SEAMLESS SHARED URBAN MOBILITY

On-demand fleet management

Project deliverable D2.2

WWW.SUM-PROJECT.EU

Co-funded by
the European Union

This project has received funding from
the European Union's Horizon Europe
research and innovation programme
under grant agreement No 101103646.

Deliverable Administrative Information

Deliverable Administration					
Grant Agreement	101103646	**Project short name**	SUM		
Deliverable no.	D2.2	**Deliverable Name**	On-demand fleet management		
Status	Final	**Due month**	M24	**Submission Date**	30/05/2025
Author(s)	David Epstein (TAU), Mor Kaspi (TAU)				
Dissemination level	PU = Public				

	Version	**Date**	**Submitted**	**Reviewed**	**Comments**
Document history	V1.0	20/05/2025	David Epstein (TAU), Mor Kaspi (TAU)	Giulia Petrarulo (INRIA), Shadi Sharif Azadeh (TUD), Oded Cats (TUD), Florian Dandl (TUM), Markus Dieter Fischer (TUM), Rafal Kucharski (JU), Olha Shulika (JU)	1st draft
	V2.0	22/05/2025	Giulia Petrarulo (INRIA)	Mor Kaspi (TAU), David Epstein (TAU), Shadi Sharif Azadeh (TUD), Oded Cats (TUD), Florian Dandl (TUM), Markus Dieter Fischer (TUM), Rafal Kucharski (JU), Olha Shulika (JU),	2nd draft
	V3.0	28/05/2025	David Epstein (TAU), Mor Kaspi (TAU)	Giulia Petrarulo (INRIA)	Final version

Legal Disclaimer

Co-funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Climate, Infrastructure and Environment Executive Agency (CINEA). Neither the European Union nor the granting authority can be held responsible for them.

TABLE OF CONTENTS

List of figures

List of tables

List of abbreviations and acronyms

Acronym	Meaning
ABM	Agent Based Modelling
ASC	Alternatives Specific Constant
CPU	Central Processing Unit
DARP	Dial-a-Ride Problem
ExMAS	Exact Matching of Attractive Shared
GTFS	General Transit Feed Specification
KPI	Key Performance Indicator
MaaS	Mobility-as-a-Service
MOD	Mobility On Demand
MSA	Method of Successive Averages
NIS	New Israeli Shekel
NSM	New Shared Mobility
OD	Origin-Destination
ORS	Open Route Service
OSM	Open Street Map
OTP	Open Trip Planner
POI	Point of Interest
PT	Public Transport
SimFLEX	Simulation Framework for Feeder Location Evaluation
SUM	Seamless Shared Urban Mobility
TAZ	Traffic Analysis Zone
VMT	Vehicle Miles Travelled
VoT	Value of Time

Project Executive Summary

The objective of the SUM project is to transform current mobility networks towards innovative and novel shared mobility systems (NSM) integrated with public transport (PT) in more than 15 European Cities by 2026, reaching 30 by 2030. Intermodality, interconnectivity, sustainability, safety, and resilience are at the core of this innovation. The outcomes of the project offer affordable and reliable solutions considering the needs of all stakeholders such as end users, private companies, public urban authorities.

Social Media links:


@SUMProjectHoEU


@SUM Project

For further information please visit **WWW.SUM-PROJECT.EU**

Deliverable executive summary

Keywords

shared mobility, on-demand transportation, mode-choice, simulation, optimization

Summary

Shared mobility services offer innovative transportation solutions that, when effectively integrated with public transportation, bridge service gaps and enhance overall mobility. This deliverable focuses on ride-pooling services, a form of ridesharing where either a city or a private enterprise manages a fleet of vehicles (often shuttles) to provide on-demand transit. A ride-pooling service matches requests with similar routes that start and end within a defined service area. Operators must decide which requests to accept and which vehicles to assign. A sustainable service pools a substantial portion of rides without significantly extending journey times due to detours and intermediate stops.

This deliverable presents two holistic frameworks developed as part of the SUM project for managing ride-pooling services. These complementary frameworks facilitate the design and evaluation of ride-pooling services that enhance urban mobility and integrate effectively with existing public transportation networks.

The first framework is centred on three core components: mode choice, service simulation, and design optimization. At its core is a feedback loop where a mode choice model predicts user demand, a high-resolution simulator (FleetPy) models service performance, and updated service quality metrics refine the demand estimates. Once this inner loop stabilizes, an outer optimization process explores service design parameters – such as fleet size and operational area – using simulation-based search heuristics. This iterative framework is demonstrated through a case study in the Yuvalim-Ganim neighbourhood in Jerusalem. Initial results demonstrate inner loop convergence, revealing consistent patterns in model parameters that capture passenger preferences and system dynamics. Notably, an increasing returns phenomenon is observed, where expanding the fleet size leads to higher mode share and a greater number of served requests, emphasizing the importance of maintaining sufficiently large fleets to support effective pooling.

The second framework enables evaluating the potential implementation of a demand-responsive ride-pooling service. The framework consists of two stages, beginning with an area-level feasibility assessment using the ExMAS tool to simulate pooling opportunities and identify optimal hub connections across various demand scenarios. The second stage integrates new shared mobility with the city's public transport system using the SimFLEX framework, which incorporates iterative mode choice modelling to evaluate the combined service's attractiveness over standard transit. Key performance indicators such as mileage reduction, waiting time, occupancy, and user satisfaction guide the comparative analysis. This framework is utilized to evaluate and compare 12 selected areas in Krakow for the potential implementation of a new demand-responsive transport service. The service is envisioned as a small-capacity bus system that picks up travellers from designated shared collection points and transports them to major transfer hubs, such as tram or railway stations, where they can continue their trip via regular public transport.

Together, these frameworks offer robust tools for designing ride-pooling services – from evaluating spatial feasibility to optimizing user adoption – serving as practical resources for urban planners aiming to improve multimodal connectivity and the overall efficiency of urban transport systems.

This document provides a detailed description of the proposed framework and offers step-by-step guidelines for implementing them in new locations, covering introduction to the code repository, software installation,

data preparation, execution, and output interpretation. To illustrate the framework's components, we use examples from the two Living Labs.

Purpose of the deliverable

The purpose of this deliverable is to present comprehensive frameworks for designing, implementing, and evaluating dynamic ride-pooling services in urban environments. It outlines the theoretical foundations, key components, and practical considerations necessary for applying the framework across different locations. By providing a clear structure, the document aims to guide practitioners and researchers through each stage of the process, from initial setup to the interpretation of system performance outcomes.

Beyond describing the methodology, the document serves as a practical manual for deployment. It details the technical steps required to implement the framework, including software installation, data preparation, scenario execution, and result analysis. By offering concrete guidelines and best practices, it seeks to ensure that users can effectively replicate and adapt the framework to the unique conditions and needs of different cities or neighbourhoods.

Any change to a transportation service places a burden on users, whether it involves road construction, adjustments to public transport timetables, or the introduction of new route plans. While new shared mobility (NSM) services are inherently flexible and lend themselves to periodic adjustments, changes to pricing structures, service areas, or operating modes can be challenging to communicate to users and often require considerable time for the system to stabilize. The primary goal of this document and its accompanying code repository is to equip cities participating in the SUM project – as well as other cities considering the implementation or redesign of ride-pooling services – with tools to systematically evaluate different service configurations. These tools aim to support realistic assessments of service performance, operational costs, and expected mode share, ultimately enabling better advance planning and more informed decision-making regarding the design of ride-pooling services.

Attainment of the objectives and explanation of deviations

The following excerpts are representative statements from the Task 2.2 description in the SUM agreement: 1) “we will apply a holistic framework for the management of shared on-demand fleets consisting of the following elements: mode choice, simulation, and optimization” and 2) “Our approach consists of the following steps: (i) creating digital twin models of real-world ride-hailing and ride-pooling operations, including user-interaction, routing, and vehicle charging strategy; (ii) calibrating the operational strategies and the user model with data available from several of our Living Labs; (iii) analysing traffic-flow and safety indicators related to boarding processes; (iv) analysing the impacts of user cancellations and developing control strategies for specific cancellation rates; (v) specifying utility-based service design models and online control strategies; and (vi) testing different scenarios to support the design of field implementations.”

We have built the frameworks as initially envisioned; both are already implementable and consist of the three main elements described above. Steps (i)–(ii) and (v)–(vi) were carried out as planned. While a more detailed representation of the boarding process and a direct modelling of user cancellations are still pending, these were not required for the initial analysis. The initial development focused on the Krakow and Jerusalem living labs. The frameworks were developed in a modular manner, enabling the addition of these components later, based on the needs of the following cities.

Intended audience

The intended audience includes SUM Living Labs, other cities, and relevant policymakers involved in planning the implementation or redesign of ride-pooling services. The dissemination level of this deliverable is “Public”.

Structure of the deliverable and links with other work packages/deliverables

This deliverable consists of both this document and an accompanying code repository. As a by-product, it enables high-resolution static demand predictions for ride-pooling services and supports the representation of existing public transportation (PT) services, allowing their interactions with ride-pooling systems to be modelled effectively. In this sense, the deliverable is closely linked to other two tasks – – task 2.1 and 2.3 – – within Work Package 2.

1. Introduction

As urban populations grow and the demand for sustainable transportation increases, cities are actively exploring innovative mobility solutions. Ride-pooling, a form of shared mobility that allows multiple passengers to share a vehicle for trips with overlapping origins or destinations, has emerged as a promising option (Shaheen & Cohen, 2019). It occupies an important space in the Mobility-as-a-Service (MaaS) paradigm, which aims to integrate various forms of transport services into a single accessible platform (Hensher et al., 2020).

Understanding how individuals choose between ride-pooling and other transportation modes is essential for designing effective services and policies. Mode choice modelling offers a structured framework for capturing user preferences and decision-making behaviour (Ben-Akiva & Lerman, 1985). When combined with simulation frameworks, these models enable researchers and planners to analyse and forecast the performance of ride-pooling systems under various scenarios (Friedrich, 2016). To effectively leverage these approaches for evaluating and identifying promising system designs, they must be integrated with appropriate optimization algorithms – while accounting for the significant computational cost associated with evaluating each design.

This introduction provides an overview of the current state of the art in the three core components essential for evaluating ride-pooling services: mode choice, simulation, and optimization. In this Deliverable, we present two frameworks developed as part of the SUM project, each designed to support the planning and design of different types of ride-pooling services. These frameworks integrate the three components in distinct, modular ways. An overview of the two frameworks is provided at the end of this section. The subsequent sections offer a detailed description of each framework, present numerical results, and highlight key insights derived from the analyses.

1.1. Literature review

Ride-pooling services, also known as dynamic carpooling or shared rides, differ from traditional ride-hailing by allowing multiple passengers to share a trip, thereby reducing costs, vehicle miles travelled (VMT), and environmental impact (Fagnant & Kockelman, 2014). Unlike fixed-route transit, ride-pooling offers flexible routing and scheduling, responding dynamically to real-time demand.

Operational models for ride-pooling vary widely. Some services operate on a fully dynamic, on-demand basis (e.g. Uber Pool, Lyft Shared), while others use hybrid models combining scheduled and flexible components (e.g. Via, MOIA). The effectiveness of these services depends on algorithms for matching riders, routing vehicles, and minimizing detours (Alonso-Mora et al., 2017). Challenges include ensuring acceptable levels of service (e.g., waiting time, travel time), achieving high vehicle occupancy, and managing user expectations (Ma et al., 2013).

Mode choice modelling is grounded in discrete choice theory, which assumes that individuals choose the option that maximizes their utility among a set of alternatives. Commonly used models include: Multinomial Logit, Nested Logit, Mixed Logit and Probit Models. These models differ in their assumptions regarding the correlation between alternatives, the number of alternatives that can be considered, representation of variation and the use of theoretical distributions (Train, 2009). These models are estimated using revealed preference data from actual travel behaviour, stated preference surveys, or a combination of both (Walker & Li, 2007).

In the context of ride-pooling, mode choice models must incorporate attributes unique to shared mobility, such as waiting time, detour length, price sensitivity, and social factors (Lavieri & Bhat, 2019, de Ruijter et al., 2023). For instance, Krueger et al. (2016) explicitly includes ride-pooling as a distinct alternative in mode

choice models, alongside public transit, walking, biking, and ride-hailing. Empirical studies highlight several key factors influencing ride-pooling adoption: time-related attributes such as travel and waiting times play a critical role in mode choice decisions (Abouelela et al., 2022); cost savings relative to private or single-passenger options enhance the appeal of ride-pooling (Alemi et al., 2018); and user preferences are shaped by demographic characteristics, trip purposes, and familiarity with shared mobility services (Dias et al., 2017).

A major challenge in modelling new shared mobility services – particularly ride-pooling – is the lack of representation in existing travel behaviour data and stated preference surveys. This is primarily because the analysis is often done prior to the implementation of these services. As a result, additional methods are needed to integrate them into models. These include approximating their attributes using comparable existing modes (Krueger et al., 2016) and incorporating performance metrics derived from simulation models (Liu et al., 2019; Alonso-González et al., 2021).

Simulation frameworks are essential tools for evaluating the performance of ride-pooling systems under varying demand, supply, and policy scenarios. Two predominant modelling approaches are commonly employed: Agent-Based Modelling (ABM) and Event-Based Simulation.

Agent-based models represent travellers, vehicles, and operators as individual agents with distinct behaviours and decision rules. This approach offers the flexibility to capture interactions, heterogeneity, and emergent system-level dynamics (Nahmias-Biran et al., 2019, Zwick et al., 2021; Engelhardt et al., 2022, Akhtar et al., 2024) For example, Bischoff et al. (2017) integrates ride-pooling services into the MATSim platform to assess their impact on Berlin's urban transport system. The findings suggest that replacing private car use with pooled rides can significantly reduce total vehicle kilometres travelled, although effects on congestion depend on adoption levels. Alonso-Mora et al. (2017) develop a real-time, high-capacity ride-pooling algorithm and simulate it using a large-scale dataset from New York City. The dynamic simulation showed that over 95% of taxi demand could be met using only 20% of the fleet, through efficient pooling. Omidvar et al. (2022) employ agent-based simulation to analyse electric, autonomous ride-pooling services under different charging strategies, emphasizing trade-offs between battery capacity, charging infrastructure, and service quality.

Event-based simulation, in contrast, models system dynamics by sequencing discrete events – such as ride request arrivals, vehicle assignments, and passenger boardings – in chronological order. This approach is particularly well-suited for evaluating dispatching algorithms and operational strategies. For instance, Bongiovanni et al. (2022) simulate the operations of autonomous ride-sharing services using an event-based framework, which is embedded in a data-driven optimization model designed to dynamically assign requests and determine vehicle routing.

Optimization approaches in ride-pooling research can be broadly categorized into two groups: operational decision-making and strategic service design. The first group focuses on daily operational decisions such as vehicle routing, request assignment, and vehicle redistribution. Central to this is the Dial-a-Ride Problem (DARP), which models the task of designing cost-minimizing routes and schedules for a fleet of vehicles departing from a common depot to serve a set of passengers with specified pickup and drop-off locations and associated time constraints. The *static* version of DARP assumes that all transportation requests are known in advance, whereas the *dynamic* version deals with real-time request arrivals that must be handled as the system operates. Operational constraints typically include vehicle capacity, tour duration, time windows, and ride-time limitations (Molenbruch et al., 2017; Ho et al., 2018).

A related stream of research addresses the matching of vehicles to ride requests under dynamic conditions, often emphasizing the development of online algorithms. Agatz et al. (2012) provide a foundational review of dynamic ride-sharing optimization techniques, distinguishing between heuristic and exact methods and highlighting the trade-offs between solution quality and computational speed. Nourinejad and Roorda (2016) use agent-based modelling to compare centralized and decentralized matching strategies, capturing both individual traveller behaviour and system-level dynamics. Yu and Shen (2019) propose a scalable hybrid

method that combines problem decomposition with approximate dynamic programming to improve real-time assignments by anticipating future demand. Şahin et al. (2022) introduce a data-driven approach that learns from historical trip data to balance detours and waiting times in matching decisions. More recently, Tuncel et al. (2023) develop an integrated model for real-time ride matching and vehicle rebalancing, using a mixed-integer programming framework informed by short-term demand forecasts to improve reliability and efficiency.

Another critical operational challenge is vehicle redistribution, especially given the spatial and temporal asymmetry of demand, which can result in supply imbalances. Efficient repositioning strategies are essential for maintaining service quality. Fagnant and Kockelman (2014) offer early insights using agent-based simulation to demonstrate how proactive relocation of shared autonomous vehicles (SAVs) can enhance service performance and reduce vehicle miles travelled. Wallar et al. (2018) formalize vehicle rebalancing as an optimization problem, proposing a real-time model that incorporates demand forecasts and pooling opportunities. Syed et al. (2021) propose a density-based method that redistributes idle vehicles using historical demand heatmaps. Engelhardt et al. (2023) apply machine learning to forecast demand and guide repositioning, improving both service availability and pooling rates. Valadkhani and Ramezani (2023) present a rolling-horizon optimization framework that integrates predictive repositioning into large-scale ride-sourcing operations, accounting for both current system conditions and anticipated demand.

The second group of studies focuses on strategic design decisions, including fleet sizing, pricing, service area configuration, and integration with public transit. Balac et al. (2020) use agent-based simulation to explore optimal fleet sizes for pooled automated vehicle services, revealing how vehicle supply influences service efficiency and user satisfaction under varying demand conditions. Zwick and Axhausen (2020) investigate how parameters such as detour thresholds and maximum waiting times affect system performance and public acceptance, highlighting important trade-offs between service quality and operational efficiency. Bahrami et al. (2022) introduce a mathematical optimization framework to determine the ideal mix of solo and pooled services in ride-hailing fleets, demonstrating that offering both options can enhance system efficiency and user welfare when priced effectively. Fan et al. (2024) explore the integration of ride-pooling as an on-demand feeder for public transit, proposing a bi-level optimization model that jointly designs service zones and dispatching strategies to minimize both user and operator costs. Akhtar et al. (2024) use a game-theoretic framework to analyse the interactions between dynamic pricing, driver incentives, and user mode choices, proposing pricing strategies that maximize revenue while accounting for fluctuating demand and traveller behaviour.

In recent years, several **Integrated Frameworks** have applied bi-level optimization integrating mode choice models within simulation environments to optimize the design and operation of mobility on demand (MoD) systems. These frameworks follow a common structure: an inner loop consisting of mode choice modelling and service simulation, and an outer loop responsible for optimizing system design variables such as fleet size, service configuration, and pricing. This coupling ensures that user behaviour and system performance inform design decisions iteratively and realistically, accounting for demand–supply interdependencies.

In Liu et al. (2019), a multinomial logit model is estimated from stated preference data to capture user preferences across different travel modes, including mobility on-demand services. The mode choice model feeds into a simulation module that evaluates MoD system performance under different demand realizations. This inner loop is embedded within an outer-loop optimization routine, where Bayesian optimization, a sequential search strategy for the global optimization of an expensive black-box function, determines the optimal fleet configuration and pricing strategy. Bansal et al. (2019) extends this framework by incorporating reliability into the choice model, showing that conveying wait-time reliability can significantly influence user preferences. Both studies use simulated service performance to inform travel time and wait time inputs in the utility function, creating a feedback loop between user choice and service quality.

Pinto et al. (2019) and Hu et al. (2024) adopt similar agent-based simulation frameworks, embedding mode choice models within iterative simulations of multimodal systems that include shared autonomous vehicle fleets and fixed-route transit. Pinto et al. (2019) solve a bi-level optimization problem in which the upper level selects fleet sizes and transit line frequencies, while the lower level simulates mode choices and vehicle operations using a detailed agent-based model. Hu et al. (2024) further enrich the inner loop by modelling supernetwork structures and service coverage through binary logit models, capturing nuanced spatial and service-level decisions. Guo and Zhao (2024) generalize this approach by optimizing pricing, fleet sizing, and transit routing while modelling individual-level mode and route choices with discrete choice models. They use scalable approximation methods to solve the large-scale problem efficiently. Collectively, these works illustrate a shift toward tightly coupled behaviour-system co-design paradigms in transportation, where mode choice models and simulation models are central elements in system optimization.

1.2. Overview of the proposed frameworks

The two frameworks presented here are both designed to support the assessment of NSM solutions in conjunction with public transport. They share a modular architecture, allowing for the flexible substitution or modification of key components such as mode choice models and simulation engines. Both frameworks rely on open-source tools, generate comparable performance indicators, and enable comparative evaluation across different deployment strategies.

Despite these commonalities, they differ in purpose and approach. The simulation-optimization framework is intended to optimize the configuration and operation of on-demand fleets. Leveraging FleetPy and embedded optimization loops, it offers mobility operators significant flexibility to fine-tune service parameters such as dispatching policies, pricing mechanisms, and fleet composition. While public transport integration is not built-in, it can be incorporated through external modules as needed.

In contrast, the ExMAS–SimFLEX framework focuses on identifying optimal locations and conditions for introducing pooled feeder services. It is specifically designed to integrate with existing public transport systems and to evaluate user adoption over time through iterative behavioural modelling. This framework is particularly well-suited for urban planners seeking data-driven, context-sensitive recommendations for the spatial deployment of ride-pooling services.

Simulation-Optimization Framework

This framework comprises three key elements: mode choice, simulation, and optimization. While many studies in mode choice rely on external data, such as mobility preference surveys, our framework adopts a novel approach. It integrates data from simulated ride-pooling operations into the mode choice model, leveraging a high-resolution simulator called FleetPy. These two components form a feedback loop: the mode choice model predicts how many travellers will select the ride-pooling service, FleetPy simulates the service and generates service quality metrics, and these metrics feed back into the mode choice model, adjusting the modal split for the next iteration. When this feedback loop converges, it provides an evaluation of a given service design. The outer layer of the framework focuses on the optimization of service design. Using a simulation-based optimization approach, this layer generates each service design, initiates the inner loop, and evaluates the results. The goal is to explore the vast parameter space intelligently to identify promising configurations. The challenge lies in balancing the computational cost of each evaluation with the desire to find reliable solutions. We address these computational challenges through parallel processing and adaptive modelling. Even with these approaches, only a limited number of possible designs can be evaluated, necessitating the development of efficient search heuristics. Figure 1 displays an overview of the framework.

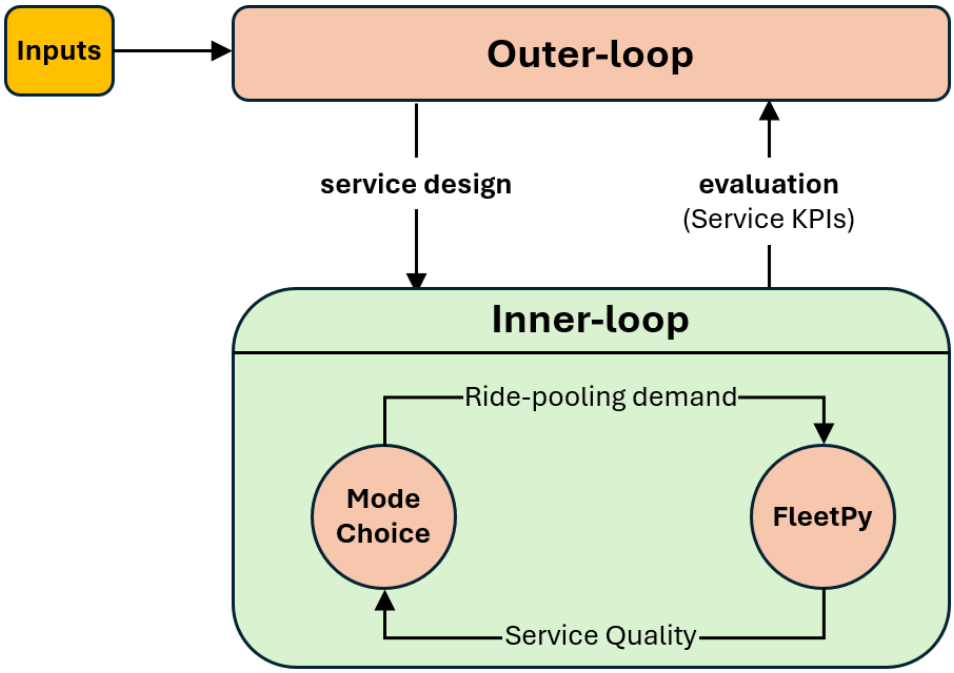

Figure 1: Overview of the Simulation-Optimization Framework

ExMAS-SimFLEX Framework

This framework follows a two-stage approach designed to comprehensively assess the feasibility and potential impact of NSM services integrated with public transport. In the first stage, we apply a parameter-free methodology to evaluate the suitability of NSM deployment in pre-selected areas based on spatial demand distribution and ride-pooling potential. Using ExMAS, we simulate multiple demand scenarios across 24 area-hub combinations and assess service performance using key performance indicators (KPIs) such as mileage reduction, passenger satisfaction, and occupancy rates. This stage identifies the most promising hub for each area and pinpoints which areas meet the minimum demand thresholds for initial implementation. In the second stage, we analyse complete door-to-door trips by integrating NSM with existing public transport through the SimFLEX framework. This involves simulating travel demand and traveller behaviour using iterative mode choice modelling and evaluating the attractiveness of the combined NSM+PT service relative to traditional PT-only alternatives. KPIs such as service attractiveness, waiting time reduction, and added value inform our evaluation of the areas with the highest adoption potential. Together, these stages provide a robust, passenger-centred methodology that supports evidence-based decision-making for urban planners and policymakers aiming to strengthen first- and last-mile connectivity in public transit networks.

2. Simulation-Optimization Framework

2.1. Input

The framework requires several key inputs to operate effectively. These include an origin-destination (OD) matrix specifying the trip volumes between all pairs of Traffic Analysis Zones (TAZs) for a given time period, and a street network, which can be obtained in a compatible format as described in the "How-To" section. Additionally, a table is needed that maps each node ID in the street network to its corresponding TAZ ID. The simulation requires extensive configuration such as information about the fleet (e.g. vehicle maximum occupancy, fuel consumption, number of vehicles) and operator policies that affect the potential for pooling requests. These are aspects of an NSM design and addressed in detail in Section 2.3.1. The simulation also requires low-level settings such as the simulation time step, which the framework sets in its configuration files and does not vary.

For the mode choice component, utility formulas must be provided for each available mode, with basic examples offered in the "How-To" section. Initial model parameters must also be supplied for all modes, including both alternative-specific constants (ASCs) and beta coefficients. Finally, initial estimates for the NSM's key performance metrics – such as expected wait time and trip delay relative to private car travel – are needed.

While not strictly required, GIS layers of the units of analysis for demand (e.g., TAZ) and for the focus neighbourhood or area would likely prove helpful. Furthermore, since calculating utility for each mode and trip requires estimating the trip time and distance, access to a routing service will be required. This is discussed further in the section dedicated to creating the sample frame.

2.2. Inner-loop

The inner loop receives a parameterized NSM design and loop control settings from the outer loop. It iterates over the design, running the NSM simulation, and updating its estimates until reaching a threshold of consistency called convergence. It then returns an evaluation of the NSM design. The following sections discuss every aspect of the inner loop as presented in Figure 2 below.

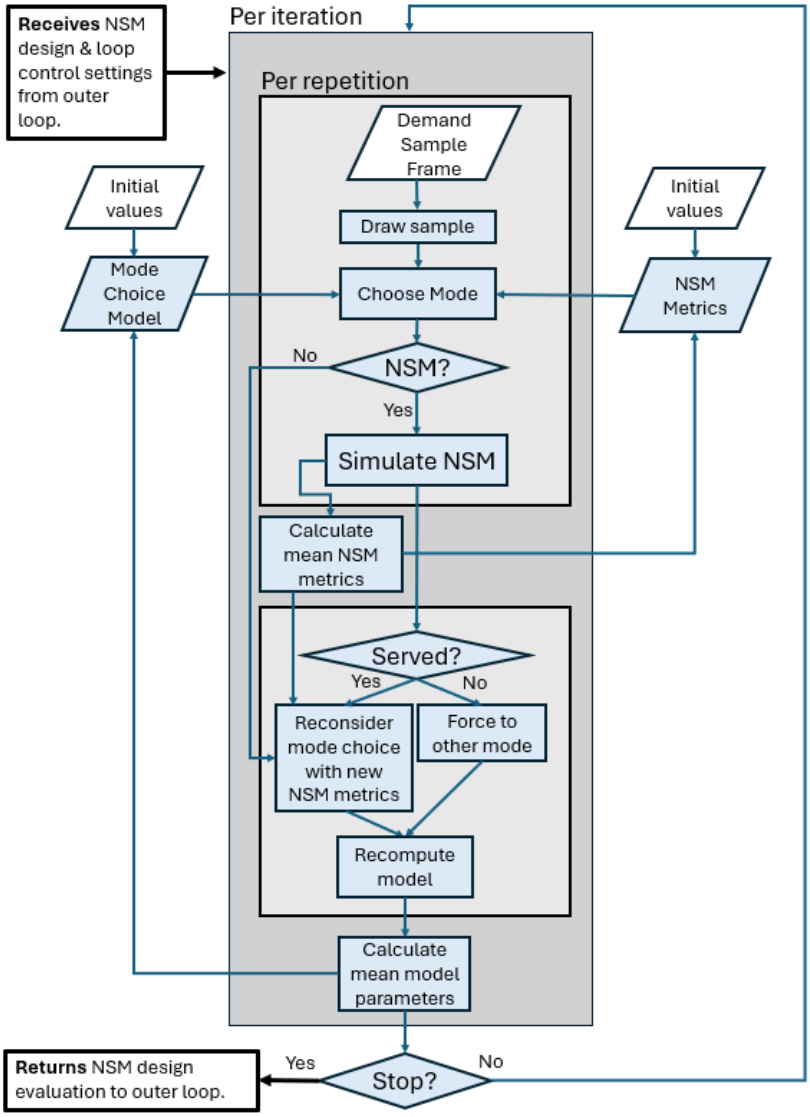

Figure 2: Overview of the inner-loop

Repetitions

An iteration does not usually involve just a single sample of demand, mode choice, and simulation. To reduce the impact of outliers, the framework permits users to specify that each iteration will include multiple repetitions with different demand samples but the same NSM design, NSM metrics, and mode choice model. After every iteration, the framework calculates the mean value of NSM performance metrics and mode choice model parameters for the next iteration. This process is made more efficient by FleetPy's excellent parallelism for disparate jobs.

Initial Values

Every time the inner loop receives a new NSM design, it resets to the same user-defined initial values for mode choice model and NSM performance metrics. These initial values affect the efficiency of the inner loop.

The mode choice model includes alternatives specific constants (ASCs) for every mode and beta coefficients for every utility parameter. The framework expects users to have initial values for these based on a recent mode choice survey, but without the hypothetical NSM under investigation. The framework is designed to estimate a new model that includes this NSM as a mode option. However, the framework requires starting estimates for the missing NSM ASC and any new betas, such as B_RISK, which adjusts the impact of low service rates (high risk) for on-demand, ride pooling services. The case study describes one way of calculating these.

Similarly, the framework requires initial values for NSM metrics that become or are used to calculate utility parameters. The framework tracks the NSM wait time, NSM delay ratio, and NSM request acceptance ratio. The NSM wait time is the time between when a traveller makes a request and when that traveller receives service. The NSM delay ratio is the NSM travel time divided by car travel time. This value is equal to 1.0 when ride pooling is not possible and over 1.0 when ride pooling occurs, causing delay. The framework requires values for these metrics to determine mode choice in the first iteration. Mode choice determines which and how many travellers select the NSM, enter the simulation, and affect the performance metrics used in the next iteration. The initial values for the NSM metrics therefore might be interpreted as representing what travellers believe about the NSM on its first day of service. If their beliefs are overly optimistic, the NSM will be overwhelmed with requests for service, leading to time consuming ridesharing delays, and many unserved users. Additionally, from a framework performance point of view, an overload of requests for NSM service results in longer simulation times. The metrics will be corrected downward over subsequent iterations. If the metrics are pessimistic, the NSM will provide a high level of service to nearly all customers. The performance metrics will be corrected upward over subsequent iterations. However, if the metrics are so pessimistic that no users select the NSM, the entire optimization process will stop prematurely. For this reason, it is better for initial NSM metrics to err on the side of optimism.

Alpha Weight

The mode choice model parameters and NSM metrics do not remain at their initial values. Each iteration produces new estimates of their true value. A user-defined *alpha* weight balances prior estimates against new estimates. Technically, this alpha is a float between zero and one with values less than 0.5 favouring the prior estimate and values greater than 0.5 favouring the new estimate. Practically, the weight controls how quickly travellers act upon new information regarding NSM performance. Conceptually, this might be interpreted variously as learning, information diffusion, or technology adoption. There is a single alpha weight for all trips and metrics. When the weight is low, travellers act slowly upon new information. Here is the formula used to update all metrics:

*next value = previous value * (1.0 - alpha) + current value * alpha*

If NSM performance metrics start far from their true values, many iterations will be required to reach convergence, costing CPU time and resulting in evaluating fewer NSM designs. Furthermore, the high number of iterations required may reach the max iteration set by the user via the outer loop, causing the inner loop to exit before reaching a stable estimate of NSM performance and raising doubts about optimization results. Conversely, when the weight is high, travellers act quickly upon new information. Even if NSM performance metrics start far from their true values, few iterations may be required to reach convergence. However, weights that are too high can cause metrics to cycle around their true values as travellers overreact. For this reason, it is better to begin experimenting with the framework using an alpha weight between 0.1 and 0.5.

Mode Choice

At the start of each iteration, the inner loop draws a sample from a large pool of potential trips. This pool is called the demand sample frame. The case study explains one way of creating a suitable sample frame from an OD matrix. Generating it takes time, but is only performed once, offline. Sampling from the frame every iteration and repetition is not instantaneous, but still much faster than generating new data. Larger sample frames produce more varied demand and robust results. Every part of the framework runs slower when there are many travellers, especially many NSM users. For this reason, even when the sample frame is very large, it is recommended to begin experimenting with the framework using small demand samples.

The framework chooses a mode for every traveller in the sample following random utility theory. It calculates the utility of every mode based on user-defined formulas and the current mode choice model and mode metrics. Only the NSM's metrics change between iterations. The framework uses utility to calculate the probability of selection for each mode and then selects one at random in accordance with those probabilities.

Simulation

The framework uses a specialized NSM simulator to evaluate the performance of a given NSM design. For on-demand ride pooling services, the framework relies on FleetPy. For every iteration and repetition, the framework turns travellers who chose the NSM into requests for NSM service and calls the simulator to determine if those requests receive service and – if yes – at what level (wait time and travel time).

FleetPy runs simulations for every repetition in an iteration, executing many jobs in parallel if configured to do so and if hardware permits. The framework then reads FleetPy's output files. To simulate the need for travellers to reach their intended destinations, the framework forces unserved NSM requests onto other modes according to the original probabilities.

Updating NSM metrics

Served NSM requests are used to compute mean performance metrics across all repetitions. One key metric is the NSM delay ratio, which is usually greater than one and represents how much longer a trip takes by NSM compared to a private vehicle due to ride pooling. Other key metrics include the wait time and the service rate. The framework computes a mean value for each metric across all repetitions and updates the inner loop's ongoing estimate of these values using the user-defined alpha weight.

Updating the Mode Choice Model

The framework applies the updated NSM metrics to the original demand sample, recalculating utilities with the original mode choice model but with a better estimate of NSM performance. It then permits travellers who received service from the NSM and travellers who never requested service to choose their travel mode again based on the new probabilities. These decisions are used only to update the model at the end of an iteration – the simulator is not run again. For each repetition, the framework runs the discrete choice package, Biogeme, on the new set of utilities and mode choices and receives new model parameters. It then calculates the mean of these and updates the mode choice model according to the user-define alpha weight.

Stop Condition

The inner loop has two stop conditions, either of which causes the loop to return the most recent NSM evaluation to the outer loop and exit. The outer loop establishes both conditions when calling the inner loop. The first condition is if the difference in the NSM mode split between the current iteration and previous iteration is lower than the specified threshold. This condition is interpreted as meaning that the inner loop has converged at a stable (enough) solution and further iterations are no longer necessary. The second condition is if the current iteration is equal to the maximum number of iterations permitted. This condition is interpreted as meaning that the inner loop has timed out.

If no stop condition fires, then the inner loop repeats from the beginning using the most recent mode choice model and NSM metrics. It draws a new random sample for each repetition, chooses travel modes for traveller, etc.

The next sections describe the two major components of the inner loop (mode choice and simulation) in more detail.

2.2.1. Mode choice

This section reviews the role of mode choice modelling in the framework, introduces random utility theory, provides examples of utility formulas, describes how the framework updates the mode choice model, and introduces the discrete choice modelling package, Biogeme, which the framework calls to generate new estimates of model parameters.

The Role of Mode Choice Modelling
Unlike a bus or train operator who could hypothetically test aspects of the service by assigning vehicles to move along the intended path without passengers, on-demand mobility operators have no default path for vehicles to follow. The framework must generate NSM service requests before it can evaluate that service.

The performance of an on-demand service depends directly on the time, origin, and destination attached to requests and the current MoD fleet state. The NSM simulator provides this dynamism on the supply-side by simulating a fleet of vehicles moving through a network to pickup and drop-off travellers. The mode choice model provides this dynamism on the demand-side by simulating individuals deciding which mode to use given details about their trips and the various modes available.

The framework draws from a sample frame based on an empirically derived OD matrix. The sample does not specify each traveller's intended mode, just the parameters necessary to compute the probability each mode has of being selected. Using a mode choice model permits the framework to change these probabilities as each traveller learns more about how the NSM is expected to perform. Mode choice modelling permits good systems to attract more users and poor systems to attract fewer users. The simulator responds, permitting systems with few requests to serve nearly all of them in a timely manner and overburdened systems to struggle with low service rates and significant delays due to ride-pooling.

Random Utility Theory
The framework follows the tenants of random utility theory, which holds that decision makers (people) are utility maximisers and choose the alternative with the highest utility. The total utility (U) of an alternative (e.g., a product or service) has both an observable, deterministic part (V) and an unobservable, stochastic part (ε).

$$U = V + \varepsilon$$

Since ε is stochastic (a random variable), researchers assume it follows a known distribution (e.g. normal) and then model choice probabilistically based on the relative sizes of the deterministic part of the equation. The alternative with the highest V would therefore have the highest likelihood of selection and be selected the most often, but its selection could not be guaranteed every time if alternatives exist with non-zero V. In this document, the term "utility" refers to this deterministic part (V) and the term "total utility" refers to both parts together.

Utility for a given alternative is composed of several parameters. Many travel-related utility formulas depend upon time and cost. Each of these variables requires a coefficient called a beta, which scales its value based on its impact on choice (amount and direction). Since travelers don't like spending money and time on travel,

both these betas are usually negative. Each mode also has an alternative ASC capturing the relative attractiveness of each mode that cannot be explained by the other parameters. Therefore, using parentheses to emphasize grouping and the prefix “B” to connote “beta”, each mode’s utility follows the pattern:

V = ASC + (B_VAR1 * VAR1) + (B_VAR2 * VAR2)

Example Utility Formulas
The following utility formulas employed during the framework’s development can serve as a starting point for more location specific and advanced ones. They assume:

Walking and biking are free save for the time spent.

A unit of time or money spent for one mode is equal to the same unit spent for another mode. This means time spent riding a bike (in a park) is the same as time spent driving a car (in traffic).

Only the NSM comes with a risk of not receiving service.

Calculating and comparing utility requires formulas for each mode that will include the alternative specific constant (ASC) for each mode, one or more variables and the beta (coefficient) for each variable. The ASC captures the relative attractiveness of each mode that the variables do not account for.

Walking

V0 = ASC_WALK + B_TIME * WALK_TIME

Biking

Traditional and electric bikes have different travel times and may require modelling as different modes.

V1 = ASC_BIKE + B_TIME * BIKE_TIME

Private car

The formula below calculates the utility of car usage as fuel consumption, which requires setting the cost of fuel per litre and fuel economy (km per litre) to a local average and adding a column to the sample frame measuring trip distance by car. More sophisticated utility formulas might include parking or even per trip fuel economy based on car ownership if such data exists.

V2 = ASC_CAR + B_TIME * CAR_TIME + B_COST * CAR_COST

Public transit

More advanced PT utility formulas might consider the availability of service near the origin and destination.

V3 = ASC_PT + B_TIME * PT_TIME + B_COST * PT_COST

New Shared Mobility (NSM)

The framework introduces risk into the NSM utility formulas measured as one minus the service rate. So, if a given ride pooling service pickup up seventy-five percent of requests, the risk is 0.25. Since risk is unpleasant when one wants travel, the beta for risk is negative.

V4 = ASC_NSM + B_TIME * NSM_TOTAL_TIME + B_COST * NSM_COST + B_RISK * NSM_RISK

Mode Choice Model Updates

The development team faced several big questions regarding mode choice modelling. First, should the prior ASCs and betas remain constant when a new mode is introduced into the transportation system? These values are often derived from surveys of travellers who did not have the opportunity to choose the NSM under investigation. Since the framework holds the entire transportation system constant and varies just the design of the NSM, it could precompute the utility of each mode for each trip using the prior ASCs and beta values. Then the framework would only need to compute the ASC and any betas associated with the NSM (and only with the NSM) like B_RISK. However, the developers assert that preferences for travel time, cost, and reliability are all deeply interconnected and change depending on the travel modes available. For this reason, the framework currently recalculates all model parameters at the end of every iteration. This approach is computationally expensive. Further analysis of how the mode choice model parameters change from iteration to iteration may reinforce this decision or lead to a better, more efficient solution.

Additionally, our approach may initially put excessive downward pressure on B_RISK. The framework does not permit travellers who were denied service and forced to use a different travel mode the chance to choose again in the “reconsider” step in Figure 2. This constraint punishes the NSM design for providing poor service. During framework development, permitting this population to also choose travel mode again resulted in convergence at service rates around 80% while denying them resulted in convergence at service rates above 90%.

The second big question was, how can demand be generated when the inner loop first starts? It needs RISK and B_RISK to calculate NSM utility. However, RISK is calculated from the NSM service rate received after simulation and B_RISK is calculated by discrete choice analysis at the end of every iteration. The framework requires the value of variables it is designed to find! In computation, these chicken and egg problems are often solved through a process variously referred to as bootstrapping or iterative calculation. The user must provide initial values and the framework will improve upon them.

If the initial values are wildly incorrect, the optimization process will take longer. If RISK (the chance of not being served) and B_RISK (its beta, assumed negative) are both too low, travellers will flock to the NSM, it will become overburdened, perform poorly, and the framework will iteratively correct the initial values. If RISK and B_RISK are too high, travellers will avoid the NSM, resulting in exceptional performance (as long as there are some requests) and iteratively more users will choose the service until it begins to struggle. Like the NSM metrics, model parameters are updated according to the alpha weight balancing prior and new values.

To estimate the model parameters, we employ the maximum likelihood method using the Biogeme package. Biogeme is an open-source tool specifically developed for estimating a broad range of discrete choice models using advanced optimization algorithms. It is widely used in transportation research due to its flexibility and robustness. Biogeme supports a variety of custom model specifications, with built-in support for several commonly used models, including Multinomial Logit (as used in this framework), Nested Logit, Mixed Logit, and Probit models. The package is implemented in Python, allowing for seamless integration into our analytical framework. See: https://biogeme.epfl.ch/ for installation instructions, getting started instructions and some examples.

2.2.2. NSM Simulation – FleetPy

FleetPy is an open-source, agent-based simulation framework developed to model NSM services with high temporal and spatial resolution. Written in Python, it is designed to simulate both user and operator behaviour

in detail, enabling the study of dynamic interactions in shared mobility systems. FleetPy supports scenarios with multiple operators and vehicle types, facilitating research into cooperative and competitive dynamics in the on-demand mobility landscape.

A core strength of FleetPy is its focus on modelling user-operator interactions in a realistic manner. For instance, users receive real-time trip offers through mobile apps and make decisions such as accepting, rejecting, or waiting, while operators control pricing, routing, and matching decisions. This dynamic, bidirectional communication structure allows for the simulation of complex system designs, including integrated public-private mobility services, or joint passenger-freight operations.

FleetPy is built with a modular and extensible architecture that supports the reuse of components across scenarios. Its key components include: a simulation core that governs event handling and temporal progression; a fleet control module for optimization strategies (e.g., dispatching, repositioning); a user demand model to represent traveller requests and decisions; a network module for spatial representation and routing; and a vehicle module to simulate state transitions and operations. This structure enables researchers to plug in custom algorithms or modify specific behaviours without overhauling the entire framework.

Because of its modularity and layered architecture, FleetPy supports flexible experimentation at different levels of abstraction. For example, developers can compare different assignment algorithms or operational policies using a shared simulation base, or test the sensitivity of system performance to variations in traveller behaviour or network design. The framework balances computational efficiency with detailed representation, making it suitable for both large-scale simulations and fine-grained analysis of NSM systems.

See: https://github.com/TUM-VT/FleetPy and Engelhardt et al. (2022) for further information on FleetPy.

2.3. Outer-loop

As outlined in previous sections, the evaluation of the inner loop is computationally intensive due to the multiple iterations required between the mode choice model and FleetPy until convergence is achieved – particularly in terms of KPIs. With a clear understanding of the inner loop's structure and the parameters that influence its behaviour, we developed a search heuristic that leverages this knowledge to efficiently explore promising service designs.

Specifically, the optimization layer differentiates between two categories of input: service design parameters, which correspond to those defined in Section 2.3.1, and evaluation settings, which govern the configuration of the inner loop itself.

It is important to note that a parallel line of research in the scientific literature explores the use of surrogate models to approximate computationally expensive simulation-based evaluations. This approach can significantly increase the number of configurations that can be assessed during the optimization process. While this method is not implemented in the current deliverable, the framework has been designed with a modular architecture, allowing for the future integration of such an “intermediate” surrogate modelling layer if desired. However, the development and calibration of such a model would likely require tailoring the framework to the specific context of the implementing city.

To maintain clarity and manageability, we implement a metaheuristic search algorithm that interacts directly with the inner loop, exploring both service and evaluation settings. As with other components of the framework, this outer loop is fully modular and can be readily adapted to incorporate alternative optimization strategies as needed.

2.3.1. Service design parameters

The design parameters encompass both strategic and tactical characteristics of the ride-pooling service. These include the network structure, service area, operational mode, fleet configuration, and operational constraints such as request response time and maximum allowable detour delay. Table 1 summarizes these parameters, including their types and defined ranges.

The network structure specifies the set of transportation links on which ride-pooling vehicles are permitted to operate, while the service area is defined as a polygon within which trip requests can be made. Both features are treated as categorical variables, meaning the search is limited to a predefined set of candidate networks and areas specified as part of the framework's input.

We consider two primary operational modes: Many-to-Many – where users can request trips between any two stops within the defined area, and Few-to-Many – where trip requests must involve at least one location from a predefined subset of hub nodes (either as the origin or the destination). In the latter mode, the input includes a list of potential hub nodes, from which a subset is selected during the optimization process.

The fleet configuration is characterized by the number and capacity of vehicles, with the current implementation assuming homogeneous fleets. From an operational perspective, when a user submits a request, the system must respond within a predefined number of seconds, indicating whether the request is accepted and, if so, which vehicle will serve it. Longer response windows generally increase the potential for pooling by enabling more flexible assignment options; however, they also significantly raise the computational load in FleetPy, as the system must evaluate a larger solution space in the internal assignment module.

Lastly, service quality and pooling effectiveness are highly sensitive to the maximum detour delay allowed for on-board passengers – i.e., the additional travel time incurred due to picking up or dropping off other riders. This parameter plays a critical role in balancing user satisfaction with system efficiency.

Table 1: Service design parameters

Setting	Sub-setting	variable type	Dictionary	Settings considered in the numerical results
Network/ Service area	Demand matrix (=Points of interest+ Hubs)	categorial	[table A, table B…]	Yuvalim-Ganim
Mode Of Operation	Mode Of Operation	categorial	MTM: many--> many (+ POI), MTF\FTM: many--> few (POI/Hubs)	MTM without POI
Fleet	Fleet size	integer	(# of vehicles)	Ranges 2-20
	Vehicle capacity	categorial	(# of seats per vehicle)	will drive additional definition
Operator response time	pooling waiting time	integer [sec]	The maximal time the system can wait before providing a response to a request	300 Seconds
De-tour delay	De-tour delay	Percentage	Percentage increase as compared to direct travel time by car	100%

2.3.2. Evaluation settings

The following parameters govern the execution of the inner loop and influence both computational efficiency and the statistical robustness of the results. These parameters are especially critical when estimating the

expected performance of a given system design. Proper tuning ensures a balance between resource usage and result reliability. Table 2 summarizes these parameters along with their types and allowable ranges.

The first parameter defines the number of simulation replications used to evaluate each service design. Increasing the number of replications improves the statistical accuracy of the evaluation by reducing the variance in the results. This parameter typically takes on discrete values, which may depend on the number of available processing cores for parallel execution. For instance, if 30 processor cores are available, the parameter might take values such as 30, 60, or 90, corresponding to one, two, or three full sets of parallel replications.

The second parameter specifies the maximum number of inner loop iterations allowed before the simulation terminates and returns an evaluation. In the current experiments, this value is fixed; however, it can be complemented with convergence criteria to enable adaptive termination, thereby reducing computational time when the system reaches stability before the iteration limit is met.

Table 2: Evaluation parameters

Settings	variable type	Range	Note
# of replication in the inner loop The number of days (or datasets) with different demand realizations given to the framework for evaluating a single NSM design	integer	1 to number of CPUs available	FleetPy can efficiently run a simulation on each CPU simultaneously, which permits adding replications without adding significant computation time.
Max iterations in the inner loop The maximum number of iterations permitted for evaluating a single NSM design.	integer	1 and higher. 20 is a good starting setting.	This serves as a timeout to prevent endless computation if convergence is not reached (according to the stop threshold).
Stop threshold for the inner loop Percent point NSM mode share difference between one iteration and the next	Float	Usually less than 1.0 (one percentage point). Even as low as 0.02.	Smaller values will require more iterations and may cause a timeout (max iterations) before convergence

2.3.3. Metaheuristic approach

We employ a nested adaptive stochastic local search metaheuristic to explore the space of ride-pooling service designs, focusing on identifying high-performing configurations through hierarchical, adaptive search. Given the strong interaction between network structure, operational mode, and other design parameters, we use a nested search strategy to manage complexity and prioritize more impactful decisions early in the process.

At each iteration of the heuristic, a network setting is randomly selected from a predefined set, guided by an adaptive probability distribution that favours previously successful configurations. Next, a mode of operation is sampled, and if a few-to-many structure is chosen, hub locations are also randomly selected from a candidate list.

Other design parameters – such as fleet size, vehicle capacity, response time window, and maximum detour – are initialized based on the best-known configuration so far. These parameters are then optimized incrementally: for each, we evaluate the impact of increasing or decreasing its value by a single unit. If an improvement is observed, we continue to explore in that same direction until no further gains are found.

To manage computational load while ensuring robust results, simulation fidelity is progressively adjusted throughout the evaluation process. Each design is initially evaluated using a single full set of parallel

simulations and a limited number of inner loop iterations (e.g., 10). During this stage, we monitor the variance of the KPIs. If high variance persists across several consecutive evaluations – indicating that differences between designs are not statistically significant – the number of replications is incrementally increased to enhance estimate precision and enable more reliable comparisons between candidate configurations.

In parallel, we track the number of inner loop iterations required for KPI values to stabilize, defined as the point at which the relative change falls below a specified threshold. If, over a user-defined number of consecutive evaluations, convergence is consistently achieved in fewer than the maximum allowed iterations, the inner loop limit is reduced to save computational effort in outlier cases. Conversely, if convergence repeatedly requires more iterations than the current limit, the maximum number of inner loop iterations is increased to ensure more accurate performance assessments. The outer loop accepts evaluation results even when the maximum number of iterations has been reached. A pseudo code of the algorithm is presented in Table 3.

Table 3: Outer-loop heuristic

Algorithm: Nested adaptive stochastic local search

Input: an initial solution x , network_distribution, modes_of_operation, hub_candidates, num_replications, inner_loop_limit

$x^b = x$; **/initializing best solution**

While stopping criteria is not met:
    $x^t.network \leftarrow$ samplenetwork(network_distribution)
    $x^t.mode \leftarrow$ samplemode(modes_of_operation)
    if $x^t.mode$ == "few-to-many":
        $x^t.hubs \leftarrow$ samplehubs(hub_candidates)
    $x^t.fleesize = x^b.fleetsize$
    $x^t.capacity = x^b.capacity$
    $x^t.res\_tw = x^b.res\_tw$
    $x^t.maxdetour = x^b.maxdetour$
    for *param* in { $fleesize, capacity, res\_tw, maxdetour$**}:**
        for direction in [+1, -1]:
            while True:
                $temp = x^t$
                $temp.[param]$ += direction
                $temp.KPI =$**inner_loop**($temp$, num_replications, inner_loop_limit)
                if $temp.KPI > x^t.KPI$
                    $x^t = temp$
                else:
                    break
    if $x^t.KPI > x^b.KPI$
        $x^b = x^t$
    network_distribution=updatenetworkdist(network_distribution)
    mode_of_operation=updatedmodedist(modes_of_operation)
    hub_candidates=updatehubdist(hub_candidates)
    if high_variance_detected_over_recent_evaluations():
        num_replications += 1

    if inner_loop_converges_quickly():
        inner_loop_limit -=1
    elif inner_loop_converges_slowly():
        inner_loop_limit += 1

return x^b

2.4. Output

The framework's output includes a set of key performance metrics, classified into service-level, behavioural, and operational indicators. **Service-level** indicators reflect user experience and include the percentage of requests served and the average waiting time. **Mode split metrics** reflect user travel behaviour by indicating the distribution of selected transportation modes – such as private vehicles, public transit, walking, cycling, and NSM services. These metrics help evaluate how attractive the shared service is relative to other available options. **Operational indicators**, such as the total driving cost (e.g., kilometres driven), measure the system's efficiency and resource utilization.

Together, these outputs provide a comprehensive basis for evaluating and comparing different service configurations. In the outer loop, we employ a composite objective function defined as a weighted sum of the main KPIs. Table 4 summarizes the main KPIs.

Table 4: Simulation-Optimization main KPIs

KPIs		variable type	Description
Service level	% of served request	number	Share of requests that were successfully fulfilled
	avg. wait time	number	Mean time from request to pickup
Mode split	% usage of NSM	number	Share of users selecting NSM over other travel modes
Operational indicators	cost (drive+ km)	number	Total kilometres driven, reflecting service cost and vehicle utilization

2.5. How to use it

2.5.1. Installation

The SUM Framework code is available at https://github.com/AUTOlab-TAU/SUM-Optimization. The framework has been tested in the following environment: python 3.11.1 (python.org), biogeme (3.2.13), fleetpy (1/7/2024), gurobipy (11.0.1), numpy (1.23.5), pandas (1.5.2), scipy (1.14.0). Further integration with its main dependencies, FleetPy and Biogeme, to permit easier installation is planned.

2.5.2. Framework Architecture

The framework consists of two main files, outer_loop.py and inner_loop.py and a number of helper files in the util folder. The outer loop calls the inner loop with a dictionary of NSM design and loop control parameters. When the inner loop's stop condition triggers, it returns a dictionary of performance indicators, which the outer loop uses to inform the next NSM design.

The framework is configurable through these variables in *util/setup.py*. Table 5 and Table 6 display the key folders and variables, respectively.

Table 5: Framework architecture – key folders

Variable	Description
base_path	Path to the home folder to the analysis
work_path	Path to folder for intermediate files
result_path	Path to folder for NSM design results for every iteration, averaged across repetitions
fleetpy_path	Path to FleetPy base folder, under which lie the *src, studies, and data* folders.
requests_file_path	Path to the sample frame

Table 6: Framework architecture - variables

Variable	Description
requests_file_ratio	A floating point number representing how much larger the sample frame is compared to actual demand.
demand_ratio	An intuitive way to scale demand. 1.0 is the demand as originally specified. 1.5 is 150% demand.
weight_alpha	Controls the balance between prior values and new values when the framework updates NSM metrics and the mode choice model. Values below 0.5 favor the prior value.
modes	A list like ["walk","bike","car","pt","nsm"] used to identify the various modes. In some cases the indexes are used (e.g. 0 = walk)
avails	A dictionary for setting what modes are available either probabilistically (if set to a float between 0.0 and 1.0 inclusive) or per traveller (if set to a column with 0 and 1 values). This can be used, for instance, to show which travellers own cars or to apply a threshold to a per TAZ or per row walkability or transit accessibility index.
init_model	The initial mode choice model in dictionary form e.g. {"ASC_WALK":4.0, "B_TIME":-0.12, ...}

2.5.3. Defining your use case

This example demonstrates setting up all aspects of an on-demand pooled taxi service within a neighbourhood, evaluating several designs, and generating basic tables and figures that show the how the designs perform and the mode share they attract.

Road Network

The FleetPy package includes a script, *osm_converter.py*, for downloading OpenStreetMap data and converting it into the node and edge files that FleetPy requires. The script is located in *FleetPy/src/preprocessing/networks/extractors*. The framework includes a short example in the *util* folder called *create_network.py* of how to call just the needed functionality to save a network using the by_name option (e.g. "Tel Aviv, Israel"). After execution, the files *edges.csv* and *nodes.csv* will be where they belong in *FleetPy\data\networks\city_name_osm\base*.

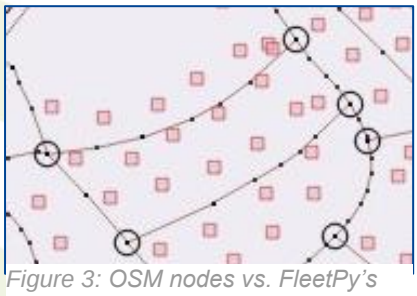

Figure 3: OSM nodes vs. FleetPy's routing network

FleetPy's simplifies the raw OSM data, leaving nodes only where streets connect to speed up routing calculations. Figure 3 shows how the original OSM nodes (dots) align with FleetPy's routing network (circles) and a GIS housing layer (squares) for a section of Jerusalem. Origins and destinations must be assigned to a node. Future versions of the framework will permit inserting nodes at specific coordinates to represent places of interest like mobility hubs, schools, and shopping areas.

OSM streets datasets in some locations lack information about the maximum or typical velocity on an edge. The converter defaults to 30km/h. While incorrect or incomplete data reduces FleetPy's accuracy and validity, the convertor permits researchers to get up and running very quickly and is highly recommended. Since FleetPy uses simple, open formats, custom solutions using city GIS data or mapping APIs are possible, but not currently part of the framework.

Demand

Configuring demand requires generating the sample frame and setting initial values for the mode choice model and NSM metrics.

Generating the Sample Frame

A framework user must create an appropriate trip sample frame for the area under investigation if a pool of potential trips does not exist. Generating the sample frame takes time, but is only performed once, offline. Sampling from it online (i.e., while the inner loop runs) is much faster. Larger sample frames require more time to compute initially and require more disk space. However, they enable more varied demand and therefore, more robust and trustworthy results. Larger samples (travel volumes) result in slower simulations and longer optimization times. While it is necessary to study the true demand volume due to its nonlinear impacts on the performance of a NSM service, relying on smaller volumes during initial setup and configuration is recommended.

The OD matrix shows the demand for travel between any two TAZ during a specific time period. The flow will often be strongly asymmetric during the morning and evening commutes. Dividing the flow by the duration permits estimating the average number of travellers per unit of time, usually per hour. Then, the Poisson distribution can be used to create synthetic data that continues to mirror, statistically, the original, empirical demand pattern. Each trip in the sample frame must be enriched with all the information necessary for calculating the utility (attractiveness) of each travel mode for that particular origin and destination.

The framework includes the script, *generate_requests_OSM.py* in the *util* folder for generating a demand sample frame (the pool of trips to sample from). This can be used to generate demand for just a particular neighbourhood or set of neighbourhoods as shown in Figure 4. The script requires:

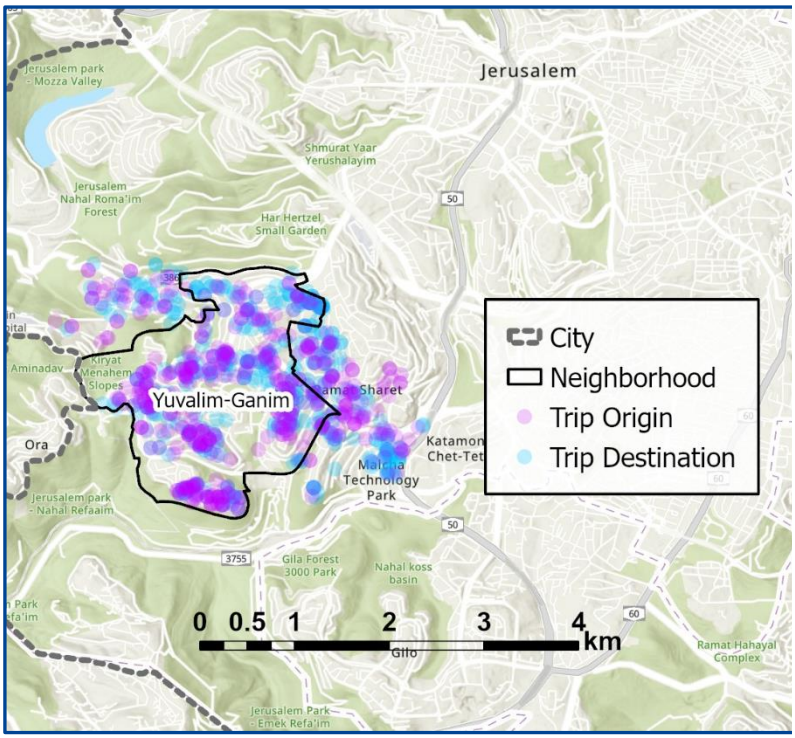

Figure 4:Origins and destinations distribution in Yuvalim-Ganim

(1) An OD matrix in the format of a table with origin and destination using TAZ ids and a time period in the last column. For the example study, the development team obtained commercial TAZ-level data from Decell to understand current demand volume. Decell differentiates between pedestrian, bike, train, and motorized. This last category combines private car and public transportation by bus and therefore cannot directly reveal mode choice preferences across all available modes.

(2) A node-TAZ mapping in the format of a table with two columns, FleetPy node IDs and corresponding TAZ ids. This can be generated using a GIS system (PostGIS, Python, or a GUI-based tool) using a "within" query on every node (point) or a "contains" query on every TAZ (polygon)

(3) A router like Open Route Service, which can be installed fairly easily using the docker image.

The script divides the demand volume by the duration to determine the average number of travellers per unit of time, usually by the hour. Then, the Poisson distribution is used to create synthetic data that continues to mirror, statistically, the original, empirical demand pattern. The framework generates a CSV file of user-defined size. It selects a random start and end node within the origin TAZ and destination TAZ and includes the time and distance for every mode. The user may then enrich the sample frame with additional data as required by the utility formulas employed. The sample frame used in the case study contained the columns shown in Table 7.

Table 7: Description of *Sample Frame* Columns

Parameter (Column)	Description	Source
orig_taz	Origin Traffic Analysis Zone	These are the fundamental parameters describing demand across all modes, obtainable through travel surveys, automatic or manual counts, and data providers (e.g., Decell). These parameters are often summarized in a TAZ origin-destination matrix for a known time period but without per trip travel times. A detailed and representative list of trips can therefore be generated via a Poisson distribution.
dest_taz	Destination Traffic Analysis Zone	
rq_time	Request time (start of trip)	
start	Starting node in FleetPy network	Depending on the NSM to be investigated, the SUM Framework's simulation module may require additional demand details. For example, FleetPy relies on a node and edge network derived from Open Street Map and the sample frame generation script assigns each trip to a random node in the origin TAZ and in the destination TAZ via a user provided lookup table.
end	Ending node in FleetPy network	
orig_x	Origin X-Coordinate	
orig_y	Origin Y-Coordinate	
dest_x	Destination X-Coordinate	
dest_y	Destination Y-Coordinate	
car_time	Private car travel time (sec)	These common parameters for utility calculation can be generated from routing services (e.g. Google Routes API or openrouteservice).The sample frame generation script provides a working example of using such a service.
car_dist	Private car travel distance (km)	
bike_regular_time	Regular bike travel time (sec)	
bike_regular_dist	Regular bike travel distance (sec)	
walk_time	Walk travel time (sec)	
walk_dist	Walk travel distance (km)	
pt_time	Public transit travel time (sec)	
pt_dist	Public transit travel distance (km)	
car_cost	Cost of car (fuel use)	User supplied (local knowledge)
pt_cost	Cost of public transit (ticket)	
request_id	Unique request id	User supplied, random, or sequential
orig_median_pt	Median PT availability in start TAZ	The SUM framework user provides the utility formulas for each mode. These formulas draw from additional columns in the sample frame. In our Jerusalem experiments, we used a locally defined indicator of public transit availability to estimate the utility of public transit.
dest_median_pt	Median PT availability in dest TAZ	
PTabove25	Boolean, both values > 25	
PTabove50	Boolean, both values > 50	
PTabove75	Boolean, both values > 75	

Setting initial values

Initial values are declared in util/setup.py. The initial values for the mode choice model and NSM metrics determine how many travellers choose the NSM during the first iteration of the inner loop. For the example study, the developers started with mode choice model parameters derived from the City of Jerusalem's most recent travel preferences survey (2017), which did not include a NSM option. The initial values are detailed in Table 8-11. The alpha weight for the case study was set to 0.25 in the same file.

Initial Mode Availability

Table 8: Initial values – mode availability

Variable	Value	Note
Walk	1.0	Assume everyone can walk
Bike	1.0	Assume everyone has access to a bike
Car	0.67	67% car ownership rate in the 2017 survey
PT	1.0	Experimented with PT available everywhere and PT available only in TAZ surpassing a given transit availability index.
NSM	1.0	Assuming anywhere-to-anywhere service with operating area defined through demand generation

Initial ASCs

Table 9: Initial values - ASCs

Variable	Value	Note
ASC_WALK	4.95	
ASC_BIKE	-1.09	Jerusalem is very hilly
ASC_CAR	4.80	
ASC_PT	4.70	
ASC_NSM	4.75	The mean of car and PT ASCs

Initial Betas

Table 10: Initial values - betas

Variable	Value	Note
B_COST	-0.35	COST in shekels. From survey.
B_TIME	-0.12	TIME in minutes. From survey.
B_RISK	0.0	RISK 0.0 to 1.0. Arbitrary. Assume indifferent to risk. Ensures high number of initial users.

Value of Time (VoT) can be computed as B_TIME/B_COST = -0.12 * 60 min/hr / -.35 = 20.57 shekels per hour, well within the range calculated by government agencies for various demographic groups in Israel.

Initial NSM Metrics

Table 11:Initial values – NSM Metrics

Variable	Value	Note
Occupancy	1.0	Anywhere-to-anywhere service with pooling due only to small size of focus area. Occupancy not currently used in utility formulas
Service_rate	0.2	
Nsm_car_time_ratio	1.5	
Nsm_wait_time	150	
Nsm_travel_time	370	
Car_time	500	

After the initial or current model and NSM metrics are used to determine mode choice, the inner loop calls the simulator.

Simulation

The framework interacts with FleetPy entirely through the filesystem. During every iteration, the inner loop draws a random sample of travellers for every repetition, determines mode choice for every repetition, and saves a separate request file in FleetPy format for every repetition containing the information about just the travellers who intend to use the NSM. It saves these rq-files ("rq" for request) in *FleetPy\data\demand\your_city_demand\mached* with names like SUM_i000_r000 to signify the first iteration and repetition. The framework overwrites these files everytime it runs. There is no reason to keep them if you already have results and are short on storage space.

After writing all the rq-files, the inner loop references them when it overwrites *scenario_config.csv* in *FleetPy\studies\your_city\scenarios*, specifying a new scenario for each request file, see for example 12. It then executes run_private_your_city.py, which can be configured to run multiple scenarios (repetitions) simultaneously using FleetPy's excellent parallelism. When all simulations have completed, the inner loop loads FleetPy's evaluation files and computes the mean of key metrics.

Table 12: Example of scenario files

scenario_name	rq_file
SUM_i000_r000	i000_r000_nsm.csv
SUM_i000_r001	i000_r001_nsm.csv
SUM_i000_r002	i000_r002_nsm.csv
SUM_i000_r003	i000_r003_nsm.csv
SUM_i000_r004	i000_r004_nsm.csv

FleetPy's wiki documents many of the package's features. While not required, if the street network is much larger than the demand area, adding a simple CSV file to *FleetPy\data\fleetctrl\initial_vehicle_distribution\your_city_osm* to focus the starting position of vehicles greatly improves NSM performance, especially when not simulating many hours of service. The file looks like Table 13. After create the file, edit the framework's util\fleetpy.py file, which describes how constant config will be overwritten. Adjust the value of *fleetpy_constant_config* to reflect the name of your file where you see the key "op_init_veh_distribution".

Table 13: example of vehicle starting node specification

node_index	probability
219	0.00034002
473	0.00034002
474	0.00034002
475	0.00034002
491	0.00034002
497	0.00034002

2.5.4. Running the framework

Once installed, configured with the proper paths, and provided all the input it requires, the framework is executed by simply running *outer_loop.py.* This script will quickly call the inner_loop with its first NSM design to evaluate, and printing a lot of information to the terminal window concerning sampling, simulating progress, and the current iteration.

2.5.5. Interpreting the output

The inner loop collects several outputs over the course of its iterations, including mode split results, NSM performance metrics, and the updated coefficients of the mode choice model. These outputs can be used to analyse the performance of a given NSM configuration and to track the convergence behaviour of the inner loop.

The following figures present the framework's results for the 20,000 travelers who participate in the 2-hour morning commute in the Yuvalim-Ganim area. The NSM service uses a fleet of 20 vehicles, each with a capacity of 20 seats, leftover from a discontinued on-demand service in Jerusalem called TikTak. The service in the case study operates in a many-to-many configuration, with an enforced operator response time of 5 minutes and a maximum detour delay limited to 100% of the direct travel time between origin and destination. The price of the service is set at 2 NIS regardless of time or distance, practically a free ride and highly attractive if dependable and time efficient.

Figure 5 shows the evolution of mode split over the iterations of the inner loop. The figure includes five curves, each representing one of the following modes: bike (red), car (lime), NSM (green), public transport – PT (blue), and walking (purple). As observed, the NSM share decreases rapidly in the initial iterations, while the shares of the other modes increase accordingly. The NSM stabilizes at around three percent of total trips after twenty iterations. This downward trend can be attributed to assuming travellers ignore the risk of not being served. The framework corrects this error. This suggests that it is at least somewhat robust against incorrect initial assumptions. Figure 6 displays the evolution of the average wait time of served requests over the inner-loop iterations. Again, the initial wait time provided to the framework was overly optimistic. In the first five iterations the wait time increased rapidly while the system was overwhelmed with requests. Then

the wait time fell slowly as the mode split began to stabilize. Figure 7 shows the travel time ratio (nsm vs car), occupancy, and service rate. Since the fleet size is fixed, occupancy reflects the number of users. Wait time (previous figure, travel time ratio, and occupancy are all positively correlated, suggesting a system struggling to meet demand. The initial low service rate further underscores this struggle.

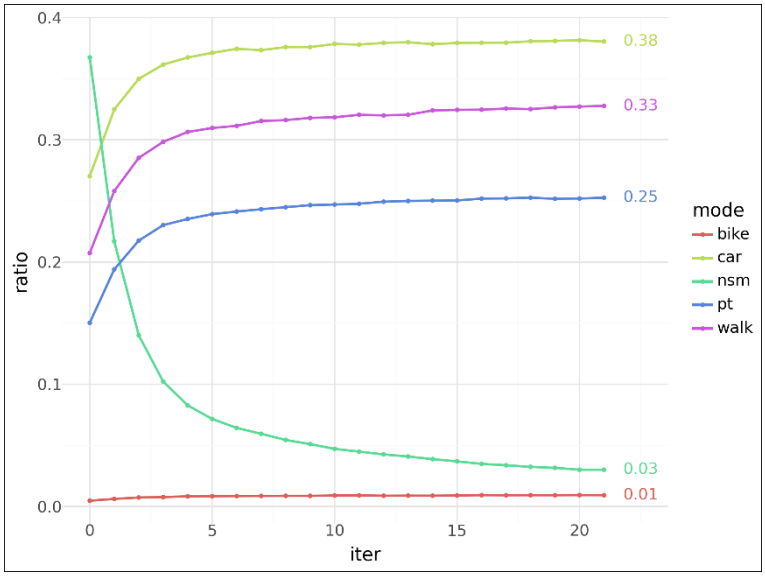

Figure 5: Change in mode split in the inner loop for a many-to-many service in the Yuvalim Ganim area. The service employs a fleet of 20 vehicles with 20 seats each. The operator is permitted a response time of 5 minutes and detour delay of 100%.

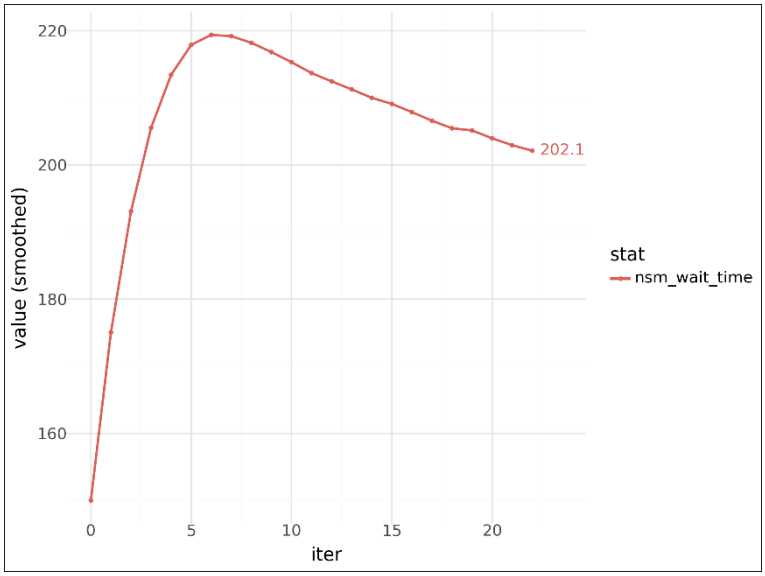

Figure 6: Change in NSM wait time in the inner loop for a many-to-many service in the Yuvalim Ganim area. The service employs a fleet of 20 vehicles with 20 seats each. The operator is permitted a response time of 5 minutes and detour delay of 100%.

Figure 8 and 9 below show the change in the mode choice model parameters as travellers receive more accurate information about the NSM service. The initial values for walking, biking, using a private car, and taking public transit were all computed from Jerusalem's 2017 mobility preferences survey, which did not have an NSM option. After the initial parameters for measuring NSM utility are used to compare it against the existing options and travellers permitted to choose amongst all options a new model is computed. This

causes the recalibration of all the ASCs seen in Figure 8. The initial estimate that the ASC for the NSM lies somewhere between that of the private car and PT proves incorrect. Figure 9 shows that even though the betas for cost and time are permitted to vary, they remain fairly stable and the new beta measuring the risk of not receiving NSM service moves significantly downward. This is not surprising because the framework forces all travellers who did not receive service to choose an alternative mode to reach their destinations before the model is recomputed. The result was shown in the earlier figures as the service's mode split dropped significantly, which permitted its service rate to eventually begin to rise.

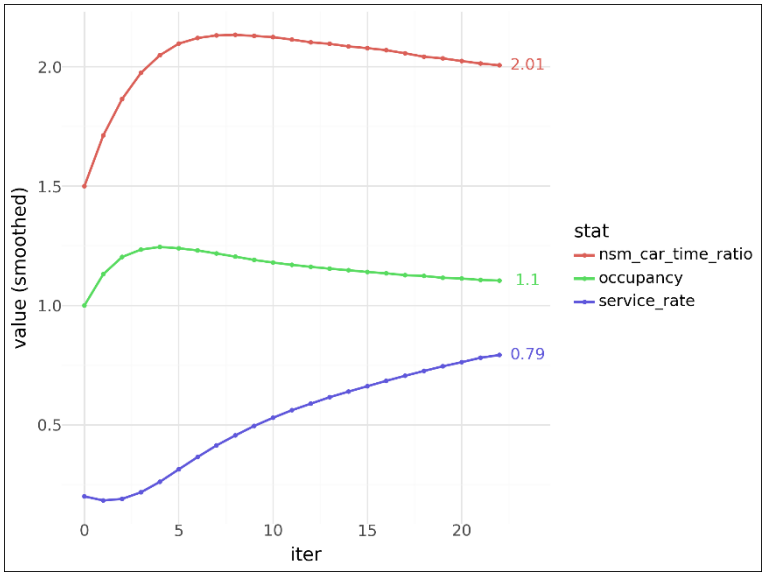

Figure 7: Change in NSM car time ratio, occupancy, and service rate in the inner loop for a many-to-many service in the Yuvalim Ganim area. The service employs a fleet of 20 vehicles with 20 seats each. The operator is permitted a response time of 5 minutes and detour delay of 100%.

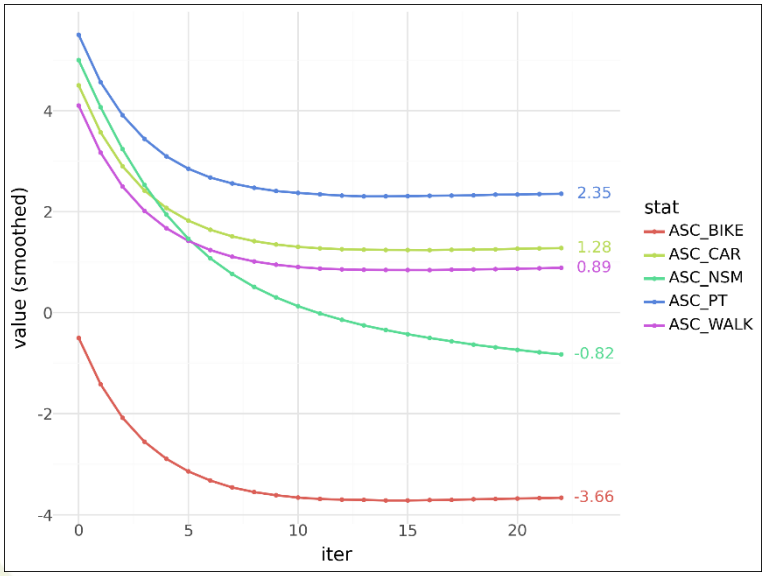

Figure 8: Change in alternative specific constants (ASCs) in the inner loop for a many-to-many service in the Yuvalim Ganim area. The service employs a fleet of 20 vehicles with 20 seats each. The operator is permitted a response time of 5 minutes and detour delay of 100%.

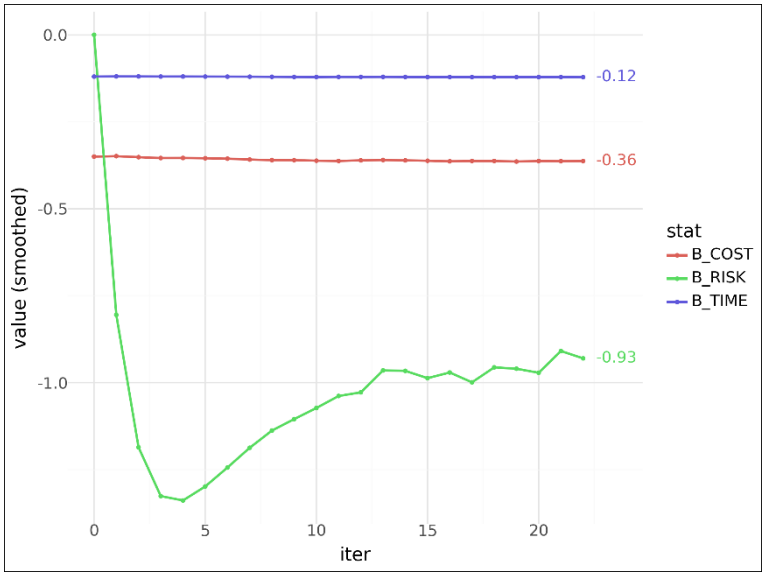

Figure 9: Change in beta coefficients in the inner loop for a many-to-many service in the Yuvalim Ganim area. The service employs a fleet of 20 vehicles with 20 seats each. The operator is permitted a response time of 5 minutes and detour delay of 100%.

Table 14 presents results for several key performance indicators – including mode split, number of served requests, acceptance rate, average vehicle occupancy (in passengers), and average passenger wait time – across configurations with 5, 10, and 20 vehicles, and service prices set at either 2 or 20 NIS. As expected, the number of NSM users increases as the service becomes more affordable or as the fleet size expands. Notably, these additional users are primarily drawn from three existing modes: walking, private car, and public transport (PT). While the goal is to attract private car users, a portion of the NSM mode share is diverted from more sustainable alternatives such as walking and PT. This substitution effect should be carefully considered in the service design to avoid the unintended "cannibalization" of PT usage.

Moreover, in all tested configurations, the acceptance rate remains consistently high, suggesting robust system performance. An additional insight is that when the fleet size is doubled, the number of served requests increases by more than a factor of two, indicating a phenomenon of increasing returns to scale in service capacity.

Table 14: Mode Split and NSM performance for varying fleet sizes and service prices

Fleet	Cost(₪)	Walk	Bike	Car	PT	NSM	Served	Acpt. Rate	Occ	Wait(min)
5	2	5,384 33.70%	147 0.92%	6,207 38.86%	4,163 26.06%	73 0.46%	63	86.37%	0.77	2:54
	20	5,386 33.71%	151 0.94%	6,238 39.05%	4,181 26.17%	19 0.12%	19	98.59%	0.66	2:22
10	2	5,317 33.29%	147 0.92%	6,157 38.54%	4,137 25.90%	216 1.35%	175	81.02%	0.96	3:10
	20	5,382 33.69%	153 0.96%	6,222 38.95%	4,202 26.30%	15 0.10%	15	99.88%	0.72	1:50
20	2	5,235 32.77%	146 0.92%	6,077 38.05%	4,035 25.26%	480 3.01%	380	79.27%	1.09	3:22
	20	5,367 33.60%	149 0.93%	6,242 39.08%	4,196 26.27%	20 0.13%	20	99.86%	0.79	1:13

3. ExMAS-SimFLEX Framework

3.1. Data

3.1.1. Demographics

For our report, we used real-world data on the population distribution within the city.

Figure 10 presents the population distribution of Krakow, while Figure 11 shows residential locations (address points).

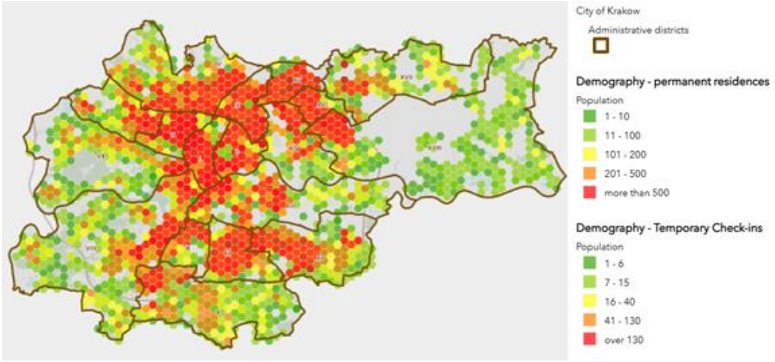

Figure 10: Demographics of Krakow[1]

For our analysis, we required specific address points along with the associated number of residents. We based our work on demographic data from October 2023, provided by ZTP Krakow.

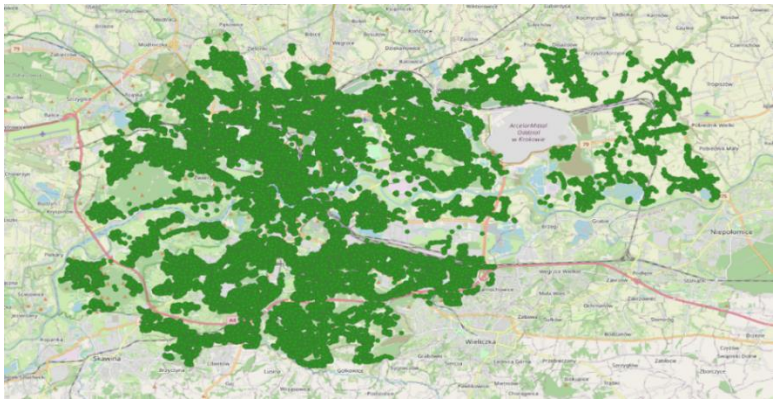

Figure 11: Address points

3.1.2. Origin-Destination Matrix (ODM)

To analyse the demand for NSM at a microscopic level, we used a traffic model (the number of trips during peak hours between city zones) obtained from the Krakow Municipality. These data allow us to determine for each starting area the distribution of its target destinations with the help of disaggregating macro-level

[1] The data is available at: msip.um.krakow.pl.

data from the ODM into synthetic agents representing individual travellers. This information is used in the second stage of our analysis.

3.1.3. Road network in the city

We rely on publicly available OpenStreetMap (OSM) data, which enables detailed analysis of travel routes by car (based on a parametrised detailed road network) and on foot (taking into account pedestrian pathways).

3.1.4. Public transport routing

The trip planning software OpenTripPlanner (OTP)[2] is used to integrate feeder buses with public transport networks and to compare the performance of this integrated system with a public transport-only alternative. It serves as a public transport routing tool, incorporating different transport modes, such as buses, trains, trams, etc. The OTP tool finds the optimal routes based on real-time and scheduled transit data, considering travel time, number of transfers, and walking distances.

3.1.5. GTFS

We use publicly available General Transit Feed Specification (GTFS) files[3], updated as of March 2024, provided by the city for PT trip planners (such as Google Maps and JakDojade). These files allow us to search for the optimal connections for a given pair of coordinates (origin-destination) and departure time. The dataset includes bus and tram timetables and stops, but unfortunately does not cover rail services within the city (SKA), which could be relevant for the Bronowice area.

3.1.6. Algorithm NSM

NSM is a demand-responsive transport service when travellers share their rides with other travellers who travel to / from the same hub. Our analysis focusses on the morning peak hour. During this period, the operator collects requests from travellers who wish to reach the hub and dispatches feeders (on-demand buses). These on-demand small-capacity buses will pick up travellers from designated pick-up points (stops) for pooled rides and drive them to transfer points (hubs) such as high-frequency tram/train stops of (Figure 12). From there, travellers reach their destination by regular and already efficient public transport.

[2] https://www.opentripplanner.org/

[3] The data is available at:https://gtfs.ztp.krakow.pl/

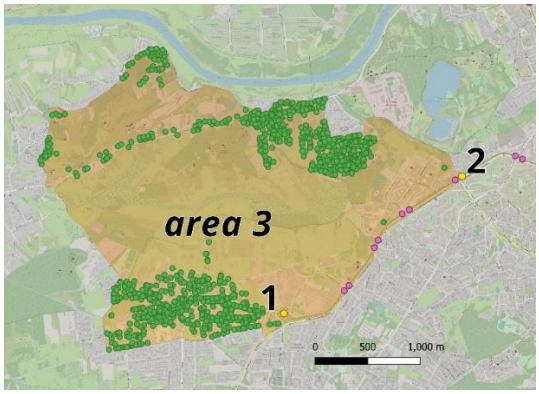

Figure 12: Example of the spatial distribution of address points (green), tram stops (in pink), and light rail hubs (yellow) for Area3

The optimization task for NSM is to find a route that allows for the picking up travellers from multiple pick-up points and transporting them to the destination while ensuring sufficient comfort (i.e., minimal delay compared to a trip without NSM). We rely on the ExMAS algorithm (Kucharski & Cats, 2020). As a fundamental assumption, we require that shared rides (consisting of more than one pick-up point) must remain attractive to travellers (i.e., their utility of NSM travel is greater than an alternative travel, e.g. by PT or own car).

Based on the solution, we can calculate key NSM characteristics (comparing a single vehicle with multiple pick-up points to a single vehicle with a single pick-up point), such as:

- Reduction in vehicle kilometres travelled (ΔT_v).
- Increase (decrease) in passenger comfort (ΔU_p).
- Average vehicle occupancy (O)[4].

The algorithm is applied in both stages of our analysis.

3.2. 'ORIGIN-HUB' ANALYSIS

In the first simplified stage of the analysis, we focus on identifying the best hubs for each area and comparing the NSM potential across different areas. We compare 12 areas, each with one to three corresponding hub[5].

At this stage, we only consider the population potential. We do not take into account travel motivations, destinations, the existing modal split or the current PT network. We assume that every traveller who uses the system wants to reach a hub. We evaluate the operating parameters of the system accordingly.

All these simplifications used in the first stage of the analysis disappear in the second stage, where (for two selected areas indicated in the first part) we take into account in detail the traffic patterns and the public transport offer.

3.2.1. Areas

The selected areas and corresponding hubs (including tram stops) are shown in Figure 13.

[4] Detailed description in our study Shulika et al., 2024 .
[5] Detailed description in our study Shulika et al., 2025 .

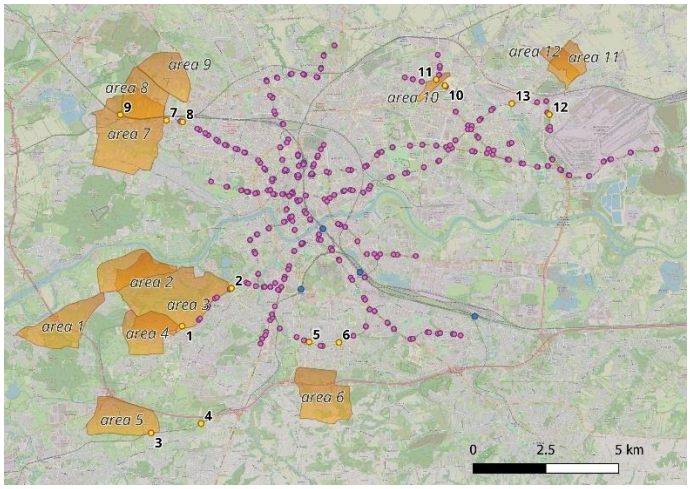

Figure 13: Krakow preselected areas (in orange) with corresponding hubs (in yellow), tram stops (in pink) and train stops (in blue)

3.2.2. Methodology

We propose a parameter-free approach that utilises probabilistic demand fractions to simulate potential demand. We consider a fraction of the residents who must be interested in the service sufficient to achieve the minimum required thresholds of a new service efficiency. Areas with the lowest resident interest required to meet efficiency thresholds are considered to be the most favourable for service implementation. After identifying the most promising area and hub combination, we establish benchmarks to understand the service's early-stage performance. This approach enables us to evaluate the potential of a new on-demand pooled transit feeder service and identify the most promising area for its launch, even without actual demand data. Our approach to selecting a preferred area to implement integrated on-demand pooled transit feeders with PT is illustrated in Figure 14.

The process starts with data collection, including population distribution, road network, candidate areas, and public transport hubs (one or more for each area). For each combination of pre-selected area and hub location(s), and at each demand fraction level and replication, we apply the utility-based ExMAS algorithm to match travellers to the pooled rides.

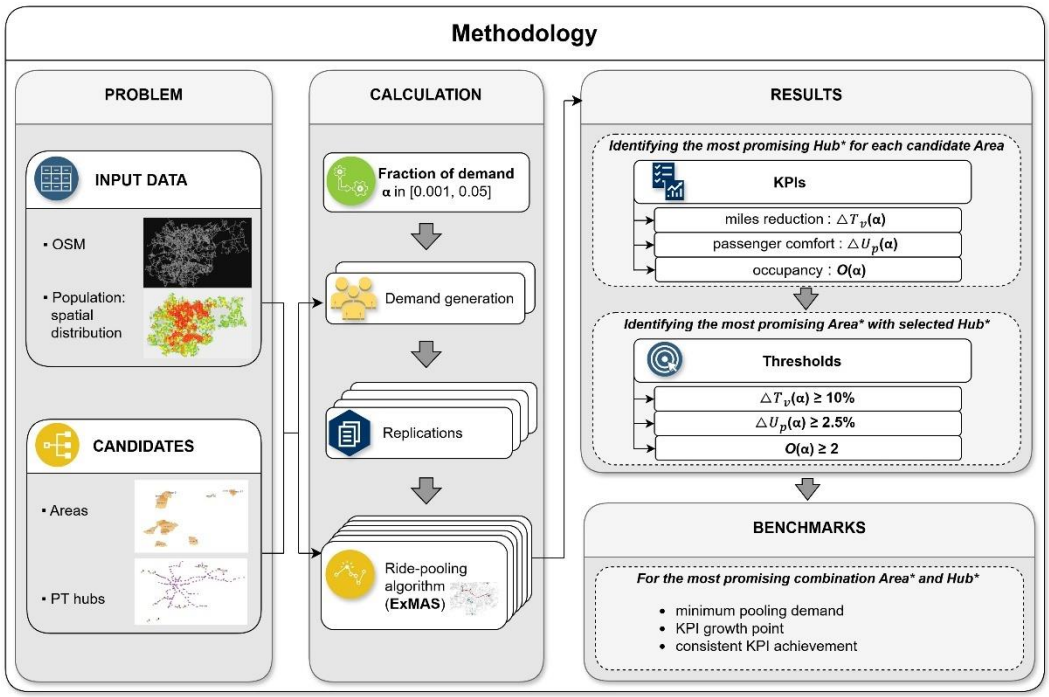

Figure 14: An overview of the applied methodology for selecting a preferred area to implement on-demand pooled transit feeders (NSM service)

We start with a scenario where all travellers use individual ride-hailing. Then we determine the extent to which ride-hailing can be replaced by ride-pooling while maintaining traveller satisfaction and vehicle mileage reduction. Travellers who cannot be efficiently matched on shared rides continue to use individual ride-hailing. Next, we evaluate the potential of on-demand service using three KPIs: mileage reduction, passenger satisfaction, and occupancy. We assess the performance of the system using KPIs derived from the simulation results.

Initially, we assess the progression of these KPIs across varying demand fractions, identifying the most promising hub within each preselected area. Following this, we compare the candidate areas by identifying the proportion of residents (fraction of demand) who must be interested in the service, denoted as the level ***α***, to meet the following minimum efficiency thresholds required to launch the new service:

- ΔT_v(vehicle hours reduction) ≥ 0.1: the launching of shared rides (instead of individual ones) allows for a reduction of vehicle kilometres by at least 10%;
- ΔU_p (travellers utility gains) ≥ 0.025: passenger comfort improves by at least 2.5% compared to individual travel. For the analysed scenario of free on-demand bus service, this measure ensures that passengers do not encounter significant discomfort associated with a new service;
- O (occupancy) ≥2: the average vehicle occupancy exceeds 2.

After selecting the most promising combination of area and hub, we go deeper to establish the following benchmarks for the selected area and hub combination, providing the municipality with valuable insights into what to expect when initiating the new service: the minimum demand fraction needed to successfully pool travellers into shared rides, the demand level at which ride-pooling potential starts to grow, and the demand level necessary for the service to consistently meet its key performance indicators. Scripts for reproducible results are available in the public repository (Github).

3.2.3. Results

Detailed results are presented in **Erreur ! Source du renvoi introuvable.**5. For each area-hub combination, we indicate the minimum NSM interest threshold (α) required to achieve each of the three efficiency levels. The minimum population fraction value '-' (Table 15) indicates that the key NSM characteristics did not meet any efficiency thresholds for α between 0% and 5%[6].

Table 15: Ranking of candidate areas

Area	Hub	Threshold						Total KPI rank score	Final area rank
		$\Delta T_v \geq 0.1$		$\Delta U_p \geq 0.025$		$O \geq 2$			
		α	rank	α	rank	α	rank		
1	1.'Czerwone Maki P+R'	0.005	11	0.01	4	0.02	4	29	10
2	2.'Norymberska'	0.002	4	0.02	7	0.02	4	15	6
3	1.'Czerwone Maki P+R'	0.001	1	0.009	2	0.01	2	5	2
4	1.'Czerwone Maki P+R'	0.003	10	0.05	11	-	11	32	11
5	1.'Czerwone Maki P+R'	0.002	4	0.009	2	0.01	2	8	3
6	6.'Nowosadecka'	0.002	4	0.01	4	0.02	4	12	5
7	9.'Kraków Mydlniki (PKP)'	0.001	1	0.01	4	0.02	4	9	4
8	9.'Kraków Mydlniki (PKP)'	0.002	4	0.02	7	0.02	4	15	6
9	9.'Kraków Mydlniki (PKP)'	0.001	1	0.005	1	0.007	1	3	1
10	10.'Dunikowskiego'	0.02	12	-	12	-	11	35	12
11	12.'Zajezdnia Nowa Huta'	0.002	4	0.02	7	0.02	4	15	6
12	12.'Zajezdnia Nowa Huta'	0.002	4	0.02	7	0.03	10	21	9

Erreur ! Source du renvoi introuvable.15 illustrates the hub selection process using our approach, with Area 3 as an example. In this figure, the higher line indicates superior performance for hub 1 compared to hub 2, establishing hub 1 as the preferred hub based on key performance indicators. Similarly, in **Erreur ! Source du renvoi introuvable.**16, the line for the most promising Area 9 frequently appears above the others, indicating its higher potential of the ride-pooling service. This visual ranking illustrates how the method supports data-driven decision-making by clearly delineating areas and hubs that best align with the new service goals.

[6] For area 9 we considered the upper range of 3%, but it does not matter, as the efficiency thresholds were exceeded earlier.

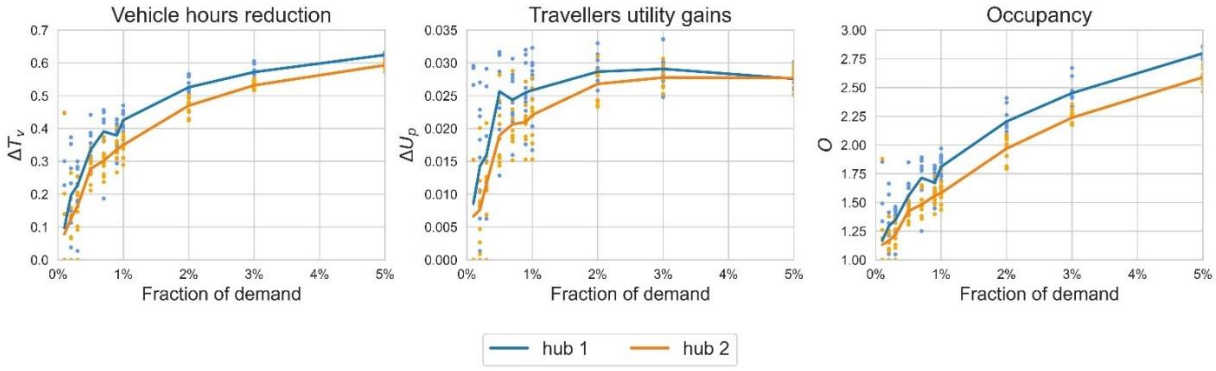

Figure 15: Three key performance indicators of ride-pooling plotted against the fraction of demand for Area 3. Lines represent the average performance across multiple simulations, while dots represent individual simulation results. Both hubs in Area~3 showed similar trends, but hub 1 has a slight edge in potential

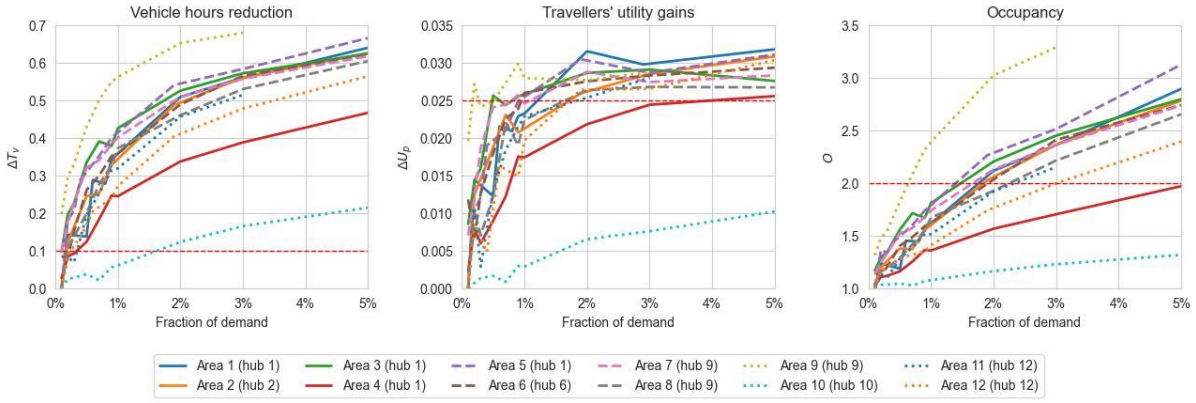

Figure 16: Three key performance indicators of ride-pooling plotted against the fraction of demand for pre-selected areas and the most promising corresponding hubs. Horizontal dashed red lines represent the set thresholds

The results identify Area 9, paired with Kraków Mydlniki (PKP), as the most promising candidate (Table 15). Higher-ranking areas, such as Area 9 (Bronowice) and Area 3 (Skotniki), exhibit a favourable balance of population density, hub distance, and infrastructure suitability for pooled transit. In contrast, lower-ranking areas, such as Area 10, with a high population density but close proximity to the Dunikowskiego hub, demonstrate the least potential, as the need for additional feeder services decreases with shorter distances to public transit options.

For the top-ranked Area 9, we establish three benchmarks to assess early-stage service performance (Figure 17). Ride-pooling potential becomes significant at demand levels of 0.05%, with efficiency thresholds met at 0.1%, 0.5%, and 0.7% for the three KPIs. These benchmarks guide municipal planning, enabling phased implementation aligned with demand growth projections.

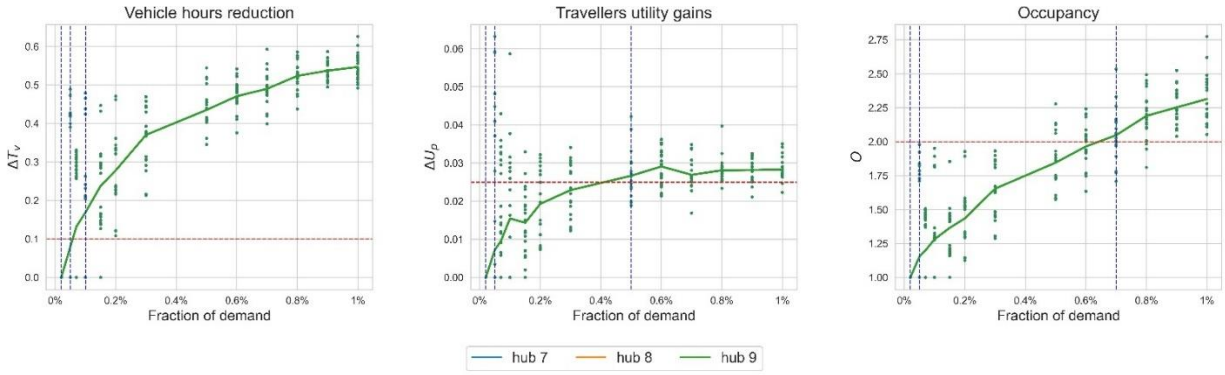

Figure 17: KPIs and three benchmarks for the combination of Area 9 and Hub 9 'Kraków Mydlniki (PKP)', plotted against demand levels. Pooled trips start at a demand fraction of 0.025%, with a ride-pooling potential increasing from 0.05%, and three KPIs reaching target levels at 0.1%, 0.5%, and 0.7%, respectively. Horizontal dashed red lines indicate established KPI thresholds, while vertical blue lines mark three benchmarks

3.2.4. Limitations

Despite its merits, our study has certain limitations. The ExMAS algorithm is limited to point-to-point ride-hailing, assessed only in comparison to solo ride-hailing. Additionally, demand must be predetermined, and the fleet is not explicitly managed.

The experiment was conducted in Krakow, a mid-sized European city, using a medium-scale sample. Our analysis focusses solely on the first-mile ride from pick-up points to hubs, simplifying the model, as the efficiency of on-demand pooled transit depends on the entire trip taken by travellers. Some areas, particularly those farther from the centre, may experience longer travel distances.

The analysis also assumes a single predetermined hub for all travellers leaving the area, without considering individual hub selection. Furthermore, the study only considers the population potential, ignoring factors such as travel motivations, goals, current transportation habits, and existing public transport options. We assume that all travellers intending to use the system aim to reach the hub, and we focus on the system operation parameters.

Future research should consider the entire journey, including both the feeder segment and the public transit segment, to provide a more comprehensive assessment of the attractiveness of public transport. Examining factors such as demographics and time-of-day variations could improve demand estimation and service predictability. Further studies could also test the scalability of this method in different urban contexts, allowing cross-regional comparisons and deeper insight into the role of on-demand feeder services in complementing traditional public transit.

By applying this methodology to various urban settings, we can identify universal patterns in shared mobility potential and inform the development of effective on-demand feeder services.

3.3. Analysis 'Integrated NSM Vs PT'

In this stage, we consider a scenario in which travellers can choose between two available options:

1. F - an integrated trip, where an on-demand bus service (NSM) as a first- or last-mile connection from an origin to a designated hub. From the hub, public transport completes the trip to the final destination (the trip from the hub to the final destination is denoted HD).
2. PT - a trip entirely serviced by the existing public transport system from the origin to the destination (denoted as OD).

We introduce SimFLEX (Simulation Framework for Feeder Location Evaluation), a methodology specifically designed to assess the feasibility and effectiveness of on-demand feeder bus services in diverse urban conditions[7]. By leveraging spatial, socio-demographic, and transportation-specific data of the analysed region, the method enables the computation of various KPIs for a given area-hub combination, allowing a comparative analysis to identify the most suitable urban area for service implementation (Figure 18). The proposed key indicators capture both operational aspects of the feeder buses, such as vehicle-hours travelled, passenger-hours, and vehicle occupancy, as well as utility-based metrics that reflect the effectiveness of the overall transport system that includes feeders as first- or last-mile solutions. These are service attractiveness, waiting time reduction, and overall added value.

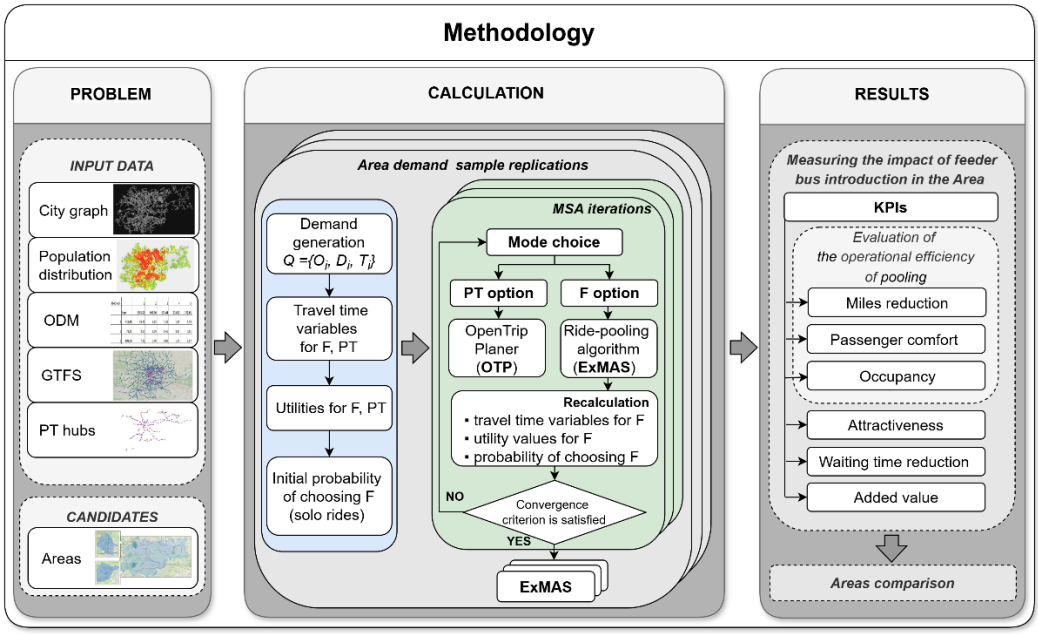

Figure 18: SimFLEX computes service performance with the following methodology. For a given service area and hub location it uses widely available inputs (such as network graph, GTFS, population distribution and OD-matrices), and it runs a series of microsimulations to obtain a wide range of performance indicators. First, we sample microscopic demand patterns for services from macroscopic models. For each single demand realisation, we simulate the travellers learning process, when they experience system performance (with unknown travel times due to detours, here sampled with ExMAS). After stabilisation (when each travellers expectations meet the realisations), we simulate extra runs to compute indicators from the stabilised system. This concludes a single run of SimFLEX, which can then be replicated (for different realisations of the demand), or used for comparisons (between areas, hubs, parameterizations, etc.).

Beyond evaluating feeder service effectiveness, SimFLEX is useful in comparing different urban areas, to introduce services where they offer the highest benefits. Additionally, it enables a sensitivity analysis of key performance indicators, notably those not well estimated (like an alternative specific constant or transfer penalty). To achieve these objectives, SimFLEX integrates a combination of computational tools, optimisation techniques and analytical methods that together enable a comprehensive assessment of feeder system performance. We propose SimFLEX as a comprehensive decision support tool developed to address the lack of location-specific methods to evaluate the potential impact and feasibility of novel transportation services. The source code for the methodology is freely accessible on GitHub and provides an adaptable

[7] Detailed description in our study Vasiutina et al., 2025 .

foundation, enabling analysts to conduct analytically guided implementation decisions by simulating diverse scenarios, quantifying performance metrics, and analysing potential outcomes.

3.3.1. Areas

We focus on two revised areas: Bronowice and Skotniki (Table 16), as shown **Erreur ! Source du renvoi introuvable.**19.

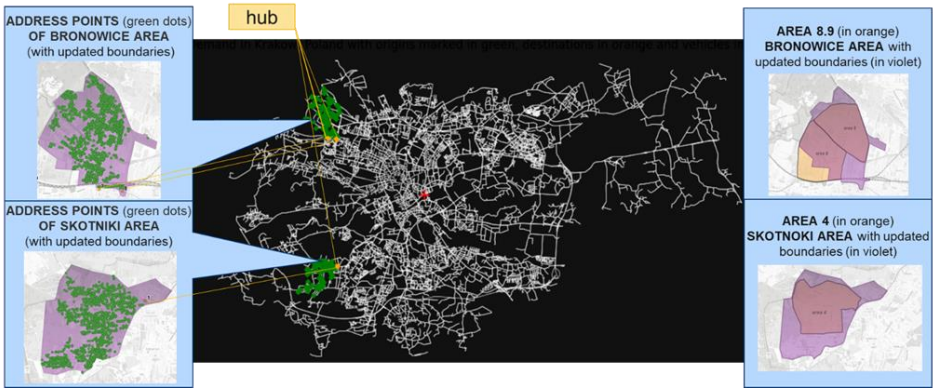

Figure 19: Updated Bronowice and Skotniki areas with marked address points (green) and hubs (yellow).

Revision of areas resulted in boundary changes for address points and population data updates. Due to the lack of GTFS data on train departures for Krakow Mydlniki PKP, we exclude this hub as a potential hub for Bronowice and consider the remaining two hubs.

Table 16: Revised areas and hubs considered

Predefined Area		Population	Hub
8	Bronowice and Bronowice Wielkie (split into two zones)	1836	Bronowice Małe
			Krakow Mydlniki PKP
9	Bronowice and Bronowice Wielkie (split into two zones)	4002	Bronowice Małe
			Bronowice SKA
			Krakow Mydlniki PKP
Bronowice		**8390**	**Bronowice Małe Bronowice SKA**
3	Skotniki Area	3719	Czerwone Maki P+R
Skotniki		**6070**	**Czerwone Maki P+R**

The basic statistics of the surveyed areas are given in Table 17.

Table 17: Basic statistics of the surveyed areas.

Statistics	Bronowice area	Skotniki area
Number of address points	1227	1229
Number of residents	8390	6070
Number of planned collection points	100	83

3.3.2. Methodology

SimFLEX is an iterative framework for assessing the effectiveness of on-demand feeder bus services. (Figure 18). It uses spatial and sociodemographic characteristics of the analysed region, along with existing transportation system parameters, to estimate travel demand via a discrete choice model for mode selection. SimFLEX is a two-loop framework: the outer loop generates multiple demand scenarios for analysis by sampling travel demand for feeder services, while the inner loop performs MSA iterations to model the traveller learning process and achieve system stabilization. First, this involves refining macroscopic data to generate detailed individual-level travel demand for agent-based simulations, which mimic the behaviour and interactions of individual agents (here, travellers). A discrete choice model allocates travellers among available transport modes based on utility functions that consider travel time, cost, and convenience factors. The daily demand for the feeder bus is computed through this mode-choice process, reflecting user preferences in response to service attributes and learning. For each demand realization (i.e., a single sampled demand scenario), the inner loop simulates the traveller learning process, updating travel time expectations and mode choices until system stabilization is achieved. Consequently, dependent on travel times, utilities mode choice probabilities are recalculated on each iteration of MSA for travellers selecting shared rides. The attributes of shared trips, including travel times, are obtained using the ExMAS framework. For trips performed by public transport, the trip parameters, such as duration, walking distance, transit, and waiting time - are obtained using the OTP software.

Once the system stabilizes, we perform additional iterations to estimate key performance indicators. In the outer loop, we perform multiple demand replications to capture variability in traveller behaviour and network parameters. These resulting metrics are then used to assess the effectiveness of introducing the feeder services in different urban areas and to conduct the comparative analysis.

SimFLEX is designed to be adaptable, allowing researchers and practitioners to modify input data (such as varying transit schedules, population distributions, and operational constraints), assumptions, and models based on specific case study requirements. The modular framework enables integration with alternative demand estimation techniques, learning methods, and routing algorithms, making it applicable to various urban settings and transportation networks. To ensure reproducibility, the complete SimFLEX implementation, including input data and computational functions, is publicly available in the GitHub repository. This allows for method validation, extensions, and further experimentation by researchers, urban planners, and policymakers interested in evaluating feeder service feasibility in different cities.

3.3.3. Performance indicators

After the system reaches a stable state, we perform additional iterations (days) to obtain statistically reliable performance metrics: operational KPIs for the feeder bus service and effectiveness indicators for the integrated feeder-public transport system. Specifically, we derive the following indicators:

- Decrease in total vehicle-hours (ΔT_v): quantifies the change in efficiency for service operators by measuring the reduction in total vehicle operation time due to ride-pooling.
- Increase in total passenger-hours travelled (ΔT_p): reflects the trade-off for travellers by indicating any increase in total travel time when shifting from solo rides to pooled services.
- Vehicle occupancy (O): assesses pooling effectiveness by comparing passenger hours with vehicle-hours, illustrating how efficiently shared rides utilize vehicle capacity.

To evaluate the overall effectiveness of the integrated feeder transit system and conduct a comparative analysis of urban areas, we incorporate the utilities for the feeder (U_F) and the direct public transport trip options (U_{PT}^{OD}), as well as the reduction in waiting time for public transport: for the feeder (t_{wt}^{HD}) and for the public transport trip option (t_{wt}^{OD}). Based on these SimFLEX outputs, we compute the following integrated system effectiveness indicators, expressed as average values over the *n*-th demand sample replication:

- Attractiveness of the integrated feeder service: measuring the relative desirability of the travel options based on the utility differences;
- Reduction in waiting time for public transport: evaluates the effect of implementing feeder services on public transport waiting times;
- Overall added value: provides a quantitative measure of the improvement in traveller experience when using the new service (Cats et al., 2022).

The proposed performance metrics are subsequently used for comparative analyses across different urban areas. This analysis helps identify locations where the introduction of feeder services would provide the greatest improvements in public transportation efficiency, accessibility, and overall user satisfaction.

3.3.4. Analysis of system stabilisation and traveller learning

Figure 20 illustrates a representative simulation run, depicting the stabilization of the average expected travel times through MSA iterations for Bronowice and Skotniki areas. The figure provides a visual representation of the learning process for a single-demand replication, showing how expected travel times stabilize over the iterations.

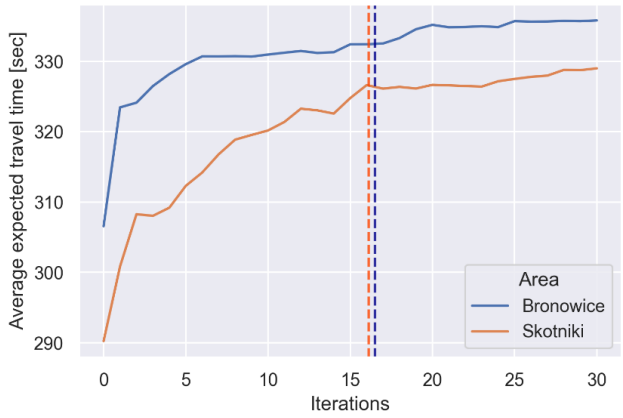

Figure 20: Stabilization of the expected average travel times through MSA over 30 iterations for Bronowice and Skotniki areas. The initial travel times at the first iteration correspond to the raw travel times estimated for solo rides, before any learning or adaptation occurs. Travel times initially vary as travellers adapt to the new feeder bus system, but gradually stabilize as the system converges. Both areas show a similar trend of initial variation followed by convergence, though the specific travel times and rates of convergence differ. The dashed lines represent the mean value of the convergence iteration, for Bronowice in blue and for Skotniki in orange, indicating the average convergence iteration around 16 (the mean over 100 demand replications)

3.3.5. Feeder choice probabilities

Figure 21 presents the probability distribution of choosing the feeder bus integrated with the public transport system for a single demand replication for the analysed areas, Bronowice and Skotniki. In both regions, a substantial portion of travellers exhibit a low probability of choosing the feeder service, with approximately 57% in Bronowice and 51% in Skotniki, indicating that in many cases, no travellers opted for the feeder bus service in the analysed replication. On the other hand, a notable fraction of travellers (around 17% in Skotniki and 10% in Bronowice) show a high probability (close to 1) of choosing the feeder service, with a slightly greater prevalence in Skotniki. The remaining probability values are distributed across intermediate ranges, with Skotniki displaying a slightly higher frequency of probabilities above 0.5. Additionally, the mean probability of the feeder choice, marked by dashed lines, is lower in Bronowice (around 0.23) compared to Skotniki (nearly 0.35), suggesting that on average the feeder service is a more attractive option for travellers in Skotniki.

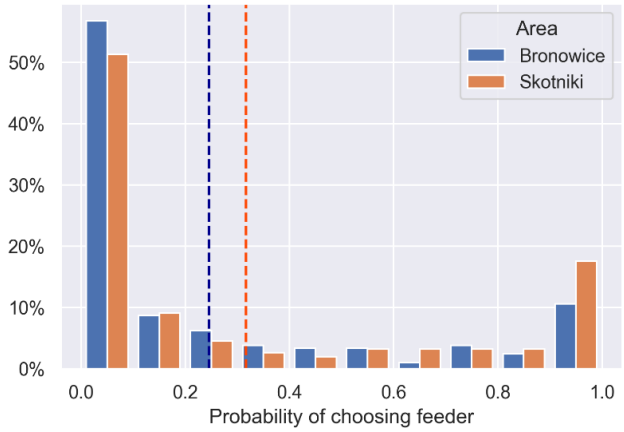

Figure 21: The probability distribution of feeder bus service choice in Bronowice and Skotniki for a single demand replication. The figure shows that in Bronowice, slightly more than half of travellers (57%) and in Skotniki roughly half (51%) have almost 0% probability of choosing the feeder. Meanwhile, a tenth of travellers (10%) in Bronowice and approximately a sixth (17%) in Skotniki have a 100% probability of choosing the feeder. The mean probability of feeder choice is 0.23 for Bronowice and 0.35 for Skotniki, as indicated by the dashed lines. These values highlight differences in the feeder service adoption between the two areas in this single demand scenario.

3.3.6. Comparative analysis of study areas

The primary metrics that describe the effectiveness of the feeder system integrated with public transport, used for Bronowice and Skotniki comparison (e.g., the probability of choosing feeders, feeder attractiveness, reduction in waiting time, overall added value along with their respective variances) are summarized in Table 18, while their distributions are depicted in Fig. 22. These indicators are the mean values of 100 demand replications, calculated after the system's stabilization (30 MSA iterations and an additional 20 iterations).

Table 18: Evaluation of feeder service potential: Comparison of Bronowice and Skotniki feeder system level indicators comparison

Area	P_F	$Var(P_F)$	ΔA	$Var(\Delta A)$	ΔT	$Var(\Delta T)$	ΔV	$Var(\Delta V)$
Bronowice	0.248	0.003	-2.690	0.034	**65.904**	107.691	0.568	0.0001
Skotniki	**0.265**	0.003	**-1.966**	0.036	28.925	224.955	**0.600**	0.0001

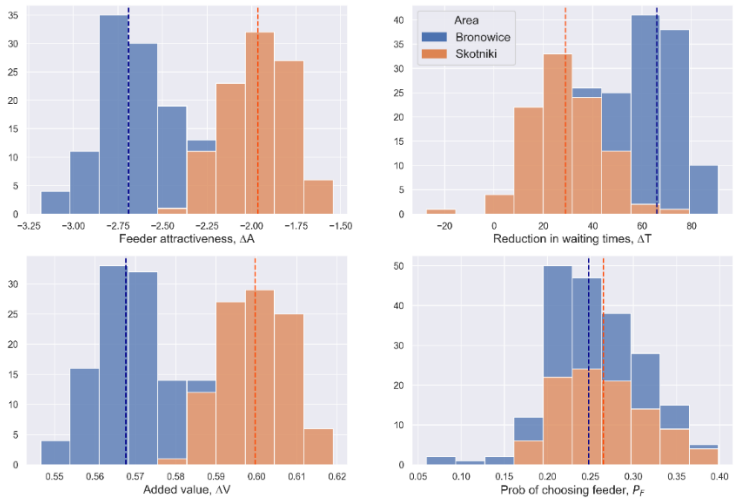

Figure 22: Distribution of the KPIs of the integrated feeder system across the 100 demand replications with their mean values (illustrated with dashed lines) for the Bronowice and Skotniki areas. Skotniki generally exhibits higher (less negative) values for feeder attractiveness, suggesting that it is perceived as more attractive compared to Bronowice. In terms of waiting-time reduction, Bronowice shows a distribution concentrated at higher positive values, indicating a more significant reduction in waiting times. For added value, Skotniki has a higher mean value and a narrower data spread, implying a slightly better added value compared to Bronowice, which shows a similar spread but with slightly lower mean values. Finally, the histograms of the probability of choosing the feeder indicate that Skotniki has a distribution skewed towards higher probabilities, suggesting a greater likelihood of choosing the feeder service compared to Bronowice.

For our comparative analysis, we also investigate the operational performance of the feeder buses across the analysed areas, Bronowice and Skotniki, by using KPIs derived with the ExMAS algorithm. Based on simulation results obtained under the configuration, the histograms (Fig. 23) depict the distributions of relative KPI changes for the feeder bus service in comparison to the solo ride baseline (calculated based on Shulika et al.2024).

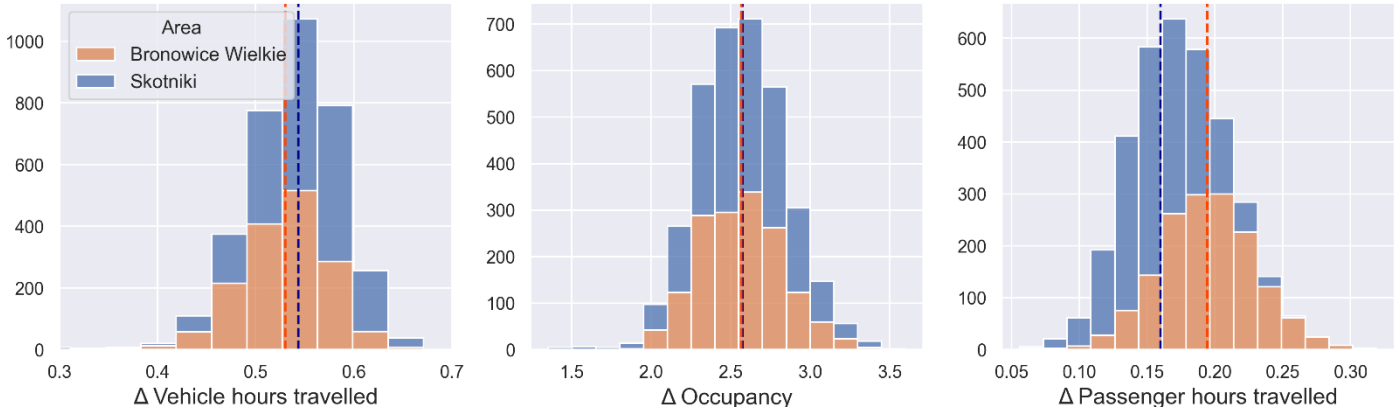

Figure 23: Distributions of relative changes in KPIs (with mean values depicted by dashed lines) following the implementation of feeder bus services compared to solo rides in Bronowice and Skotniki. For vehicle-hours travelled, Bronowice displays a distribution with a higher mean increase, indicating a greater impact on vehicle operation. Similarly, Bronowice vehicle occupancy is higher on average in Bronowice, suggesting better utilisation of vehicle capacity. Conversely, for passenger-hours travelled, Skotniki shows a distribution with a higher mean increase, indicating a greater impact on overall passenger travel time despite the similar distribution shapes. These differences highlight the varying effects of the feeder system across the two areas, with Bronowice experiencing improved vehicle utilization and Skotniki showing a larger increase in passenger travel time.

3.3.7. Sensitivity analysis of KPIs to ASC variations

A significant advantage of the SimFLEX methodology is that it enables the analysis of the sensitivity of outputs to variations in the alternative-specific constant. ASCs represent the average effect of unobserved factors on mode choice utility, and they are typically unknown a priori for new service implementations.

To conduct the sensitivity analysis, we perform simulations with five area sample replications for each ASC value (N = 5). We vary the ASC parameter across a range from 0 to 5, with 50 evenly spaced points sampled from a uniform distribution within this interval. This approach allows for the exploration of a wide range of potential user preferences and assessing the stability and reliability of method outputs.

The results of this analysis, presented in Fig. 24, reveal distinct patterns in how the choice of ASC affects the primary KPIs across the two analysed areas: Bronowice and Skotniki.

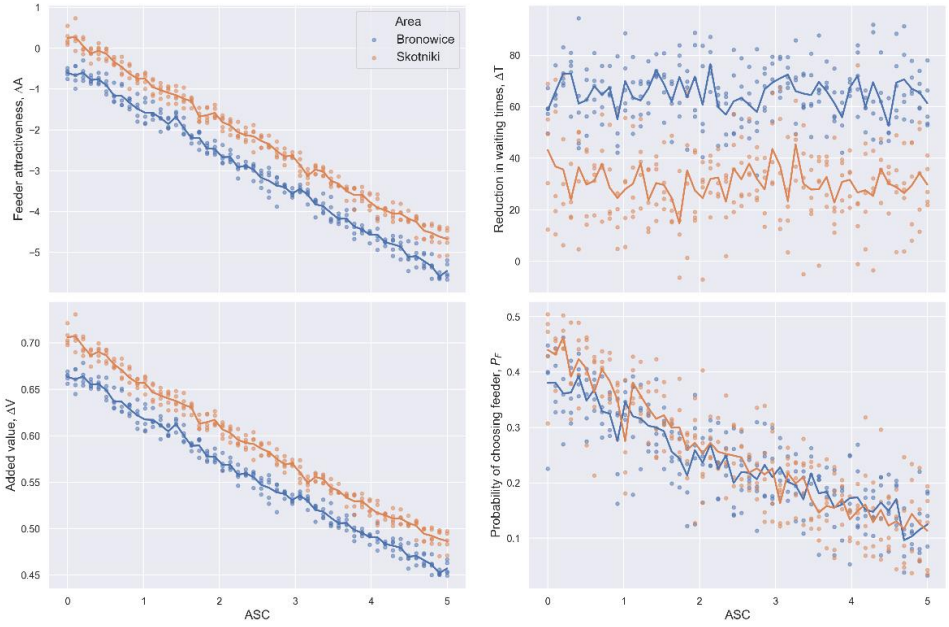

Figure 24: The results remain consistent when the behavioural parameter (ASC) varies. The impact of ASC variations (ranging from 0 to 5) on four key metrics: feeder attractiveness, reduction in public transport waiting times, overall added value, and the probability of choosing the feeder service. Each data point represents the average value of the respective metric across five replications per ASC value, with solid lines indicating trends for each area

3.4. Discussion

Our first analysis indicated Bronowice (area 9) due to its larger population (Table 15). With the fixed level of the population fraction interested in the service, the demand for SUM was significantly higher. For the revised areas, the difference between the number of inhabitants decreased. Additionally, we conducted more detailed analyses considering the full paths from the beginning to the end of the journey. In the current form, we can indicate that **Skotniki** is the area with greater potential, based on the following received data.

In the comparative analysis of the two revised areas (Bronowice and Skotniki), we took into account four main measures: the probability of using NSM, the attractiveness of NSM, reduction in waiting time for PT, and added value (Table 16).

A quantitative comparison of the KPIs of the integrated feeder transit system between Bronowice and Skotniki reveals significant differences in service potential (Table 16). Skotniki exhibits a higher average probability

of feeder service selection (0.265 compared to 0.248 in Bronowice) and a greater (approximately by 30%) feeder service attractiveness (-1.966 compared to -2.690 in Bronowice). Furthermore, Skotniki demonstrates a higher added value (up to 7%) (0.600 compared to 0.568 in Bronowice). In contrast, Bronowice shows substantially greater potential for reducing traveller waiting times (nearly by 77%), with an average reduction of 65.904 seconds compared to 28.925 seconds in Skotniki. These numerical differences support the data in Figure 22, highlighting the distinct advantages and challenges of feeder service implementation in each area.

The analysis of KPIs reveals distinct patterns in the potential for on-demand feeder bus service implementation in Bronowice and Skotniki (Fig. 22). Skotniki demonstrates a higher average probability of users choosing the feeder service, coupled with a greater service attractiveness and added value. Conversely, Bronowice exhibits a significant potential for reducing traveller waiting times for public transport. However, the lower average probability of feeder service selection and the distribution of choice probabilities suggest that the service may face challenges in attracting users. The concentration of zero-probability values in Bronowice indicates that a substantial portion of the population may consistently opt for alternative modes of transport. Based on these findings, Skotniki appears to be a more suitable area for the feeder service implementation. The higher average probability of feeder selection, greater attractiveness, and added value suggest a greater likelihood of successful integration and user adoption of feeders in this area.

Based on the analysis of operational KPIs, the implementation of feeder bus services in Bronowice and Skotniki reveals distinct impacts. Bronowice exhibits a higher average vehicle occupancy and increased vehicle-hours travelled. However, Skotniki shows a larger increase in passenger-hours travelled, suggesting potentially greater delays for travellers. Therefore, while Bronowice achieves better vehicle utilization, Skotniki results in an increased time burden for travellers. In terms of operational efficiency, neither area clearly outperforms the other, as each demonstrates a trade-off between vehicle utilization and passenger travel time.

To further assess the stability and reliability of our method outputs, we conduct a sensitivity analysis across a range of ASC values. The results consistently confirm Skotniki as the superior candidate for the feeder service implementation across all ASC variations (Fig. 24). This consistency strengthens our findings that Skotniki's advantages result from differences in travel demand and service characteristics, not specific ASC assumptions.

However, the decision whether Bronowice or Skotniki is a more suitable area for the implementation of the feeder service depends on the priorities of the transportation authorities and the specific goals of the deployment of the service. If minimizing travel time is the primary objective, Bronowice appears to be the preferable choice. However, this choice can cause lower overall service adoption of services and potential long-term benefits due to lower attractiveness of services.

Conversely, if increasing accessibility and consistent service usage are prioritized, Skotniki is a more appropriate choice. The higher feeder service attractiveness and the added value in Skotniki suggest a greater likelihood of consistent ridership and integration with the existing transportation system. The simulation code and input data are available in the repository Github repository.

3.5. How to use it

A link to the code repository: SimFLEX: Methodology for comparative analysis of urban areas for implementing new on-demand feeder bus services

Getting Started

- Prepared the data (see datasets in Defining a Use Case).
- Start the OpenTripPlanner server before running the framework
- Define initial experiment configurations (the default file with configurations)

Defining a Use Case

Each use case must include the following datasets (in the `data/` directory):

- graphml file with road network graph (`Krakow.graphml`)
- Zoning data: city zones in `.geojson` format (e.g., `krk_zones.geojson`), centroids of the zones (e.g., `krk_centroid.geojson`)
- Urban area boundaries for analysis (e.g., `sum_areas_B+S.shp`)
- Demographics: CSV with address and population (e.g., `krk_demographic.csv`)
- Origin–Destination Matrix: OD demand (e.g., `krk_ODM.xlsx`), defining travel demands between zones
- Feeder hub coordinates: a dictionary with `(lat, lon)` tuples for each area
- zip with GTFS file for the area and date that we query (e.g., `gtfs.zip`)

Running Simulations

Once the files are ready and OTP is running, use the following steps in `simulations.ipynb`:

1. Load the street network and configuration file
2. Load and preprocess input data:
   - city zones, centroids, areas
   - demographics
   - OD matrices

3. Run the main simulation:

```
results = fncs.simulate_MSA(gdf_areas, df_demo, gdf_centroid,
                            od, od_probs, hubs, inData, params,
                            "http://localhost:8080/otp/routers/default/plan",
                            degree, N, max_iter, ASC, results_period)
```

Interpreting Results

Each simulation outputs key travel and behavioural indicators:

Table 19: SimFLEX outputs - key travel and behavioural indicators

Metric	Description
`tw_PT_OD`, `tw_PT_HD`	Waiting times for PT from O to D, from H to D
`u_PT_OD`, `u_SUM_OD`	Utilities of PT-only and feeder alternatives
`p_SUM`	Probability of choosing the feeder option
`ΔA`	Feeder attractiveness
`ΔV`	Added value
`avg_ts_pt`, `avg_ts_sh`	Average travel times by mode across iterations
`converged_is`	Convergence iteration index per run
`kpis_res`	ExMAS-based KPIs such as occupancy and vehicle hours

You can compute per-area:

- Averages and variances of indicators
- KPIs: passenger hours, vehicle hours, occupancy (via `ExMAS`)
- ASC sensitivity over multiple replications
- Travel time stabilization using the method of successive averages (MSA)

Output Files

- `results/main_results/*.csv`: average results, KPIs, ASC batch runs
- `results/images/*.png`: visualizations

Visualizations:

- KPI histograms by area
- Probability distribution of feeder choice
- ASC sensitivity analysis
- MSA convergence plots

Based on obtained results compare feeder service performance for each preselected area to interpret which areas benefit most from feeder deployment.

4. Conclusions

This deliverable presents two complementary frameworks for analysing ride-pooling services. The simulation-optimization framework is tailored for designing and operating ride-pooling fleets, while the ExMAS-SimFLEX framework supports planning the integration of on-demand feeder services with existing public transit systems.

Both frameworks have been applied in real-world contexts. The simulation-optimization framework was applied to the Yuvalim-Ganim neighbourhood in Jerusalem, with preliminary results presented to Israel's National Public Transit Authority to inform future deployment. The analysis underscores the importance of carefully defining service areas when operating with small fleets, confirms that low pooling demand can be met with smaller vehicles, and reveals an increasing returns effect – where fleet expansion significantly increases pooling and the number of served requests – highlighting the need for sufficiently large fleets to enable effective pooling.

The ExMAS-SimFLEX framework was deployed in Krakow to guide the selection of neighbourhoods for initial feeder service implementation. The analysis identified Skotniki and Bronowice as strong candidates, with Skotniki emerging as the more promising option due to its higher service attractiveness, user uptake, and added value – factors that support long-term adoption and integration. Although Bronowice showed greater potential for reducing waiting times and better vehicle efficiency, its lower overall service appeal may limit its effectiveness.

Both case studies are ongoing, and comprehensive findings will be included in the Living Labs deliverables at the conclusion of the SUM project.

5. References

Abouelela, M., Tirachini, A., Chaniotakis, E., & Antoniou, C. (2022). Characterizing the adoption and frequency of use of a pooled rides service. *Transportation research part C: emerging technologies*, *138*, 103632.

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

Akhtar, U., Ghasemi, F., & Kucharski, R. (2024). Optimizing Ride-Pooling Revenue: Pricing Strategies and Driver-Traveller Dynamics. *arXiv preprint arXiv:2403.13384*.

Alemi, F., Circella, G., Handy, S., & Mokhtarian, P. (2018). What influences travelers to use Uber? Exploring the factors affecting the adoption of on-demand ride services in California. *Travel Behaviour and Society*, *13*, 88-104.

Alonso-González, M. J., Cats, O., van Oort, N., Hoogendoorn-Lanser, S., & Hoogendoorn, S. (2021). What are the determinants of the willingness to share rides in pooled on-demand services?. *Transportation*, *48*(4), 1733-1765.

Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., & Rus, D. (2017). On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, *114*(3), 462-467.

Balac, M., Hörl, S., & Axhausen, K. W. (2020). Fleet sizing for pooled (automated) vehicle fleets. *Transportation Research Record*, *2674*(9), 168-176.

Bansal, P., Liu, Y., Daziano, R., & Samaranayake, S. (2019). Can Mobility-on-Demand services do better after discerning reliability preferences of riders?. *arXiv preprint arXiv:1904.07987*.

Bischoff, J., Kaddoura, I., Maciejewski, M., & Nagel, K. (2017). Simulation-based analysis of travel behavior changes using free-floating carsharing. *Transportation Research Procedia*, 25, 3345–3355. https://doi.org/10.1016/j.trpro.2017.05.283

Bongiovanni, C., Kaspi, M., Cordeau, J. F., & Geroliminis, N. (2022). A machine learning-driven two-phase metaheuristic for autonomous ridesharing operations. Transportation Research Part E: Logistics and Transportation Review, 165, 102835.

Bahrami, S., Nourinejad, M., Nesheli, M. M., & Yin, Y. (2022). Optimal composition of solo and pool services for on-demand ride-hailing. Transportation Research Part E: Logistics and Transportation Review, 161, 102680.

de Ruijter, A., Cats, O., Alonso-Mora, J., & Hoogendoorn, S. (2023). Ride-pooling adoption, efficiency and level of service under alternative demand, behavioural and pricing settings. *Transportation planning and technology*, *46*(4), 407-436.

Dias, F. F., Lavieri, P. S., Garikapati, V. M., Astroza, S., Pendyala, R. M., & Bhat, C. R. (2017). A behavioral choice model of the use of car-sharing and ride-sourcing services. *Transportation*, *44*, 1307-1323.

Engelhardt, R., Dandl, F., Syed, A. A., Zhang, Y., Fehn, F., Wolf, F., & Bogenberger, K. (2022). Fleetpy: A modular open-source simulation tool for mobility on-demand services. *arXiv preprint arXiv:2207.14246*.

Engelhardt, R., Mahmassani, H. S., & Bogenberger, K. (2023). Predictive vehicle repositioning for on-demand ride-pooling services. *arXiv preprint arXiv:2308.05507*.

Fagnant, D. J., and Kockelman, K. M. (2014). The Travel and Environmental Implications of Shared Autonomous Vehicles, Using Agent-Based Model Scenarios. *Transportation Res. Part C: Emerging Tech.* 40, 1–13. doi:10.1016/j.trc.2013.12.001

Fan, W., Gu, W., & Xu, M. (2024). Optimal design of ride-pooling as on-demand feeder services. *Transportation Research Part B: Methodological*, *185*, 102964.

Guo, X., & Zhao, J. (2024). Design of Transit-Centric Multimodal Urban Mobility System with Autonomous Mobility-on-Demand. *arXiv preprint arXiv:2404.05885*.

Hensher, D. A., Mulley, C., Ho, C., Wong, Y., Smith, G., & Nelson, J. D. (2020). *Understanding Mobility as a Service (MaaS): Past, present and future*. Elsevier.

Ho, S. C., Szeto, W. Y., Kuo, Y. H., Leung, J. M., Petering, M., & Tou, T. W. (2018). A survey of dial-a-ride problems: Literature review and recent developments. Transportation Research Part B: Methodological, 111, 395-421.

Hu, S., Hyland, M. F., Saha, R., Berkel, J. J., & Veen, G. V. (2024). Flexible Agent-based Modeling Framework to Evaluate Integrated Microtransit and Fixed-route Transit Designs: Mode Choice, Supernetworks, and Fleet Simulation. *arXiv preprint arXiv:2405.19527*.

Krueger, R., Rashidi, T. H., & Rose, J. M. (2016). Preferences for shared autonomous vehicles. *Transportation research part C: emerging technologies*, *69*, 343-355.

Lavieri, P. S., & Bhat, C. R. (2019). Modeling individuals' willingness to share trips with strangers in an autonomous vehicle future. *Transportation research part A: policy and practice*, *124*, 242-261.

Liu, Y., Bansal, P., Daziano, R., & Samaranayake, S. (2019). A framework to integrate mode choice in the design of mobility-on-demand systems. *Transportation Research Part C: Emerging Technologies*, *105*, 648-665.

Ma, S., Zheng, Y., & Wolfson, O. (2013, April). T-share: A large-scale dynamic taxi ridesharing service. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)* (pp. 410-421). IEEE.

Molenbruch, Y., Braekers, K., & Caris, A. (2017). Typology and literature review for dial-a-ride problems. Annals of Operations Research, 259, 295-325.

Nahmias-Biran, B. H., Oke, J. B., Kumar, N., Basak, K., Araldo, A., Seshadri, R., ... & Ben-Akiva, M. (2019). From traditional to automated mobility on demand: a comprehensive framework for modeling on-demand services in SimMobility. *Transportation Research Record*, *2673*(12), 15-29.

Nourinejad, M., & Roorda, M. J. (2016). Agent based model for dynamic ridesharing. *Transportation Research Part C: Emerging Technologies*, *64*, 117-132.

Omidvar, B., Lee, S., & Ramezani, M. (2022). Performance evaluation of electric shared autonomous vehicles in ride-pooling service using agent-based simulation. *Transportation Research Part D: Transport and Environment*, 105, 103222.

Pinto, H. K., Hyland, M. F., Mahmassani, H. S., & Verbas, I. Ö. (2019). Joint design of multimodal transit networks and shared autonomous mobility fleets. *Transportation Research Procedia*, *38*, 98-118.

Şahin, A., Sevim, İ., Albey, E., & Güler, M. G. (2022). A data-driven matching algorithm for ride pooling problem. *Computers & Operations Research*, *140*, 105666.

Shaheen, S., & Cohen, A. (2019). Shared ride services in North America: definitions, impacts, and the future of pooling. *Transport reviews*, *39*(4), 427-442.

Shulika, O., Bujak, M., Ghasemi, F., & Kucharski, R. (2024). Spatiotemporal variability of ride-pooling potential–half a year New York City experiment. *Journal of Transport Geography*, *114*, 103767.

Syed, A. A., Dandl, F., Kaltenhäuser, B., & Bogenberger, K. (2021). Density based distribution model for repositioning strategies of ride hailing services. *Frontiers in Future Transportation*, *2*, 681451.

Train, K. E. (2009). *Discrete choice methods with simulation*. Cambridge university press.

Tuncel, K., Koutsopoulos, H. N., & Ma, Z. (2023). An integrated ride-matching and vehicle-rebalancing model for shared mobility on-demand services. *Computers & Operations Research*, *159*, 106317.

Valadkhani, A. H., & Ramezani, M. (2023). Dynamic ride-sourcing systems for city-scale networks, Part II: Proactive vehicle repositioning. *Transportation Research Part C: Emerging Technologies*, *152*, 104159.

Walker, J. L., & Li, J. (2007). Latent lifestyle preferences and household location decisions. *Journal of Geographical Systems*, *9*, 77-101.

Wallar, A., Van Der Zee, M., Alonso-Mora, J., and Rus, D. (2018). "Vehicle Rebalancing for Mobility-On-Demand Systems with Ride-Sharing," in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) October 1–5, 2018 Madrid, Spain:IEEE, 4539–4546.

Yu, X., & Shen, S. (2019). An integrated decomposition and approximate dynamic programming approach for on-demand ride pooling. *IEEE Transactions on Intelligent Transportation Systems*, *21*(9), 3811-3820.

Zwick, F., & Axhausen, K. W. (2020, September). Impact of service design on urban ridepooling systems. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)* (pp. 1-6). IEEE.

Zwick, F., Kuehnel, N., Moeckel, R., & Axhausen, K. W. (2021). Agent-based simulation of city-wide autonomous ride-pooling and the impact on traffic noise. *Transportation Research Part D: Transport and Environment*, *90*, 102673.