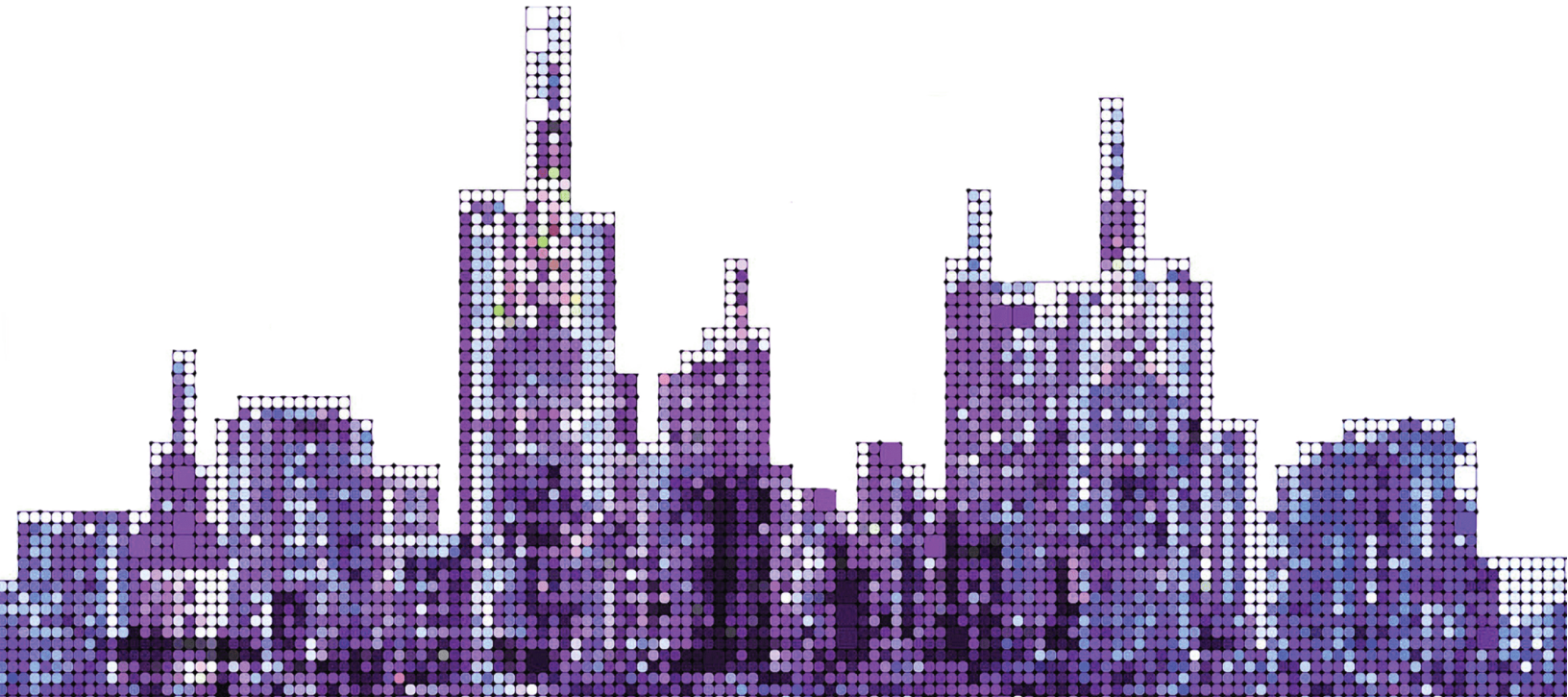
C2SMART
CONNECTED CITIES WITH SMART TRANSPORTATION
A USDOT University Transportation Center

New York University
Rutgers University
University of Washington
University of Texas at El Paso
The City College of New York

ONE-TO-MANY SIMULATOR INTERFACE WITH VIRTUAL TEST BED FOR EQUITABLE TECH TRANSFER

July 2023



TECHNICAL REPORT DOCUMENTATION PAGE

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
4. One-to-Many Simulator Interface with Virtual Test Bed for Equitable Tech Transfer		5. Report Date July 2023
		6. Performing Organization Code:
7. Author(s) Joseph Chow, Kaan Ozbay, Xuegang (Jeff) Ban, Hai Yang, Haggai Davis III, Ethan Wong, Farnoosh Namdarpour		8. Performing Organization Report No.
9. Performing Organization Name and Address Connected Cities for Smart Mobility towards Accessible and Resilient Transportation Center (C2SMART), 6 Metrotech Center, 4th Floor, NYU Tandon School of Engineering, Brooklyn, NY, 11201, United States		10. Work Unit No. 11. Contract or Grant No. 69A3551747119
12. Sponsoring Agency Name and Address Office of Research, Development, and Technology Federal Highway Administration 6300 Georgetown Pike McLean, VA 22101-2296		13. Type of Report and Period Final report, 3/1/22-7/31/23 14. Sponsoring Agency Code
15. Supplementary Notes		

16. Abstract

After five years of R&D, researchers have a number of independent simulation tools to evaluate different algorithms. A broad API was developed to handle interfacing any simulation with a multi-agent demand simulator. This was tested on the existing MATSim-NYC (which will be enhanced to include freight and parcel delivery activities) and aBEAM implementation, BEAM-NYC, for three use cases in electric transit, freight, and traffic. Each of these use cases considers equity impacts on different population segments (by income level, having disabilities, age level). The project jointly conducts some of the case studies in NYC and Seattle, enabling deeper insights of evaluated cases and promote tech transfer and collaborations to broader communities (including agencies, the industry, and the public).

17. Key Words	18. Distribution Statement No restrictions. This document is available to the public through the National Technical Information Service, Springfield, VA 22161. http://www.ntis.gov		
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 56	22. Price

Form DOT F 1700.7 (8-72) Reproduction of completed page author

One-to-Many Simulator Interface with Virtual Test Bed for Equitable Tech Transfer

C2SMART Center is a USDOT Tier 1 University Transportation Center taking on some of today's most pressing urban mobility challenges. Using cities as living laboratories, the center examines transportation problems and field tests novel solutions that draw on unprecedented recent advances in communication and smart technologies. Its research activities are focused on three key areas: Urban Mobility and Connected Citizens; Urban Analytics for Smart Cities; and Resilient, Secure and Smart Transportation Infrastructure.

Some of the key areas C2SMART is focusing on include:

Disruptive Technologies

We are developing innovative solutions that focus on emerging disruptive technologies and their impacts on transportation systems. Our aim is to accelerate technology transfer from the research phase to the real world.

Unconventional Big Data Applications

C2SMART is working to make it possible to safely share data from field tests and non-traditional sensing technologies so that decision-makers can address a wide range of urban mobility problems with the best information available to them.

Impactful Engagement

The center aims to overcome institutional barriers to innovation and hear and meet the needs of city and state stakeholders, including government agencies, policy makers, the private sector, non-profit organizations, and entrepreneurs.

Forward-thinking Training and Development

As an academic institution, we are dedicated to training the workforce of tomorrow to deal with new mobility problems in ways that are not covered in existing transportation curricula.

Led by the New York University Tandon School of Engineering, C2SMART is a consortium of five leading research universities, including Rutgers University, University of Washington, the University of Texas at El Paso, and The City College of New York.

c2smart.engineering.nyu.edu

PI: Joseph Y. J. Chow
New York University
0000-0002-6471-3419

Co-PI: Kaan Ozbay
New York University
0000-0001-7909-6532

Co-PI: Xuegang (Jeff) Ban
University of Washington
0000-0003-3605-971X

Hai Yang
New York University
0000-0003-2143-4636

Haggai Davis III
New York University
0000-0001-9886-4090

Ethan Wong
New York University

Farnoosh Namdarpour
New York University
0000-0002-9362-7456

Disclaimer

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated in the interest of information exchange. The report is funded, partially or entirely, by a grant from the U.S. Department of Transportation's University Transportation Centers Program. However, the U.S. Government assumes no liability for the contents or use thereof.

Acknowledgments

In addition to the funding support from C2SMART, some of the involved researchers were supported by Volkswagen/MOIA. Information shared by Volkswagen/MOIA is gratefully acknowledged. In addition, support from the NYU's Summer Undergraduate Research Programs is much appreciated.

Executive Summary

Simulation plays a crucial role in transportation studies. Through decades of development, more than 70 transportation focused simulation tools have been introduced, each dedicated for a specific area of transportation. One of the challenges in utilizing them is that it is difficult to incorporate multiple simulation tools in a unified setting and generate collaborative output. Given the increasing complexity of transportation systems and the incorporation of novel transport services, it is essential to tackle such difficulties and establish a systematic approach to create a multi-simulator environment.

The introduction of MATSim enables efficient simulation of large-scale transportation systems involving multi-modal movements. By default, it functions as an agent-based simulator, providing detailed trip information for individual entities traveling on roads. Its functionality can be expanded by integrating customizable extensions. Each extension is dedicated to a specific transportation service, and it can be seamlessly integrated with the existing MATSim environment. As a result, MATSim and its extensions have evolved into an ecosystem that facilitates the detailed evaluation of complex transportation systems.

However, the MATSim ecosystem has some shortcomings. One limitation is the restricted functionality of MATSim extensions. Since the extensions are added as modules within the MATSim environment, their capabilities are constrained by the baseline design of MATSim. For instance, due to the simplified traffic behavior in MATSim, it is not possible to incorporate extensions involving advanced traffic control algorithms. Another drawback is the barrier to extension development. If there exists an external simulator specifically designed for a new transportation service that needs to be integrated into MATSim, it must be translated into an extension. This translation process is comparable to developing a new extension from scratch, requiring familiarity with MATSim code and expertise in Java. Consequently, these requirements limit the accessibility of the MATSim ecosystem to a broader user base.

To overcome these shortcomings, we initiated a project focusing on developing APIs that integrate MATSim with other independent external simulators. These APIs eliminate the need to convert external simulators into dedicated MATSim extensions while expanding the application of MATSim in broader scenarios. The project is a collaborative effort between the NYU and UW teams. The NYU team developed an API focusing on incorporating MATSim with external simulators dedicated to ride-pooling services. The UW team developed a platform that integrates MATSim with the traffic simulator SUMO, and the platform is used to evaluate how an advanced

traffic control system applied on a local scale can impact city-wide traffic. The case studies using the two products showcase their potential in future transportation studies. They provide opportunities for large-scale transportation analysis with enhanced accuracy while lowering the obstacles in jointly utilizing MATSim and external simulators.

The API developed by the NYU team is called the Fleet-Demand platform (*FD platform*). It enables the communication between MATSim and external fleet-based simulator, which facilitates the collaborative process between the two parts. In this project, ride-pooling simulators were used to demonstrate the overall process. Before running the *FD platform*, an initial trip plan and configuration file are prepared. In the configuration file, ride-pooling services are defined as a teleportation mode, and the teleportation speed reflects the initial estimate of the average speed of ride-pooling trips. When running the *FD platform,* the initial trip plan and configuration file are input into MATSim, generating a trip itinerary containing estimated ride-pooling trip information. Then, they are extracted from the trip itinerary and fed into the external ride-pooling simulator. The original trip itinerary is subsequently updated with the realized trip information generated from the ride-pooling simulator. In addition, the teleportation speed defined in the configuration file is also updated by the new average speed obtained from the external simulation result. The updated trip itinerary and configuration file are used as new input in the next *FD platform* cycle, which represents a new round of MATSim and external simulation run. The cycle terminates once a predetermined stopping criteria is met. The final updated trip itinerary is the final output of the *FD platform* run, which can be used for further analysis.

The *FD platform* was used in several case studies to showcase its capability. We defined an area in midtown Manhattan as the ride-pooling service region with two fleets serving the area. In the first case study, we created a trip plan with 5,027 agents by using part of the *MATSim-NYC* population. We ran the *FD Platform* in juxtaposition with running DRT extension enabled MATSim using the same trip plan. We adjusted the parameters used in the two systems to ensure identical ride-pooling service strategies adopted in the simulation runs. The stopping criteria was met after 50 *FD platform* cycles. 10% difference was observed between the outputs generated from the two systems in terms of share of total ride-pooling trips. When coming to the total ride-pooling trip time, the difference was less than 1.5%, showing that the API can adequately mimic a fully integrated MATSim simulation without having to run the extension.

After proving the reliability of *FD platform*, we showcased how it can be applied in evaluating ride-pooling service strategies. In the same service area, heterogeneous service strategies were defined for the two fleets. In total, 3 scenarios were created by assigning difference levels of

service speed and cost to the two fleets. The ride-pooling trips defined in the previous trip plan were evenly split between using the two fleets. The final *FD platform* outputs were able to reflect the trip choice differences made by users when facing heterogeneous ride-pooling services. Overall, users favor cheaper service when same levels of trip speed are provided by the two fleets. However, such preference becomes negligible when neither fleet can provide faster service compared to other modes, and less users choose the ride-sharing service. Interestingly, faster service is still preferable even if it costs more than the slower service, showing the importance of providing higher quality services.

We also conducted a case study exploring its application in social equity analysis. We created a new trip plan containing 35,000 agents randomly selected from the *MATSim-NYC* population file. In addition, those agents were divided into two subpopulation groups: "Low_Income" and "Not_Low_Income". The "Low_Income" group consists of 20% of the population while the remaining population is defined as "Not_Low_Income". 6 scenarios of ride-pooling service strategies were created to study the social equity impact caused by different ride-pooling services. After running the *FD platform* under the 6 scenarios, we found that for the broader trip makers, the social equity impact imposed by ride-pooling services is limited due to the small number of actual ride-pooling trips. However, for the actual ride-pooling passengers, the level of inequality increases between the two groups when faster and cheaper services are provided. Because "Not_Low_Income" is the dominant population, a more accessible service would disproportionately benefit the majority. However, when analyzing the utility gain within each group, we found that a faster and cheaper service benefited both groups, and the relative benefit brought to the "Low_Income" group was higher. Therefore, it is essential to bring cheap while convenient services to communities with a higher percentage of low-income households.

The platform developed by the UW team is called the *VTD simulator*. In this project, two applications were discussed to showcase its capability in traffic associated studies. The first one studied the effectiveness of applying a multiscale traffic signal control system on a local scale. By using the *VTD simulator*, the efficiency gain of applying the proposed multiscale method on an intersection was accurately quantified, and it was compared with the output generated by using the traditional signal control method. After running the *VTD simulator* under different volume scenarios, fuel-saving performance improves by at least 22.71%. In addition, vehicle trajectories were plotted under each scenario to offer more insights regarding how the multiscale method changed the traffic patterns and resulted in such high efficiency gain.

In addition to local scale scenarios, a traffic control problem involving traffic information on the regional-level network was also discussed. The Downtown Seattle network was used and the speed, flow, and density information for each link was generated by running the *VTD simulator*. The generated data was then used as input to obtain the macroscopic fundamental diagrams (MFDs). By combining the MFDs with region partitioning, a MFD-based perimeter control plan could be generated, which could be further implemented in the *VTD simulator* for further traffic evaluation.

The project has resulted in a python product of the *FD platform* and a dataset containing test parameters and results from NYC case study. A paper was prepared for product development. It was presented in the University Luxemburg invited talk and the Transit Techies presentation. The work has supported a PhD student for portions of his dissertations and one undergraduate summer project at C2SMART.

Table of Content

List of Figures

List of Tables

1. Introduction

1.1. Project Background

Agent-based simulation has long been an influential tool in traffic and transportation research. It offers a detailed perspective on individual entities within the system, which can then be aggregated into mesoscopic and macroscopic level results. In traffic studies, this method is commonly employed for traffic flow modeling (Lee, 2007; Agarwal et al., 2015; Naiem et al., 2010; de Souza et al., 2019), route choice modeling (Wahle et al., 2002; Klügl et al., 2003; Bazzan and Klügl, 2005), and signal control (Jang et al., 2018; Santos et al., 2018), among other applications. Moreover, its usage extends beyond traffic studies to the wider field of transportation. Examples include innovative transportation applications (Liu et al., 2017; Mualla et al., 2018; Yao et al., 2020), pricing scheme evaluations (Zheng et al., 2012; Zheng et al., 2016; He et al., 2021), and transportation service design (Ronald et al., 2015; Basu et al., 2018; Hörl, 2017). For a more comprehensive overview of the scope and applications of agent-based simulation, readers are encouraged to consult Bazzan and Klügl (2014).

Numerous agent-based simulation tools have been developed to address the increasing demand for transportation studies. Nguyen et al. (2021) identify over 70 simulation tools used in transportation research, with 40 of these being agent-based. Among these, MATSim (Horni et al., 2016) is one of the most popular for transportation studies due to its ability to capture multi-modal movements efficiently and accurately in large-scale systems. To enhance its adaptability, customizable extensions can be incorporated to better simulate specific transport modes. For instance, extensions for transit and demand-responsive transport (DRT) have been employed to assess public and on-demand transportation services (Viergutz and Schmidt, 2019; Ciari et al., 2016; Balac and Hörl, 2021), while the freight extension is utilized to analyze and evaluate freight movements (Martins-Turner et al., 2020; Bean and Joubert, 2020; Schroeder et al., 2012). However, the currently available extensions offer limited functionalities, and developing new compatible extensions for MATSim requires advanced knowledge and expertise in Java, which restricts its accessibility to a broader user base.

Without the APIs, each external simulator needs to be converted into a compatible extension that can be integrated into the multiagent simulator as demonstrated in Figure 1.1(a). Driven by these limitations, we developed two application programming interfaces (APIs) to enhance MATSim's functionality by connecting it to external local simulators without the need for

dedicated extensions. The two APIs focus on different aspects of the transportation study, fleet-based transport services and road traffic operations, as summarized in Figure 1.1(b).

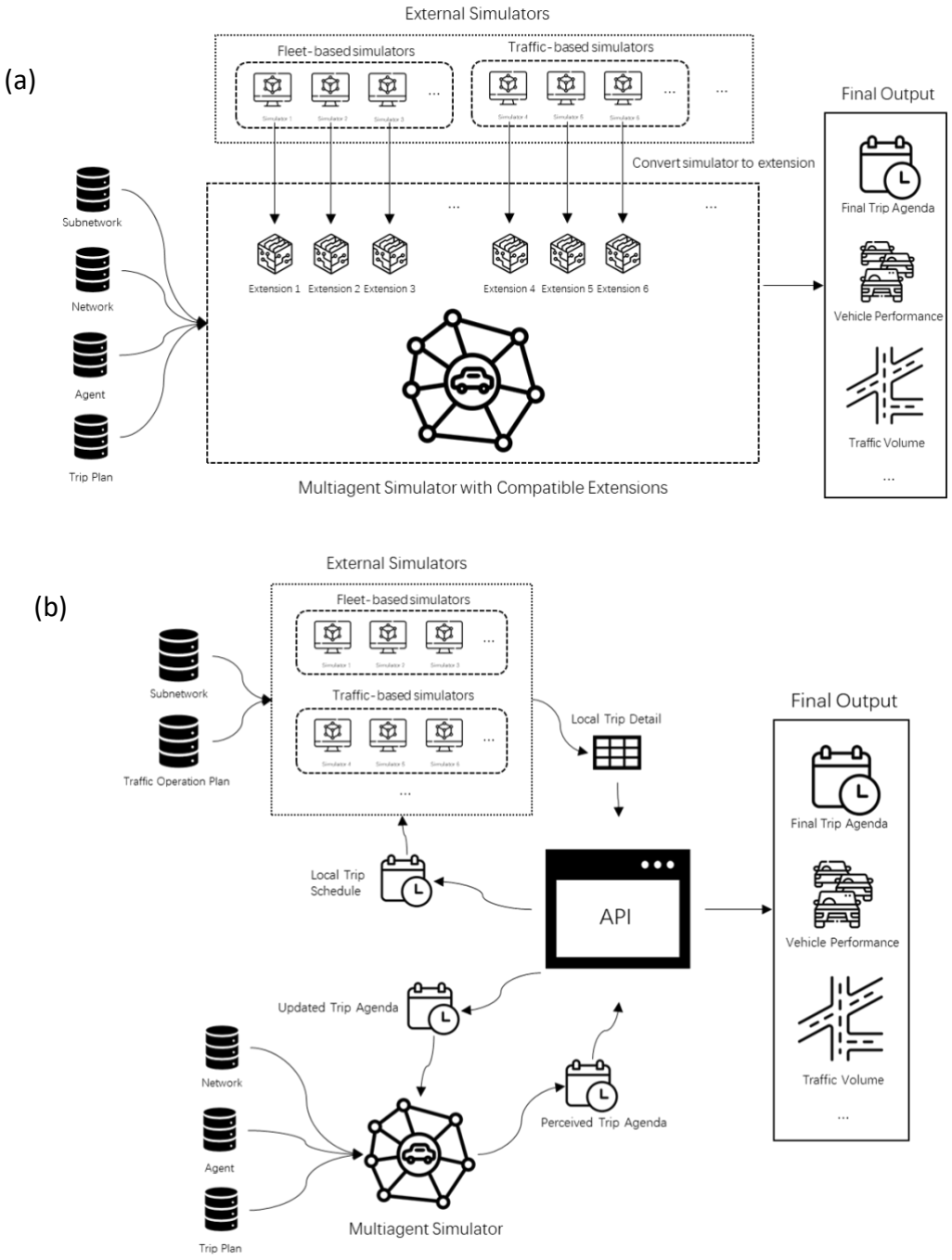


Figure 1.1. Overview of system interfaces with external simulators (a) without API and (b) using proposed MATSim APIs.

Written in Python, the APIs allow an external simulator that satisfies specific requirements to interface with MATSim without having to custom develop the simulator within the MATSim environment as an extension. Instead, the API integrates MATSim's day-to-day demand adjustment and mesoscopic traffic and transit network simulation with the external simulators.

As illustrated in Figure 1.1(b), the API can incorporate multiple external simulators. Figure 1.2 illustrates how the API should be able to incorporate two fleet-based external simulators and one traffic-based external simulator with the multiagent simulator. The multiagent simulator simulates the whole city-scale network while the external simulators focus on transportation services and traffic patterns in specific regions.

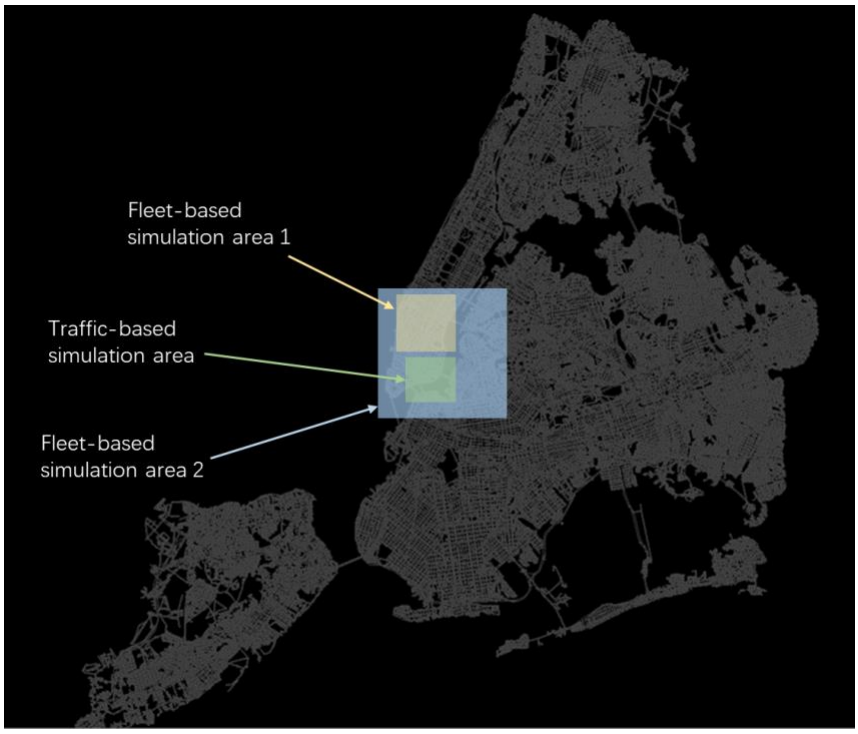


Figure 1.2. One scenario of a multiagent simulator (city-scale) interacting with external simulators (local-areas) using the API.

The external simulators can be coded in languages other than Java. The APIs facilitate information exchange between MATSim and local simulators, ensuring that the final output is generated collaboratively by the involved simulation tools. In this project, we develop a fleet-based API, named "*Fleet-Demand (FD) Platform*", to integrate with a Java-based ride-pooling simulation platform.

We run the *FD Platform* under various scenarios, alongside MATSim with the DRT extension enabled, to show that the API can adequately mimic a fully integrated MATSim simulation without having to run any extension. We further show how the API can link MATSim with more than one ride-pooling operation. Realistic scenario data obtained from a modified MATSim-NYC (He et al., 2021) is used to evaluate how changes in ride-pooling services would impact user equity among different population segments.

At the same time, we developed a traffic focused API named "*Vehicle-Traffic-Demand (VTD) Simulator*" to simulate the multiscale traffic at the vehicle, local traffic network, and large urban traffic network scales. We apply the *VTD simulator* to study coupled traffic-vehicle control with connected and automated vehicles (CAV), as well as to study macroscopic traffic dynamics.

Although only specific external simulators are involved in this project, the overall architecture can be easily adapted to incorporate other types of local simulators dedicated to different transport modes and services.

1.2. Terminology

In this section, we list all terminologies and corresponding definitions for clarity.

API: Application Programming Interface. In this project, the proposed APIs are platforms connecting MATSim with other simulators. The platform developed by the NYU team is called the FD Simulator, and the one developed by the UW team is called the VTD Simulator.

Core module: Modules that are included in the default MATSim setting. They allow MATSim to generate trip itineraries for each agent with the mode choice of car, public transit, walk, and bike.

DRT: Demand Responsive Transport. In this project, it is equivalent to the ride-pooling service.

Extension-enabled MATSim: MATSim simulation that involves additional extensions other than the core modules. The added extensions are dedicated to simulating transportation modes that the core modules are not able to handle. Those extensions require additional installation and activation for use.

FD Platform: A platform that connects MATSim and a ride-pooling simulator. Its output can reflect the system-wide transportation dynamics involving ride-pooling services.

External simulator: A simulation that only captures the supply side dynamics of a local system within a study area simulated by MATSim. An external simulator runs on a smaller scale and is designed to generate more detailed trip information and/or vehicle behaviors. In this project, two types of external simulators are involved. One is a ride-pooling simulation which simulates a fleet of vehicles that offering on-demand microtransit service. The other one is a microscopic traffic simulation, SUMO, which simulates vehicle behaviors in the road network and generates vehicle trajectories and corresponding traffic dynamics.

MATSim: Multi-Agent Transport Simulation. An open-source framework for implementing large-scale agent-based transport simulations (Horni et al., 2016). In this project, the word "MATSim" alone means that only the core modules are utilized in the simulation run**.**

MOD: Mobility on Demand. A form of transportation service that responds to on-demand requests from users, offering more flexible service. MOD services can include ride-pooling/microtransit, taxis, carshare, micromobility services like bikeshare and e-scooters, among others.

Ride-pooling: A mobility service that involves fleet-based on-demand microtransit service. Passengers request pickups and drop-offs from virtual stops in real time. A vehicle can be shared by multiple users when there are overlapping parts among their requested trips.

SUMO: Simulation of Urban Mobility. An example of a local traffic simulator that is integrated with MATSim using one of the proposed APIs.

VTD simulator: A traffic simulator platform that connects MATSim and SUMO and is dedicated to traffic dynamics and traffic demand. The result of the VTD simulator can be used to evaluate macroscopic traffic patterns in a detailed, controlled road network with heterogenous vehicle movements.

1.3. Research Objectives

NYU and UW propose two objectives for this project and are responsible for one of the two proposed APIs. The NYU team is responsible for developing FD Simulator. Following a similar objective, the UW team aims to develop a traffic-based API named *VTD simulator* (Ban et al., 2022) to develop, test, and evaluate a multiscale coupled vehicle-traffic control modeling framework for urban traffic with CAVs. The team also plans to investigate macroscopic level

traffic dynamics or macroscopic fundamental diagrams, by using the *VTD simulator*, currently developed for downtown Seattle.

1.4. Potential Applications

The developed products provide versatile usage scenarios. In this section, we list some potential applications that could be valuable for future transportation study.

Emerging Transportation Service Evaluation: Introducing a novel transportation service or bringing drastic changes to an existing service can highly influence a transportation system and users' choice of modes. For emerging transportation services, it is costly to implement the simulation of the operation in MATSim. Instead, the proposed APIs allow a researcher to implement an emerging service cheaply as a supply-side only simulator, and integrate that "local simulator" with MATSIM through the proposed API. By connecting external fleet-based simulators dedicated for certain transportation services with MATSim, with the general public and the existing transportation modes can be properly evaluated. The list below illustrates some potential scenarios.

- Evaluate how the introduction of automated vehicle fleets in mobility services would change users' travel patterns
- Explore the impact of charging station locations on shared electric vehicle services and how the impact could influence the overall service quality and the service usage
- How competition among MOD service providers could impact users' travel preferences

Large Scale Traffic Dynamic Evaluation: The incorporation of external traffic simulators enables MATSim to generate traffic dynamics sensitive to local adjustments such as the change of traffic control plan in the city center area. In such a way, the macroscopic traffic pattern can be evaluated on the large network level.

1.5. Research Challenges

Several challenges need to be overcome when developing the API. We list the main challenges in this section and the rest of the report presents key steps addressing these challenges:

- Design of the architecture of the API: what are the requirements for a local simulator to have in order to work with the API?

- Design of the architecture of the API: what are the requirements for the API to ensure consistency and convergence in simulation at both the local simulator and the MATSim levels, considering differences in scale, behavioral response, and detail of the simulated dynamics?
- How we can be certain that the API produces cohesive and realistic results
- In what way the API can be used and why it is beneficial

1.6. Report Organization

This report will focus on the two APIs developed by both the NYU team and the UW team. The remainder of this report is structured as follows: Section 2 describes the overall design of the two APIs. Section 3 outlines the development process for the *FD platform* and its core underlying principles. In Section 4, we present the *FD platform*'s outputs for various scenarios and compare them with results from the MATSim "DRT" extension. The scenarios are based on the modified MATSim-NYC population. We further conduct an equity study that assesses the effects of changes in ride-pooling services, which is one type of MOD service. In Section 5, we present details of using the *VTD simulator* for coupled traffic-vehicle control with CAVs, and for investigating the macroscopic traffic dynamics. For detailed development process for the *VTD simulator*, the reader can refer to Ban et al. (2022). Section 6 concludes the report, highlighting the dissemination of this work and outlining future applications and extensions of the API. Section 7 presents some information regarding technology transfer.

2. General API Design

In this section, we present the design of the two APIs. In general, MATSim simulates a large-scale transportation network with multi-modal movements while the external simulator generates more precise trip details of the involved agents in a focused subregion. By combining both simulators, large-scale multilevel simulation with improved precision can be achieved. As shown in Figure 1.1, there are commonalities in the two APIs that describe a class of APIs applicable to connecting any multiagent travel demand simulation with local supply side simulators. In addition to MATSim and its variants (BEAM (Bae et al., 2019) and predecessors (TRANSIMS (Smith et al., 1995)), other multiagent travel demand simulations include SimMobility (Adnan et al., 2016) and POLARIS (Auld et al., 2016). These commonalities are explicitly defined in

Table 2.1. Features of interfaces between multiagent travel demand and local simulators

Feature	Description
Subnetwork definition	One or more subnetworks of the multiagent travel demand simulator can be defined in which the separate local simulation is conducted. A subnetwork can potentially represent the whole study area.
Traveler arrivals	A mechanism is used to translate the travelers interested in using the system in the local simulator into a series of arrivals in that simulator
Performance updates	A mechanism is used to export the locally simulated system's performance for each traveler back into the multiagent simulator.
Within-day simulation	All local simulators should take the inputs from the multiagent simulation for one day and simulate the dynamics of the system within that day.

Table 2.2. Select types of local simulators that can be connected with proposed API class

Type	Description
Fleet simulator	A fleet of vehicles are simulated in terms of their routing logistics to serve customers. The subnetwork can be a service region. Travelers arriving can be either deterministically assigned as reservations or stochastically as real time arrivals. The fleet can be operated as a fixed route transit system or in an on-demand system, or somewhere in between (see Yoon et al., 2022). Performance includes traveler journey times, operator fleet vehicle miles traveled (and their impacts on GHG emissions, costs for travelers).
Traffic simulator	A road infrastructure system simulates the traffic control system within the subnetwork, where travelers entering the system interact with their passenger vehicles. The simulation can capture advanced ITS, traffic control devices, and road operating policies. Performance includes travel times through the subnetwork for each traveler.
Information system or mobile app simulator	An information system simulator simulates either a centralized or decentralized system in which information is shared with travelers in a dynamic manner, such as location-based services or crowdsourced systems (e.g., crowdshipping). A system that requires dynamic responses in service operations would have to include those operations in the simulation as well. Performance includes comparisons of perceived information at start of the day with the realized information.
Energy use simulator	An energy use simulator should capture exchanges in energy (e.g., charging, vehicle-to-vehicle smart grid, see Nourinejad et al., 2016). This would require exporting the within-day simulated trajectories to realize the demand for energy exchanges. Performance would be costs and energy consumed, including any schedule deviations made to exchange energy.

2.1. Fleet-based simulator

In designing the fleet-based API, the following requirements are made for the selection of a local simulator.

- The external simulator must use a subnetwork file aligned with the multiagent simulation network as input
- The local simulator must be able to simulate fleet trips in the same period as the multiagent simulation
- The external simulator must use agent's local trip start time, start location, and end location as input
- The external simulator must properly update agent's local trip agenda in terms of total trip time (from start of the trip request to end of the trip arrival)

As a result, the API design ensures the following features:

- *FD Platform* should extract feasible trip information from the multiagent simulation output and translate it into each local simulator's inputs
- *FD Platform* should properly update the trip agenda for the multiagent simulation with all the local simulators' outputs for a single simulated day and ensure the updated trip agenda follows the service requirements (e.g., service areas, service schedule, etc.)
- *FD Platform* should ensure the repeated interaction between the multiagent simulation and the local reaches a steady state
- *FD Platform* should ensure the final output is consistent with the output produced by only using multiagent simulation under the same simulation scenario if that simulator exists as an extension
- *FD Platform* should be able to run multiple local simulators with different service strategies
- *FD Platform* should generate outputs that properly reflect the service differences among local simulators

In the case of the fleet-based simulator, information on trip legs that involve the studied transportation mode are generated first using the given assumptions. Those trip legs are then extracted and used as input for the external simulator, which generates realized trip leg details. The details are subsequentially used to update the trip leg results previously generated by

MATSim. As such, the updated trip itinerary can properly reflect the user experience involving the studied mode.

As illustrated in Figure 2.1, an agent's chosen trip involves a public transit ride from home to work, a MOD service trip to have dinner, and another public transit ride back home. The mode choices are made by selecting the plan that maximizes the user's utility between the 2 perceived trip itineraries. An initial estimation of the MOD trip time is 15 minutes with a given average travel speed of 15 miles/hr. After providing the trip start location and start time, the simulator generates a realized trip time of 20 minutes with a travel speed of 12 miles/hr. The result is then used to update the previous itinerary. The updated itinerary is therefore a combined result of MATSim and the external simulator.

2.2. Traffic simulator

A similar logic is applicable when coming to the local traffic simulator. Instead of updating a trip leg, it updates part of the vehicle tour to reflect more complex road network conditions in the focused area. Figure 2.2 illustrates such logic. The process requires two MATSim simulations and one SUMO simulation in between. Since the two simulations are different, outputs from the two MATSim simulations may also be different.

We provide a simple example as follows. We define the areas that only MATSim simulation covers as M, and the areas that are covered by SUMO as S. After running the first MATSim simulation, the result shows an agent traveling from area M (at 8:10 am) through area S (8:20 to 8:30 am) and finishes the overall trip in area M (at 8:40 am). The overall trip will consequently be divided into two trips in MATSim and one trip in SUMO, as illustrated in Figure 2.2, in the following order: (i) trip #1 traveling in M (8:10 to 8:20 am), (ii) trip of traveling in S (8:20 to 8:30 am), and (iii) trip #2 traveling in M (8:30 to 8:40 am). These trips are later converted into agent plans as input for MATSim and route demand as input for SUMO.

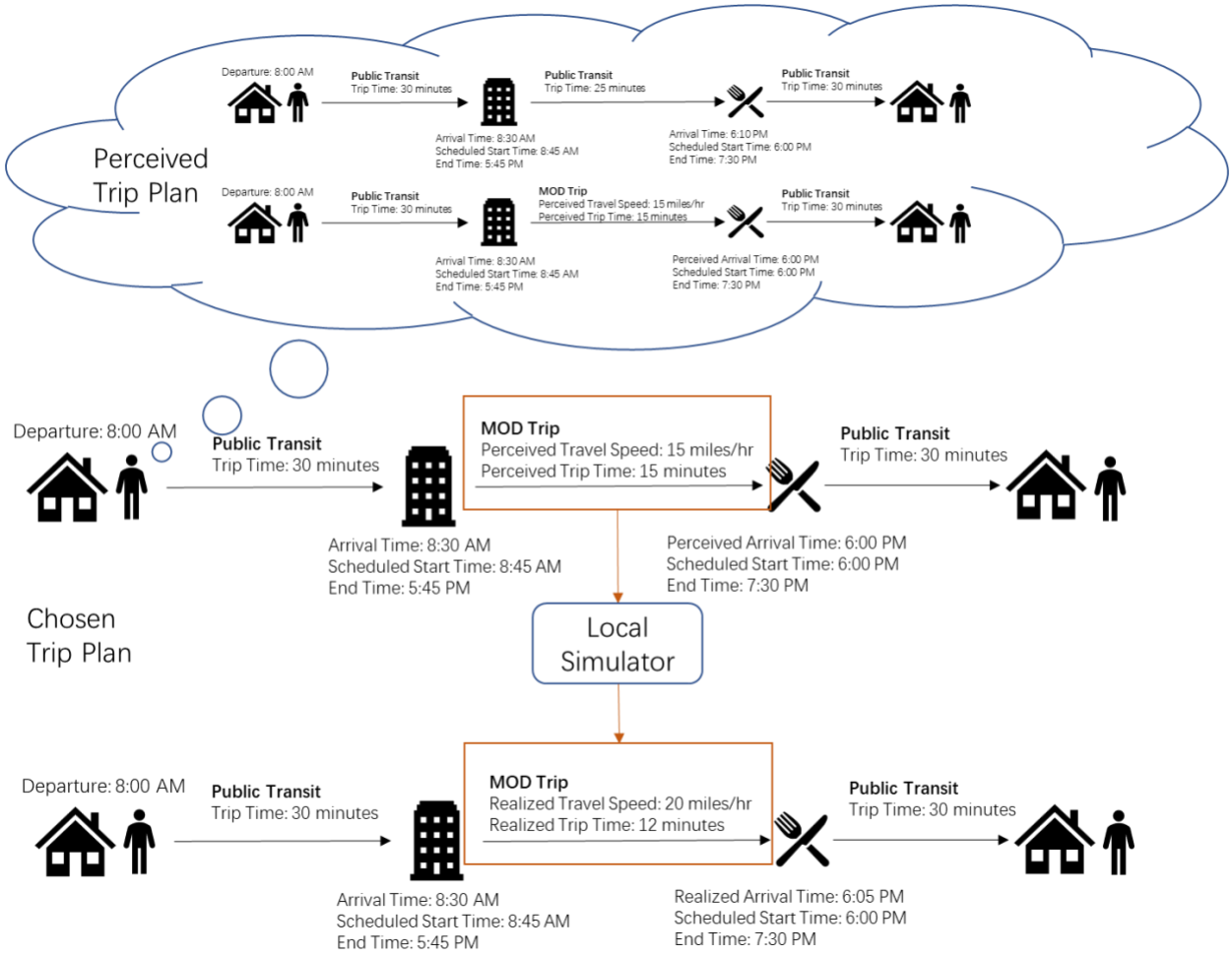


Figure 2.1. Illustration of trip itinerary update using the fleet-based local simulator.

Without the API, MATSim estimates the arrival time given to the route demand in SUMO is 8:30 am and the travel time in the SUMO area is 10 min. The agent's movement will then be simulated in SUMO, and the travel time of the agent may be different from 10 min since SUMO considers more detailed traffic dynamics and congestion patterns. Assume SUMO simulation generates 12 min travel time for the agent, implying that the agent will finish the trip in SUMO and return to MATSim at 8:32 am. That is, in the second MATSim simulation, trip #2 in MATSim will start at 8:32 am instead of 8:30 am as MATSim initially estimated. In this case, the first MATSim simulation is just used to generate initial agent schedules to divide their trips if necessary. Results of the second MATSim simulation are used for analysis.

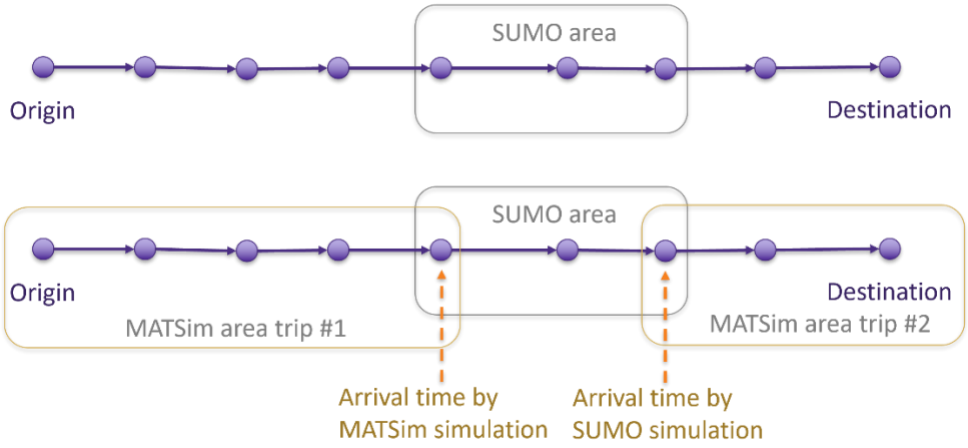


Figure 2.2. Illustration of vehicle tour update using the local traffic simulator.

Both examples presented before are simplified cases. In real world applications, multiple trip legs or multiple sections of the vehicle tour can be handled at the same time. In the case of fleet-based simulators, heterogenous fleets as well as multiple transportation modes can be incorporated in one API run.

3. Proposed structure of the FD simulator

The primary objective of the FD simulator development is to efficiently facilitate communication between MATSim and one or more project-specific externally simulated ride-pooling services, to simulate their interaction without the need for converting each local simulator into a dedicated MATSim extension. Assuming there are K non-identical ride-pooling services operating in a service region modeled by MATSim, the *FD Platform* can link MATSim to each of those services where each service is independently simulated using a calibrated ride-pooling simulator. We use MATSim version 14.0 for the *FD platform* and use proprietary code for the external ride-pooling simulators.

In summary, configuration and the initial trip plan file are inputted into MATSim. This process produces a preliminary passenger itinerary. Subsequently, trip details associated with ride-pooling services are extracted from this itinerary and transformed into a trip request log. This log is called by the local simulator to execute the service. The external simulator's output contains the realized ride-pooling service details. The initial passenger itinerary and configuration file are then modified based on the ride-pooling service outcomes for the subsequent day of MATSim simulation. This process is iterated multiple times until a consistent output is achieved.

Figure 3.1 illustrates the workflow of the *FD Platform* as previously described, and is a detailed breakdown of the concept shown in Figure 2.1. It also indicates the key variables involved in each element. All variables involved in the *FD Platform* process are listed in Table 3.1. We also list some terminologies for clarifications.

Activity: an activity can be described as both a trip leg and an exercise performed at a specific location, such as work, home, or study.

Trip plan: a file describes agents' intended activities.

Itinerary: an itinerary contains details of the activities after executing a trip plan through MATSim.

MATSim simulation day: one day of activities simulated in MATSim. Multiple days can be simulated consecutively in one MATSim run.

Simulation cycle: a full iteration involving one MATSim run, request log creation process, ride-pooling simulation run, and passenger itinerary modification process. In one *FD platform* run, multiple simulation cycles are required.

The following subsections dive into each key process illustrated in Figure 3.1 and explain how the variables are defined and adjusted. Section 3.1 addresses the preparation of the configuration file that depicts the ride-pooling services. Section 3.2 explains the network files and operation parameters required for the external ride-pooling simulator. Section 3.3 presents how ride-pooling trips are extracted from the MATSim itinerary and converted into a MOD trip request log for the following local simulator run. Finally, Section 3.4 outlines the method of updating the itinerary and configuration file according to the output from the external simulator. All sections are illustrated by screenshots from a running setup to facilitate the explanation.

Table 3.1. Table of variables

Indices

c	:	Simulation cycle
m	:	Day inside MATSim simulation
k	:	Ride-pooling service
i	:	Trip
e	:	Agent
j	:	Attribute
a	:	Activity
n	:	Node
l	:	Link

MATSim parameters

K	:	Set of ride-pooling services
F_c	:	Configuration file used for MATSim run during c^{th} cycle
$I_{k,c}^0$	:	Set of ride-pooling service k's trips initially assigned in P_c^0
P_c^0	:	Initial trip plan of the whole population for c^{th} cycle
T_c	:	Itinerary of the whole population generated from MATSim during c^{th} cycle
$I_{k,c}$	:	Set of ride-pooling service k's trips included in T_c
$t_{i,c}^o$	:	Start time of trip i generated from MATSim during c^{th} cycle
d_i	:	Beeline distance of trip i
f_k	:	Distance factor of ride-pooling service k
$s_{k,c}$	:	Average speed of ride-pooling service k in MATSim during c^{th} cycle
$P'_{e,c}$	:	Executed trip plan of agent e from MATSim during c^{th} simulation cycle
$\boldsymbol{P}_{e,c}$	:	Set of agent e's trip plans from day 1 to day $m-1$ generated from MATSim during c^{th} cycle
$P_{e,c}^m$	:	Trip plan of agent e in m^{th} day of MATSim run during c^{th} cycle
$U_{e,c}^m$	:	Total utility score of agent e's trip plan $P_{e,c}^m$
C_a	:	Constant utility score associated with activity a
u_a^j	:	Per unit utility score of attribute j performing activity a
$x_{a,e,m,c}^j$	:	Amount of attribute j when performing a in m^{th} day MATSim run during c^{th} cycle for agent e

M : Number of days involved in each MATSim run
$t'_{i,c}$: Perceived travel time of trip i generated from MATSim in c^{th} cycle

External simulator parameters
G_k : Subnetwork depicting the service area of ride-pooling service k
N_k : Set of all nodes in G_k
J_k : Set of all links in G_k
$L_{k,c}$: Request log of service k in c^{th} cycle
$L'_{k,c}$: Trip record after running the external simulator using $L_{k,c}$in c^{th} cycle
$I^*_{k,c}$ Set of trips within the service region for ride-pooling service k among $I_{k,c}$
$I^{\neg}_{k,c}$: Set of trips falling outside of the service region for ride-pooling service k among $I_{k,c}$
$I^-_{k,c}$: Set of trips failed to be served by service k in the local simulator run in c^{th} cycle
$I^+_{k,c}$: Set of trips served by service k in the local simulator run in c^{th} cycle
(x_i^o, y_i^o) : Coordinates of trip i's start point
(x_i^d, y_i^d) : Coordinates of trip i's end point
R_k : Set of parameters defining the operation condition of ride-pooling service k
$t_{i,c}$: Realized travel time of trip i generated from the external simulator in c^{th} cycle

FD platform parameter
T'_c : Final itinerary in the c^{th} cycle
$p_{k,c}$: Share of ride-pooling service k in T'_c
p_K : Total mode share of ride-pooling services belonging to set K
μ_n^C : n-cycle mode share average by C^{th} cycle
δ : Stopping threshold
T_F : Final trip itinerary of the *FD platform*
$U_{e,F}$: Final utility score obtained from T_F for each agent e

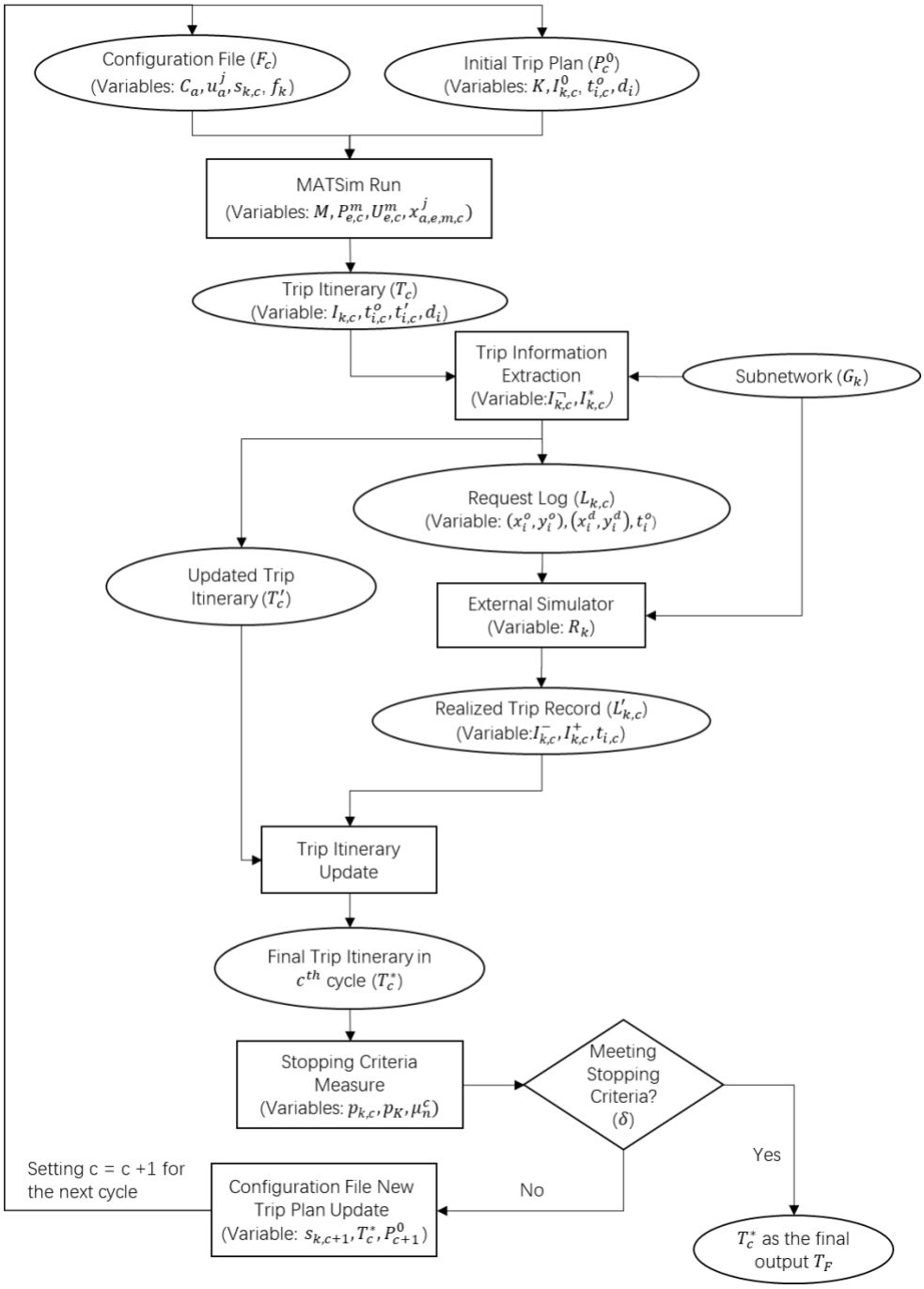


Figure 3.1. Workflow of the FD simulator process connecting MATSim and one or more local ride-pooling simulators.

3.1. MATSim Input File Preparation

At the beginning of the *FD platform* run, one MATSim run is required. To initiate the MATSim run, an initial trip plan and a configuration file are needed. We denote c as the current cycle iteration. When $c = 0$, the initial trip plan P_0^0 can be created by extrapolating travel surveys. Readers can refer to other MATSim studies such as He et al., (2021) for more information. When $c \geq 1$, the initial trip plan P_c^0 is the modified trip itinerary T'_{c-1} generated from cycle $c - 1$, which will be elaborated in the following subsections. In this subsection, we focus on the preparation of configuration file F_c that defines the parameters for ride-pooling services.

Figure 3.2 is an example of part of the initial trip plan P_0^0. In this example, trips associated with a specific ride-pooling service is labeled as "drt" (demand-responsive transport), and the "drt" trips are initially assigned in P_0^0 for agents prior to the start of the day. The *FD Platform* is used to avoid employing a dedicated MATSim extension like DRT for simulating every ride-pooling service's trip activity. The "DRT" extension is further elaborated in section 4 with examples. Instead, it defines each ride-pooling service being simulated externally as a separate teleportation mode in the configuration file, such as the "drt" mode illustrated in Figure 3.3.

The teleportation speed serves as an initial estimate for the aggregated average speed of the ride-pooling trip from the origin to the destination. This is not a vehicular speed; rather, it reflects the operational speed of transporting the passenger. For example, a service that provides ride-pooling may directly transport a passenger at the same speed as a passenger car on the shortest path in one simulation, but in another simulation, it may detour to drop off another passenger and take twice as long.

Let us assume the set of ride-pooling services is K, and each service is k. $I_{k,c}^0$ is the set of trip legs using service k initially assigned in P_c^0. After running one full cycle with P_c^0, itinerary T_c^* is generated with the set of realized trips $I_{k,c}^+$ during the c^{th} cycle. For each ride-pooling trip i from $I_{k,c}^+$, we define the trip start time as $t_{i,c}^o$, the trip duration as $t_{i,c}$, the beeline distance of the trip as d_i, and a distance factor f_k. f_k is held constant during the whole *FD platform* running process. The teleportation mode speed $s_{k,c}$ reflects the average performance of service k over the trip set $I_{k,c-1}^+$ obtained from the final trip itinerary T_{c-1}^*. When c = 0, $s_{k,0}$ can be estimated by using the trip details obtained from a similar mode such as private cars. The value of $s_{k,c}$ is updated using Eq. (3.1) when $c \geq 1$.

$$s_{k,c} = f_k \sum_{i \in \boldsymbol{I}_{k,c-1}} \frac{d_i}{t_{i,c-1}} \tag{3.1}$$

As illustrated in Figure 3.3, the initial teleportation speed of service "drt" $s_{drt,0}$ is the value assigned to "teleportedModeSpeed", and f_r is the value assigned to "beelineDistanceFactor".

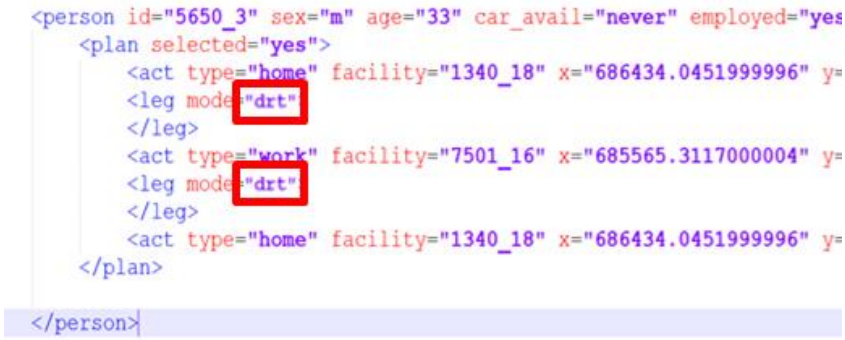


Figure 3.2. Screenshot of a population plan with assigned ride-pooling service (labeled as "drt").

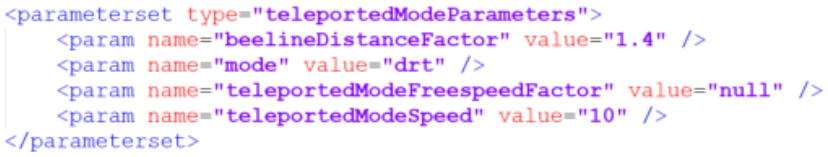


Figure 3.3. Screenshot of a configuration file with ride-pooling service (labeled as "drt") defined as a teleportation mode.

Additionally, the scoring function for the ride-pooling trip is specified in the configuration file. This scoring function incorporates the necessary parameters for the attributes, as illustrated in Eq. (3.2).

$$U_{e,c}^{m} = \sum_{a} (c_a + \sum_{j} u_a^j x_{a,e,m,c}^j) \tag{3.2}$$

$U_{e,c}^{m}$ is the total utility score of agent e's trip plan $P_{e,c}^{m}$ generated from the m^{th} day of MATSim run in the c^{th} cycle, c_a is a constant score value associated to activity a, u_a^j is the utility score gain of per unit attribute j associated with activity a, and $x_{a,e,m,c}^{j}$ is the amount of attribute j associated with activity a during the m^{th} day of MATSim run in the c^{th} cycle. As illustrated in Figure 3.4, the constant score value of performing a "drt" trip c_{drt} is the value assigned to "constant", and all per unit utility scores associated to the "drt" trips u_{drt}^{j} are defined in the rest of the elements.

The score function parameters need to be estimated beforehand or assumed relative to parameters in existing modes of an estimated model. By defining these essential elements, the files are prepared for input into MATSim for subsequent steps. For a complete guide regarding score functions in MATSim, the reader can refer to the MATSim user guide (Rieser et al. 2014). This approach can be applied to other modes of interest as well.

```
<parameterset type="modeParams" >
    <param name="constant" value="-2.482011986" />
    <param name="marginalUtilityOfDistance_util_m" value="0.0" />
    <param name="marginalUtilityOfTraveling_util_hr" value="0.0" />
    <param name="mode" value="drt" />
    <param name="monetaryDistanceRate" value="0" />
</parameterset>
```

Figure 3.4. An example of the score function associated with ride-pooling service (labeled as "drt").

To ensure sufficient mode changes inside each MATSim run, multiple days of simulation are required to generate a set of perceived trip plans so that each agent e can choose the one optimizing its own score $U_{e,c}$. However, the trip plan executed on the last day should be the optimized trip plan chosen from previous days. We define M days of simulation are involved in each MATSim run. Therefore, the generated trip plan set $\boldsymbol{P}_{e,c}$ in the first $M-1$ days for agent e would be $\{P_{e,c}^{1}, \dots, P_{e,c}^{M-1}\}$ during the c^{th} cycle. In day M, trip plan $P'_{e,c}$ with the highest utility score $U_{e,c}^{m}$ in $\boldsymbol{P}_{e,c}$ for each agent e will be executed. The final trip itinerary of all agents T_c will be used in the following steps. The multi-day simulation and final day execution is presented in Algorithm 1.

Algorithm 1. Multi-day simulation and final day execution in MATSim

Input : Initial trip plan P_c^0 and configuration file F_c

Output: Trip itinerary T_c

For $m \leq M - 1$ **do**

    Generate trip plan $P_{e,c}^m$ for each agent e

    Calculate utility score $U_{e,c}^m$ for each agent e using Eq. (3.2)

    Store $P_{e,c}^m$ in the trip plan set $\boldsymbol{P}_{e,c}$ along with the utility score $U_{e,c}^m$

Select $P_{e,c}^{'}$ from $\boldsymbol{P}_{e,c}$ with the highest $U_{e,c}^m$ for each agent e

Execute the selected $P_{e,c}'$ across the population and generate itinerary T_c

3.2. File Preparation for External simulators

Before running the external simulators, dedicated subnetwork files need to be prepared. For each ride-pooling service k, a subnetwork G_k is defined to depict the service area. The subnetwork files should contain the information of all nodes N_k and links L_k. Each node $n \in N_k$ should be labeled with a unique node ID and have its coordinates (x_n, y_n) stored using the same coordinate system as what MATSim has. Each directed link $l \in L_k$ should be defined by start and end point IDs along with unique link ID. The file should also contain link attributes such as distance d_l and travel speed s_l to define the link travel time t_l. The link travel time should properly reflect the traffic condition in the MATSim simulation environment. Therefore, it is recommended to extract the subnetwork information directly from some sample MATSim simulation results without the ride-pooling services. It also ensures the continuity of the network when connecting MATSim and the external simulators. By using the subnetwork files, G_k can be constructed for external simulation use.

Besides the subnetwork files, a set of parameters R_k defining each service k's operation is needed. R_k is a set of parameters defining the constraints and the strategies service k would have. For example, R_k can includes the number of vehicles, the capacity of each vehicle, the maximum detour each vehicle can have, etc. With each service k having a unique set of R_k, heterogenous services can be defined, and the *FD simulator* can reflect the service heterogeneity from agents' trip choices. In section 4.1, we present an external ride-pooling simulator involved in this project with the details of utilized R_k.

3.3. Ride-pooling Trip Extraction and Conversion

After running MATSim in the c^{th} cycle and outputting T_c, trips associated with ride-pooling service k can be extracted for the external simulator run. Figure 3.5 illustrates part of the itinerary containing two legs of the "drt" trip as defined in subsection 3.1. We define $I_{k,c}$ to be the set of executed trips using service k after running MATSim in the c^{th} cycle. For trip i belonging to $I_{k,c}$, the trip duration $t'_{i,c}$ of ride-pooling service k is obtained using the teleportation speed $s_{k,c}$ and beeline distances d_i between the trip's endpoints. However, the generated ride-pooling trip legs are considered as "perceived trips". In other words, they are hypothetical trips, and the trip duration $t'_{k,c}$ are what agents perceive they will experience based on historical information. To obtain realistic ride-pooling trip information that adheres to the specified ride-pooling service settings R_k for service k, an additional simulation run on the external simulator is required.

The request log for each ride-pooling service k in the c^{th} run is defined as $L_{k,c}$. Each row i represents the i^{th} trip generated from MATSim that day in chronological order. It is possible that one person may generate multiple trips requesting service from one ride-pooling service, or even request from multiple different services. The essential information needed to create the ride-pooling request log for the ride-pooling simulator input includes person ID, trip ID, trip start $(x^o_{i,c}, y^o_{i,c})$ and end $(x^d_{i,c}, y^d_{i,c})$ coordinates, as well as trip start time $t^o_{i,c}$, all of which can be directly extracted from the itinerary T_c. Only trips that both start and end inside subnetwork G_k are extracted to $L_{k,c}$. The set of those included trips is defined as $I^*_{k,c}$ .We define $I^{\neg}_{k,c}$ as the set of trips belonging to $I_{k,c}$ while being excluded from being written into $L_{k,c}$. All trips in $I^{\neg}_{k,c}$ are assigned to use other viable alternatives. In the current version of *FD platform*, they are all switched to use public transit. An independent mode choice model deciding which alternative mode to be switched to can be incorporated in future development. The updated T_c with mode switch of trips in $I^{\neg}_{k,c}$ is denoted as T'_c.

As illustrated in Figure 3.6, two trips extracted from T_c are perceived to use service k. Since one end of trip 1 is not inside the service region, it is not included in $L_{k,c}$. Instead, it will be switched to use public transit.

```
<activity type="home" facility="1340_18" x="686434.0451999996" y="4824844.6183" end_time="07:40:13" >
</activity>
<leg mode="drt" dep_time="07:40:13" trav_time="00:09:14">
    <attributes>
        <attribute name="routingMode" class="java.lang.String">drt</attribute>
    </attributes>
    <route type="generic" start_link="54_2" end_link="50_2" trav_time="00:09:14" distance="1569.2000265296067"></route>
</leg>
<activity type="work" facility="7501_16" x="685565.3117000004" y="4824006.562899999" start_time="07:55:13" end_time="17:16:19" >
</activity>
<leg mode="drt" dep_time="17:16:19" trav_time="00:02:19">
    <attributes>
        <attribute name="routingMode" class="java.lang.String">drt</attribute>
    </attributes>
    <route type="generic" start_link="50_2" end_link="54_2" trav_time="00:02:19" distance="1569.2000265296067"></route>
</leg>
<activity type="home" facility="1340_18" x="686434.0451999996" y="4824844.6183" >
</activity>
```

Figure 3.5. Screenshot of an itinerary output.

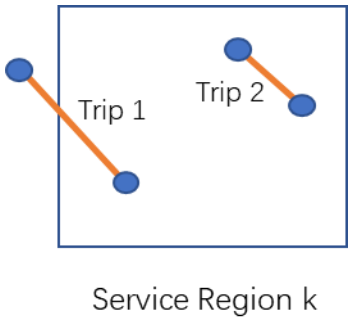


Figure 3.6. Trip selection based on service region constraint.

3.4. Itinerary and Configuration File Update

After feeding $L_{k,c}, G_k$, and R_k into the external simulator, realized trip record $L'_{k,c}$ will be generated. Figure 3.7 showcases part of the realized ride-pooling trip record $L'_{drt,c}$ following the external simulator run using the service log $L_{k,c}$. For each trip i in $L'_{k,c}$, the start point $(x^o_{i,c}, y^o_{i,c})$ and end point $(x^d_{i,c}, y^d_{i,c})$ locations, as well as the trip start time $t^o_{i,c}$, remain unchanged throughout the cycles. As a result, the only trip information that requires updating is the actual travel time $t_{i,c}$. However, due to service constraints, some trips in $L_{k,c}$ might be infeasible in the external simulation run. For example, trip requests could be rejected due to the limited number of available vehicles or violation of service constraints such as additional detour distance.

We define the set of unrealized trips as $I^-_{k,c}$ and the set of realized trips as $I^+_{k,c}$. For trip i belonging to $I^-_{k,c}$, it will be switched to use other feasible alternatives. In the current version, all unrealized trips are switched to use public transit. For trip i belonging to the realized trip set $I^+_{k,c}$, the

realized travel time $t_{i,c}$ will be used to update the MATSim itinerary T_c' by replacing the perceived travel time $t_{i,c}'$. As highlighted in Figure 3.8, the travel time of "drt" trips shown in Figure 3.5 is updated based on the simulation output as shown in Figure 3.7. The updated T_c' becomes T_c^*, and it will be used as the initial trip plan P_{c+1}^0 for MATSim run in the next cycle.

Traveler	Trip Start	Trip End	Travel Time
5650_3	7:40:13	7:45:44	0:05:31
5650_3	17:16:19	17:19:10	0:02:51

Figure 3.7. Realized MOD trips after the external simulator run.

```
<activity type="home" facility="1340_18" x="686434.0451999996" y="4824844.6183" end_time="07:40:13" >
</activity>
<leg mode="pt" dep_time="07:40:13" trav_time="0:05:31">
    <attributes>
        <attribute name="routingMode" class="java.lang.String">drt</attribute>
    </attributes>
    <route type="generic" start_link="54_2" end_link="50_2" trav_time="0:05:31" distance="1569.2000265296067"></route>
</leg>
<activity type="work" facility="7501_16" x="685565.3117000004" y="4824006.562899999" start_time="07:55:13" end_time="17:16:19" >
</activity>
<leg mode="pt" dep_time="17:16:19" trav_time="00:02:51">
    <attributes>
        <attribute name="routingMode" class="java.lang.String">drt</attribute>
    </attributes>
    <route type="generic" start_link="50_2" end_link="54_2" trav_time="00:02:51" distance="1569.2000265296067"></route>
</leg>
<activity type="home" facility="1340_18" x="686434.0451999996" y="4824844.6183" >
</activity>
```

Figure 3.8. Illustration of the updated itinerary.

Besides updating the itinerary T_c^*, the average speed of service k for the MATSim run in the next cycle, denoted as $s_{k,c+1}$, needs to be adjusted in the configuration file to reflect the updated service. $s_{k,c+1}$ is calculated by using Eq. (3.1), which incorporates d_i and $t_{i,c}$ for trip i belonging to $I_{k,c}^+$.

3.5. Full Simulation Cycle and the Stopping Criteria

After getting the updated trip plan P_{c+1}^0 and configuration file with the new teleportation speed $s_{k,c+1}$, a new cycle of *FD platform* run can be initiated. Starting with a new MATSim run, it employs the updated speed $s_{k,c+1}$ to generate improved approximations of the service k's trip travel time, offering agents a more accurate perceived trip duration $t_{i,c+1}'$ for trip i using service k.

The entire cycle described from subsections 3.1 to 3.4 is repeated until a stopping criteria is met. In the current *FD platform*, the stopping criteria is set as the difference of n-cycle moving average of all ride-pooling services reaching a threshold. The mode share of ride-pooling service k is denoted as p_k. Therefore, the total mode share among all K ride-pooling services P_K can be calculated using Eq. (3.3).

$$p_{K,c} = \sum_{k \epsilon K} p_{k,c} \tag{3.3}$$

The n-cycle average till the C^{th} cycle is denoted as μ_n^C, which is calculated using Eq. (3.4).

$$\mu_n^C = \frac{\sum_{C-n}^{C} p_{K,c}}{n} \tag{3.3}$$

The *FD platform* terminates when the absolute difference between μ_n^{C-1} and μ_n^C is lower than a predefined threshold δ. For example, if we use 5-cycle moving average and set δ to be 0.5%, the *FD platform* terminates when average mode share from cycle $C-6$ to $C-1$ deviates by less than $\pm$0.5% from cycle $C-5$ to C. When the cycle terminates at cycle C, the final itinerary T_C^* generated from the C^{th} cycle would be the final output of the whole *FD platform*. We denote the final output as T_F and the final utility score for each agent e obtained from T_F as $U_{e,F}$. They can be further analyzed to study the impact of ride-pooling services.

Error! Reference source not found. is a screenshot of the stabilized simulation result visualization. Through multiple tests, we determined that at least 30 cycles are needed to reach a relatively steady state. As a result, the entire process takes considerably longer to execute compared to only running MATSim with added extensions. We consider this extended execution time to be the main drawback of this tool.

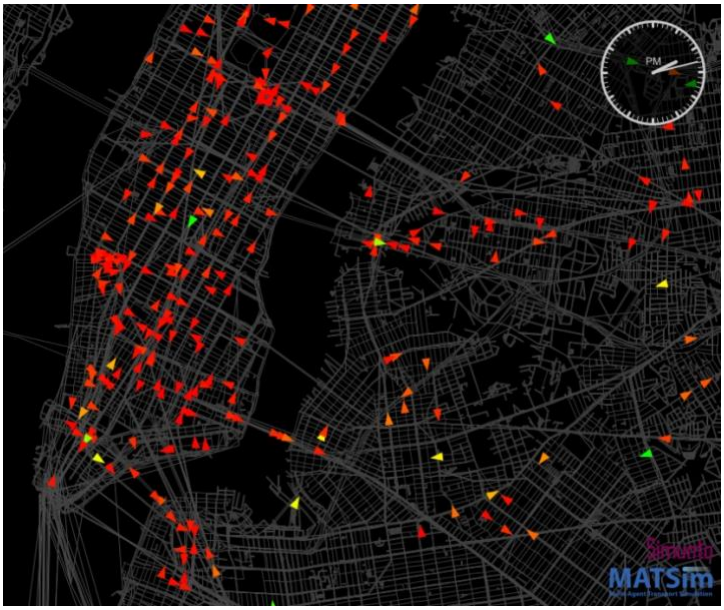


Figure 3.9. Screenshot of the stabilized simulation result.

3.6. Summary of the Algorithm

The whole *FD platform* process is summarized in Algorithm 2.

Algorithm 2: The whole *FD platform* process

Input : Initial trip plan P_c^0 and configuration file F_c.
Output: Final trip itinerary T_F.

1 Start with the cycle $c = 0$.
2 Prepare initial trip plan P_c^0 and configuration file F_c containing $s_{k,c}$, c_a, and u_a^j.
3 Prepare network file G_k and R_k for each ride-pooling service k.
4 Run **Algorithm 1** with M simulation days using P_c^0 and F_c as input and generate itinerary T_c.
5 Check each $i \in I_{k,c}$ in T_c; if $(x_{i,c}^o, y_{i,c}^o)$ and $(x_{i,c}^d, y_{i,c}^d)$ are both inside G_k, extract $(x_{i,c}^o, y_{i,c}^o)$, $(x_{i,c}^d, y_{i,c}^d)$, and $t_{i,c}^o$ to create request log $L_{k,c}$ for service k; otherwise, change the mode of i to public transit.
6 Run external simulator for service k using $L_{k,c}$, G_k, and R_k as input and obtain $L'_{k,c}$
7 For each trip i in $L'_{k,c}$, if trip $i \in I_{k,c}^+$, replace $t'_{i,c}$ in T_c by the realized trip duration $t_{i,c}$; otherwise, change the trip mode of i to public transit
8 Obtain T'_c by fully updating T_c
9 Compute $s_{k,c+1}$ using $L'_{k,c}$ and update the configuration file to be F_{c+1}
9 Set $c = c + 1$
10 **If** $c \leq n + 1$

11 Set $P_c^0 = T'_{c-1}$, return to step 3

12 **Else**

13 Compute the absolute different of the n-cycle average ride-pooling mode share between cycle $c-1$ and cycle c: $abs(\frac{\sum_{C-n}^{C} p_{K,c}}{n} - \frac{\sum_{C-1-n}^{C-1} p_{K,c}}{n})$

14 If the result is larger than δ, Set $P_c^0 = T'_{c-1}$, return to step 3; Otherwise, the algorithm terminates and use T'_{c-1} as the final output

4. Experiments/Evaluation Using the FD Simulator

In this section, we present various case studies based on the modified MATSim-NYC file. The section is organized as follows: Subsection 4.1 describes the external simulator involved in the *FD platform*; Subsection 4.2 outlines the general settings of the defined ride-pooling service; Subsection 4.3 presents a case study comparing the results produced by the *FD platform* and the "DRT" extension; Subsection 4.4 features a case study involving two heterogeneous ride-pooling services; and Subsection 3.5 showcases an equity study based on the *FD platform* output.

4.1. External Ride-pooling Simulator

In this project, the *FD platform* uses an external ride-pooling simulator jointly developed by NYU and Volkswagen/MOIA team. MOIA is a German ride-pooling service provider under the umbrella of the Volkswagen Group, whose goal is to provide sustainable and affordable mobility services. The simulator represents MOIA's experiment to explore innovative algorithms for more efficient ride-pooling service. Table 4.1 lists the key adjustable parameters R_k included in the simulator. With such a wide range of parameters, a high degree of flexibility in service strategies can be achieved, allowing for broader exploration of service approaches.

Table 4.1. Summary table of local simulator's adjustable parameters

PARAMETER	DESCRIPTION
VEHICLE NUMBER	Number of Vehicles in a service fleet
CAPACITY	Vehicle capacity
MAX_T_DISTANCE	Maximum travel distance in terms of seconds for finding candidate vehicles
MAX_WAIT_T	Maximum waiting time for passengers to be picked up
MAX_TT_P	Maximum added travel time proportional to
MAX_TT_ADDED	Maximum travel time added to the direct travel time
BASE_DWELL_TIME	Dwell time at each stop
THETA	Weight of vehicle travel time in the cost function

4.2. Ride-pooling Service Setting

In real-world scenarios, ride-pooling services typically operate within a defined service area to minimize costs while maximizing service quality. In our case studies, the network encompassing Manhattan, Queens, and Brooklyn serves as the overall simulation environment, with a portion of lower Manhattan designated as the ride-pooling service region for two ride-pooling service fleets, “drt_1” and “drt_2”. Figure 4.1 illustrates the whole network with the highlighted area being the ride-pooling service region. The subnetwork for “drt_1” (G_{drt_1}) is identical to the subnetwork for “drt_2” (G_{drt_2}). They both consist of 3,421 nodes and 6,325 directed links. We define all 3,421 nodes as service stops, indicating that vehicles can pick up passengers at these nodes. Only agents located within a 1,500-foot radius of the service stops are considered accessible users. Both fleets consist of nine identical vehicles with a capacity of six passengers. In the following subsections, we adjust the service strategies R_{drt_1} and R_{drt_2} employed by the two fleets to explore how users respond to service changes.

Figure 4.1. Whole simulation network and defined ride-pooling service region.

4.3. FD Platform vs. "DRT" Extension

The "DRT" module (Bischoff et al. 2016) is an official MATSim extension designed for demand-responsive transport, including certain ride-pooling services. By integrating it into MATSim, the extension adds specific MOD service entities such as ride-pooling vehicles to the simulation system and generates corresponding travel itineraries. Table 4.2 compares the extension's parameters with those of the external simulator. The adjustable parameters in the extension are a subset of those in the external simulator, which means that we can use the *FD platform* to emulate the "DRT" extension's functionality by aligning their operational settings.

In this setting, we define "drt_1" and "drt_2" using identical operational strategies. Therefore, R_{drt_1} is identical to R_{drt_2}. The corresponding values of parameters are shown in the third column of Table 4.2. The "max_t_distance" is expected to be the same as maximum wait time by default. By setting "theta" to 1, the local simulator optimizes service operations by solely minimizing total vehicle travel time, which is the same optimization method utilized by the "DRT" extension. Aligning the results generated by the two systems would demonstrate the *FD platform*'s ability of producing reliable results, thereby proving its credibility for future use.

Table 4.2. Parameter comparison between the FD simulator and "DRT" extension

Local Simulator Parameter	Corresponding "DRT" Parameter	Value
Vehicle Number	**VEHICLE NUMBER***	9 per fleet
Capacity	**CAPACITY***	6 passengers
max_t_distance	**N/A****	900 seconds
max_wait_t	**maxWaitTime**	900 seconds
max_tt_p	**maxTravelTimeAlpha**	0.4***
max_tt_added	**maxTravelTimeBeta**	1200 seconds
base_dwell_time	**stopDuration**	60 seconds
theta	**N/A****	1

* Vehicle number and capacity are defined in the vehicle file in MATSim
** N/A means not tunable in the "DRT" extension.
*** MAX_TT_P = 0.4; maxTravelTimeAlpha = max_tt_p + 1 = 1.4

In the test case, we reduce the MATSim-NYC population size to include 5,027 agents for shorter simulation times and easier evaluations. In the initial trip plan, 932 trips legs are assigned to use ride-pooling services. The number of ride-pooling trips will change gradually when running the simulation. We set the stopping criteria as the consecutive 5-cycle averages of total number of ride-pooling trip legs showing less than 1% fluctuation for the *FD platform*. As a result, we conducted 50 runs of the *FD platform* cycle. For the "DRT" enabled MATSim run, there is not stopping criteria setup in its configuration. Therefore, we executed the same number of iterations of the "DRT" enabled MATSim run. As mentioned in section 3.1, multiple days of MATSim simulation are necessary within each *FD platform* cycle to generate sufficient number of perceived trip plans, which subsequently ensuring sufficient mode changes across cycles. In this case study, 5 days of simulation were executed within each MATSim run inside one cycle.

Figure 4.2 displays the trend of the total number of ride-pooling service trips generated by both systems. There is a sudden drop of number of ride-pooling trips after 40th "DRT" enabled MATSim iteration. This phenomenon occurs because among the 50 iterations, only the first 80% iterations are set to generate new trip plans and the last 20% iterations are set to only select the trip plans maximizing agents' utility score in the previously generated trip plans. In other words, trip plans generated from the first 40 iterations are not strictly following the score maximization rule while the last 10 iterations do. A roughly 10% difference between the two results in terms of number of ride-pooling trips is observed once they stabilize. In terms of total trip hours, the FD simulator generates 236 hours of ride-pooling service trips, while the "DRT" extension generates 233 MOD trip hours, resulting in a better match. We believe that the differences observed in the test case are tolerable in simulation, and the alignment of results between the two systems provides a high level of confidence for the subsequent case studies.

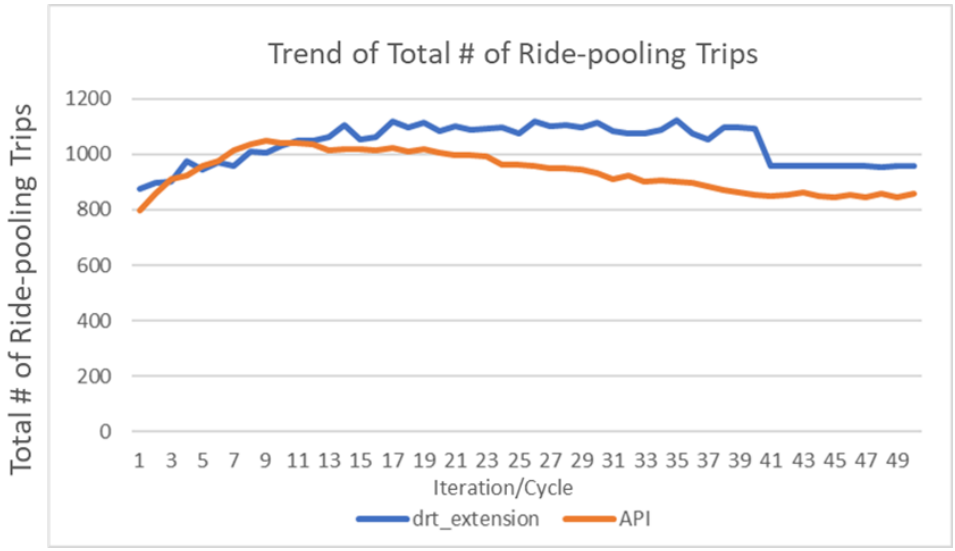


Figure 4.2. Trend of total number of MOD services generated from the FD simulator and "DRT" extension.

4.4. Heterogeneous Ride-pooling Services

As shown in Table 4.1, the parameter "theta" determines how the operator optimizes its operation. A theta value of 1 means minimizing only the vehicle travel time, while a value less than 1 indicates that the system takes passenger wait time into account in the objective function by the weight of 1 minus theta. Consequently, setting a lower theta level would enhance the user experience by reducing their wait time. In other words, the service would appear "faster" from the users' perspective. In the following case study, we define a ride-pooling fleet with a theta value of 0.5 as providing a "fast" service. In contrast, the fleet offers a "slow" service when its theta value is 1. After testing both values of theta, the average trip time offered by "fast" service is 30% shorter than what the "slow" service can provide.

In this scenario, there is a flat cost imposed on using both "drt_1" and "drt_2", while the cost of using "drt_1" is 20% higher than that of "drt_2". We evenly assigned the ride-pooling trips $I^0_{drt\_1,0}$ and $I^0_{drt\_2,0}$ in the initial trip plan P^0_0 described in subsection 3.3. The disparity between the constant value C_{drt_1} and C_{drt_2} defined in the configuration file F_0 reflects such cost difference, with C_{drt_2} posing smaller negative value to the overall utility score. All other utility values u^j_a for both services are identical. The initial teleportation speed $s_{drt\_1,0}$ and $s_{drt\_2,0}$ are defined to be the same.

We create three scenarios and run them with the same P_0^0 to examine how heterogeneous services influence user's mode choices. Within each cycle, two separated external simulator runs are involved. Each external simulator run is dedicated for the trips served by each ride-pooling fleet. The cycle terminates when the absolute difference between μ_{C-1}^5 and μ_C^5 is less than 1%. $\mu_{C-1}^5$and μ_C^5 are calculated using Eq. (3.3). In all three scenarios, the cycle terminates when C is equal to 50.

Error! Reference source not found. displays the number of ride-pooling trips generated from the three scenarios. In scenario 1, both ride-pooling fleets set their theta values to 0.5, offering "fast" service to customers in the region. The ridership of "drt_2" is 25% higher than "drt_1", demonstrating a strong user preference for lower-cost service. However, the ridership gap nearly disappears when both fleets provide "slow" service in scenario 2, revealing that the speed of the ride-pooling service has a more substantial effect on user's travel choices. This effect is further evidenced in scenario 3, where "drt_1" provides "fast" service while the service speed of "drt_2" is "slow". Higher ridership is still observed even though customers are facing increased trip costs.

The three studies scenarios demonstrate the capability of the *FD platform* in properly reflecting users' preferences based on their own valuation of trip attributes when facing heterogenous services. In such way, the *FD platform* can provide insights related to the competition among different services and how travelers respond to potential service changes.

4.5. Equity analysis

Social welfare has long been a crucial aspect of transportation design. Often, transportation services cater to more privileged population segments, benefiting only a few and resulting in less equitable service. We are interested in how the hypothetical ride-pooling service included in our study would impact social equity between low-income groups and others. In the following equity study, we randomly select 35,000 agents from the original MATSim-NYC population and divide them into two subpopulations based on their income level: "Low_Income" and "Not_Low_Income". We define 20% of the population as "Low_Income". The value-of-time (VOT) is calibrated by fitting a discrete choice model developed by Ren and Chow (2022) using Manhattan trips provided by Replica. The VOT for "Not_Low_Income" is $9.98/hr, while for "Low_Income", it is $5.68/hr. To study the impact of service changes on user equity, we create six service scenarios summarized in Table 4.3. In each scenario, the same operation strategy is applied to both fleets. Thus, we do not consider service differentiation. To measure equity

differences between the two subpopulations, we employ the formulation presented in Karsu and Morton (2015) as follows:

$$Inequity\ Score = \sum_{i \in I} \sum_{j \in J} |y_i - y_j| \quad (4.1)$$

In Eq (4.1), the value of y represents the utility score obtained from the MATSim output. I and J correspond to the two subpopulations. As described in subsection 3.5, utility scores $U_{e,F}$ can be directly obtained from T_F. We denote the inequality score as ISQ. Therefore, Eq (4.1) can be written as:

$$ISQ = \sum_{i \in Low\_Income} \sum_{j \in Not\_Low\_Income} |U_{i,F} - U_{j,F}| \quad (4.1)$$

A lower ISQ indicates a higher level of equity achieved by the system.

Table 4.3. Service scenarios for equity study

$5 flat fee, fast service	$7 flat fee, fast service	$10 flat fee, fast service
$5 flat fee, slow service	$7 flat fee, slow service	$10 flat fee, slow service

In the initial trip plan, 1316 ride-pooling trips are evenly assigned between the two fleets. The same stopping criteria as the one defined in subsection 4.4 is applied. As a result, we run 50 cycles for each service scenario. The itinerary outputs generated from the last 10 cycles are used to calculate the inequity score and its distribution. The users included in the inequity score calculation are those with trips that both start and end within the ride-pooling service area regardless of the trip mode. Consequently, we only consider users that can be potentially impacted by the MOD service.

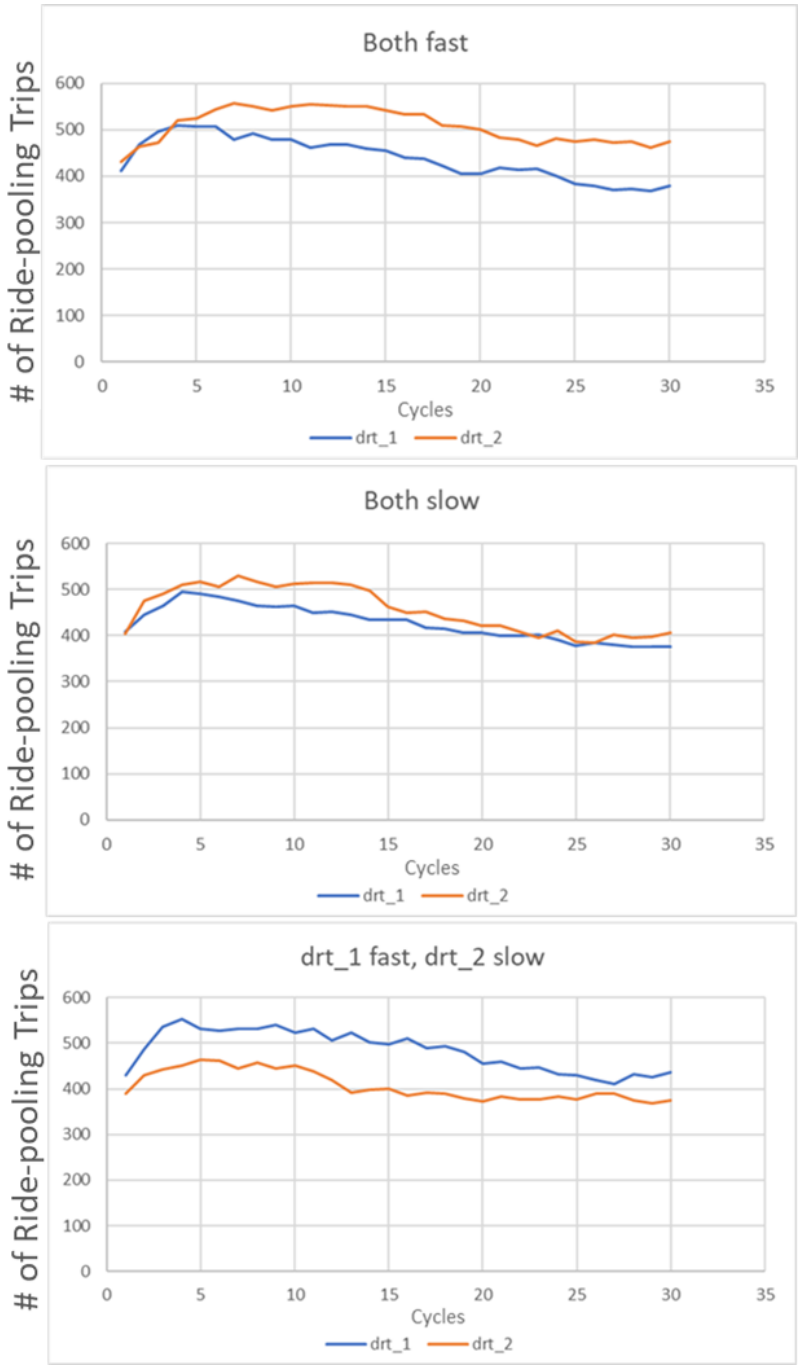


Figure 4.3. Trend of total number of ride-pooling services when (1) both fleets provide "fast" service, (2) both fleets provide "slow" service, (3) "drt_1" provides "fast" service, "drt_2" provides "slow" service.

Figure 4.4 is a box plot illustrating the normalized inequality score distributions. The overall level of inequity score does not change drastically when facing service changes, indicating the limited impact on travel equality among the general public that the ride-pooling service could impose. This is because only a small subset of users actually uses the ride-pooling service. To further investigate how ride-pooling services impact actual users, we select only the actual ride-pooling service users to compute the inequality score. The results are illustrated in Figure 4.5. The impact of ride-pooling service changes on inequity score is much more noticeable when only considering the actual ride-pooling users. Faster service speed and lower service price both result in higher levels of inequality. In an area where higher-income users are the dominant population, the addition of more convenient services could disproportionately benefit the majority. In our case, the "Low_Income" subpopulation only constitutes 20% of the population, making a cheaper and faster service favors the higher-income subpopulation more.

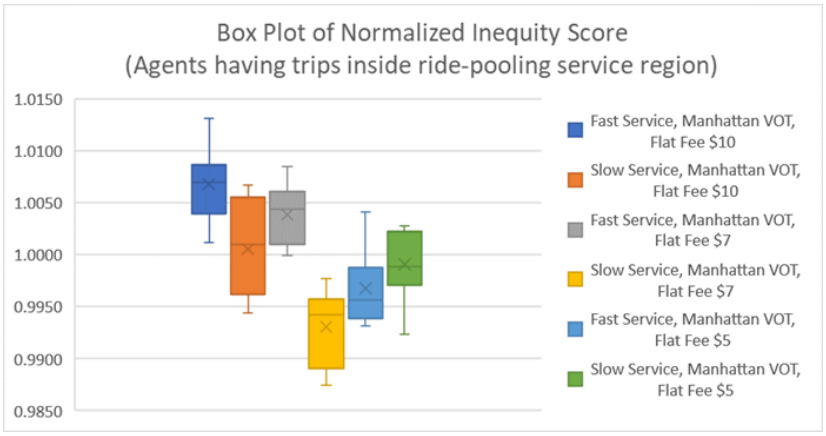


Figure 4.4. Normalized inequality score among all potential users.

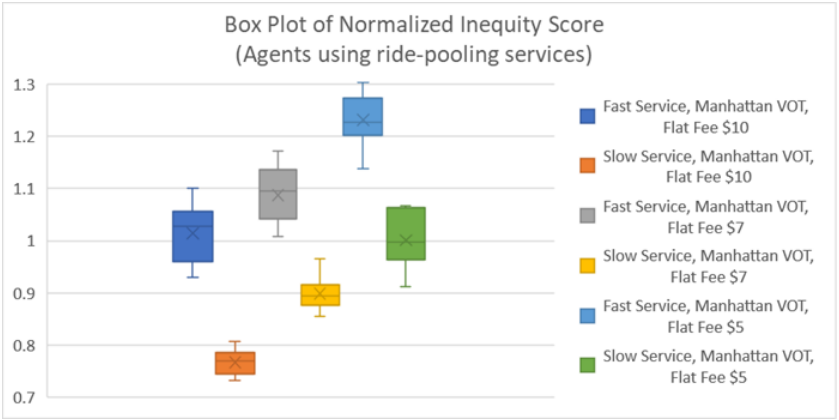


Figure 4.5. Normalized inequality score among actual MOD trip users.

Even though a faster and cheaper ride-pooling service could increase the level of transport inequality, the absolute benefit brought to both subpopulations should need to be evaluated. Two subpopulations not being benefitted equally does not mean that the service does not benefit the “Low_Income” subpopulation at all. Therefore, we calculate the average utility scores among the two subpopulations that use the service. The higher the score indicates the more benefit brought to the users. Figure 4.6 shows the normalized results. In general, a faster and cheaper service benefits both subpopulations. In fact, the relative benefit brought to “Low_Income” subpopulation by switching from high-cost service to low-cost service while remaining fast service speed is more significant, indicating the importance of a cheaper and faster transportation service for everybody.

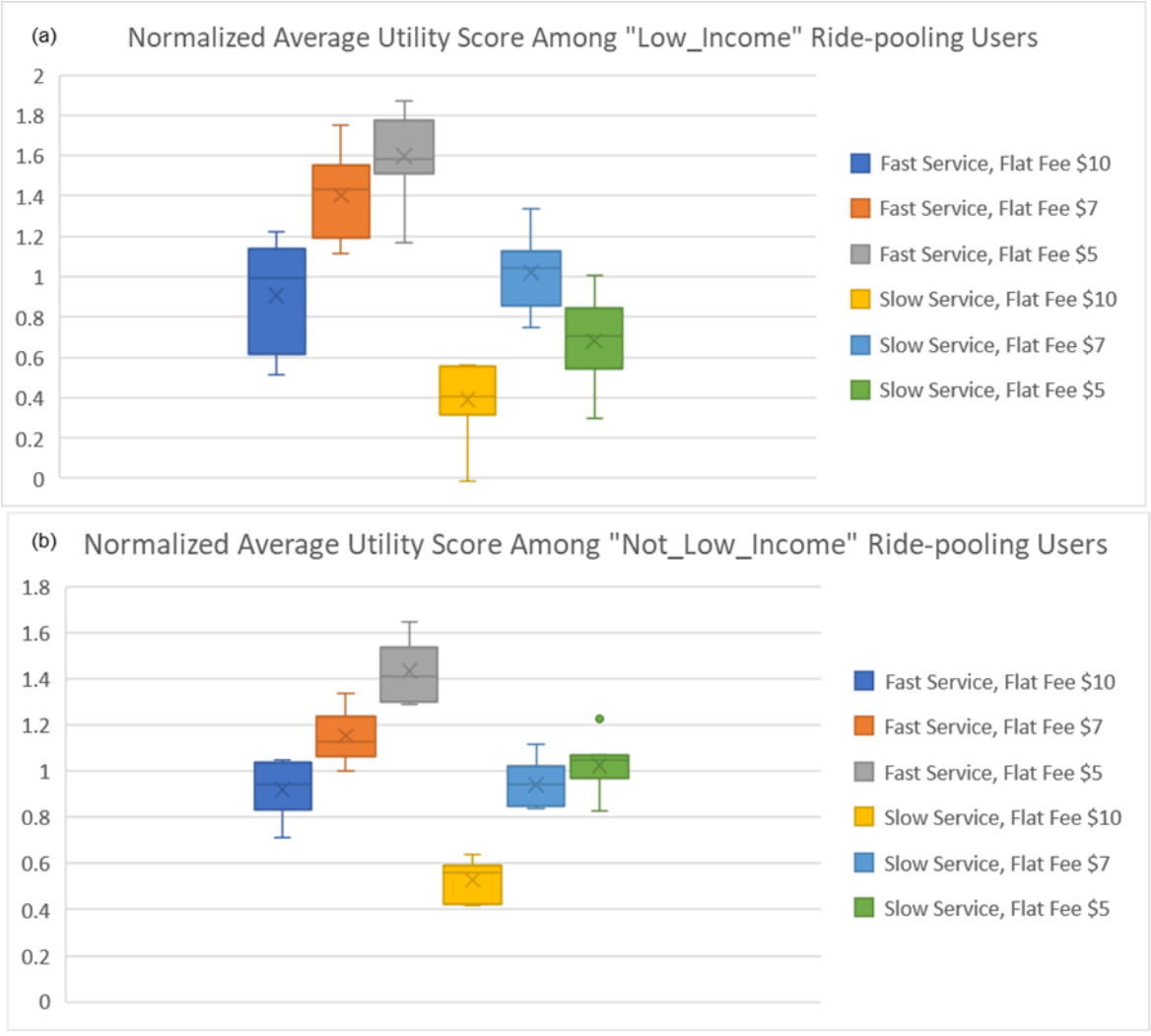


Figure 4.6. Normalized average utility score among a. “Low_Income” b. “Not_Low_Income”.

5. Experiments/Evaluation Using the VTD Simulator

The detailed structure of the VTD simulator can be found in Ban et al. (2022), and we omit the details here. It is worthwhile noticing that the key of developing the integrated multiscale VTD simulator is to properly handle data transmissions and communications between different scales, noticeably between the vehicle scale (Unity) and the traffic scale (SUMO), and between the traffic scale (SUMO) and the demand scale (MatSim). During this process, inconsistencies may arise, as illustrated earlier in Section 2, which need to be handled properly. In this section, we present two applications based on the VTD simulator. The first one is to test and evaluate the multiscale traffic-signal control problem (Guo and Ban, 2023) and the second one is to investigate the macroscopic traffic dynamics for downtown Seattle based on MFD.

5.1. Multiscale traffic-vehicle control

Urban transportation system is multi-modal and multi-scale in both spatial and temporal domains. Even when we limit ourselves to the control of vehicular traffic, the multi-scale nature of urban traffic control (UTC) is evident as it contains traffic control at multiple levels (scales): from vehicle control to intersection or freeway segment control, corridor and sub-network (regional) level control, and to network (global) control. Traffic at each scale has its own behavior (i.e., ``dynamic'') while interacting with traffic at other scales, especially the "neighboring" scales (e.g., intersection control dynamics interact with the dynamics at the vehicle scale and corridor scale more than those at other scales). Furthermore, each scale has specific temporal and spatial contexts when control is concerned. Using the temporal scale as an example: vehicle control often occurs in sub-seconds or even milliseconds when safety-critical applications are considered (e.g., collision avoidance), intersection control runs in seconds, and corridor control can run in minutes, while sub-network and regional control often happen in even larger time intervals. Recently, a general multiscale modeling and control framework is developed for UTC (Guo and Ban, 2023), which is applied to the two-scale coupled traffic-vehicle control problem. We omit the modeling details in this report and focus on detailed model evaluation using the VTD simulator.

The evaluation is conducted using a real-world intersection at Fairview Avenue and Denny Way (one of the major signalized intersections in downtown Seattle), modeled in the VTD simulator. The intersection is shown in Figure 5.1. We made some modifications to the geometry of the intersection to fit the multiscale model. After the revision, the intersection becomes a typical

four-leg intersection with four approaches (each with three incoming lanes including a dedicated left-turn lane) and twelve movements. There are eight candidate phases for the studied intersection, which is shown in Figure 5.2 (Roess et al., 2004). As an illustration, “NS” represents the “Northbound & Southbound Straight” phase.

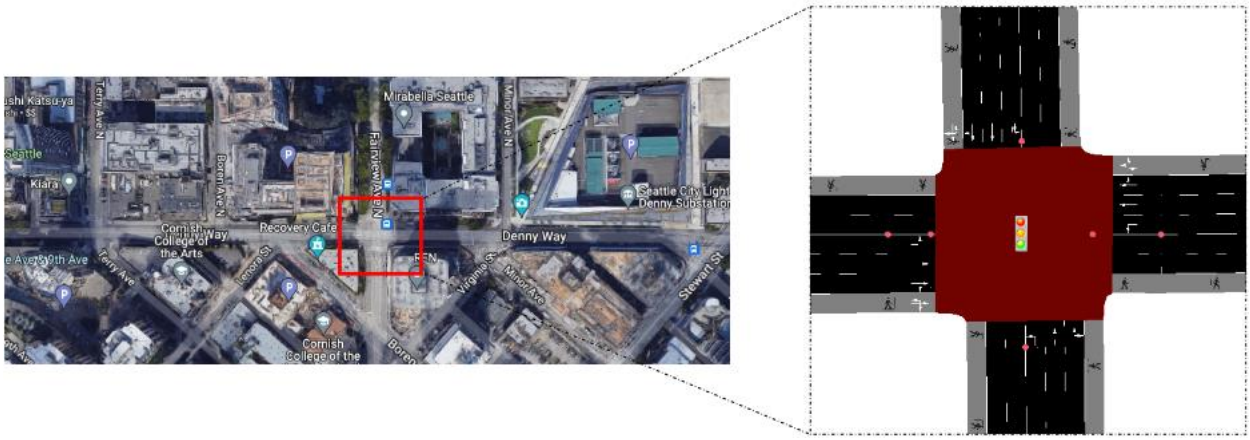


Figure 5.1. The intersection at Fairview Avenue and Denny Way, Seattle.

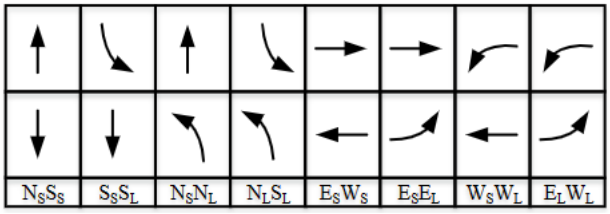


Figure 5.2. The candidate signal phases.

We compare the proposed multi-scale method with the traditional actuated signal control method. Actuated signal control relies on loop detectors to obtain information of surrounding vehicles, and extend the green time if there are incoming vehicles under the current phase until the green time reaches the maximum green time or an arrival gap is detected. It can adjust the green time for each phase based on real-time traffic conditions using simple logic, making it the most popular method in real-world advanced traffic signal control. Hereafter in this paper, the actuated signal control and the proposed method are denoted as Actuated and Multiscale, respectively.

For each leg around the intersection, the real volumes are around 250 veh/hour for normal hours and 400 veh/hour for peak hours. We consider these values as the medium volume scenario and

vary the values to create the low- and high-volume scenarios (which are shown in Table 5.1). In addition, we create asymmetric volume scenarios by increasing the eastbound and northbound volumes and reducing the westbound and southbound volumes to test the performance under unbalanced volumes. The simulation period is set to 30 minutes. In order to capture dynamic volumes, the volumes increase from minimum to maximum linearly through the simulation period.

Table 5.1. Volume settings for the experiments

	symmetric						asymmetric					
	low		medium		high		low		medium		high	
	min	max	min	max	min	max	min	max	min	max	min	max
WE straight	100	200	250	400	400	600	250	400	400	600	550	800
WE left/right turns	33	67	83	133	133	200	83	133	133	200	183	267
SN straight	100	200	250	400	400	600	250	400	400	600	550	800
SN left/right turns	33	67	83	133	133	200	83	133	133	200	183	267
EW straight	100	200	250	400	400	600	125	200	200	300	275	400
EW left/right turns	33	67	83	133	133	200	41	67	67	100	91	133
NS straight	100	200	250	400	400	600	125	200	200	300	275	400
NS left/right turns	33	67	83	133	133	200	41	67	67	100	91	133

Table 5.2 shows the performance of the two control methods. The time loss of a vehicle is calculated as the time lost due to traveling at a speed below the maximum speed. For the low volume scenario, compared with the Actuated method, the Multiscale method can improve the performance considering the fuel consumption, waiting time, time loss, and queue length. For the medium volume scenario, the level of performance improvement of the Multiscale method is similar to that of the low volume scenario, but with a small increment for all metrics. For the high-volume scenario, the performance improvement becomes even higher. For example, the fuel-saving performance improves by 35.22% and 45.31%, respectively, by two fuel consumption calculation methods (one using the internal fuel consumption model in SUMO and one fitted using real data). The same pattern can be observed in the table for waiting time, time loss, and queue length. Similar trends can also be found under the asymmetric volume scenarios from Table 5.2; we omit the detailed discussions for brevity. In summary, compared with the Actuated method, the Multiscale method can reduce fuel consumption, waiting time, time loss, and queue

length under all volume scenarios tested in the table, and the reductions increase as the volume increases.

Table 5.2. Experiment results

Volume type	Metrics	Symmetric volumes			Asymmetric volumes		
		Actuated	Multiscale		Actuated	Multiscale	
			value	change		value	change
low	Avg. fuel I (mg/m)	0.11	0.08	-27.27%	0.11	0.08	-27.27%
	Avg. fuel II (mg/m)	0.11	0.09	-18.18%	0.12	0.09	-25.00%
	Avg. waiting time (s)	12.02	6.15	-48.84%	13.48	6.65	-50.67%
	Avg. time loss (s)	23.58	11.85	-49.75%	26.02	12.84	-50.65%
	Avg. queue length [m]	6.47	5.24	-19.01%	8.21	5.87	-28.50%
medium	Avg. fuel I (mg/m)	0.11	0.08	-27.27%	0.11	0.08	-27.27%
	Avg. fuel II (mg/m)	0.12	0.09	-25.00%	0.13	0.10	-22.71%
	Avg. waiting time (s)	16.31	7.89	-51.62%	17.44	8.50	-51.26%
	Avg. time loss (s)	29.57	15.02	-49.21%	31.54	15.9	-49.59%
	Avg. queue length [m]	10.19	6.63	-34.94%	12.30	7.83	-36.34%
high	Avg. fuel I (mg/m)	0.13	0.09	-35.22%	0.13	0.09	-30.77%
	Avg. fuel II (mg/m)	0.18	0.10	-45.31%	0.17	0.10	-40.89%
	Avg. waiting time (s)	41.31	10.89	-73.64%	33.96	11.06	-67.43%
	Avg. time loss (s)	28.18	19.22	-31.80%	50.96	19.62	-61.50%
	Avg. queue length [m]	28.19	9.81	-65.20%	26.56	10.43	-60.73%

In order to better understand the performance of different control methods, we show the aggregated trajectories of all WE straight vehicles under the high symmetric volume in Figure 5.3. For each vehicle, the time when it enters the simulation is set as zero. We plot every vehicle's trajectory in the same figure and refer it as the "aggregated trajectories" which can help clearly show the overall performance of all vehicles. The figure shows that the Multiscale method can largely reduce the delay and the maximum queue length of WE straight vehicles. In fact, we can observe similar trends for the vehicles of other movements, which are omitted here for brevity. Compared with the Actuated method, the Multiscale method can better balance the signal phases and timings for different movements, thus reducing the travel times and queue lengths. Another observation is that, although we did not explicitly include any vehicle platoon model in the Multiscale method, there are platoon patterns shown in the results. To see this, we plot every single trajectory of the WE straight vehicles during the first 500 seconds under the Multiscale control in Figure 5.4. We can see that, from time to time, vehicles did arrive in platoons, e.g.,

around t=150, 200, 250, 300, etc. The Multiscale method designs signal timing in such a way that vehicles in the same platoon often pass the intersection (and wait in the same queue if needed) during the same green time. This implies that the method can somehow capture the arrival patterns of the vehicles, and design signal timing to maintain the arrival platoons when vehicles passing the intersection.

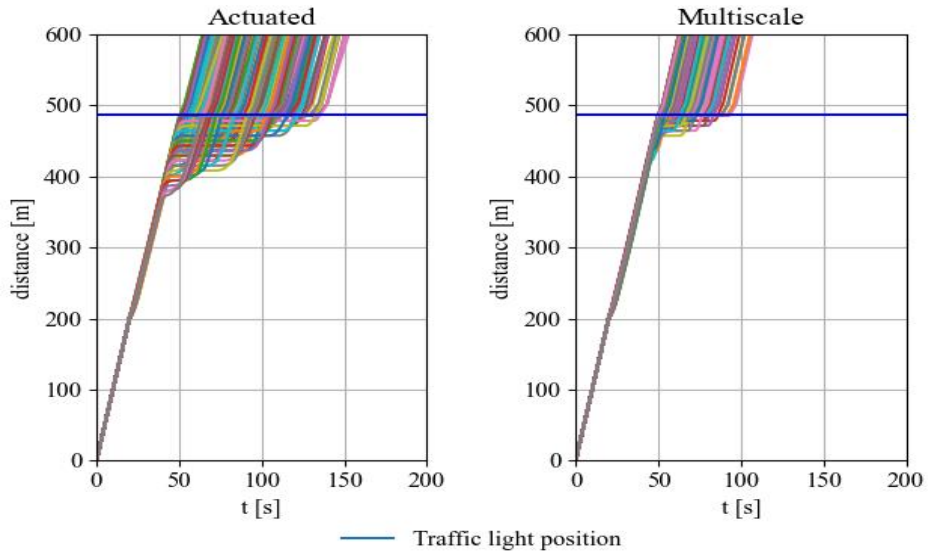


Figure 5.3. Trajectories of all WE straight vehicles for different control methods.

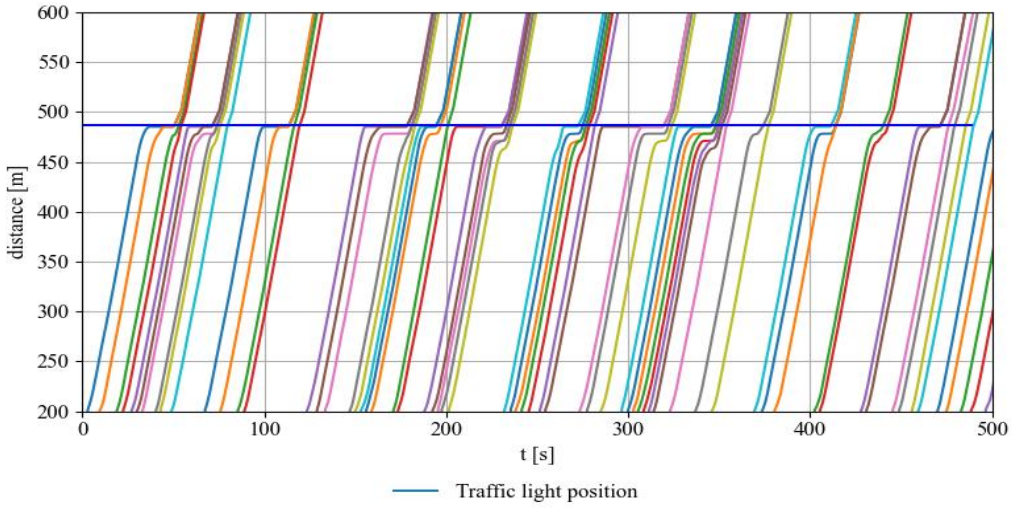


Figure 5.4. Trajectories of all WE straight vehicles for different control methods.

5.2. Partition and MFD calibration for Downtown Seattle

Real-world urban traffic control needs to be done for a city-scale network in order to increase the efficiency of the whole network. However, traditional methods involve partitioning the network into mostly-corridor-level independent subnetworks and then controlling each subnetwork separately. While optimizing a city-scale network, dividing the entire network into such subnetworks is almost impossible. Concepts such as the macroscopic fundamental diagram (MFD; see Geroliminis and Daganzo (2008)) may provide new insights of research for network optimization.

In this regard, perimeter control methods have been proposed by first partitioning a city network into homogeneous regions, fitting an MFD to each region, and then optimizing the flow in/out of each region so that the total number of vehicles in a region is kept below some critical value (Geroliminis et al, 2013). There is an extensive literature for concept, methods, and algorithms for MFD-based perimeter control. One of the most recent works is proposed by Guo and Ban (2020). Their method considers traffic dynamics and intersections of two neighboring regions at the boundaries (i.e., the buffer zones) via the point-queue model. The buffer zone and point-queue model can better capture congestion and interactions in the boundary of two regions.

While most studies were tested on hypothetical networks, this study aims to divide a real-world city network into regions and calibrate the MFD curve of each region. First, we partition the Downtown Seattle traffic network into subnetworks (regions) by applying an improved partitioning algorithm initially proposed by Ji and Geroliminis (2012). The network and data are from the VTD simulator (Ban et al., 2022). An MFD is then trained and examined for each region, using one-minute simulation outputs, to calibrate and validate the homogeneity of the partitioned regions. Figure 5.5 shows a flowchart of these processes.

The input data for the proposed work is the simulation of all links in Downtown Seattle network. The network consists of 7374 unique street edges in total consisting of 208 edges of highways (I5 and SR 99 tunnel) and 3120 edges of minor streets which are either alleys or parking entrances. The range of the network is north to Mercer Street, south to South Atlantic St/Edgar Martinez Dr St, West to Alaskan Way, and East to 1^{2t}h Ave. The data was generated in traffic simulation using SUMO (Ban et al., 2022) for 5AM to 10AM period, and included speed, flow, and density information for each edge. We ran one simulation and aggregated the output on 5 consecutive one-hour intervals. We also ran an identical simulation and aggregated the outputs on every

minute of simulation. We used the output generated for an hour as data for the partitioning, while the output generated for each minute was used as data to plot the MFD of the region.

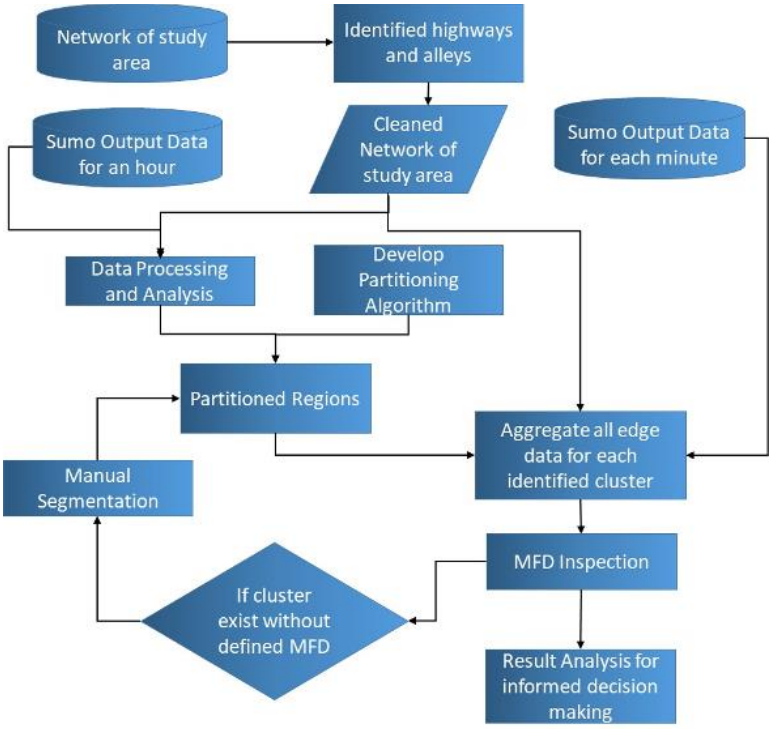


Figure 5.5. Overview of the methodology framework.

The main network used in the simulation consists of various types of roads, including highways, arterials and main streets, and minor streets (alleys or parking entrances). The network used in the algorithm, however, only consisted of main streets. The cleaned network included 4,100 edges, excluding major highways and minor streets, without compromising connectivity (Figure 5.6). These roads were removed from the algorithm network as they have different rules and dynamics and do not require traffic control strategies. Alleys and highways could also act as outliers in our analysis and adversely affect the homogeneity of resulting regions. Additionally, it is shown in the past that a well-defined MFD does not exist on freeway systems (Geroliminis and Jie, 2011). Yet, for implementation of the perimeter control strategies, we might need to include highway segments as one region since the flow between highways and downtown clusters is not negligible.

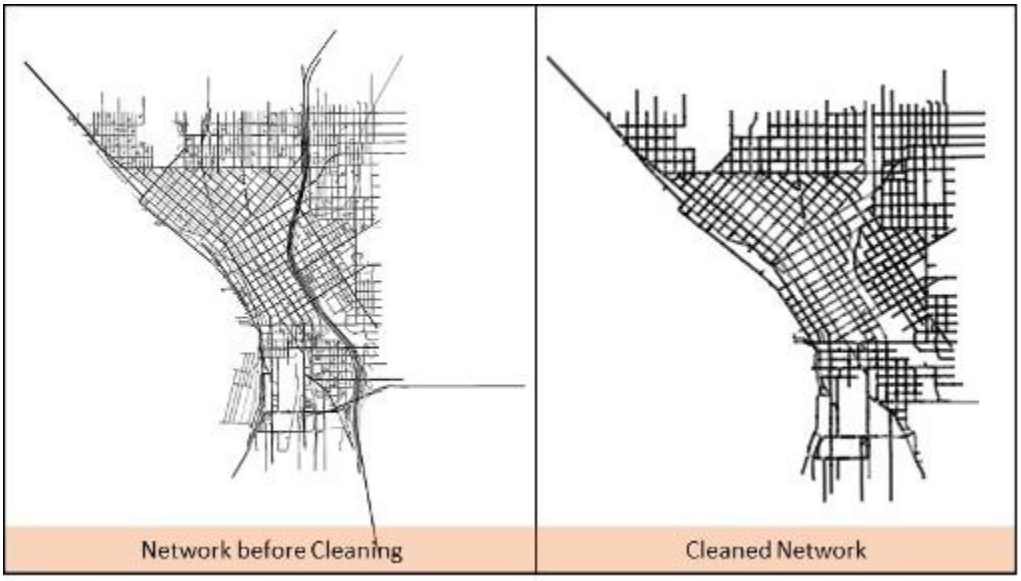


Figure 5.6. The network before and after removing unrequired edges.

Region partitioning is implemented mainly based on the method proposed by Ji and Geroliminis (2012). They developed a heuristic method with three consecutive algorithms: (1) initial segmentation, (2) merging, and (3) boundary adjustment. In the initial segmentation, the Normalized Cut algorithm (nCut) is used to divide a region into smaller regions by turning the entire network into a graph and solving a generalized eigenvalue system. This process occurs several times to produce multiple regions. Each time, the parent region is chosen based on a function of variance of densities and the number of links in the region, and the one with the highest value is chosen each time. To impose connectivity, a similarity matrix W is defined in which for any connected links *i* and j, $W_{ij} = exp(\frac{-(d_i - d_j)^2}{\sigma_d{}^2})$, where d stands for the set of densities and σ_d is the standard deviation of the densities. The merging algorithm reduces unnecessary cuts in the nCut algorithm by combining two neighbor regions with the least difference in mean densities. We also need to consider the number of boundary links so that the new region is more likely to form a compact shape. In our implementation, the neighbors with less than 8 links in their boundary are not considered as neighbors. Until this point, the boundary adjustment algorithm has not been explored yet in this work, which ensures the boundaries of each two cluster belong to the best region.

Moreover, some metrics need to be calculated each time when either the initial segmentation or the merging algorithm is running to help us assess the homogeneity of the result and find the

optimal number of clusters. This helps find some valid candidates for the MFD investigation step. These metrics are the average of NcutSillhutte (average NS) (Ji and Geroliminis, 2012), Normalized total variance TV_N (Saeedmanesh and Geroliminis, 2016), and coefficient of variations (COV) of densities in each region. The smaller these metrics are, the better the result of the clustering is.

After partitioning, we fit an MFD to each region. An ideal output of a good partitioning would be well-defined MFDs for all regions with different shapes for each two neighbor regions. In this report, to show the calibrated MFDs, the plot is shown between the distance-weighted average flow and the distance-weighted average density. By investigating the MFD and homogeneity indices of each region, we aim to find a set of clusters with well-defined MFDs. To achieve this goal, we first perform the partitioning on the cleaned network. The analysis is done on the data for 8AM-9AM. This is the interval with most traffic jams in the morning. Figure 5.7 (a) depicts the network with darker links showing higher average lane density. Plots b, c, and d from Figure 5.7 are the output of the initial partitioning after 1, 8, and 11 times of cutting, respectively. Furthermore, Figure 5.7.I), 5.7(f), and 5.7(g) show the resulting regions after 3, 6, and 8 times of performing the merging algorithm after 11 times of performing initial partitioning. Table 5.3 shows homogeneity metrics for each step of cutting and
Table 5.4 shows the same metrics for merging.

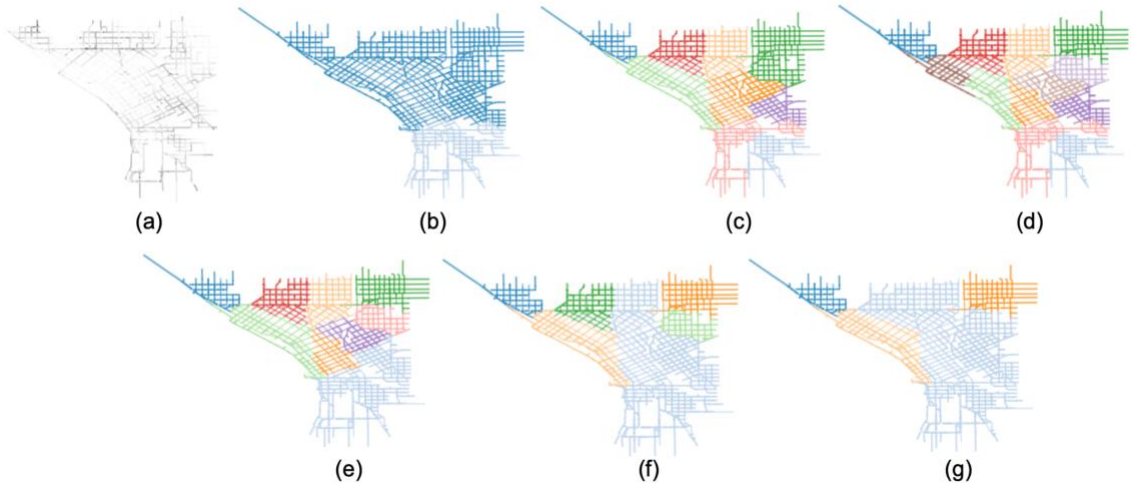


Figure 5.7. Results of partitioning algorithm in different stages.

Table 5.3. Homogeneity metrics for 11 times for running initial partitioning algorithm

Ncut step	1	2	3	4	5	6	7	8	9	10	11
NS	0.988	1.051	1.077	1.0958	1.1056	1.1218	1.0973	1.1412	1.0926	1.1188	1.1055
TVn	0.995	0.995	0.995	0.994	0.987	0.96	0.96	0.958	0.948	0.948	0.948
average COV	1.291	1.321	1.3	1.307	1.312	1.32	1.309	1.293	1.319	1.357	1.342

Table 5.4. Homogeneity metrics for 11 times for running merging algorithm

Merging step	1	2	3	4	5	6	7	8	9	10	11
NS	1.1218	1.111	1.0769	1.0975	1.0633	1.011	0.999	0.9266	1.0346	0.974	-
TVn	0.948	0.948	0.948	0.948	0.949	0.953	0.953	0.957	0.971	0.995	1
average COV	1.353	1.369	1.329	1.348	1.368	1.394	1.447	1.526	1.428	1.32	1.324

Now, based on Table 5.3,

Table 5.4, and our judgment, we choose some candidates to check their MFDs and find the best combination of cuts and merges. In this regard, four candidates were chosen, each candidate is made of 11 times of initial partitioning and then performing the merging algorithm for 3, 6, 7, and 8 times. Our investigation of MFDs showed that the partitioning after 11 times of running Ncut algorithm and 8 times of merging algorithm, the result with lowest NS value, shows us the most well-defined MFDs out of all candidates. shows the MFDs of our selected partitioning.

There are some factors that we need to consider in our further attempts for finding further improved partitioning. One is that some MFDs might not have a peak since our simulation is only for 5AM to 10AM and the regions with no peak might not reach their maximum flow in this time frame. In figure 8(b), two regions have peaks densities that are above 40 veh/km, while in the other two regions, the highest density is only around 20 veh/km.

Second, there are several ways of comparing density (or number of vehicles) and flow, and other types of MFD plots are worth trying. Thirdly, there are a number of links in the margins of our network which usually have None values for density in the simulation result. This is due to the fact that these links are not connected to any defined TAZs. Removing them in our analysis might help us obtain better MFDs.

Lastly, the algorithm used in this work is heuristic and creating best regions with least heterogeneity is not guaranteed. Therefore, changes in the algorithm are needed in order to find better partitioning. First of all, the effect of the boundary adjustment algorithm is yet to be explored. Moreover, one change could be interactively cutting and merging regions based on our point of view and find regions with lower heterogeneity. This helps design better criteria for finding the proper region to cut in each initial partitioning step, or finding the best two neighboring regions for the merging algorithm. Another way of improving the method is to iteratively cut each region for each step of the cutting algorithm or merge each two neighboring regions in each step of the merging algorithm, and then simply select the best iteration for that step.

After obtaining the partitioned regions and well-calibrated MFD for each region, the next step is to apply the MFD-based perimeter control proposed in Guo and Ban (2020), assuming that travelers follow the instantaneous dynamic user equilibrium (IDUE) when they travel from their origins to the destinations, and then evaluate the performance of the perimeter control method. Actual investigations of this and the detailed results will be done in future research.

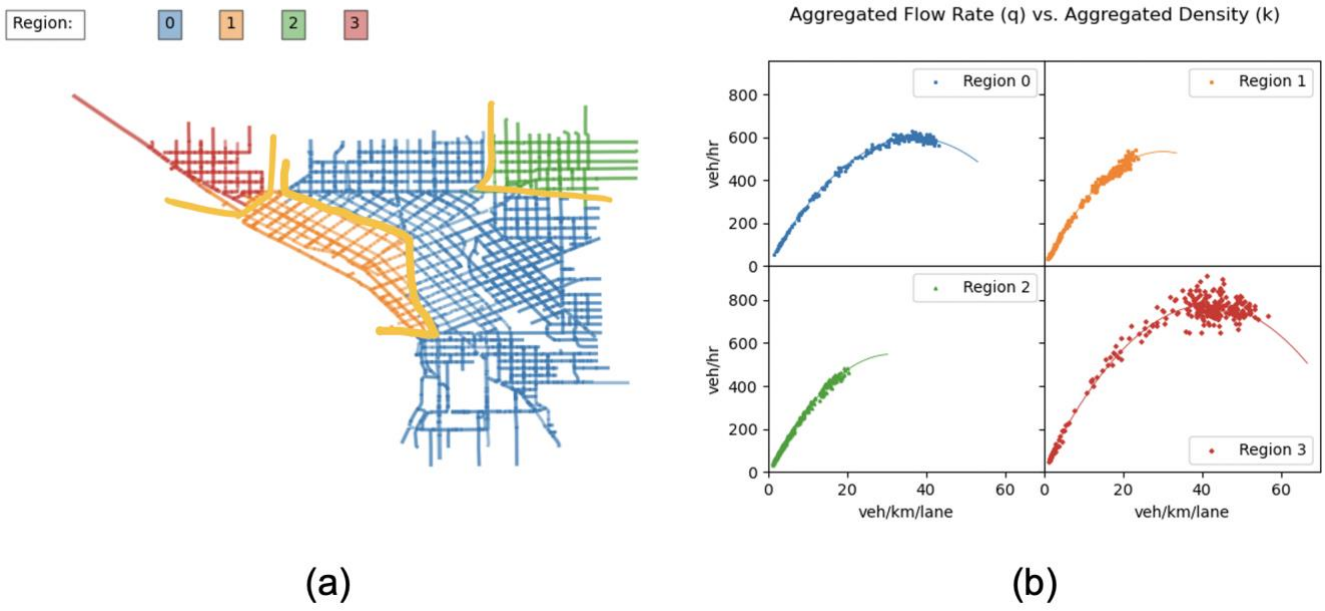


Figure 5.8. Selected partitioning results and the MFDs for each region.

6. Conclusion and Recommendation

This project focuses on developing APIs that integrate MATSim with other independent external simulators, eliminating the need to convert external simulators into dedicated MATSim extensions. The development process is a collaborative effort between the NYU and UW team. The NYU team establishes an API framework by incorporating a ride-pooling simulator with MATSim to showcase the successful integration between MATSim and local fleet-based simulators. The end product is called the *FD platform*. The UW team developed a platform that integrates MATSim with traffic simulator SUMO and it is called the VTD simulator. Both of these products hold vast potential for application in future transportation studies, offering opportunities for large-scale transportation analysis with enhanced accuracy, without the need to modify MATSim functionalities.

The core idea behind connecting MATSim and external simulators is to update portions of the trip information obtained from the initial MATSim simulation using output from external simulators. In the case of fleet-based simulators, the trip leg information involving the studied transportation service is initially estimated based on given parameters. It is subsequently

updated after corresponding trip legs are further simulated in local simulators. The updated trip itineraries are used as new input in the next cycle. A similar concept is applied to traffic simulators. As demonstrated in VTD, the vehicle trip details are initially estimated by MATSim. By fetching parts of the trips that cross the focused area and updating them using SUMO, more accurate traffic information that reflects local traffic changes can be acquired.

Several experiments have been conducted using both products to showcase their capabilities and possible applications. The NYU team designates a service region where ride-pooling services are provided and assesses how changes in service strategies can impact users' travel behavior. The *FD platform* is capable of generating comparable results when pitted against the "DRT" extension output with identical settings. Furthermore, the *FD platform* can capture behavior changes due to service heterogeneity, showcasing its potential use in transportation service evaluations. Additionally, an equity analysis is performed based on the *FD platform*'s results, unveiling a method of using the *FD platform* to investigate social welfare effects resulting from transportation service changes. The UW team conducts two test cases, one for coupled traffic-CAV control and coordination at signalized intersections, and the other on using VTD for generating data to divide downtown Seattle into homogeneous regions and further to calibrate the MFD curve of each region.

The API frameworks can be easily adapted to accommodate simulators targeting various modes of transportation. Moreover, it can include more than one simulator within a single running cycle, demonstrating its significant potential for future applications in simulating large multi-modal systems involving multiple innovative transportation services.

7. Tech Transfer

The following products resulted from this project.

Table 7.1. Summary of research outputs

Output type	Description	Link/source
Paper	A simulation platform for connecting MATSim and external ride-pooling simulators	n/a
Code	FD Platform	https://github.com/BUILTNYU/Fleet-Demand-Platform
Code	VTD Simulator	n/a
Presentation	C2SMART Workshop	n/a
Presentation	Transit Techies presentation,	n/a
Presentation	University Luxemburg invited talk, March 2, 2023, "BUILT Lab Research Overview"	n/a
Data	FD Simulator test parameters and results from NYC case study	https://zenodo.org/record/8114990

References

Adnan, M., Pereira, F.C., Azevedo, C.L., Basak, K., Lovric, M., Raveau, S., Zhu, Y., Ferreira, J., Zegras, C. and Ben-Akiva, M., 2016. *SimMobility: A Multi-scale Integrated Agent-Based Simulation Platform* (No. 16-2691).

Agarwal, A., Zilske, M., Rao, K. R., & Nagel, K. (2015). An elegant and computationally efficient approach for heterogeneous traffic modelling using agent-based simulation. *Procedia Computer Science*, 52, 962-967.

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. *Transportation Research Part C: Emerging Technologies*, *64*, 101-116.

Bae, S., Sheppard, C., Campbell, A., Waraich, R., Feygin, S., Bilal, Z., Asif, M., Aria, D., Serdyuk, D., Balayan, A. and Sharma, R., 2019. *Behavior, Energy, Autonomy, Mobility Modeling Framework*. 7 Summits IT AG LTD, Zurich, Zurich; Skylite Networks, Fremont, CA; Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States).

Basu, R., Araldo, A., Akkinepally, A. P., Nahmias Biran, B. H., Basak, K., Seshadri, R., ... & Ben-Akiva, M. (2018). Automated mobility-on-demand vs. mass transit: a multi-modal activity-driven agent-based simulation approach. *Transportation Research Record*, 2672(8), 608-618.

Ban, X., Angah, O., Zhang, Y., Guo, Q., 2022. A Multiscale Simulation Platform for Connected and Automated Transportation System. Final Report Submitted to C2smart Tier 1 UTC, New York University.

Bazzan, A. L., & Klügl, F. (2014). A review on agent-based technology for traffic and transportation. *The Knowledge Engineering Review*, 29(3), 375-403.

Bean, W. L., & Joubert, J. W. (2020). A case study of off-hour delivery collaboration and cost-sharing between freight receivers and carriers. *Transportation Research Procedia*, *46*, 141-148.

Bischoff, J., Soeffker, N., & Maciejewski, M. (2016). A framework for agent-based simulation of demand responsive transport systems. Technische Universität Berlin.

Chen, Y., Zheng, N., & Vu, H. L. (2021). A novel urban congestion pricing scheme considering travel cost perception and level of service. *Transportation Research Part C: Emerging Technologies*, 125, 103042.

Ciari, F., Balac, M., & Axhausen, K. W. (2016). Modeling carsharing with the agent-based simulation MATSim: State of the art, applications, and future developments. *Transportation Research Record*, *2564*(1), 14-20.

de Souza, F., Verbas, O., & Auld, J. (2019). Mesoscopic traffic flow model for agent-based simulation. *Procedia Computer Science*, 151, 858-863.

Geroliminis, N. and Daganzo, C. (2008). Existence of urban-scale macroscopic fundamental diagrams: Some experimental findings, *Transportation Research Part B: Methodological*, 42(9), pp. 759-770.

Geroliminis, N., and Sun, j. (2011). Properties of a well-defined macroscopic fundamental diagram for urban traffic. *Transportation Research Part B: Methodological*, Vol. 45, no. 3, pp. 605-617.

Geroliminis, N., Haddad, J., Ramezani, M. , 2013. Optimal perimeter control for two urban regions with macroscopic fundamental diagrams: a model predic tive approach. *IEEE Trans. Intell. Transp. Syst.* 14 (1), 348–359

Guo, Q. and Ban, X. (Jeff) (2020) 'Macroscopic fundamental diagram based perimeter control considering dynamic user equilibrium', *Transportation Research Part B: Methodological*, 136, pp. 87–109.

He, B. Y., Zhou, J., Ma, Z., Wang, D., Sha, D., Lee, M., ... & Ozbay, K. (2021). A validated multi-agent simulation test bed to evaluate congestion pricing policies on population segments by time-of-day in New York City. *Transport Policy*, 101, 145-161.

Hörl, S. (2017). Agent-based simulation of autonomous taxi services with dynamic demand responses. *Procedia Computer Science*, 109, 899-904.

Hörl, S., & Balac, M. (2021). Synthetic population and travel demand for Paris and Île-de-France based on open and publicly available data. *Transportation Research Part C: Emerging Technologies*, *130*, 103291.

Horni, A., Nagel, K., & Axhausen, K. W. (2016). Introducing matsim. In *The multi-agent transport simulation MATSim* (pp. 3-7). Ubiquity Press.

Jang, I., Kim, D., Lee, D., & Son, Y. (2018, October). An agent-based simulation modeling with deep reinforcement learning for smart traffic signal control. *In 2018 International Conference on Information and Communication Technology Convergence (ICTC)* (pp. 1028-1030). IEEE.

Ji, Y. and Geroliminis, N. (2012) 'On the spatial partitioning of urban transportation networks', *Transportation Research Part B: Methodological*, 46(10), pp. 1639–1656.

Karsu, Ö., & Morton, A. (2015). Inequity averse optimization in operational research. *European journal of operational research*, 245(2), 343-359.

Klügl, F., Bazzan, A. L., & Wahle, J. (2003, July). Selection of information types based on personal utility: a testbed for traffic information markets. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems* (pp. 377-384).

Lee, T. C. (2007). An agent-based model to simulate motorcycle behaviour in mixed traffic flow (Doctoral dissertation, Imperial College London (University of London)).

Liu, J., Kockelman, K. M., Boesch, P. M., & Ciari, F. (2017). Tracking a system of shared autonomous vehicles across the Austin, Texas network using agent-based simulation. *Transportation*, 44, 1261-1278.

Martins-Turner, K., Grahle, A., Nagel, K., & Göhlich, D. (2020). Electrification of urban freight transport-a case study of the food retailing industry. *Procedia Computer Science*, *170*, 757-763.

Mualla, Y., Bai, W., Galland, S., & Nicolle, C. (2018). Comparison of agent-based simulation frameworks for unmanned aerial transportation applications. *Procedia computer science*, 130, 791-796.

Naiem, A., Reda, M., El-Beltagy, M., & El-Khodary, I. (2010, March). An agent-based approach for modeling traffic flow. *In 2010 The 7th international conference on informatics and systems* (INFOS) (pp. 1-6). IEEE.

Nguyen, J., Powers, S. T., Urquhart, N., Farrenkopf, T., & Guckert, M. (2021). An overview of agent-based traffic simulators. *Transportation research interdisciplinary perspectives*, 12, 100486.

Nourinejad, M., Chow, J. Y., & Roorda, M. J. (2016). Equilibrium scheduling of vehicle-to-grid technology using activity-based modelling. *Transportation Research Part C: Emerging Technologies*, *65*, 79-96.

Oh, S., Seshadri, R., Azevedo, C. L., Kumar, N., Basak, K., & Ben-Akiva, M. (2020). Assessing the impacts of automated mobility-on-demand through agent-based simulation: A study of Singapore. *Transportation Research Part A: Policy and Practice,* 138, 367-388.

Ren, X., & Chow, J. Y. (2022). A random-utility-consistent machine learning method to estimate agents' joint activity scheduling choice from a ubiquitous data set. *Transportation Research Part B: Methodological*, 166, 396-418.

Rieser, M., Dobler, C., Dubernet, T., Grether, D., Horni, A., Lammel, G., ... & Nagel, K. (2014). MATSim user guide. *Zurich: MATSim*.

Ronald, N., Thompson, R., & Winter, S. (2015). Simulating demand-responsive transportation: a review of agent-based approaches. *Transport Reviews*, 35(4), 404-421.

Saeedmanesh, M., & Geroliminis, N. (2017). Dynamic clustering and propagation of congestion in heterogeneously congested Urban Traffic Networks. *Transportation Research Part B: Methodological*, *105*, 193–211.

Santos, F., Nunes, I., & Bazzan, A. L. (2018). Model-driven agent-based simulation development: A modeling language and empirical evaluation in the adaptive traffic signal control domain. *Simulation Modelling Practice and Theory*, 83, 162-187.

Schroeder, S., Zilske, M., Liedtke, G., & Nagel, K. (2012). Towards a multi-agent logistics and commercial transport model: The transport service provider's view. *Procedia-Social and Behavioral Sciences*, *39*, 649-663.

Smith, L., Beckman, R., & Baggerly, K. (1995). *TRANSIMS: Transportation analysis and simulation system* (No. LA-UR-95-1641). Los Alamos National Lab. (LANL), Los Alamos, NM (United States).

Viergutz, K., & Schmidt, C. (2019). Demand responsive-vs. conventional public transportation: A MATSim study about the rural town of Colditz, Germany. *Procedia Computer Science*, *151*, 69-76.

Wahle, J., Bazzan, A. L. C., Klügl, F., & Schreckenberg, M. (2002). The impact of real-time information in a two-route scenario using agent-based simulation. *Transportation Research Part C: Emerging Technologies*, 10(5-6), 399-417.

Yao, F., Zhu, J., Yu, J., Chen, C., & Chen, X. M. (2020). Hybrid operations of human driving vehicles and automated vehicles with data-driven agent-based simulation. *Transportation Research Part D: Transport and Environment*, 86, 102469.

Yoon, G., Chow, J. Y., & Rath, S. (2022). A simulation sandbox to compare fixed-route, semi-flexible transit, and on-demand microtransit system designs. *KSCE Journal of Civil Engineering*, *26*(7), 3043-3062.

Zheng, N., Rérat, G., & Geroliminis, N. (2016). Time-dependent area-based pricing for multimodal systems with heterogeneous users in an agent-based environment. *Transportation Research Part C: Emerging Technologies*, 62, 133-148.

Zheng, N., Waraich, R. A., Axhausen, K. W., & Geroliminis, N. (2012). A dynamic cordon pricing scheme combining the macroscopic fundamental diagram and an agent-based traffic model. *Transportation Research Part A: Policy and Practice*, 46(8), 1291-1303.