, Appendix A)
if $u > 1$ then
    return TRUE
end if
if $u < 0$ then
    $v$ needs to travel in reverse direction
    Calculate $D_{rev}(r.v)$ (Equation $A3$, Figure 3, Appendix A)
    if $D_{rev}(r.v) > D_{rev}^{max}$ then
        return FALSE
    end if
else if $u \leq 1$ then
    Calculate $D_{detour}(r, v)$ (Equation $A4$, Figure 3, Appendix A)
    if $D_{detour}(r, v) \leq D_{detour}^{max}$ then
        return FALSE
    end if
end if
return TRUE
```

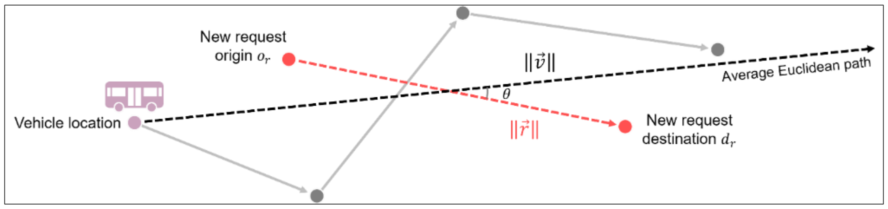


Figure 2. Directionality check for finding feasible request-vehicle pairs.

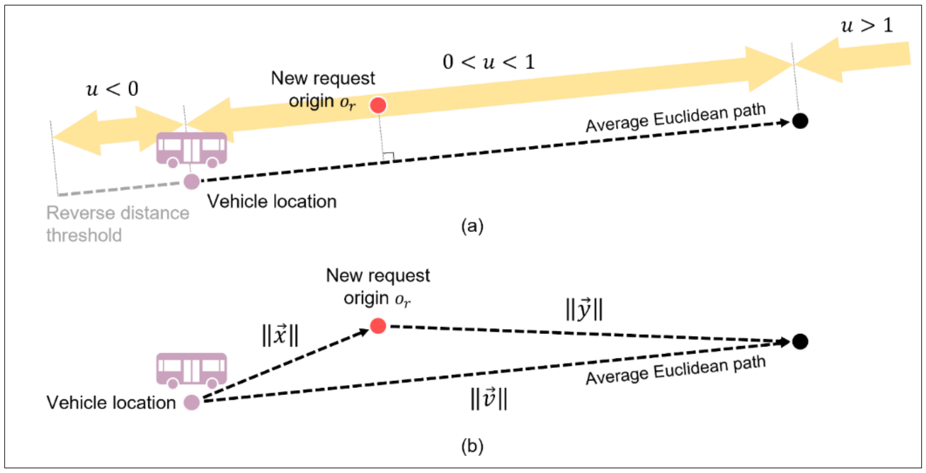


Figure 3. Vehicle path detour check for finding feasible request-vehicle pairs, considering (a) PU travel direction threshold and (b) PU travel detour threshold.

4.2.2 Finding Optimal PUDO Sequence after Insertion

This step involves finding the optimal order of PUDOs after the new unmatched request is inserted into each candidate vehicle obtained in the previous stage. The insertion location and the revised optimal PUDO sequence is found using a greedy algorithm based on R-tree similar to Gurumurthy and Kockelman (2022). The procedure is described in Algorithm 3 and illustrated in Figure 4.

Algorithm 3 – Find optimal PUDO sequence after inserting request r into (candidate) vehicle v

```
Input: Unassigned request $r$, Candidate vehicle $v$
Output: Optimal PUDO Sequence $S^*_{r,v}$, Time to complete sequence $t_{S^*_{r,v}}$
Procedure:
Insert $v$'s current link location coordinates into R-tree
Insert coordinates of $v$'s current PUDO links into R-tree
Insert $o_r$ and $d_r$ into R-tree
Let $x_v$ be the current link location coordinates of $v$
Let $S^*_{r,v}$ be an empty FIFO queue denoting the optimal PUDO sequence after inserting $r$ into $v$
while R-tree is not empty do
    $y_v$ = Query from R-tree the nearest location to $x_v$ that satisfies precedence constraints
    Push $y_v$ to $S^*_{r,v}$
    Delete $y_v$ from R-tree
    $x_v = y_v$
end while
return $S^*_{r,v}$ , $t_{S^*_{r,v}}$
```

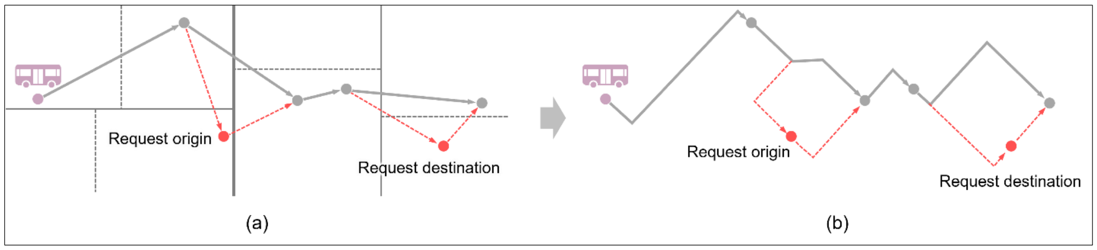


Figure 4. R Tree Query to find optimal PUDO sequence: (a) PUDO Sequencing and (b) Updated Path

4.2.3 Evaluation of Time-Window Constraints

In this step, time window constraints are checked for each request (new and previously assigned) in the revised PUDO sequence obtained from Algorithm 3. Starting from the vehicle's current link location and the current simulation time, we estimate arrival times at each PUDO link using a Euclidean distance approximation for each trip leg in the candidate vehicle's revised PUDO sequence. We use the average of hourly zonal speeds at the origin and destination zones of each trip leg in the tour (at the current simulation time) to estimate the arrival times at each PUDO link. The subroutine checks the following two time window constraints for all requests in the revised PUDO sequence for each (r, v) match candidate:

1. Latest PU time constraint - The latest PU time constraint mandates that the vehicle arrival time at the PU link for any request must be no later than the latest allowable PU time of the request corresponding to that PU link. The PU link is assumed to be the origin link for the new request that is inserted into the candidate vehicle (before PUDO links adjustment). For requests that have already been assigned to the vehicle, PU links in the PUDO sequence could either be their origin link or an adjusted origin based on the decision made in Section 4.3 while matching the request to the vehicle. The latest allowable PU time of a request is the request time plus the value of the t_{max}^{w} parameter. The constraint is expressed in Eqn. 1 and Eqn. 2.

$$t_{r,v}^{w} = \tau_{r,v}^{PU} - \tau_{r} \tag{1}$$

$$t_{r,v}^{w} \leq t_{max}^{w} \tag{2}$$

2. Maximum in-vehicle travel time delay constraint – This constraint mandates that the in-vehicle travel time for each request in the revised PUDO sequence (both previous assigned and new request) from PU time to DO time should not be more than a delay threshold. The subroutine includes two delay thresholds, one is an absolute delay value, and the other is a relative delay value. The constraint is expressed in Eqn.3 to Eqn. 5.

$$t_{r,v}^{iv} = \tau_{r,v}^{DO} - \tau_{r,v}^{PU} \tag{3}$$

$$t_{max}^{iv} = min\left(t(o_r, d_r) + t_{max_{abs}}^{iv}, t(o_r, d_r) \times \left[1 + t_{max_{rel}}^{iv}\right]\right) \tag{4}$$

$$t_{r,v}^{iv} \leq t_{max}^{iv} \tag{5}$$

Euclidean distance along with the average zonal speed at the origin and destination of the request at the time of matching are used to approximate the direct automobile travel time. (r, v) match candidates that fail to meet either of the time window constraints after inserting the new request are discarded in this step.

4.2.4 Insertion Cost Calculation for Feasible Candidates

In this step, the cost of adding the new request to a candidate vehicle's tour is calculated for each feasible (r, v) match candidate that fulfilled all time window constraints, and directionality and detour related constraints listed in the previous steps. We use the cost associated with each (r, v) match candidate in the optimal matching stage (described in Section 4.4). The insertion cost is calculated as a factor of change in total request wait time and in-vehicle travel time for all requests associated with the candidate vehicle based on the revised optimal PUDO sequence obtained upon inserting the new request into the candidate vehicle. This is expressed in the following equation:

$$C_{v,r_{(o_r,d_r)}} = w_{wt} * t_{r,v}^{w} + w_{ivtt} * \left(t_{r,v}^{iv} - t(o_r, d_r)\right) + \sum_{\forall r' \in R_v} \left[w_{wt} * \Delta t_{r',v}^{w} + w_{ivtt} * \Delta t_{r',v}^{iv}\right] \tag{6}$$

The first term in the expression represents the PU wait time for the new request r if matched with vehicle v. The second term represents the increase in in-vehicle travel time for the new request r based on match candidate (r, v) compared to the direct travel time between the request origin and destination if the person chose to drive alone instead of choosing a C2C-RP service. The terms in the expression after Σ represent the change in wait times and in-vehicle travel times for all other requests that are in vehicle v's PUDO sequence during the time of matching other than the newly inserted request r.

4.3 PUDO Links Adjustment

As the name suggests, the PUDO links adjustment subroutine, involves selecting PU and DO links for a request-vehicle match or a candidate request-vehicle match, depending on whether the PUDO links adjustment subroutine occurs before ($\gamma_{C2C} = 1 - TRUE$) or after ($\gamma_{C2C} = 0 - FALSE$) the optimal request-vehicle matching subroutine. The purpose of the PUDO links adjustment subroutine is to decrease vehicle detours and detours for in-vehicle travelers. Figure 5 displays an overview of the PUDO links adjustment subroutine for a request-vehicle. The following five subsections describe the components of the subroutine in more detail.

As a small note, the subroutine does not consider very short trips, i.e., trips with a trip origin to destination Euclidean distance less than the total maximum walking range ($2 \cdot D_{walk}^{max}$). Additionally, the PUDO links adjustment procedure sequentially adjusts PU links and then DO links for a request-vehicle pair.

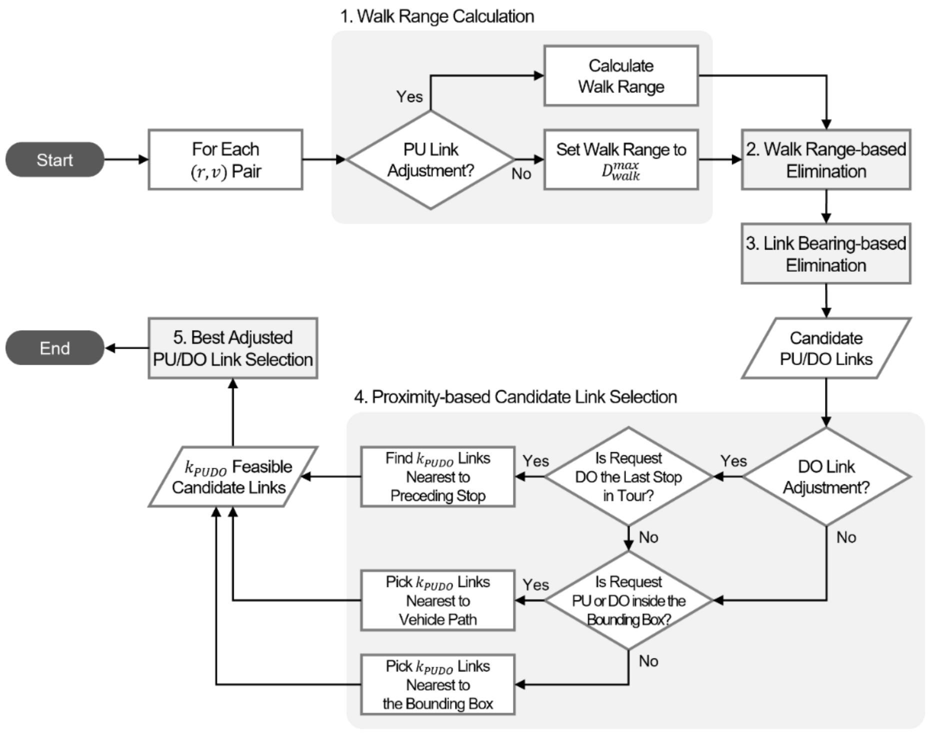

1. Walk Range Calculation
Start
For Each (r, v) Pair
PU Link Adjustment?
Yes
No
Calculate Walk Range
Set Walk Range to D_{walk}^{max}
2. Walk Range-based Elimination
3. Link Bearing-based Elimination
Candidate PU/DO Links
4. Proximity-based Candidate Link Selection
DO Link Adjustment?
Is Request DO the Last Stop in Tour?
Is Request PU or DO inside the Bounding Box?
Find k_{PUDO} Links Nearest to Preceding Stop
Pick k_{PUDO} Links Nearest to Vehicle Path
Pick k_{PUDO} Links Nearest to the Bounding Box
k_{PUDO} Feasible Candidate Links
5. Best Adjusted PU/DO Link Selection
End

Figure 5. Overview of PUDO links adjustment procedure (Repeated sequentially for PU link adjustment and DO link adjustment)

4.3.1 Walk Range Calculation

The walk range calculation subroutine determines the maximum distance a traveler can walk to PU links from their trip origin (D_{walk}^{PU}) and from DO links to their destinations (D_{walk}^{DO}), respectively, for a given request-vehicle pair. While we have a hard upper-bound for the maximum walk distance to PU links and from DO links (D_{walk}^{max}) to prevent travelers from having to walk long distances, there is another important consideration when considering PU links for a request-vehicle pair. This consideration is how far away the vehicle is from the request origin location at the current simulation time for a request-vehicle pair. If the vehicle is very close to the request origin, then ‘allowing’ the request to walk 300-500 meters to a PU location would almost certainly require the vehicle to wait for the request at the PU location for several minutes. Having vehicles, particularly non-idle vehicles, wait at PU locations for travelers to arrive via walking can significantly reduce the efficiency (and therefore productivity) of the vehicle fleet. Although not captured in the decision model or simulation environment, having vehicles wait at PU locations can also negatively impact traffic flow on the PU link.

The walk range for DO links is always D_{walk}^{max} because a vehicle will never need to wait for a request after dropping them off.

$$D_{walk}^{DO} = D_{walk}^{max} \tag{7}$$

The estimated PU walk range for a new request D_{walk}^{PU} depends on the (i) initial distance between the request origin link and the vehicle’s current link, (ii) the average walk speed, and (iii) the average vehicle speed during the matching time step. The initial distance between the request origin and the vehicle’s current position is approximated using Euclidean distance. Average walk speed (s_w) is an input parameter. The average vehicle speed (s_v) is approximated as the average of the zonal speeds at the request origin and the current vehicle link at the current simulation time.

Figure 6a illustrates the approach used to estimate the PU walk range for the new request. The Euclidean path assumption between the request origin and the vehicle’s current link at the current decision epoch is made irrespective of the PUDO sequence of the vehicle. We make this assumption so as not to over-estimate the initial distance of separation between the request origin and vehicle position, and hence minimize vehicle wait time at the adjusted PU link by not setting a high value for D_{walk}^{PU}. Based on this assumption, the following equation describes the initial Euclidean distance between the vehicle position and the request origin (D):

$$D = s_w t + s_v t$$

where t is the approximate time that the request and vehicle will occupy the same point in Euclidean space if they were to travel directly toward each other. From the above equation, $s_w t$ is the walk range of the request such that the vehicle does not have to wait at the adjusted PU link for the request to arrive. Rearranging the above equation to solve for t yields the following:

$$t = \frac{D}{s_v + s_w} \tag{8}$$

Therefore, the PU walk range for the new request is obtained as follows:

$$D_{walk}^{PU} = s_w t = \frac{D \cdot s_w}{s_v + s_w} \tag{9}$$

PU walk range is capped at a maximum value of D_{walk}^{max} set as an input parameter. Therefore, the effective maximum PU walk range for a request in a request-vehicle pair:

$$D_{walk}^{PU} = min\left(D_{walk}^{PU}, D_{walk}^{max}\right) \tag{10}$$

4.3.1 Walk Range-based PU/DO Link Elimination

The walk range-based PU/DO link elimination subroutine is the first among several PU/DO link elimination subroutines. The initial set of PU and DO link candidates for each (potential) request-vehicle match is the set of all walkable links in the network. The link elimination subroutines remove links from this set for each request-vehicle pair.

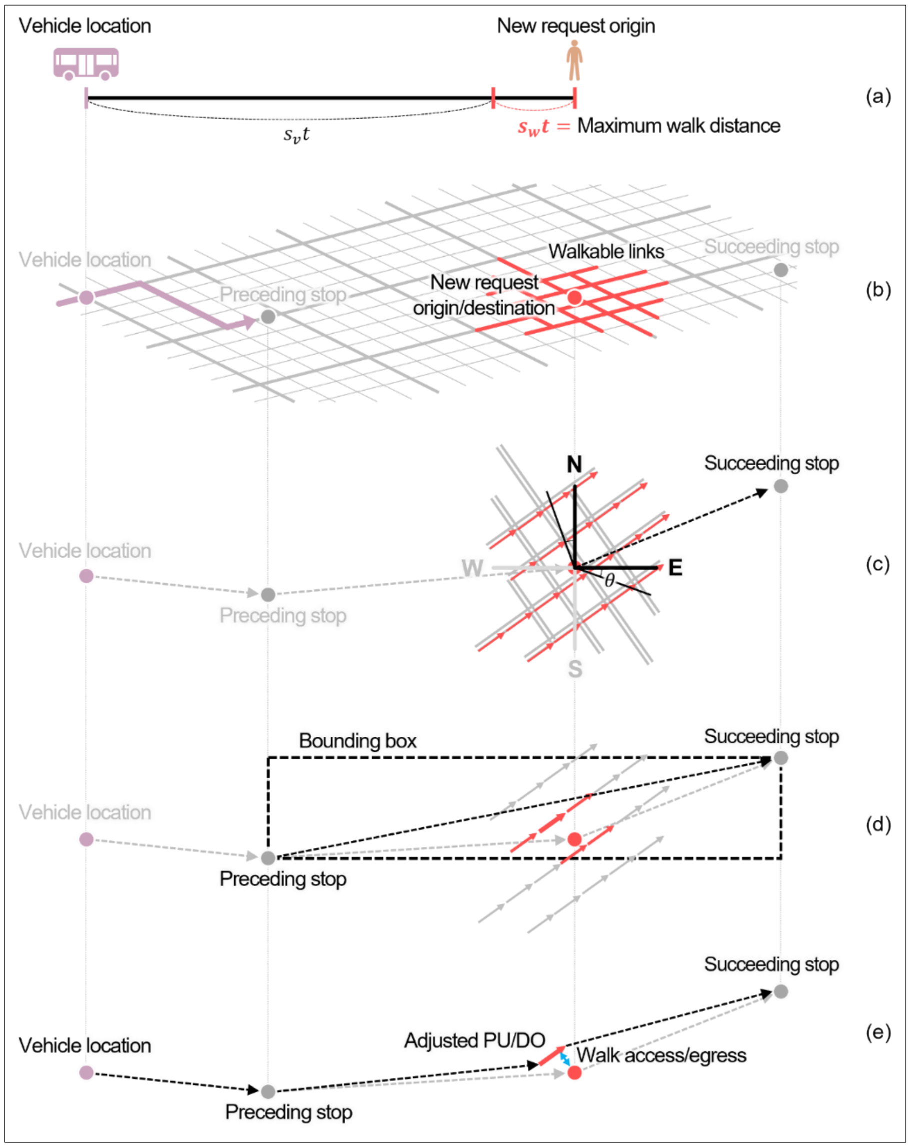


Figure 6. PU/DO Links Adjustment: (a) determining the maximum walk range for PUs, (b) eliminating links considering walk range, (c) eliminating links based on their bearing and the vehicle's current planned path, (d) determining k_{PUDO} candidate links nearest to the vehicle path when the request origin/destination is within the bounding box, and (e) choosing a pair of PU and DO links that minimize cost

Figure 6b displays the subroutine. The process applies to both PU locations and DO locations, so we will only describe the process for PU locations. Prior to the simulation (i.e., offline), we create one-Origin to many-destination Dijkstra trees for every walk link in the network. The Dijkstra trees only extend from each link to the links within D_{walk}^{max}. Given the PU walk range (D_{walk}^{PU}) for a request-vehicle pair determined in the walk range calculation subroutine, the walk range-based PU/DO link elimination subroutine eliminates all links from the request origin link's Dijkstra walk tree that are not within D_{walk}^{PU}.

4.3.2 *Link Bearing-based PU/DO Link Elimination*

The link bearing-based PU/DO link elimination subroutine further eliminates PU and DO link candidates for a request-vehicle pair. Figure 6c displays this link elimination subroutine. The overall solution algorithm does not call this link elimination subroutine for the new request's DO location if the request's destination would be the vehicle's last planned stop at the current iteration of the current decision epoch.

The link bearing-based PU/DO link elimination subroutine considers the future direction of travel of the vehicle after picking up or dropping off a new request. The future travel direction of the vehicle is obtained by measuring the bearing of the vector connecting the new request's link (origin o_r for PU link adjustment and destination d_r for DO link adjustment) and the matched vehicle's preceding stop. The subroutine classifies the vector into one of 4 directional quadrants based on its relative direction with respect to the North-South and East-West direction axes. Based on this classification, the subroutine eliminates links that do not have a bearing within the same directional quadrant as the future vehicle path, plus a bearing buffer of θ_{buf} in the upper and lower bounds of the quadrant.

4.3.3 Proximity-based Candidate Link Selection

The proximity-based candidate link selection subroutine involves selecting up to k_{PUDO} feasible candidate PU links for a request's origin and up to k_{PUDO} feasible candidate DO links for a request's destination. For each origin and destination, the available k_{PUDO} are those links remaining after walk range-based and link bearing-based link eliminations.

As stated earlier, the PUDO links adjustment algorithmic step adjusts PU and DO links sequentially for an (r, v) match or candidate match, in which it adjusts PU links before DO links. The procedure to choose k_{PUDO} candidate links for PU and DO link adjustment for a request-vehicle (candidate) match is described as follows:

1. Construct an R-Tree of all feasible candidate PU/DO links (PU links for PU link adjustment, DO links for Dropoff link adjustment) that have fulfilled the walk distance range and link bearing constraints as described in Section 4.3.1 and 4.3.2.
2. Perform a find nearest links search query on this R-Tree to find k_{PUDO} candidate links based on the following criteria:
    a. For DO link adjustment where the new request's destination is the last stop in the vehicle's tour, query the k_{PUDO} nearest feasible candidate links from the R-Tree closest to the stop preceding the new request's DO link. This is illustrated in Figure 7b.
    b. For all other cases, construct a minimum bounding box rectangle using the maximum and minimum coordinates of the stops preceding and succeeding the new request's link (origin link for PU link adjustment, destination link for DO link adjustment) in the vehicle's tour. Also construct a polyline object denoting the vehicle's Euclidean path connecting the stops preceding and succeeding the new request's link. The k_{PUDO} nearest link search query is performed based on the location of the new request link with respect to the bounding box:
        i. If the new request link is within the bounding box, then query the k_{PUDO} nearest candidate links from the R-Tree closest to the vehicle path. This is illustrated in Figure 6d.
        ii. If the new request link is outside the bounding box, then query the k_{PUDO} nearest candidate links from the R-Tree closest to the bounding box. This is illustrated in Figure 8.

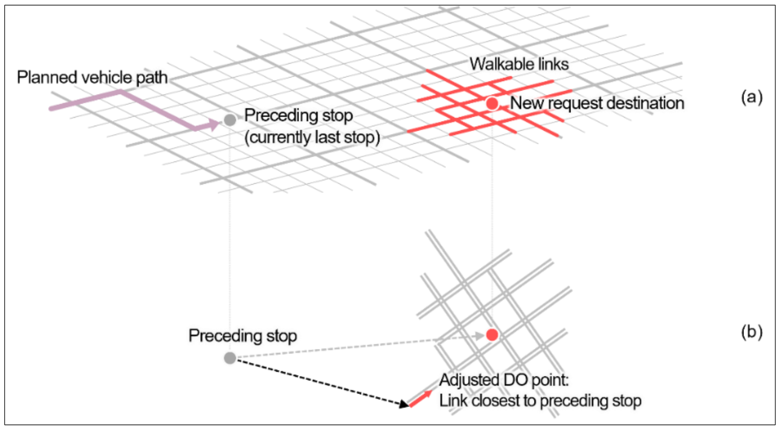

Planned vehicle path
Walkable links
New request destination
Preceding stop (currently last stop)
(a)
Preceding stop
(b)
Adjusted DO point: Link closest to preceding stop

Figure 7. DO Links Adjustment in Case of Last DO in vehicle tour: (a) Finding Walkable Links and (b) Finding the Closest Link from Preceding Stop

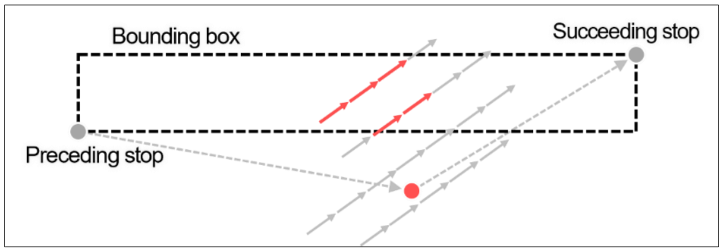


Figure 8. PU/DO Links Adjustment in the Outside Bounding Box Case

4.3.4 Best adjusted PU/DO Link Selection

This step involves choosing the best link for PU/DO link adjustment by evaluating from the k_{PUDO} feasible candidate links returned from the R-Tree query in the previous step. The PU/DO link is adjusted if it results in a reduction in total vehicle travel time to complete the sequence of PUDOs in its tour (including serving the new request) as known during the matching time step. PU/DO link adjustment is not performed if none of the candidate PU or DO links returned from the previous step results in a travel time reduction for the vehicle. The procedure to select the best candidate PU/DO link is described below in Algorithm 4. Sections 4.3.1 to 4.3.4 are performed first for PU link adjustment for a (r, v) pair and repeated for DO link adjustment.

Algorithm 4 – Optimal Adjusted PU/Dropoff Link for New (r, v) pair

```
Input: Unassigned request $r$, Vehicle $v$, PU adjustment or DO adjustment (Boolean)
Output: Adjusted PU/DO Location, PUDO Sequence, Insertion Cost and Travel time after PU/DO adjustment
Procedure:
Let $S^*_{r,v}$ be the Optimal PUDO Sequence for vehicle $v$ after inserting request $r$ (from Algorithm 3)
Let $t_{S^*_{r,v}}$ be the minimum total travel time for the vehicle $v$ to complete the optimal PUDO sequence $S^*_{r,v}$
Let $C_{v,r_{(o_r,d_r)}}$ be the insertion cost of adding request $r$ to trip sequence of vehicle $v$ before PUDO links adjustment
Let $C^*_{v,r_{(o'_r,d_r)}}$ be the insertion cost of adding request $r$ to trip sequence of vehicle $v$ after PU Link adjustment
Let $C^*_{v,r_{(o'_r,d'_r)}}$ be the insertion cost of adding request $r$ to trip sequence of vehicle $v$ after DO Link adjustment
Let $L_{PU/DO}$ be the set of optimal PU/DO links returned from Section 4.3.3
Let $l^*$ be the optimal adjusted PU/DO link
Initialize values based on whether PU or DO link is being adjusted
if PU Link Adjustment then
    $S^*_{r,v}$ = $S^*_{r,v}$ returned by Algorithm 3 before PU Link adjustment
    $t_{S^*_{r,v}}$ = $t_{S^*_{r,v}}$ returned by Section 4.3 before PU Link adjustment
    $L_{PU/DO}$ = $L_{PU}$ feasible alternative PU links returned by Section 4.3.3
    $l^* = o_r$
    $C^*_{v,r_{(o'_r,d_r)}} = C_{v,r_{(o_r,d_r)}}$ returned by Section 4.2.4 before PU Link adjustment
end if
if Dropoff Link Adjustment then
    $S^*_{r,v}$ = $S^*_{r,v}$ returned by Algorithm 4 after PU Link Adjustment
    $t_{S^*_{r,v}}$ = $t_{S^*_{r,v}}$ returned by Algorithm 4 after PU Link Adjustment
    $L$ = $L_{DO}$ feasible alternative Dropoff links returned by Section 4.3.3
    $l^* = d_r$
    $C^*_{v,r_{(o'_r,d'_r)}} = C_{v,r_{(o'_r,d_r)}}$ returned by Algorithm 4 after PU Link adjustment
end if
for each $l \in L$ do
    if PU Link Adjustment then
        $o'_r = l$
    end if
    if Dropoff Link Adjustment then
        $d'_r = l$
    end if
    Find updated optimal PUDO sequence $S'_{r,v}$ and updated vehicle travel time $t_{S'_{r,v}}$ using Algorithm 3
    Find updated request travel times $t^{w}_{r',v}$ and $t^{iv}_{r',v}$ $\forall r' \in v_r \cup \{r\}$ from Section 4.2.2 for the updated PUDO sequence $S'_{r,v}$
    Check for violation of time window constraints $\forall r' \in v_r \cup \{r\}$ based on the new optimal PUDO sequence $S'_{r,v}$ after adjusting PU link (Section 4.2.3)
    Choose the PU/DO link that results in the least vehicle travel time without violating time window constraints
    if no time window constraints violated and $t_{S'_{r,v}}$ < $t_{S^*_{r,v}}$ then
        $t_{S^*_{r,v}}$ = $t_{S'_{r,v}}$
        $l^* = l$
        Update insertion cost for the $(r, v)$ pair after adjusting PU or DO link based on Section 4.2.4
        if PU Link Adjustment then
            $C^*_{v,r_{(o'_r,d_r)}} = C'_{v,r_{(o'_r,d_r)}}$
        end if
        if Dropoff Link Adjustment then
            $C^*_{v,r_{(o'_r,d'_r)}} = C'_{v,r_{(o'_r,d'_r)}}$
        end if
    end if
end for
Finalize adjusted PU/DO link, Insertion Cost,  PUDO sequence corresponding to $t_{S^*_{r,v}}$
if PU Link Adjustment then
    $o'_r = l^*$
    $C_{v,r_{(o'_r,d_r)}} = C^*_{v,r_{(o'_r,d_r)}}$
    return $o'_r$, $C_{v,r_{(o'_r,d_r)}}$, $S^*_{r,v}$, $t_{S^*_{r,v}}$
end if
if Dropoff Link Adjustment then
    $d'_r = l^*$
    $C_{v,r_{(o'_r,d'_r)}} = C^*_{v,r_{(o'_r,d'_r)}}$
    return $d'_r$, $C_{v,r_{(o'_r,d'_r)}}$, $S^*_{r,v}$ , $t_{S^*_{r,v}}$
end if
```

4.4 Optimal Request-Vehicle Matching

The optimal matching of requests to vehicles is performed either before or after PUDO links adjustment based on the γ_{C2C} parameter. If PUDO links adjustment is performed before matching, then the cost value associated with each candidate (r, v) match is the insertion cost $C_{v,r_{(o'_r,d'_r)}}$ returned by Algorithm 4 after adjusting the PU and DO links for r if assigned to v. Thus, the PUDO links adjustment is also a deciding factor in the optimal (r, v) matching process, since the objective function reflects the change in insertion cost for each (r, v) candidate upon adjusting the PUDO links. If PUDO links adjustment is performed after matching, then the cost value for each (r, v) match candidate is the insertion cost before PUDO links adjustment calculated in Section 4.2.4. In this case $C_{v,r_{(o'_r,d'_r)}} = C_{v,r_{(o_r,d_r)}}$. The objective of each iteration of the optimal request-vehicle matching procedure is to match unassigned requests to available vehicles such that the total insertion cost is minimized:

$$Z = min \sum_{\forall r \in R^u} \sum_{\forall v \in V_r} \left(C_{v,r_{(o'_r,d'_r)}} - P \right) \cdot X_{v,r} \qquad (11)$$

where R^u is the set of all unassigned requests at the beginning of each iteration of the optimal matching stage, and V_r is the set of all feasible vehicle candidates for request r, where $V_r \subset V$. $C_{v,r_{(o'_r,d'_r)}}$ is the insertion cost of assigning request r to vehicle v with the following PUDO links (o'_r, d'_r). P is a penalty cost incurred for not making an assignment; P is a large positive number. $X_{v,r}$ is the binary decision variable denoting whether request r is matched with vehicle v.

The matching problem is subject to two sets of constraints. The set of constraints in Eqn. 12 suggest that each request $r \in R^u$ can be assigned to at most one vehicle. The set of constraint in Eqn. 13 limit each vehicle to be assigned to at most one unassigned request.

$$\sum_{v \in V} X_{v,r} \leq 1 \qquad \forall r \in R^u \qquad (12)$$

$$\sum_{r \in R^u} X_{v,r} \leq 1 \qquad \forall v \in V \qquad (13)$$

The C2C-RP problem is solved iteratively (Sections 4.2 to 4.4) at each decision epoch until there are no unassigned requests left ($R^u = \emptyset$) or no feasible candidate vehicles for each unassigned request ($V_r = \emptyset, \quad \forall r \in R^u$).

5 Computational Experiment Set-up

To evaluate the performance of the proposed C2C-RP decision policy and algorithmic approach, we constructed a large number of computational experiments. This section describes the simulation environment wherein we embed the proposed C2C-RP decision policy and algorithmic approach (Section 5.1), the road network model (Section 5.2), the performance metrics for analysis (Section 5.3), and the scenarios for testing (Section 5.4). The four MOD service types evaluated are C2C-RP, C2C-RH, D2D-RP, D2D-RH. To operate D2D services we use the same decision policy and solution algorithm as the C2C service, except that PUDO links are not adjusted (i.e., Section 4 excluding section 4.3). Similarly, to operate ride-hailing services, we use the same decision policy and solution algorithm as the ride-pooling service, except that we only consider idle vehicles in the matching subproblem.

5.1 POLARIS Simulation Environment

We use the large-scale agent-based simulation framework called POLARIS (Auld et al., 2016) to compare the four MOD service types and evaluate the C2C-RP decision policy and algorithmic approach. POLARIS integrates supply and demand and allows the simulation of MOD services in a congestible network with full feedback (Gurumurthy et al., 2020). POLARIS and its modules can simulate activities and trips in a large metropolitan region with over 10 million people in under 5 hours.

POLARIS is a high-performance C++ codebase for agent-based modeling of transportation demand and supply. The tool consists of several modules for population synthesis, long-term and short-term planning, vehicle routing, traffic flow, and has functionality to model transit, MOD services, and freight at a high-level of detail. The population synthesis module creates person agents for the target region based on underlying demographic information as sourced from the Census and American Community Survey (ACS) in the United States. Consistent with agent-based modeling, each individual makes travel and activity decisions based on its individual characteristics and information available to the individual regarding the state of the system (e.g., prevailing travel times, modal attributes, destination attributes, etc.).

5.1.1 Corner-to-Corner Routing Module in POLARIS

The MOD module in POLARIS models operations through a centralized operator. The operator maintains control over all fleet vehicles, tracks their real-time information, and runs algorithms to determine the next set of instructions to pass to the vehicles. To enable easy incorporation of new algorithms, a generic structure exists in POLARIS for different aspects of MOD control: such as matching, repositioning, charging (in the case of EVs), and parking. These strategies are custom-coded to either serve a single purpose like a matching algorithm or the modeler can control several aspects of operation simultaneously like the joint control of matching, repositioning, and charging of fleet vehicles (Dean et al., 2022).

We now briefly describe the general flow of information and control in POLARIS for a MOD request to be received and served. The fleet operator is informed of requests that originate from person agents. Requests encompass information on time of request, PU location, DO location, and the estimated fare, travel time, and costs associated with serving it. The matching subroutine receives trip requests from the operator and can either immediately assign an available vehicle or batch the requests over a pre-defined duration and then solve an optimization-based request-vehicle matching problem. Once the matching subroutine is complete, an update regarding the match is passed on to both the person agent and the assigned vehicle (if matched). Assuming the match is successful, the vehicle is instructed to route itself to the PU location and then, on picking the traveler, proceed to the DO location. While en-route, new requests can be added to the existing vehicle trip, with the vehicle reporting available seats remaining.

The C2C service in this paper requires travelers to walk to their PU locations and from their DO locations. Trips in POLARIS are typically modeled to start and end at activity locations. Activity locations represent buildings and places that are typical origins and destinations. Depending on network density incorporated in the model, some level of aggregation may exist in the number of activity locations used to represent the underlying origins and destinations. Route computation considers all possible links associated with the origin and destination activity locations as candidates for the shortest path, and finally results in a link-to-link vehicle trajectory. Once in the network, vehicles follow the trajectory subject to constraints of the traffic flow model. Trips are completed when the vehicle enters the last link of its trajectory. While Gurumurthy and Kockelman (2022) show the benefits of aggregation at the activity location level for C2C MOD service, their approach does limit flexibility in assigning travelers to PUDO locations in the region. Hence, in the current study, we allow traveler PUDOs in every link in the network. This approach both reduces overall VMT and improves traveler experiences compared to the approach in Gurumurthy and

Kockelman (2022), as the later study requires all travelers to walk to PUDO locations, even if these walking trips do not improve vehicle utilization and productivity. Changes in PUDO links are transmitted to the person and vehicle agents at the end of each batching interval once the iterative procedure terminates.

5.2 Study Network

We use the default network in POLARIS – the Bloomington IL network (Figure 9) – to evaluate the C2C-RP algorithm. The network includes 3057 intersections, 4527 directional drive links, 6885 walk links and 185 Traffic Analysis Zones (TAZs). The walk network is broken down into links with a maximum length of 250 meters. There are 2,833 activity locations in the network which represent request origins and destinations. The default PU link and DO link for each new request is the link closest to the request origin and destination activity locations respectively.

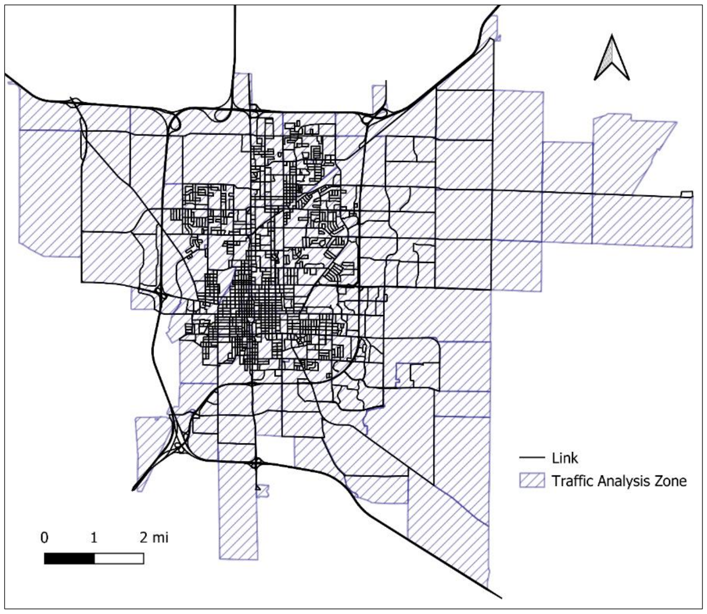


Figure 9. Bloomington, IL Network

5.3 Metrics for Analysis

To compare the four MOD service types, we consider two cost dimensions, operator costs and user costs. We use average vehicle kilometer traveled (VKT) per served request as the main operator cost metric. We normalize by 'served requests' to prevent outcomes where an efficient and productive service generates more VKT while serving a large number of requests than an inefficient and unproductive service that generates lower VKT while serving significantly fewer requests.

The main customer cost metric is request-to-destination time, which as the name suggests, measures the total time between a traveler requesting a vehicle and the same traveler arriving at their destination location. In C2C services, request-to-destination time includes time waiting to be assigned, access walking time, waiting time at the PU link, in-vehicle time, and egress walking time. The results will explicitly delineate the time travelers spend in each state.

We also include matching rate, the percentage of all MOD traveler requests who are served by the MOD service, as a key performance metric. This metric is necessary for a holistic comparison of MOD service types; without matching rate, a service type or decision policy that prioritizes *easy-to-serve* requests may perform well in terms of VKT per served request and request-to-destination time for served request, compared with other policies that attempt to serve all requests.

The final metric we use is average vehicle occupancy (AVO) per VKT. AVO per VKT includes empty vehicle kilometers as well as vehicle kilometers with one passenger, two passengers, three passengers, and four passengers. This metric does not perfectly align with either operator costs or user costs, but it is a metric of interest to transportation planners and policymakers.

5.4 Scenarios

This section describes the scenarios developed to help answer the study's two main research questions. We first introduce the baseline scenarios, specifically which parameters we vary in the baseline scenarios and which parameters are fixed. Next, we describe the additional scenarios developed to provide insights into various service design decisions, the inefficiencies associated with travel time uncertainty, and various algorithmic strategies and decision policies.

Table 2 shows that in the set of baseline scenarios, the only two parameters that vary are the MOD service type and the fleet size. Maximum walk range, walking speed, and the sequencing of matching and PUDO links selection are fixed. These baseline scenarios aim to compare and contrast the four MOD service types under various supply-demand ratios, where the demand is fixed, and the supply varies across scenarios. The results of the baseline scenarios are described in Section 6.1. The number of available seats for ride-pooling services (C2C-RP and D2D-RP) is fixed at 4.

After the baseline scenarios, we create several additional sets of scenarios. The first additional set of scenarios involves varying the maximum walk range parameter D_{walk}^{max} alongside the MOD service type and fleet size to perform sensitivity analysis on user and operator costs. We vary D_{walk}^{max} between 250 and 1000 meters in an increment of 250 meters. The results are presented in Section 6.2.

The second additional set of scenarios varies 'walking' speed s_w alongside MOD service type and fleet size. In these scenarios, there we use a baseline s_w of 5 km per hour speed, and the 'fast walking' (or e-scooter) speed of 20 km per hour. Results of this analysis are provided in Section 6.3.

The third additional set of scenarios varies the sequence of Request-Vehicle matching and PUDO links selection sub problems, alongside MOD service type and fleet size. This is done by enabling/disabling the Boolean flag γ_{C2C}. The two alternative orders are adjust PUDO links after matching and adjust PUDO links before matching. Section 6.4 describes the results of this analysis.

Finally, we also run a set of scenarios to analyze the computation time and scalability of the implemented solution methodology for the C2C-RP problem (Section 6.5). This is done by analyzing the total computational time and stage-wise computational time by varying D_{walk}^{max}, s_w and γ_{C2C} parameters across different MOD service types for 2 different fleet sizes (5,000 and 10,000 vehicles).

To ensure consistency while comparing across several scenarios, the total demand, origin, destination as well as the request initiation times are kept fixed across all scenarios. Repeating the same scenario with all parameters unchanged yielded a less than 0.1% change in values of the metrics evaluated. The spatial distribution of the initial location of vehicles at the beginning of simulation is also same across scenarios. Fleet repositioning is also disabled so as to control for idle vehicle movements from biasing the output metrics evaluated across the four different MOD types.

Table 2: Varying and Fixed Parameters in Baseline Scenarios (Section 6.1).

Fixed or Varying?	Parameter	Parameter Values
Varying Params	MOD Service Type	D2D-RH ($c_{veh} = 1$)
		D2D-RP ($c_{veh} = 4$)
		C2C-RH ($c_{veh} = 1$)
		C2C-RP ($c_{veh} = 4$)
	Fleet Size (veh.)	1000, 1500, 2000, 3000, 4000, 5000, 6000, 7500, 9000, 10000
Fixed Params for PUDO Links Adjustment (C2C-RP and C2C-RH)	D_{walk}^{max}	1,000 m
	γ_{C2C}	FALSE (Adjust PUDO link after matching)
	s_w	5 km/h
	k_{PUDO}	5 link candidates
	θ_{buf}	30°
Fixed Params for R-V Matching (All Four MOD Services)	k_{veh}	8 vehicles
	k_{veh}^{idle}	2 vehicles
	θ_{max}	30°
	D_{dir}^{max}	3000 m
	D_{rev}^{max}	3000 m
	D_{detour}^{max}	6000 m
	t_{max}^{w}	1200 seconds
	$t_{max_{abs}}^{iv}$	900 seconds
	$t_{max_{rel}}^{iv}$	50%
	w_{wt}	1.0
	w_{ivtt}	1.0
Simulation Parameters	Analysis Period	24 hours
	Δ	30 seconds
	# Requests	221,711 (Fixed) 1 request = 1 traveler
	Initial Fleet Location	Distributed inversely proportional to TAZ area
	Fleet Repositioning	FALSE

6 Results and Discussion

6.1 Operator-User Cost Trade-off in MOD Systems – Baseline Scenario

This section aims to answer this study's main research question: what are the trade-offs between operational costs and user costs across D2D-RH, D2D-RP, C2C-RH, and C2C-RP? Figure 10b shows significant operator cost differences between the four MOD services, particularly when the supply (i.e., vehicle fleet size) is low. With a fleet size of 2000 vehicles, C2C-RP has an average VKT per request of nearly 5.5, which is lower than the D2D-RP at 6.0, and substantially lower than C2C-RH at 7.0 and D2D-RH at over 7.6. The operator cost benefits of pooled-rides at low fleet sizes are even more impressive when considering the significantly higher matching rates for ride-pooling than ride-hailing. Hence, there are clear operational cost benefits associated with pooled rides, as found in many other studies (Hyland and Mahmassani, 2020).

Figure 10b also shows there is undoubtedly an operational benefit associated with C2C service over D2D service. Interestingly, this gap in VKT per request seems to remain steady, or even increase, as the fleet size increases. This is an important finding, particularly the quantification of the VKT per request gap between C2C and D2D services. While the gap is not as large as the ride-pool vs. ride-hail gap for matching rate in Figure 10a, at low to medium fleet sizes, C2C service does have a significantly higher matching rate than D2D, especially at lower values of fleet size.

Figure 10c shows the clear downsides of ride-pooling and C2C service, relative to ride-hailing and D2D service, respectively in terms of total user travel time. Given the low matching rates for both ride-hail services at fleet sizes below 3000, we will compare the services in terms of request-to-destination time for fleet sizes 3000 and larger. At a fleet size of 3000 vehicles, D2D-RH service has an average request-to-destination time of slightly over 8 minutes, whereas the C2C-RH service is around 13 minutes. The gap between these two services is nearly all due to egress walk distance/time. There is also an approximately 4-minute gap for request-to-destination time between C2C-RP and D2D-RP.

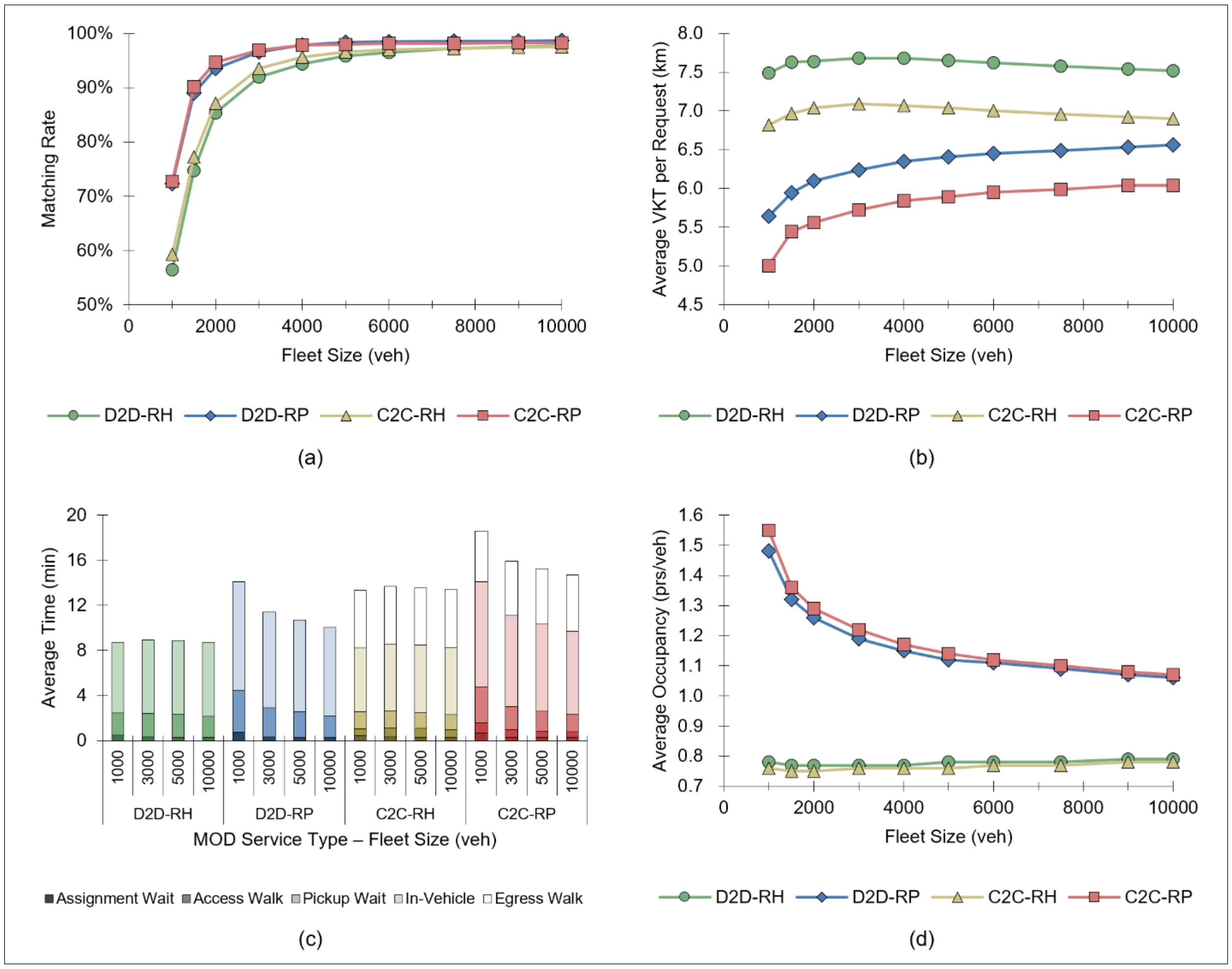


Figure 10. Performance metrics for MOD service types: (a) Matching Rate, (b) Average VKT per Request, (c) Average Travel Time broken down by Segment, and (d) Average Vehicle Occupancy

Figure 10d is consistent with Figure 10a and Figure 10b and the features of each service design—the AVO per request is significantly higher for ride-pool compared to ride-hail. There is also a gap between C2C and D2D, especially at low fleet sizes, but it is noticeably smaller than the gap between ride-pool and ride-hail.

Additionally, Figure 10d shows that AVO decreases steadily as fleet size increases. This is simply the result of the decision policy using all the vehicles in the fleet to reduce traveler wait times as shown in

Figure 10c. Interestingly, AVO per request of C2C-RP is slightly greater than D2D-RP for low fleet sizes, whereas, AVO per request of C2C-RH is slightly lower than D2D-RH (or nearly the same) for all fleet sizes. This could potentially be because adjusting PUDO links for a ride-hailing service does not increase sharing of trips in the system since a ride-hail vehicle serves only one request at a time.

In summary, regarding the original research question, there is a clear trade-off between operator cost and user costs in these four MOD service designs. The gap between ride-hail and ride-pool is the most significant in terms of operator costs, but there is also a significant gap between C2C and D2D (5 to 10% reduction in VKT per served request for C2C services compared to D2D). In terms of user costs, there is not a huge gap between ride-hail and ride-pool, but there is a significant gap between C2C and D2D. However, the gap between C2C and D2D is almost entirely due to the egress walking distance/time.

The implications of the gap between C2C and D2D being from egress walking distance are several. First, from a behavioral perspective, some users may find this egress walking to be both highly inconvenient and onerous, while other users may find a 3-or 4-minute walk to be only slightly inconvenient and even pleasant, depending on their schedule, the weather, and various other factors. The heterogeneity of travelers in terms of the willingness to walk suggests there is likely a role for multiple service offerings from the same MOD service provider.

Second, from a decision policy/algorithm and service design perspective, it is possible to put more weight on the disutility of walking in the algorithm, in order to decrease average egress walking distances. However, this algorithmic change would likely increase the operator costs for a C2C service. Similar to applying more weight to the disutility of walking in the decision policy function, it is also possible to explicitly limit the maximum total and/or egress walking distance for travelers. The next section analyzes variations in this parameter in terms of both operator and user costs.

Figure 11 illustrates the effect on user and operator costs if MoD operators offer pooled C2C services with only PU Link adjustments (C2C-RP-PU) or only DO Link adjustments (C2C-RP-DO) compared to C2C-RP and D2D-RP for the baseline scenario parameters as listed in Table 2. Figure 11a shows that VKT per served request for C2C-RP-PU is slightly less than D2D-RP while it is slightly more for C2C-RP-DO compared to C2C-RP. This is also reflected in the segment wise travel times shown in Figure 11, with total travel time of C2C-RP-PU being slightly more than D2D-RP, while total travel time of C2C-RP-DU being slightly less than C2C-RP. This trend can be explained when comparing the % of requests with PU Links adjusted with those with DO links adjusted (Figure 11c and Figure 11d) as well as the average access and egress walk distances in their respective cases (Figure 11e and Figure 11f). Even though the maximum walk range is 1,000 m for both C2C-RP-PU and C2C-RP-DO, the average access walk distance for adjusted PU links in C2C-RP-PU is less than half of the egress walk distance for adjusted DO links in C2C-RP-DO. This is because, the algorithm proposed in this paper calculates the effective maximum PU walk range (D_{walk}^{PU}) for each request-vehicle pair in such a way that instances of vehicles having to wait for the request at the adjusted PU link is reduced (Section 4.3.1). Whereas this does not apply for DO links adjustment. This also explains the higher instances of requests with DO links adjustment (Figure 11d) compared to those with PU links adjustment (Figure 11c).

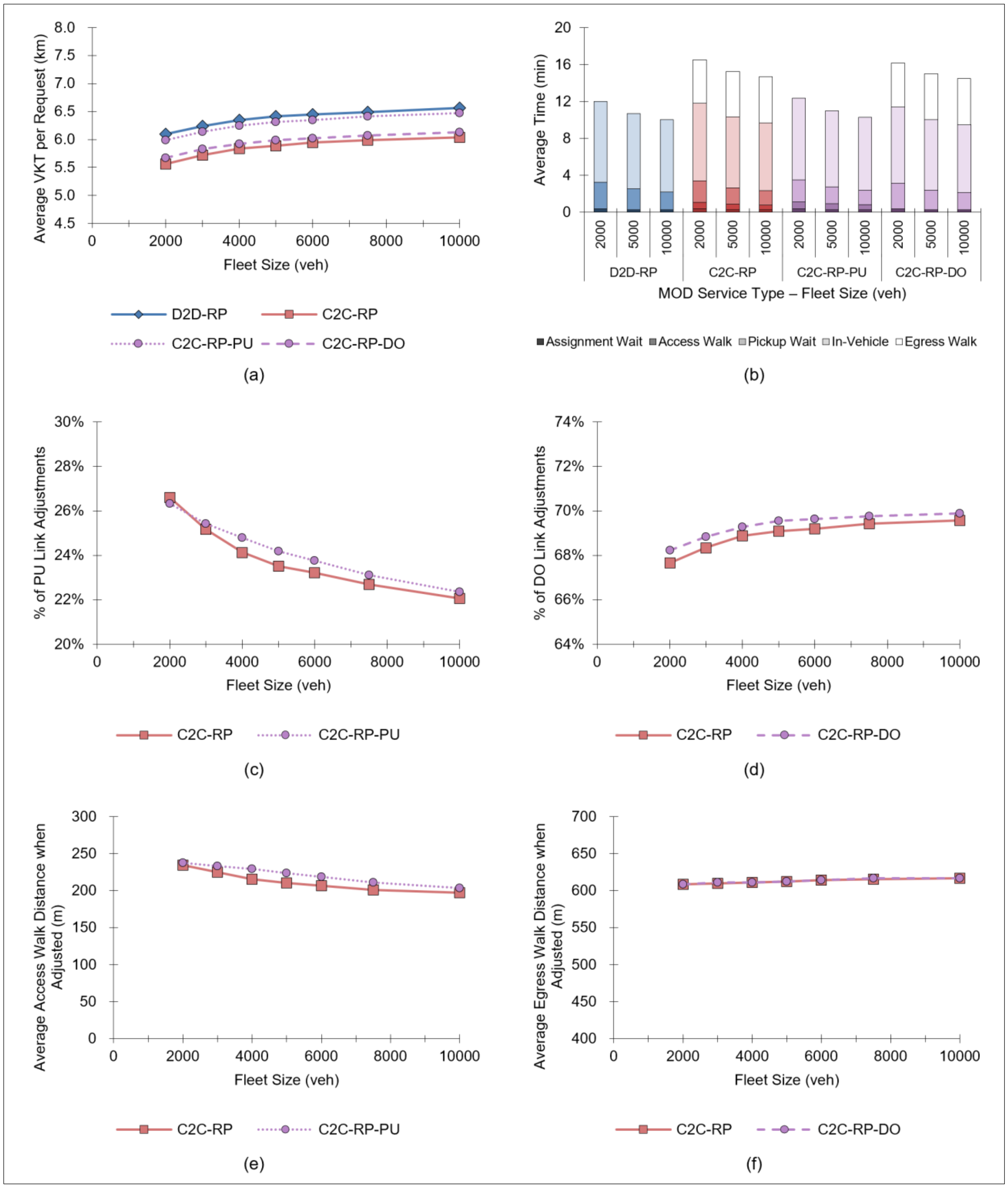


Figure 11. Performance metrics with selective 'PU Only' or 'DO Only' link adjustments (a) Average VKT per Request, (b) Average Travel Time broken down by Segment, (c) % of Requests with PU Links Adjusted, (d) % of Requests with DO Links Adjusted, (e) Average Access Walk Distance when PU Link is Adjusted, and (f) Average Egress Walk Distance when DO link is Adjusted

The percentage of instances as well as average access walk distance for adjusted PU links in C2C-RP-PU scenario is slightly more than C2C-RP. This could be because, since DO links are not adjusted in C2C-

RP-PU, it offers slightly more flexibility for the operator to adjust PU links without violating time window constraints. The prevalence of higher instances of DO links adjustment in C2C-RP-DO compared to C2C-RP can also be explained similarly. Additionally, the access walk range and percentage of requests with PU links adjustment decreases with increasing fleet size for both C2C-RP-PU and C2C-RP. This is because an unassigned request is closer to an available vehicle as fleet size increases. On the other hand, % of requests with DU links adjustment increases with increasing fleet size (Figure 11d) as more vehicles means less waiting time giving more flexibility to adjust DO links. It is also interesting to note that the average egress walk distance for requests with DO links adjustment remains more or less flat across fleet sizes. This could be because unlike calculating the effective maximum PU walk range, DO walk range calculation is not affected by proximity of requests to a vehicle.

6.2 Sensitivity Analysis with respect to Walk Range

Figure 12 displays the computational results for variations in maximum walking distance D_{walk}^{max} (i.e., maximum walk range for PU as well as DO), across three fleet sizes and both C2C MOD services. The figure also shows the results for the two D2D MOD services, but these services always have zero walk range. Walk speed (s_w) is kept fixed at 5 kmph.

The matching rate results in Figure 12a indicate that for these three fleet sizes, walking range does not have significant practical impact on matching rate, although longer walking ranges do slightly increase matching rate.

Figure 12b and Figure 12c show that as walking range decreases from 1000m to 250m, VKT per request increases and request-to-destination time decreases for both C2C-RH and C2C-RP. Interestingly, the relationship is strongly linear for both C2C services, both performance metrics, and all three fleet sizes. However, the results indicate that there is a step change improvement in VKT per request when moving from D2D to *some walking*, i.e., a 250m walking range. This latter result suggests that even allowing a small walking distance can produce significant operational efficiencies in the system. This should not be surprising, but it is important, as having travelers walk even a short distance can prevent the worst-case PUDO links for vehicles. A small amount of walking can also ensure the PUDO links are on links with the same bearing as the vehicle's direction of travel after picking up or dropping off a request.

Figure 12d shows that walking range does not have a significant impact on AVO for ride-pool or ride-hail nor for any fleet size. Even though Figure 12d showed that the AVO for C2C-RP with a baseline walk range of 1000 meters is slightly more than D2D-RP for low fleet sizes, Figure 12d denotes that change in AVO for C2C-RP services for every 250 meter increment in maximum walk range is very minimal.

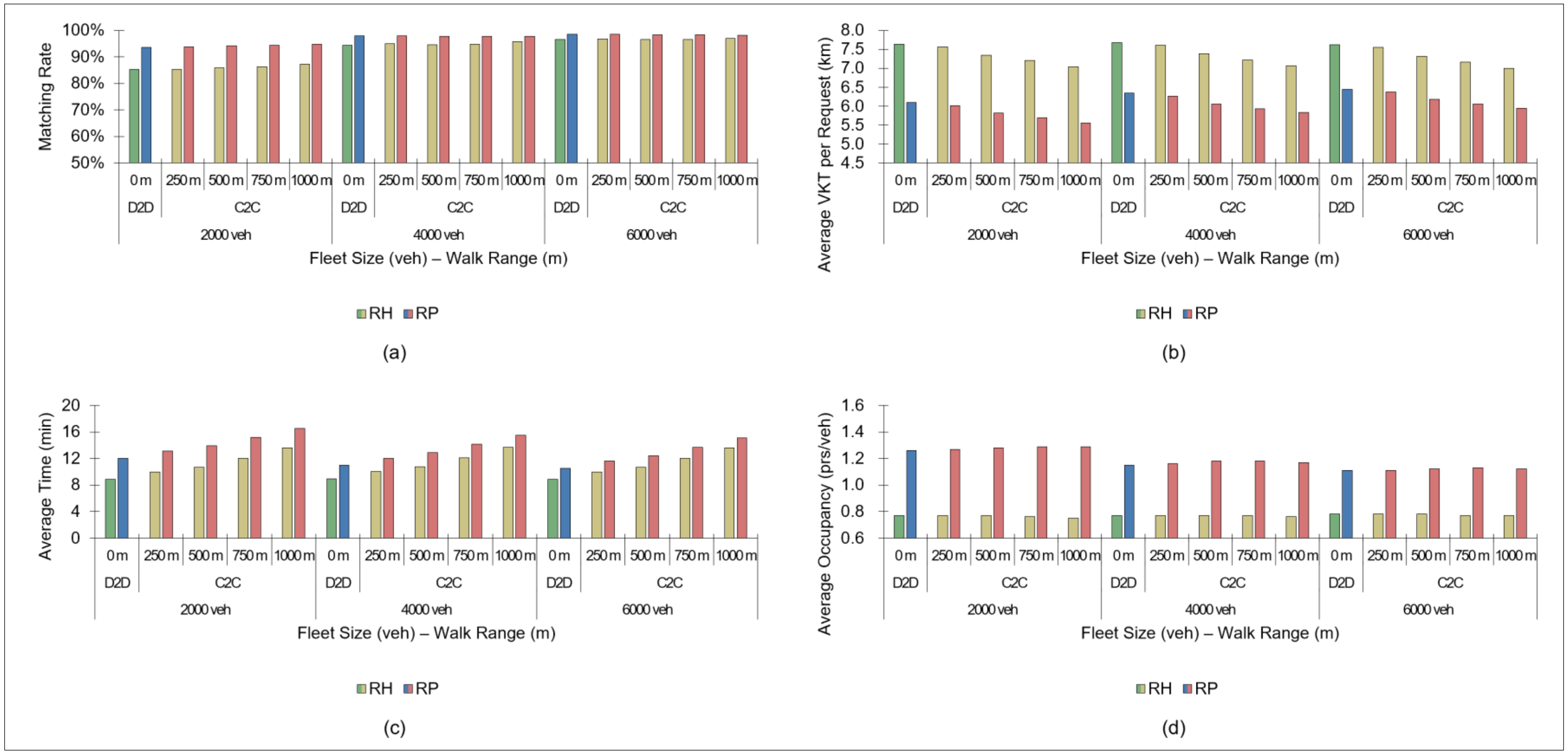


Figure 12. Sensitivity Analysis with respect to Walk Range: (a) Matching Rate, (b) Average VKT per Request, (c) Average Request-to-destination Time, and (d) Average Vehicle Occupancy

6.3 What about e-scooters? Sensitivity Analysis with respect to Walk Speed

This section analyzes the change in fleet performance and user cost with respect to changes in walk speed (s_w). We compare the baseline walking speed of 5km/h with a much faster 'walking' speed of 20km/h. This analysis serves two purposes, namely, to assess the potential benefits of using ubiquitous—personal or shared—e-scooters as access and egress modes to/from adjusted PUDO links, and to illustrate the role of travel time uncertainty across service performance metrics.

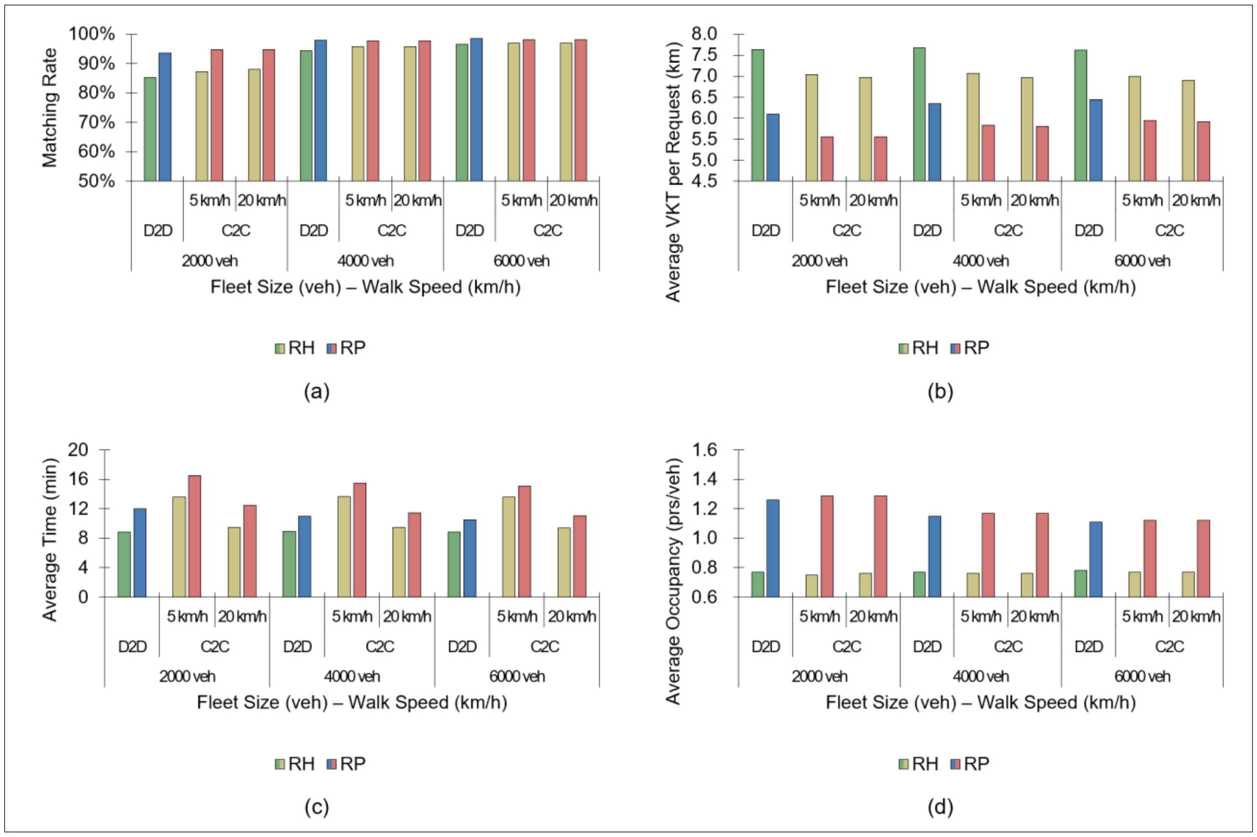


Figure 13. Sensitivity Analysis with respect to Walk Speed: (a) Matching Rate, (b) Average VKT per Request, (c) Average Request-to-destination Time, and (d) Average Vehicle Occupancy

Figure 13 shows that increasing walking speed increases matching rate slightly, decreases average VKT per request, significantly decreases request-to-destination time, and does not significantly impact AVO, compared to results presented for baseline walk speed in Figure 10. Figure 13c shows that the increased walking speed decreases request-to-destination time by around 5 minutes. This five-minute reduction in travel time effectively makes C2C competitive with D2D for both ride-hailing and ride-pooling along this dimension, while retaining a significant advantage in terms of VKT per request.

Figure 14 aims to provide insights on how/why faster walking speeds dramatically improve request-to-destination time, while also improving VKT per request and matching rate. Figure 14a shows the matching rate as a function of walking range, walking speed and fleet size. Naturally, as all three of these input parameters increase, particularly fleet size, the matching rate increases. Figure 14b shows the frequency (i.e., percentage) of users with adjusted PU links (i.e., a different PU link than their trip origin). As walk range and walk speed increase, the frequency of PU link adjustments increases. The relationship between fleet size and PU link adjustments is not monotonic. Matching rate and frequency of PU link adjustments are relevant background information, for the two main sets of results in Figure 13c and Figure 13d.

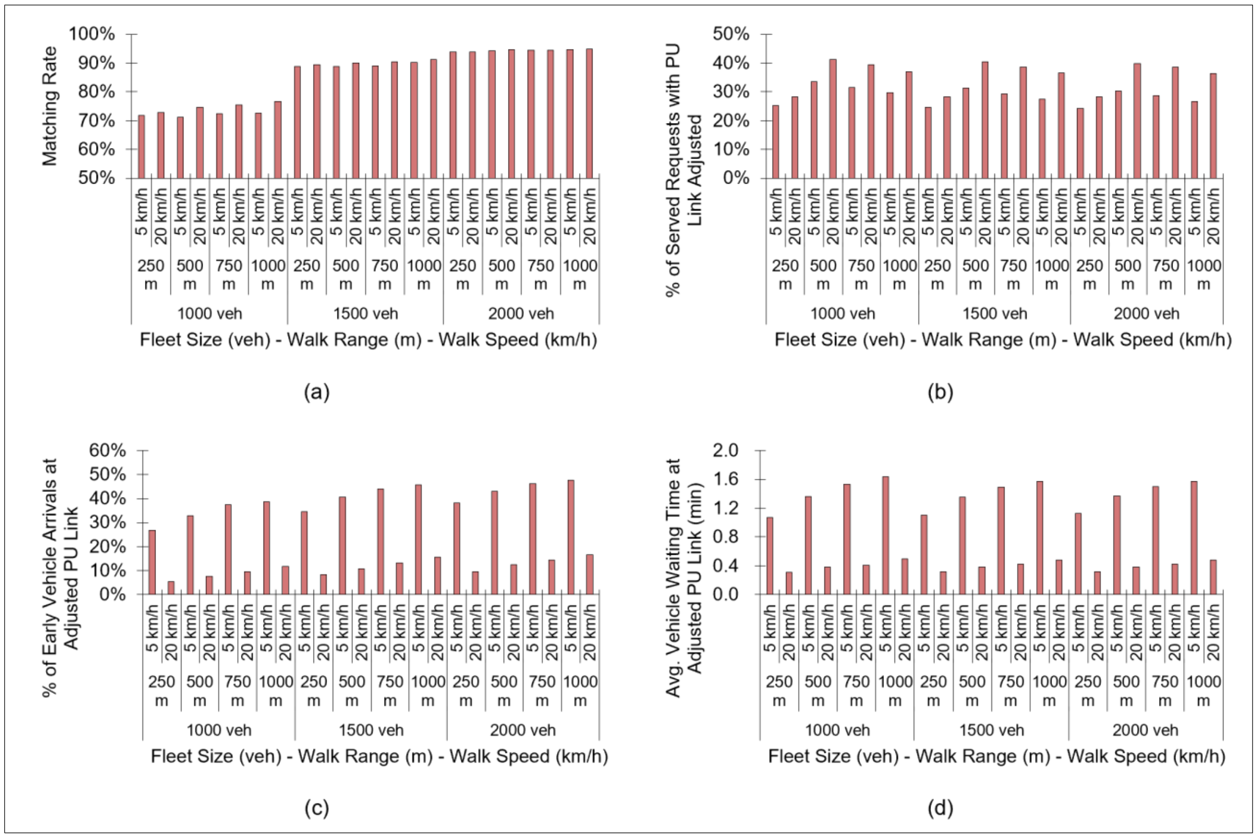


Figure 14. PU Link Adjustments and Vehicle Waiting at PU Link (C2C-RP Scenarios): (a) Matching Rate, (b) PU Link Adjustment Rate, (c) Rate of Vehicles Waiting for Requests with Adjusted PU links, and (d) Average Vehicle Wait Time at Adjusted PU Link for Early Arrivals

Figure 14c shows the probability a traveler's assigned vehicle had to wait at the traveler's PU link for the traveler to arrive, conditional on the traveler having an adjusted PU link. In such instances, the vehicle arrived earlier than the traveler at the adjusted PU link and hence has to wait for the traveler to complete their access walk trip to the adjusted PU link. The results clearly indicate that faster walking speeds drastically reduce the probability of a traveler's vehicle having to wait for them to arrive at the adjusted PU link. Similarly, Figure 14dshows that even when a vehicle waits for a traveler in the faster walking speed case, which is much less likely to happen, the vehicle waiting time is significantly lower than the case with slower walking speeds.

Figure 14c and Figure 14d collectively illustrate why faster walking speeds can improve operational efficiency and significantly reduce request-to-destination travel time—the faster walking speeds significantly reduce instances where vehicles wait for travelers, and in the case where vehicles do wait for travelers, the wait time is quite short. This combination reduces the amount of lost or unproductive vehicle time in the system. This could also be the reason why matching rate shown in Figure 13a is slightly higher for 20 kmph walk speed compared to 5 kmph walk speed, especially for low fleet sizes.

As mentioned in the Wang et al. (2022) review article, the issue of vehicles waiting for travelers at PU locations in C2C systems, is an overlooked issued. It is also an issue that does not or should not arise when travel times in the simulation environment are deterministic. However, travel times in congested real-world networks are not deterministic, they are uncertain. The POLARIS model used in this study is what enables us to capture this critical feature of C2C systems that is less relevant in D2D systems. We are concerned that other road network simulation models that do not capture travel time uncertainty may over-estimate the benefits of C2C services. Similarly, as highlighted in Section 4.3.1, a good estimate of the effective pick up walk range (D_{walk}^{PU}) value is required to reduce instances where vehicles have to wait for the request to arrive at the adjusted PU link. Using the network routed travel times would give a more accurate value for vehicle speed s_v in Eqn. 9 to estimate D_{walk}^{PU}, however it is computationally more intensive to perform this for each (r, v) pair compared to using a heuristic. Overestimating s_v would restrict the value of D_{walk}^{PU} thereby curtailing performance gains that could have been attained with a longer PU walk range. On the other hand, underestimating the value of s_v would lead to overestimating D_{walk}^{PU} thereby resulting in vehicles arriving early at the adjusted PU link and waiting for the traveler to arrive. Vehicles having to wait too long at the curbside to pick up a traveler may also impact curb space utilization as well as congestion on the adjacent links. This study could be extended in the future to evaluate such significant externalities.

6.4 Comparison of Sequencing Request-Vehicle Matching and PUDO Links Adjustment

This section compares two different algorithmic approaches (i.e., decision policies) for solving the C2C-RP problem. This is controlled by the γ_{C2C} parameter which determines whether PUDO Links adjustment is performed either after or before Request-Vehicle matching. In the baseline approach—labelled C2C-RP-A where the ‘A’ stands for ‘after’—we assign travelers to PUDO links *after* we assign them to vehicles (For each (r, v) match). In the alternative approach—labelled C2C-RP-B where the ‘B’ stands for ‘before’—we assign travelers to PUDO links for each (r, v) match candidate *before* we assign them finally to vehicles. The C2C-RP-B approach is computationally more intensive than the C2C-RP-A approach. This is because, the PUDO links adjustment procedure described in Section 4.3 is repeated for every feasible (r, v) candidate for each request in C2C-RP-B, whereas the procedure is performed only for optimal (r, v) match pairs in C2C-RP-A. Since the effect of PUDO Links adjustment is also factored into the insertion cost used in the optimal Request-Vehicle matching module (Section 4.4) for the C2C-RP-B strategy, it should be pareto-improving in terms of operator and user costs. However, this depends on the accuracy of changes in insertion cost caused due to PUDO links so as to lead to a better optimal (r, v) match after PUDO Links adjustment.

Figure 15a shows that there is basically no difference between the two algorithmic approaches in terms of matching rate. At low fleet sizes, C2C-RP-A does match slightly more travelers to vehicles. Figure 15b shows that C2C-RP-B produces significantly lower VKT per request than C2C-RP-A across all fleet sizes. This is because the optimal Request-Vehicle matching module is able to make better optimal (r, v) pair matches since the best PUDO links have been chosen for each (r, v) candidate. The gap between PUDO links adjustment before match, and PUDO links adjustment after match is not significant for the ride-hailing case in Figure 15b. Figure 15c shows that at low fleet sizes, C2C-RP-B slightly increases request-to-destination time relative to C2C-RP-A. This change seems to mostly arise from slightly higher access and egress walk times (or distances) for C2C-RP-B strategy compared to C2C-RP-A. This is because the current formulation of the optimal matching objective function (Eqn. 11) does not include PUDO walk distances. Hence the optimal minimum insertion cost matching solution may have slightly longer access and/or egress walk distances. For ride-hailing, the gap between the two algorithmic policies is insignificant. Finally, Figure 15d shows that vehicle occupancy is higher for C2C-RP-B than C2C-RP-A, which could also be explained as C2C-RP-B factoring in PUDO links adjustment to make better optimal matching choices.

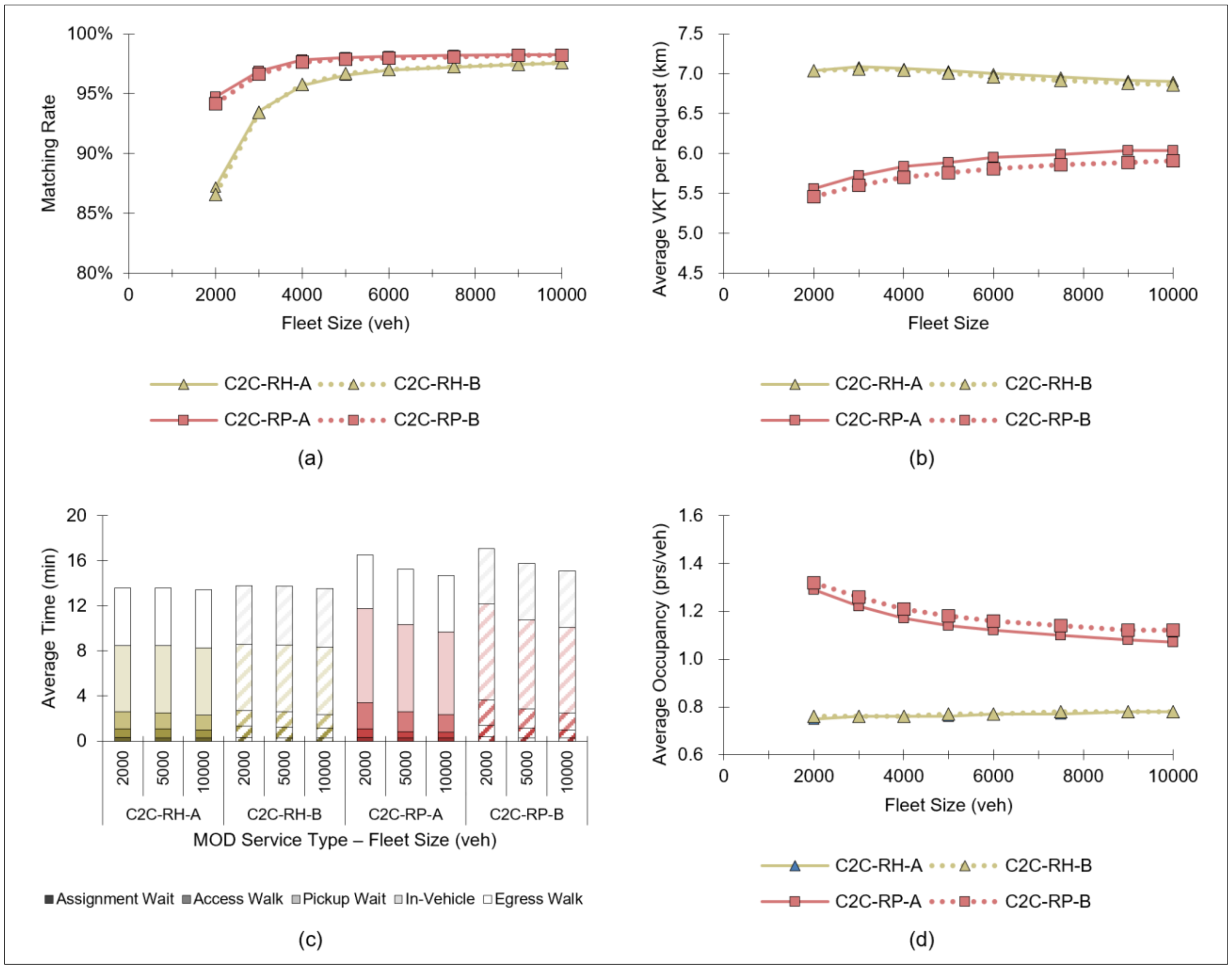


Figure 15. PUDO Links Adjustment after/before Traveler-Vehicle Matching: (a) Matching Rate, (b) Average VKT per Request, (c) Average Travel Time broken down by Segment, and (d) Average Vehicle Occupancy

6.5 Computational Time Results

This section aims to illustrate the scalability of the proposed decision policy and algorithmic approaches for the C2C-RP problem. All scenarios in this section were run on University of California Irvine's High Performance Computing Cluster (HPC3). The system resources allocated to run the simulation on the cluster are 32 logical processors and 128 GB of RAM. Figure 16 shows that for up to 10,000 vehicles serving up to 220,000 requests over the course of a day, in a detailed mesoscopic transportation system simulation model of Bloomington IL, model run times do not exceed 40 minutes. Additional test runs in the Chicago network with similar fleet sizes and traveler requests indicate that model run time does not explode in a bigger network.

Figure 16 further shows how computational run time changes with respect to several key parameters. As expected, larger fleet sizes require longer run times than shorter fleet sizes. Similarly, longer walk ranges require longer run times. The reason for this finding is that longer walk ranges increase the number of feasible PUDO links, thereby increasing the size of the PUDO links selection problem. In contrast, Figure 16 shows that run time increases slightly with higher walk speed. This can be attributed to the increase in effective PU walk range value D_{walk}^{PU} with a higher value of s_w which increases the search space of feasible PU link candidates.

Finally, Figure 16 shows that the pareto-improvement in operator and user costs when using the adjust PUDO links before match decision policy (C2C-RP-B), does come at the expense of computational time (i.e., a third ‘cost’). The computational run time is significantly larger for C2C-RP-B than C2C-RP-A, since PUDO links evaluation is performed for each (r, v) candidate instead of just (r, v) match pairs (Explained in Figure 17). Hence, in practice, fleet operators can choose between C2C-RP-A and C2C-RP-A depending on the computational power they have available and the size of individual problem instances. For the largest problem instances (10,000 vehicles, 1000 meters walk range and 5 km/h walk speed), C2C-RP-A takes nearly 20% more computation time compared to D2D-RP (26 minutes vs 21 minutes). C2C-RP-B on the other hand takes nearly 70% more computation time compared to D2D-RP (36 minutes vs. 21 minutes).

Figure 17 displays a breakdown of computational run time in each submodule of the C2C-RP solution implemented in POLARIS, for the largest problem instance—10,000 vehicles, 5 km/h walk speed, 1,000-meter walk range. As the figure shows, much of the computational time is spent in the module for determining feasible request-vehicle match candidates (‘R-V Candidate Prep Time, Section 4.2). For the adjust PUDOs before matching case, the time spent adjusting PUDOs for each request is also quite high. Compared to the adjust PUDO links after matching case, the former case takes eight times longer, because PUDO links are adjusted for all eight candidate vehicles associated with each request (k_{veh} parameter in Section 4.2), rather than just the single matched vehicle. The computational time will also increase if the number of candidate PU/DO links being evaluated (k_{PUDO}) is increased.

Fortunately, it is straightforward to parallelize the feasible request-vehicle search module as well as the PUDO links adjustment module in the code. The algorithm determines candidate vehicles for each request independently. Similarly, the algorithm adjusts PUDO links for each (r, v) match or match candidate independent of the other (r, v) matches/match candidates. The only C2C-RP module that cannot be parallelized is the request-vehicle matching optimization, but this module consumes very little computational resources compared to other stages as shown in Figure 17.

The ‘other processing time’ incorporates the rest of the modules in POLARIS including traffic simulation and vehicle pathfinding. Much of this run time is independent of the C2C-RP module.

Given that the runtime is relatively short to begin with, and the two MOD submodules that take the most time can be parallelized, it is clear that the proposed decision policy and algorithm strategy for the C2C-RP problem is highly scalable.

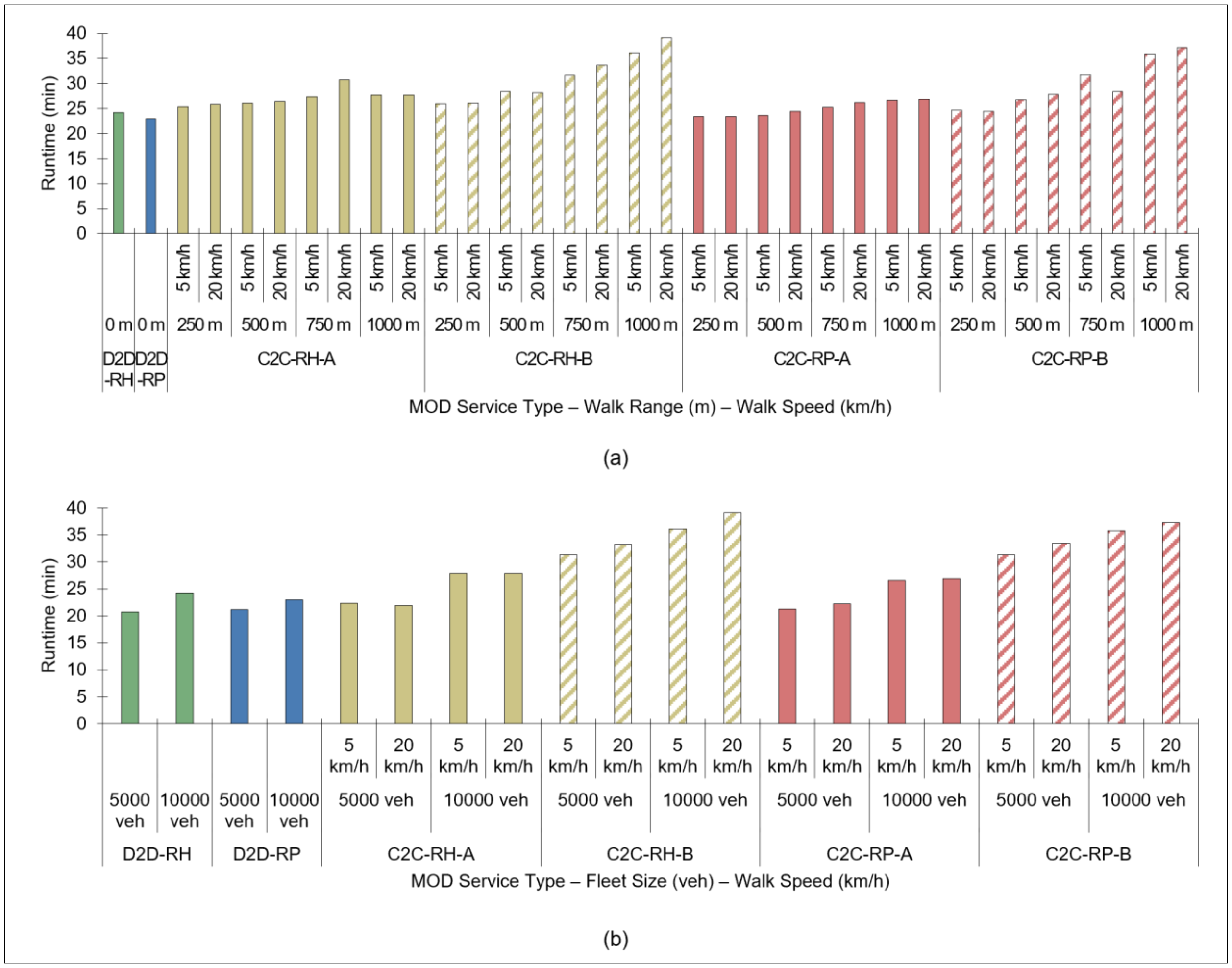


Figure 16. Computational Time by (a) Walk Range and (b) Fleet Size

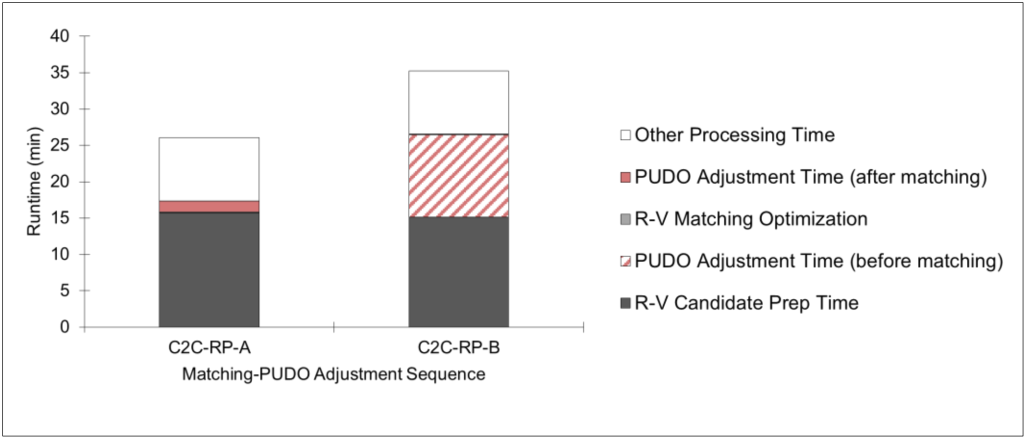


Figure 17. Computational Time by Matching-PUDO Links Adjustment Sequence (10,000 vehicles, 5 km/h walk speed, 1,000 m walk range)

7 Conclusion

MOD services serve a sizable portion of demand in many urban areas. The D2D-RH service still dominates the MOD market, which is problematic because ride-hailing includes significant deadheading miles and low vehicle occupancies. Several MOD service variants offer many of the benefits of D2D-RH with significantly fewer vehicle miles and higher vehicle occupancies. These variants include D2D-RP, C2C-RH, and C2C-RP. One of the two goals of this study is to compare these four MOD service variants in terms of user costs and operator costs. The second goal of this study is to develop an effective and scalable decision policy and solution algorithm to solve the C2C-RP operational problem.

The C2C-RP operational problem is a complex highly dynamic sequential decision problem with a very large decision space. The C2C-RP operator needs to frequently assign new requests to vehicles and to PU and DO links. We propose a decision policy that decomposes the decision problem into two subproblems, the request-vehicle matching problem and the PUDO links adjustment problem. The decision policy utilizes the location and planned itineraries of every vehicle in the fleet, the status of each request in the system, forecasted link travel times, and geospatial information to solve both subproblems in every decision epoch. The proposed decision policy dynamically assigns new requests to vehicles, sequences and re-sequences traveler PU and DO tasks, schedules and re-schedules traveler PU and DO tasks, and selects PU and DO locations for each request-vehicle match or candidate match. We also vary the order in which we iteratively solve the two subproblems. This paper includes several algorithmic contributions related to addressing the C2C-RP operational problem, delineated in Section 2 relative to the existing literature and described in Section 4.

We use the POLARIS agent-based transportation systems simulation model to test the proposed decision policy for the C2C-RP problem and to compare the four MOD service variants. The computational results illustrate that the proposed decision policy is both operationally effective and scalable. The most computationally demanding components of the decision policy and solution algorithm can easily be parallelized to further reduce run time.

The computational results also indicate that ride-pooling provides significant fleet operational benefits over ride-hailing services, in terms of VKT per request served, while only slightly increasing traveler request-to-destination times. The benefits for C2C services compared to D2D services are relatively smaller, with improvements in VKT per request coming at a large increase in traveler request-to-destination time. Moreover, combining ride-pooling and C2C service appears to provide additive benefits, compared to D2D-RH.

Additional computational experiments indicate that increasing the maximum customer walking range does not provide significant additional benefits in terms of VKT per request. Rather it is allowing walking legs at all, that provides much of the operational benefit, as allowing some walking can prevent highly inefficient PU and DO locations for vehicles.

Given this is one of only a handful of studies to address the C2C-RP operational problem, there are several remaining areas for future research. First, in the proposed decision policy and solution algorithm, we chose to avoid making extensive calls to a pathfinding algorithm to determine high-quality estimates of travel times between vehicles and requests. However, future research should consider the added operational effectiveness benefits and computational costs of employing a pathfinding algorithm. Second, in the current study, we do not consider vehicles blocking lanes of traffic or utilizing curb space when picking up and dropping off travelers. Future research should consider this possibility, as it may indicate a larger operational benefit for C2C services over D2D services. Third, future research should consider incorporating predictive models of supply and demand into the fleet operator's decision policy at each decision epoch. Fourth, while the current study performs extensive computational experiments in Bloomington, IL, future research should compare the four MOD services and evaluate the proposed decision policies effectiveness in different regions/networks. Preliminary results for Chicago suggest that the benefits of C2C-RP relative to D2D services are greater in Chicago, IL than Bloomington IL.

Acknowledgement

This article and the work described were sponsored by the U.S. Department of Energy (DOE) Vehicle Technologies Office (VTO) under the Systems and Modeling for Accelerated Research in Transportation (SMART) Mobility Laboratory Consortium, an initiative of the Energy Efficient Mobility Systems (EEMS) Program. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.

The first author also received partial funding for this research from the Gordon Hein Scholarship. The authors would like to express their gratitude to the scholarship sponsor as well as the Graduate Division at University of California Irvine for awarding the scholarship.

The authors have no competing interests to declare.

References

Aissat, K., Oulamara, A., 2014. A Priori Approach of Real-Time Ridesharing Problem with Intermediate Meeting Locations. Journal of Artificial Intelligence and Soft Computing Research 4, 287–299. https://doi.org/10.1515/jaiscr-2015-0015

Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., Rus, D., 2017. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. Proceedings of the National Academy of Sciences 114, 462–467. https://doi.org/10.1073/pnas.1611675114

Araldo, A., di Maria, A., di Stefano, A., Morana, G., 2019. On the Importance of Demand Consolidation in Mobility on Demand, in: 2019 IEEE/ACM 23rd International Symposium on Distributed Simulation and Real Time Applications. IEEE. https://doi.org/10.1109/DS-RT47707.2019.8958669

Auld, J., Hope, M., Ley, H., Sokolov, V., Xu, B., Zhang, K., 2016. POLARIS: Agent-based modeling framework development and implementation for integrated travel demand and network and operations simulations. Transp Res Part C Emerg Technol 64, 101–116. https://doi.org/10.1016/j.trc.2015.07.017

Balardino, A.F., Santos, A.G., 2015. Heuristic and Exact Approach for the Close Enough Ridematching Problem. Advances in Intelligent Systems and Computing 420. https://doi.org/10.1007/978-3-319-27221-4_24

Czioska, P., Kutadinata, R., Trifunović, A., Winter, S., Sester, M., Friedrich, B., 2019. Real-world meeting points for shared demand-responsive transportation systems. Public Transport 11, 341–377. https://doi.org/10.1007/s12469-019-00207-y

Czioska, P., Mattfeld, D.C., Sester, M., 2017. GIS-based identification and assessment of suitable meeting point locations for ride-sharing, in: Transportation Research Procedia. Elsevier B.V., pp. 314–324. https://doi.org/10.1016/j.trpro.2017.03.038

Dean, M.D., Gurumurthy, K.M., de Souza, F., Auld, J., Kockelman, K.M., 2022. Synergies between repositioning and charging strategies for shared autonomous electric vehicle fleets. Transp Res D Transp Environ 108, 103314. https://doi.org/10.1016/J.TRD.2022.103314

Fielbaum, A., Bai, X., Alonso-Mora, J., 2021. On-demand ridesharing with optimized pick-up and drop-off walking locations. Transp Res Part C Emerg Technol 126, 103061. https://doi.org/10.1016/j.trc.2021.103061

Gurumurthy, K.M., de Souza, F., Enam, A., Auld, J., 2020. Integrating Supply and Demand Perspectives for a Large-Scale Simulation of Shared Autonomous Vehicles. Transportation Research Record: Journal of the Transportation Research Board 2674, 181–192. https://doi.org/10.1177/0361198120921157

Gurumurthy, K.M., Kockelman, K.M., 2022. Dynamic ride-sharing impacts of greater trip demand and aggregation at stops in shared autonomous vehicle systems. Transp Res Part A Policy Pract 160, 114–125. https://doi.org/10.1016/j.tra.2022.03.032

Hyland, M., Mahmassani, H.S., 2020. Operational benefits and challenges of shared-ride automated mobility-on-demand services. Transp Res Part A Policy Pract 134, 251–270. https://doi.org/10.1016/j.tra.2020.02.017

Li, N., Kong, L., Shu, W., Wu, M.Y., 2020. Benefits of Short-Distance Walking and Fast-Route Scheduling in Public Vehicle Service. IEEE Transactions on Intelligent Transportation Systems 21, 3706–3717. https://doi.org/10.1109/TITS.2019.2931798

Lotze, C., Marszal, P., Schröder, M., Timme, M., 2022. Dynamic stop pooling for flexible and sustainable ride sharing. New J Phys 24, 023034. https://doi.org/10.1088/1367-2630/ac47c9

Lyu, Y., Lee, V.C.S., Ng, J.K.Y., Lim, B.Y., Liu, K., Chen, C., 2019. Flexi-Sharing: A Flexible and Personalized Taxi-Sharing System. IEEE Trans Veh Technol 68, 9399–9413. https://doi.org/10.1109/TVT.2019.2932869

Martin, S., Taylor, S.J., Yan, J., 2021. Trading flexibility for adoption: From dynamic to static walking in ridesharing, in: SSRN. https://doi.org/http://dx.doi.org/10.2139/ssrn.3984476

Martínez, L.M., Viegas, J.M., Eiró, T., 2015. Formulating a New Express Minibus Service Design Problem as a Clustering Problem. Transportation Science 49, 85–98. https://doi.org/10.1287/trsc.2013.0497

Miklas-Kalczynska, M., Kalczynski, P., 2020. Self-organized carpools with meeting points. Int J Sustain Transp 15, 140–151. https://doi.org/10.1080/15568318.2019.1711468

Mourad, A., Puchinger, J., Chu, C., 2019. A survey of models and algorithms for optimizing shared mobility. Transportation Research Part B: Methodological. https://doi.org/10.1016/j.trb.2019.02.003

Narayanan, S., Chaniotakis, E., Antoniou, C., 2020. Shared autonomous vehicle services: A comprehensive review. Transp Res Part C Emerg Technol 111, 255–293. https://doi.org/10.1016/J.TRC.2019.12.008

Sarma, N.J.S., Nam, D., Hyland, M.F., de Souza, F., Yang, D., Ghaffar, A., Verbas, I.O., 2020. Effective and Efficient Fleet Dispatching Strategies for Dynamically Matching AVs to Travelers in Large-scale Transportation Systems, in: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ITSC45102.2020.9294340

Simonetto, A., Monteil, J., Gambella, C., 2019. Real-time city-scale ridesharing via linear assignment problems. Transp Res Part C Emerg Technol 101, 208–232. https://doi.org/10.1016/j.trc.2019.01.019

Stiglic, M., Agatz, N., Savelsbergh, M., Gradisar, M., 2015. The benefits of meeting points in ride-sharing systems. Transportation Research Part B: Methodological 82, 36–53. https://doi.org/10.1016/j.trb.2015.07.025

Wang, Z., Hyland, M.F., Bahk, Y., Sarma, N.J.S., 2022. On Optimizing Shared-ride Mobility Services with Walking Legs. ArXiv.

Zardini, G., Lanzetti, N., Pavone, M., Frazzoli, E., 2022. Analysis and Control of Autonomous Mobility-on-Demand Systems. The Annual Review of Control, Robotics, and Autonomous Systems 5, 633–658. https://doi.org/https://doi.org/10.1146/annurev-control-042920-012811

Zheng, Y., Li, W., Qiu, F., Wei, H., 2019. The benefits of introducing meeting points into flex-route transit services. Transp Res Part C Emerg Technol 106, 98–112. https://doi.org/10.1016/j.trc.2019.07.012

Zwick, F., Kuehnel, N., Moeckel, R., Axhausen, K.W., 2021. Agent-based simulation of city-wide autonomous ride-pooling and the impact on traffic noise. Transp Res D Transp Environ 90. https://doi.org/10.1016/j.trd.2020.102673

Appendix A Solution Methodology

Finding Feasible Request-Vehicle match candidates

Direction and Vehicle Detour Checks

The first step in finding feasible vehicle candidates for each new unassigned request r involves subjecting each vehicle v to an elimination process based on direction compatibility and vehicle detour constraints. The procedure is described as follows:

- If v is idle and $v \notin V_r$ then add v to V_r
- If v is not idle and has seats available, then add v to V_r if all the following conditions are also met:
    - Directionality check: Check if the angle between the vectors representing the average `future path of vehicle v and the Euclidean path from the origin to the destination of request r is within a threshold. The average future path of v is formed by creating a unit vector from the current vehicle location in the direction of the average coordinates of all future PUDO links that the vehicle currently plans to visit (to pick up or drop off passengers), scaled by the total Euclidean distance of the remaining tour.

$$\theta = \cos^{-1} \frac{\vec{r} \cdot \vec{v}}{||\vec{r}||\ ||\vec{v}||} \tag{A1}$$

Where θ is the angle between the vehicle's current average Euclidean path and the requests Euclidean path, $\vec{r}$ denotes the vector connecting the origin and destination of request r and $\vec{v}$ is the vector denoting the current average planned path of vehicle v. Vehicles that are on the final leg of their current tour (i.e., the total distance remaining for v to complete its current tour is less than a minimum distance, are exempt from the directionality check.

- Vehicle path detour check: This is done to ensure that the candidate vehicle v does not detour beyond a certain threshold to pick up the new request. This is done based on the relative location of the request origin with respect to the vehicle's current average planned path, found by calculating the below parameter:

$$u = \frac{\vec{x} \cdot \vec{v}}{||\vec{v}||^2} \tag{A2}$$

where $\vec{x}$ denotes the vector that starts from the vehicle's current location and ends at request's origin, and $||\vec{v}||$ denotes the Euclidean length of the vehicle's current average path vector. The vehicle path detour check is done based on the u parameter as follows:

- If $u < 0$, this means that the vehicle v needs to travel in a direction opposite to its current planned path to PU the new request r. Vehicle v is skipped if

$$|u \cdot v_x| + |u \cdot v_y| > dist_{rev}^{max} \tag{A3}$$

where v_x, v_y denote the x and y components of $\vec{v}$ respectively, and $veh\_rev\_dist\_thresh$ denotes the maximum distance between the vehicle's current location and the projection of the request origin onto the upstream of vehicle path $\vec{v}$. If the above reverse distance threshold is met, the vehicle v is also subjected to the next detour constraint.

- If $u \leq 1$, then add v to V_r if

$$||\vec{x}|| + ||\vec{y}|| - ||\vec{v}|| \leq dist_{detour}^{max} \qquad (A4)$$

Where $\vec{y}$ denotes the vector connecting the request origin and the end point of the average current vehicle path vector $\vec{v}$, and $||\vec{x}||$, $||\vec{y}||$ and $||\vec{v}||$ respectively denote the Euclidean distance between current vehicle location and request origin, request origin and end point of $\vec{v}$ and vehicle's current average Euclidean path length.

If $u > 1$, vehicle v is added to the candidate vehicle list V_r for request r, since the request origin is in downstream of the vehicle's current path end.