THEMA
théorie économique,
modélisation et applications

THEMA Working Paper n°2024-03
CY Cergy Paris Université, France

METROPOLIS2: Bridging Theory and Simulation in Agent-Based Transport Modeling

Lucas Javaudin, André de Palma

March 2024

METROPOLIS2: Bridging Theory and Simulation in Agent-Based Transport Modeling

Lucas Javaudin* André de Palma†

March 25, 2024

Abstract

Transport simulators can be used to compute the equilibrium between transportation demand and supply within complex transportation systems. However, despite their theoretical foundations, there is a lack of comparative analysis between simulator results and theoretical models in the literature. In this paper, we bridge this gap by introducing METROPOLIS2, a novel mesoscopic transport simulator capable of simulating agents' travel decisions (including mode, departure-time, and route choice), based on discrete-choice theory within a dynamic, continuous-time framework. We demonstrate METROPOLIS2's functionality through its application to the single-road bottleneck model and validate its ability to replicate analytical results. Furthermore, we provide a comprehensive overview of METROPOLIS2 in large-scale scenarios. Finally, we compare METROPOLIS2's results with those of the original METROPOLIS1 simulator in a simulation of Paris, highlighting its speed and ability to converge to an equilibrium.

Keywords: transport simulation; agent-based modeling; bottleneck; dynamic traffic assignment; discrete-choice models

JEL Codes: C63; R4

*THEMA, CY Cergy Paris Université; lucas.javaudin@cyu.fr
†THEMA, CY Cergy Paris Université; andre.de-palma@cyu.fr

1 Introduction

Transportation science explores the complex interplay between demand and supply within transportation systems. Demand encompasses individuals, seeking to travel from one location to another, each with their own preferences such as value of time and scheduling constraints, along with some limitations imposed by factors like car ownership or public-transit subscription. Supply encompasses the transportation infrastructure, comprising a road network defined by regulations (e.g., speed limits, traffic signals, intersection priorities) and congestion dynamics (e.g., bottlenecks, shock waves, queueing), a public transit system with associated timetables, available parking spaces, etc.

A large portion of transportation science literature is dedicated to analyzing equilibrium properties and evaluating transport policies through analytical models. These models, while tractable, only capture certain facets of equilibrium dynamics. For example, the bottleneck model (Vickrey, 1969; Arnott et al., 1990, 1993) sheds light on the impact of schedule constraints and road tolls using a model featuring a single road and a continuum of identical individuals. With the advancement of computing power in recent decades, transportation simulators are being used to analyze specific scenarios or policies, considering the complex interplay of various effects (Nguyen et al., 2021). These simulators range from microscopic models examining traffic infrastructure impacts at the neighborhood level (Lopez et al., 2018) to macroscopic models investigating aggregated effects at city or national levels (McNally, 2007).

Mesoscopic models like METROPOLIS (de Palma et al., 1997) and MATSim (Horni et al., 2016) adopt an intermediary approach between microscopic and macroscopic simulations. Relying on an agent-based methodology, they account for heterogeneous effects while simplifying congestion modeling (e.g., omitting lane changing or car following behaviors) to enhance scalability for large-scale scenarios. This paper introduces METROPOLIS2, a novel mesoscopic transport simulator that extends the legacy of its predecessor, METROPOLIS, by drawing inspiration from similar mesoscopic modeling frameworks.

METROPOLIS2's key characteristics include the simulation of agents' travel decisions (mode, departure-time, and route) within a dynamic, continuous-time framework. The simulator is adapted to simulate and evaluate transport policies at the city or national level, such as low-emission zones, infrastructure changes (e.g., alteration in speed limit, introduction of new public-transit lines) or pricing adjustments (e.g., fuel cost taxation, modification in public-transit fares). Grounded in economic and discrete-choice theories, METROPOLIS2 is well-suited for computing agents' surplus in cost-benefit analyses. Moreover, it can derive insights from analytical examples, as demonstrated in Section 3 and Appendix A.

METROPOLIS2 draws inspiration from and extends the transport simulator METROPOLIS (de Palma et al., 1997; de Palma and Marchal, 1999, 2002). To avoid confusion, the original simulator METROPOLIS is referred to as METROPOLIS1 throughout the paper. Compared to its predecessor, METROPOLIS2 offers additional features including trip chaining, more flexible utility specifications, various vehicle types and explicit bottleneck queues. Moreover, it relies on state-of-the-art routing algorithms (Batz et al., 2013; Geisberger and Sanders, 2010), resulting in improved speed.

The primary difference between METROPOLIS2 and MATSim (Horni et al., 2016), a popular agent-based transport simulator with an activity-based approach, lies in the convergence algorithm used to reach an equilibrium. While MATSim employs a co-evolutionary algorithm where agents select the best alternative from a small agent-specific choice set at each iteration, with a low probability of switching to a different random alternative outside of this choice set, METROPOLIS2 agents choose the best alternative from the complete choice set at each iteration. Consequently, MATSim tends to produce a more stable system with fewer changes between iterations but converges slower to an equilibrium, compared to METROPOLIS2. MATSim may also become trapped in suboptimal equilibria if the parameters of the co-evolutionary algorithm are not properly adjusted. For an in-depth comparison between MATSim, METROPOLIS1 and METROPOLIS2, refer to Section 2.

The paper is structured as follows. Section 2 conducts a literature review on meso-

scopic transport simulators and similar models. Section 3 introduces a simplified version of METROPOLIS2, tailored to simulate the single-road bottleneck model, highlighting its foundation in economic and discrete-choice theory and its ability to replicate results from analytical models. Section 4 provides an overview of the general version of METROPOLIS2, including a presentation of the convergence algorithm, an explanation of the features modeled on the demand and supply side, and a discussion of the equilibrium concept. Section 5 delves into implementation details for the simulator, such as programming language, code parallelization and testing. Section 6 presents a comparison with METROPOLIS1 using a simulation of Paris's urban area. Finally, Section 7 offers concluding remarks.

2 Literature Review

This section starts with a comparison of METROPOLIS2 with two other transport simulators: METROPOLIS1 and MATSim. Additional related works are discussed at the end of the section.

METROPOLIS1 is a dynamic mesoscopic transport simulator, whose original version was presented by de Palma, Marchal, and Nesterov (1997). METROPOLIS1 is the main source of inspiration for METROPOLIS2. METROPOLIS1 can simulate mode choice (between car and public transit), departure-time choice (with a Continuous Logit) and route choice for a population of agents all performing a single trip. In METROPOLIS1, congestion is simulated using an event-based model relying on speed-density functions. Many applications of METROPOLIS1 have been proposed in the last decades (e.g., road pricing with de Palma et al. 2005, vehicle-emission pricing with Vosough et al. 2022, ride-sharing with de Palma et al. 2022).

MATSim is defined as "an open-source framework for implementing large-scale agent-based transport simulations."[1] A detailed description of this simulator is provided in Horni, Nagel, and Axhausen (2016). MATSim has been used in many cities and regions around the

[1] `https://matsim.org/`, last accessed on 23d January 2024.

world and its architecture is close to the one of METROPOLIS2, which justify including it in the detailed comparison.

Tables 1, 2, 3 and 4 provide a comparison between the three simulators METROPOLIS2, METROPOLIS1 and MATSim on various areas. Table 1 compares the simulators with respects to some general and software-related characteristics. While METROPOLIS2's development started in 2022, METROPOLIS1 was initially released around 1997 and the Java version of MATSim was released in 2010 (with preliminary versions dating back from the 1990s, see Nagel 1996). The three simulators each use different programming languages and different input / output formats.

Table 1: Comparison of 3 transport simulators: General and software

	METROPOLIS2	METROPOLIS1	MATSim
Reference	This paper	de Palma et al. (1997)	Horni et al. (2016)
Initial release	2022 (version 0.1.0)	Around 1997	2010 (version 0.1.0)
Language	Rust	C++	Java
Open source	Undecided	No	Yes
Input / output format	CSV or Parquet	MySQL	XML

Table 2 compares how demand is defined in the simulators. METROPOLIS1 generates the population from origin-destination matrices, while METROPOLIS2 and MATSim represent the population as a list of agents. Therefore, METROPOLIS1 defines trips between zones, while METROPOLIS2 and MATSim define them between network nodes or links (but zone-to-zone trips can also be simulated as a special case). The latter two models differ in that, for each agent, METROPOLIS2 considers a list of trips, while MATSim considers a list of activities. However, since converting between trips and activities is straightforward (activities represent the time spent between trips and trips represent the way to connect activities), this difference has no practical consequences. One limitation is that METROPOLIS2 assumes that the activity duration is exogenous.

With regards to the definition of the utility function, METROPOLIS2 uses a polynomial

function of travel time, with a linear schedule-delay cost at origin and at destination (see Appendix E.1). Other specifications could be easily added in the future. METROPOLIS1' uses the α-β-γ model (Arnott et al., 1990), with a schedule-delay cost either at origin or at destination (this is a special case of METROPOLIS2). MATSim's definition of utility (referred to as "score") is based on Charypar and Nagel (2005): utility depends on activities' duration, activities' schedule delays, trips' travel time and trips' euclidean distance. The parameters of these utility functions can be specific to each agent in METROPOLIS2, while they are defined at the class- (or subpopulation-) level in METROPOLIS1 and MATSim.

Table 2: Comparison of 3 transport simulators: Demand definition

	METROPOLIS2	METROPOLIS1	MATSim
Population	Agents with 1+ trips	Origin-destination matrix by subpopulation (single trip)	Agents with 1+ activities
Origin / destination definition (for road modes)	Node	Zone centroid	Link
Activity duration	Exogenous	N/A	Endogenous
Utility / generalized cost	Function of travel time and schedule-delay costs[a]	Generalized cost (Vickrey model)	Charypar-Nagel utility function[b]
Preference classes	Parameters defined at the agent-level	Distribution of parameters at the class-level	Constant parameters at the class-level

[a] The default utility function is a polynomial function of travel time with early / late schedule-delay costs at origin and at destination. Other specifications can be easily added.
[b] Utility of activities (logarithmic function of duration with early and late penalties) and utility of trips (linear function of travel time and euclidean distance).

Table 3 provides a comparison of the three simulators in terms of their choice models. A fundamental difference lies in the fact that both METROPOLIS2 and METROPOLIS1 are grounded in discrete-choice theory, whereas MATSim employs a co-evolutionary algorithm. In METROPOLIS2 and METROPOLIS1, agents select the alternative that maximizes their utility among all available alternatives (given the expected network conditions). In contrast, MATSim maintains a list of plans (typically 5) in the "memory" of each agent. Each plan represents an alternative (a combination of modes, departure times and routes). Agents in

MATSim then select one of these plans based on their perceived utility, or they "evolve" by choosing a new plan (usually generated by applying a random perturbation to an existing plan). While the co-evolutionary algorithm in MATSim offers a more realistic depiction of agent behavior (although this is not the primary rationale behind its use, as noted by Flötteröd 2016), it comes with certain drawbacks: (i) convergence to an equilibrium may be slower compared to METROPOLIS1 and METROPOLIS2 due to agents' slower adaptation and (ii) computing agent-level surplus for cost-benefit analysis is not as straightforward (see Kickhöfer and Nagel 2016).

METROPOLIS1 and METROPOLIS2 are very similar, with two notable exceptions. First, METROPOLIS2 offers more flexibility than METROPOLIS1: while METROPOLIS1 is limited to a binary Logit mode choice and a Continuous Logit departure-time choice, METROPOLIS2 allows for various discrete-choice models. Second, route selection in METROPOLIS1 involves making decisions at every intersection, while, in METROPOLIS2, the complete route is selected prior to departure from the origin (which enables the use of more efficient routing algorithms).

Table 3: Comparison of 3 transport simulators: Choice models

	METROPOLIS2	METROPOLIS1	MATSim
Basic principle	Discrete-choice theory	Discrete-choice theory	Co-evolutionary algorithm
Mode choice	Any discrete-choice model (random or deterministic)	Binary Logit model	Random shifts (when mode innovation is triggered)[a]
Departure-time choice	Various specifications[b]	Continuous Logit	Random shifts (when time innovation is triggered)
Route choice	Fastest path at the time of departure	Choice at each intersection (deterministic or stochastic)	Fastest path at the time of departure (when route innovation is triggered)
Destination choice	Yes (with exogenous destinations)	No	With a module (Horni et al., 2011)

[a] An alternative mode-choice model based on discrete-choice theory is available as a MATSim module (see Hörl et al. 2018 and Hörl et al. 2019).

[b] The choice models include Multinomial Logit and Continuous Logit. Departure-time choice can also be deactivate (i.e., departure time is exogenous).

Table 4 compares how congestion is modeled and simulated in the three simulators. In this comparison, we only consider the QSim mobsim[2] for MATSim (the default and most used mobsim). MATSim, with QSim, uses a time-step based model, which means that at each time step (1 second by default) all the vehicles currently on the network get an "update" to move them. In contrast, METROPOLIS1 and METROPOLIS2 use an event-based model, which means that vehicles are "updated" only when an event happens (e.g., they reach the end of their current link). Apart from the typical road vehicles (such as car and trucks), MATSim has the capability to simulate public-transit vehicles. This means that (i) buses can contribute traffic congestion and are also susceptible to being affected by congestion caused by other vehicles and (ii) agents are forced to wait when a train vehicle is full. Contrarily to METROPOLIS1 and MATSim, METROPOLIS2 represents time as a continuous variable, which allows for greater accuracy in the computation.[3]. Congestion is simulated in METROPOLIS1 via speed-density functions (the speed of a vehicle entering a road is a function of the density of vehicles currently on this road) and in MATSim via bottleneck queues (the outflow of vehicles exiting a road is limited by the road's capacity; a queue builds up if the flow is larger than the capacity). METROPOLIS2 can make use of either speed-density functions or bottleneck queues, or a combination of both. The three models support queue propagation: when the available space on a road is smaller than the incoming vehicle length, the vehicle is stuck at his current location until enough space is freed. Both METROPOLIS2 and MATSim support different vehicle types, with different length, passenger car equivalent, speed limits and road restrictions. Finally, both METROPOLIS1 and MATSim support tolls and time-varying networks (e.g., capacity of a road is decreased at an exogenous time).

We now discuss SimMobility (Adnan et al., 2016), an open-source C++ agent-based

[2]A MATSim mobsim, or mobility simulation, is a program responsible for the network loading of the model, i.e., it is a program which simulates the movement of vehicles on the network.

[3]In METROPOLIS2, for a road with a capacity of 1440 vehicles per hour, at most 1 vehicle each $3600/1440 = 2.5$ seconds is allowed to enter / exit the road. This is not easy to simulate in a discrete-time model without introducing some randomness or rounding errors.

Table 4: Comparison of 3 transport simulators: Supply

	METROPOLIS2	METROPOLIS1	MATSim (QSim)
Model	Event-based	Event-based	Time-stepping (1s step by default)
Simulated modes	Road and teleport[a]	Road and teleport[a]	Road, teleport[a] and public-transit
Time	Continuous	Discrete (1s step)	Discrete (1s step)
Speed-density functions	Yes	Yes	No
Bottleneck queues	Yes	No	Only for outflow
Queue propagation (spillback)	Yes (configurable backward propagation speed)	Yes (instantaneous backward propagation)	Yes (15 km/h backward propagation speed)
Vehicle types	Yes (with different length, PCE, speed limit)	No	Yes (with different length, PCE, maximum speed limit)
Road restrictions	Yes	No	Yes
Tolls	Partial[b]	Yes	Yes
Time-varying network	No	Yes	Yes

[a] A "teleport" mode is a mode without any interaction with the other agents. For example, walking and bicycling are usually considered as teleport modes since the travel time is constant. Public transit can also be considered as a teleport mode if one assumes that the schedules are not impacted by car traffic or in-vehicle congestion.

[b] In METROPOLIS2, simulating a single tolled road is possible with a strategy employing road restrictions and vehicle types: agents can select between a vehicle restricted from the tolled road or a vehicle able to take any road but incurring a fixed monetary penalty (equivalent to the toll amount). Expanding to multiple tolled roads involves an increasing complexity and running time, scaling combinatorially with the number of tolls.

simulator that shares many similarities with METROPOLIS2. SimMobility is divided in three main components: short-, mid- and long-term. While the short-term component mirrors a microscopic model and the long-term component represents a land-use model, the mid-term component, SimMobilityMT (Lu et al., 2015), adopts a mesoscopic approach akin to that of METROPOLIS2. The research focus of SimMobilityMT has primarily revolved around mobility-on-demand applications (Basu et al., 2018; Oh et al., 2020). Similar to METROPOLIS2, the demand model of SimMobilityMT employs discrete-choice methods. SimMobilityMT uses a fully econometric activity-based model to simulate the daily activity patterns of agents, relying on a hierarchy of discrete-choice models. In contrast to METROPOLIS2's deterministic route choice model, SimMobilityMT employs a path-size Logit model for route choice. Moreover, it allows for within-day decisions, enabling agents to adjust their routes or activities based on observed network conditions throughout the (simulated) day. Regarding the supply model, both METROPOLIS2 and SimMobilityMT simulate congestion using a combination of speed-density functions and bottlenecks, incorporating queue propagation. Both simulators are also able to simulate different vehicle types on the road network. However, METROPOLIS2 uses a continuous-time event-based model, whereas SimMobilityMT relies on a time-stepping model. Finally, while METROPOLIS2 represents the demand-supply equilibrium as a solution to a fixed-point problem, the literature concerning SimMobilityMT cited so far does not explicitly mention the notion of equilibrium.

For an extended literature review on agent-based transport simulators, the interested readers can refer to the work of Nguyen et al. (2021).

Our work shares similarities with the study conducted by Otsubo and Rapoport (2008), who propose a discrete version of the bottleneck model and observe that the solution to the symmetric mixed-strategy Nash equilibrium diverges from the equilibrium solution of the continuous model. In Section 3, we investigate a discretized version of the bottleneck model with a finite number of agents. Our model differs from theirs in two key aspects: (i) we main-

tain time as a continuous variable, and (ii) we introduce stochasticity in the departure-time decision process. Moreover, our findings shows that the results remain consistent whether considering a finite number of agents or a continuum of agents (see Appendix A).

Finally, Guo et al. (2018) investigate the bottleneck model and conclude that an equilibrium cannot be attained through a day-to-day learning process like proportional swap. However, our findings present a more optimistic outlook, demonstrating that we can achieve numerical proximity to an equilibrium with METROPOLIS2, using the same bottleneck model with three differences: (i) a finite number of agents, (ii) the introduction of stochasticity in the departure-time choice, and (iii) discretization of the travel-time function.

3 From Models to Simulations

This section explains the challenges encountered in transport simulations and provides an overview of how METROPOLIS2 operates on a basic example. All the analysis in this section rely on the bottleneck model.

3.1 The Bottleneck Model

The so-called bottleneck model is a transport model which was derived analytically by Vickrey (1969) and Arnott et al. (1990). The model consists in a single road, a single mode (car) and identical individuals whose utility is a linear function of travel time and schedule delay. Numerous extensions of the bottleneck model have been proposed in the literature (Li et al., 2020). We provide below a brief description of this bottleneck model.

There is a continuum of N individuals traveling from an origin A to a destination B, via a single road. This road consists in a free-flow travel time t^{f}, followed by a bottleneck of capacity s with a vertical queue, as illustrated in Figure 1. The travel time from A to B can thus be expressed as

$$T(t) = t^{\mathrm{f}} + \frac{Q(t + t^{\mathrm{f}})}{s},$$

where t is the departure time from A and $Q(t)$ is the length of the bottleneck's queue at time t.

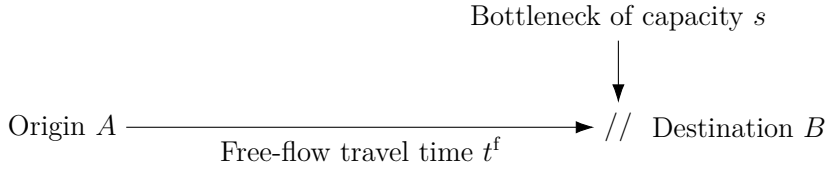

Figure 1: Illustration of the road network in the bottleneck model

Individuals choose their departure time from the origin to maximize their utility, given by

$$V(t) = -\alpha \cdot T(t) - \beta \cdot [t^* - t - T(t)]_+ - \gamma \cdot [t + T(t) - t^*]_+, \tag{1}$$

where α is the value of travel time, β is the value of time for being early at destination, γ is the value of time for being late at destination, t^* is the desired arrival time at destination and $[x]_+ = \max(x, 0)$.

Departure time is the only decision variable in the bottleneck model, as there is neither mode choice (all individuals travel by car) nor route choice (only one road connects A to B). An equilibrium is reached when no individual has an incentive to unilaterally change their departure time. The analytical solution to the equilibrium of the model is found by relying on the following property:

$$\begin{cases} V(t) = \bar{V} & \text{if } \overline{r}^{\mathrm{d}}(t) > 0, \\ V(t) \leq \bar{V} & \text{if } \overline{r}^{\mathrm{d}}(t) = 0, \end{cases}$$

where $\overline{r}^{\mathrm{d}}(t)$ is the equilibrium rate of departures at time t and $\bar{V}$ is the equilibrium utility level. This property implies that at equilibrium, all individuals get the same utility level. The analytical solution of the model is presented in Arnott et al. (1990). The equilibrium rate of departures and arrivals and the equilibrium utility function are represented on Figure 2.

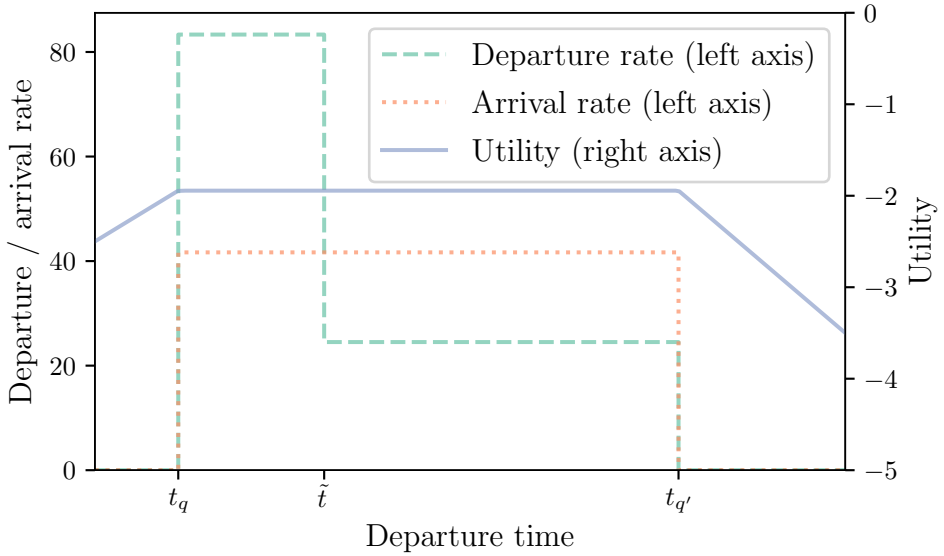

Figure 2: Analytical solution of the bottleneck model

Note: t_q is the time at which congestion starts to appear, $\tilde{t}$ is the departure time such that the individual arrives on time, $t_{q'}$ is the time at which congestion ends.

3.2 Challenges of Simulation

Simulating a model, even as basic as the bottleneck model, presents several challenges that requires specific adaptations and approximations. First, the analytical bottleneck model assumes a continuum of identical individuals, allowing for departure times to be represented as a continuous distribution. In practice, discretization of the model is required for simulations. Indeed, simulations are limited to a finite number of distinct individuals, typically referred to as "agents". Each agent is treated as a separate entity with their own departure time. Consequently, departure-time distribution in simulations becomes discrete, whereas the analytical model assumes a continuous distribution. See Otsubo and Rapoport (2008) for a numerical solution of the bottleneck model with discrete agents. Note that considering a discrete number of separate individuals is actually more realistic than a continuum of individuals and it allows to more easily deal with individual heterogeneity.

Second, the analytical model assumes a population of perfectly identical individuals all choosing a departure time that maximizes their deterministic utility. Equilibrium conditions,

as demonstrated by Arnott et al. (1990) and illustrated on Figure 2, reveal that there is a continuous range of departure times that yield maximum utility. While the analytical solution of the model provides a single solution for the rate of departures at equilibrium, there exits an infinity of solutions regarding the order in which agents depart. In simulations, whenever there are multiple departure times maximizing utility, there is no easy way to spread the individuals over these departure times so that the equilibrium conditions are satisfied. To address this, randomness is introduced in the simulation's decision-making process. This stochastic element encourages individuals to spread their departure times across the equilibrium departure-time range. One approach to introduce randomness is by adding a random component to the utility, similar to the methods used in discrete-choice models. In this vein, de Palma et al. (1983) analyze the bottleneck model where departure-time choice is modeled as a Continuous Logit model.

Third, travel-time function $T(t)$ and utility function $V(t)$ are continuous functions that may not always have explicit analytical forms. In such cases, simulations need to resort to numerical approximations. One common approach is to represent these continuous functions as piecewise linear functions. This approximation involves dividing the function into linear segments, with approximation errors depending on the length of these linear segments.

3.3 Simulating the Bottleneck Model

METROPOLIS2 is able to simulate various small-scale and large-scale models with the goal of finding an equilibrium of the simulated model. For the bottleneck model, achieving equilibrium entails finding a departure time t_n^{d}, for each agent n, such that no agent has an incentive to choose a different departure time, given the travel-time function $T(t)$ derived from the agents' collective departure times. We now describe how the bottleneck model presented in Section 3.1 can be simulated.

We assume a population of N discrete agents, all traveling from origin A to destination

B. The utility of agent $n \in \{1, \ldots, N\}$ is given by

$$U_n(t) = V(t) + \varepsilon_n(t),$$

where t is the departure time from the origin, $V(t)$ is the deterministic utility function, defined as in equation (1), and $\varepsilon_n(t)$ is an agent-specific random component of utility. All agents are identical except for the random components $\varepsilon_n(t)$. The single road connecting origin A to destination B has a free-flow travel time t^{f} and a congested travel time described by a bottleneck with capacity s. This means that, at most s cars can cross the bottleneck during each time unit, or, equivalently, that two cars cannot follow each other in less than $1/s$ time units. As discussed in Section 3.2, the N agents are treated as separate entities and the travel-time function T and the utility functions U_n are both represented as a piecewise linear functions.

The model simulated is split in a demand model, corresponding to the travel behavior of the agents simulated, and a supply model, corresponding to the road-network infrastructure operations.

Demand model The demand model consists in the agents choosing a departure time for their trip, given the expected travel-time function from origin to destination.

Following de Palma et al. (1983), we assume that the distribution of $\varepsilon_n(t)$ is such that the probability that agent n chooses departure time t is given by the Continuous Logit model (Ben-Akiva and Watanatada, 1981; Ben-Akiva et al., 1985):

$$p_n^{\mathrm{d}}(t) = \frac{e^{V(t)/\mu}}{\int_{t^0}^{t^1} e^{V(\tau)/\mu} \,\mathrm{d}\,\tau}, \tag{2}$$

where μ represents the randomness of departure-time choice (with $\mu \to 0$ corresponding to a deterministic choice and $\mu \to \infty$ corresponding to a pure random choice) and $[t^0, t^1]$ represents the period of possible departure times.

The departure time of an agent n can be drawn from the probability distribution defined by $p_n^{\mathrm{d}}(t)$ by relying on inverse sampling, which works as follows. Let F_n^{d} be the cumulative distribution function of the chosen departure time, i.e.,

$$F_n^{\mathrm{d}}(t) = \int_{t^0}^{t} p_n^{\mathrm{d}}(\tau)\,\mathrm{d}\,\tau,$$

and let $u_n \sim \mathcal{U}_{[0,1]}$ (uniform random number between 0 and 1). The chosen departure time is then simulated as $(F_n^{\mathrm{d}})^{-1}(u_n)$. Figure 3 illustrates how the departure time of an individual is drawn from the distribution F_n^{d} using inverse sampling. Note that, even though the utility function is represented as a piecewise linear function, time itself is not discretized and thus the departure time is a continuous variable.

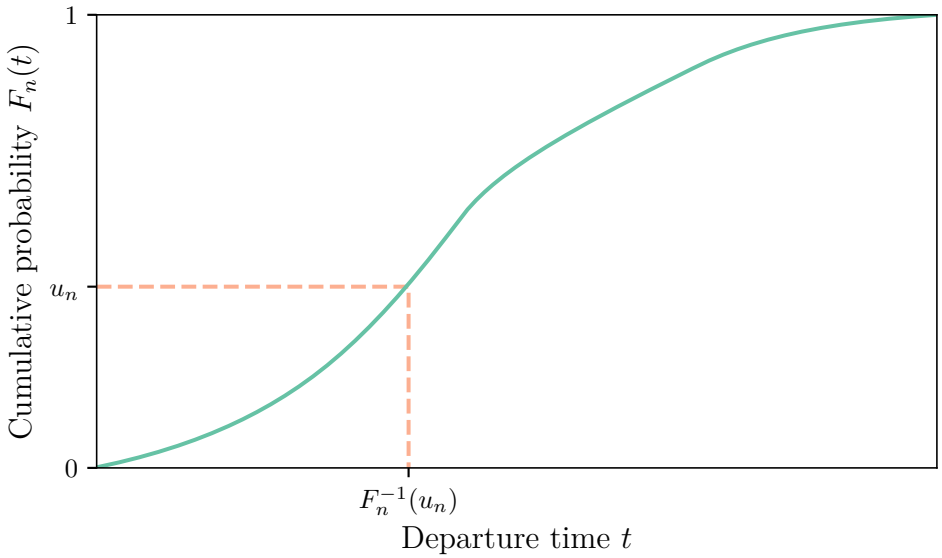

Figure 3: Illustration of the use of inverse sampling to draw a departure time from a cumulative distribution function

The logsum formula of the departure-time choice is defined as

$$\text{logsum} = \mu \cdot \ln \int_{t^0}^{t^1} e^{V(\tau)/\mu}\,\mathrm{d}\,\tau. \tag{3}$$

It represents the expected maximum utility that the agent can get from the trip, which can

be used as a measure of the surplus of the agents.

We denote by

$$f^{\text{demand}} : T \mapsto f^{\text{demand}}(T) = \{(F_n^{\text{d}})^{-1}(u_n)\}_{1 \leq n \leq N}, \tag{4}$$

the function that returns the departure times chosen by all agents, given the expected travel-time function T.

Supply model The supply model consists in the road infrastructure which constrains the arrival times of the agents at destination. In the model investigated here, the road infrastructure consists in a single road with a free-flow travel time t^{f} and a road bottleneck, as depicted in Figure 1.

In METROPOLIS2, road bottlenecks can be in two states: open or closed. When a vehicle reaches an open bottleneck, it can cross it instantaneously and then the bottleneck is closed for a period of $1/s$ time units, where s is the bottleneck's capacity. When a vehicle reaches a closed bottleneck, the vehicle is pushed to the back of a point queue.[4] When the bottleneck opens again, the vehicle at the front of the queue can cross instantaneously and the bottleneck closes again.

The bottleneck thus defined satisfies the following two properties. First, two cars cannot cross the bottleneck in less than $1/s$ time units. This implies that no more than s cars can cross the bottleneck in 1 time unit, which is consistent with the definition of the bottleneck capacity s. Second, the first-in-first-out property is satisfied as the next car to exit the bottleneck queue is the car that entered the queue the earliest.

In METROPOLIS2, the trips are simulated using an event-based model, which executes all the events occurring during the simulated period in chronological order. The events to be executed include, e.g., "the car of agent n is reaching the bottleneck at time t" or "the bottleneck is re-opening at time t'". This event-based model will yield the arrival

[4]Horizontal queues with queue propagation can also be simulated in METROPOLIS2, see Section 4.

times of all the agents, denote by $\mathbf{t}^{\mathrm{a}} = \{t_n^{\mathrm{a}}\}_{1 \le n \le N}$, given their departure times, denoted by $\mathbf{t}^{\mathrm{d}} = \{t_n^{\mathrm{d}}\}_{1 \le n \le N}$.

In METROPOLIS2, the travel-time function, T, is derived from the simulated departure times, $\mathbf{t}^{\mathrm{d}}$, and arrival times, $\mathbf{t}^{\mathrm{a}}$. This function is represented as a sequence of M breakpoints, denoted by $\{(x_1, y_1), \ldots, (x_m, y_m), \ldots, (x_M, y_M)\}$. The x-values, corresponding to departure times, are defined as $\{t^0, t^0 + \delta, t^0 + 2\delta, \ldots, t^0 + (M-1)\delta\}$, where δ denotes the length of the breakpoint intervals. The parameter M is defined as $\lfloor (t^1 - t^0)/\delta \rfloor + 1$ to ensure coverage of the entire simulated period $[t^0, t^1]$. The y-values, corresponding to travel times, are computed as weighted averages of the simulated travel times within the corresponding departure-time interval. Formally, for $1 \le m \le M$, the value y_m is computed as

$$y_m = \sum_n w_n(x_m) \cdot (t_n^{\mathrm{a}} - t_n^{\mathrm{d}}),$$

where the weights are given by

$$w_n(x_m) = \max\left(0, 1 - \frac{|x_m - t_n^{\mathrm{d}}|}{\delta}\right).$$

Here, the weight of agent n decreases linearly with the distance between the departure-time breakpoint x_m and the agent's departure time. The weight is equal to 1 when $t_n^{\mathrm{d}} = x$ and it becomes zero when t_n^{d} is outside the interval $[x_m - \delta, x_m + \delta]$. Given the sequence of breakpoints (x_m, y_m) representing the travel-time function, METROPOLIS2 then relies on linear interpolation to compute the travel-time for any departure time $t \in [t^0, t^1]$. Figure 4 provides an illustration of how the travel-time function is computed from the departure times and travel times of the agents.

We denote by

$$f^{\mathrm{supply}} : \mathbf{t}^{\mathrm{d}} \mapsto f^{\mathrm{supply}}(\mathbf{t}^{\mathrm{d}}) \tag{5}$$

the function that returns the travel-time function resulting from the departure times chosen

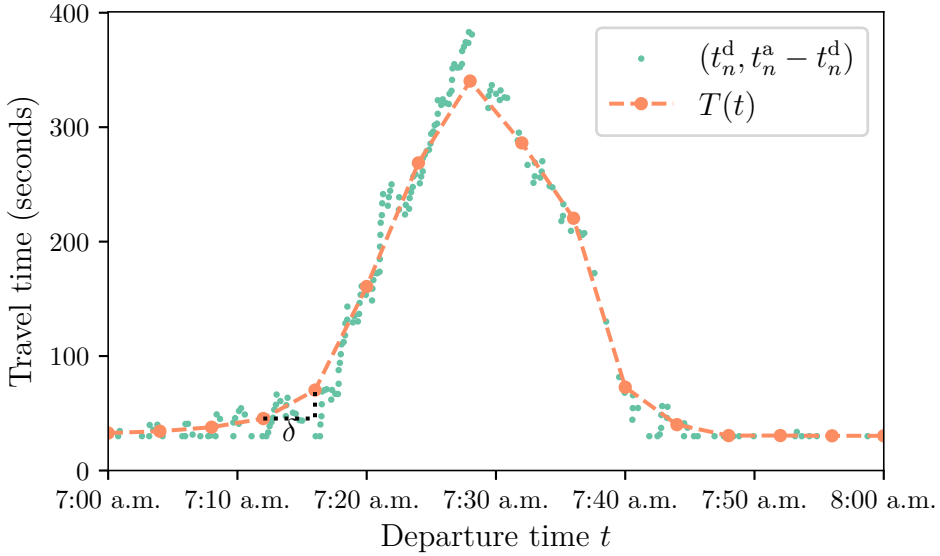

Figure 4: Agents' travel times and interpolated travel-time function (with $N = 200$ and $\delta = 4$ minutes)

by the agents.

Equilibrium The simulator METROPOLIS2 reaches an equilibrium of the model when it finds a set of departure times, $\overline{\mathbf{t}}^{\mathrm{d}}$, which result, in the supply model, in the same travel-time function that was used, in the demand model, to simulate these departure times, i.e.,

$$f^{\mathrm{demand}}(f^{\mathrm{supply}}(\overline{\mathbf{t}}^{\mathrm{d}})) = \overline{\mathbf{t}}^{\mathrm{d}}, \tag{6}$$

where f^{demand} and f^{supply} are the functions defined in equations (4) and (5), respectively. We denote by $\overline{T}$ the equilibrium travel-time function associated to $\overline{\mathbf{t}}^{\mathrm{d}}$, i.e.,

$$f^{\mathrm{supply}}(f^{\mathrm{demand}}(\overline{T})) = \overline{T}. \tag{7}$$

The goal of METROPOLIS2 is therefore to find a set of departure times which solves the fixed-point problem of equation (6), or, equivalently, to find a travel-time function which solves the fixed-point problem of equation (7).

An iterative procedure to find such an equilibrium is depicted in Algorithm 1. The algorithm starts from an initial value for the travel-time function, T_0 (e.g., the no-congestion travel-time function). Each iteration consists in simulating the departure times chosen by the agents, given the current value of the travel-time function (demand model, line 4), then simulating the travel-time function resulting from the current value of the departure times (supply model, line 5). The iterative process stops when a fixed point is reached (line 6).

Algorithm 1: Pseudo-code of a naive algorithm to find an equilibrium of the bottleneck model

Input: T_0

1 $\kappa \leftarrow 0$;

2 **repeat**

3 $\quad \kappa \leftarrow \kappa + 1$; `/* Update the iteration counter */`

4 $\quad \mathbf{t}^{\mathrm{d}}_{\kappa} = f^{\mathrm{demand}}(T_{\kappa-1})$; `/* Run the demand model */`

5 $\quad T_{\kappa} = f^{\mathrm{supply}}(\mathbf{t}^{\mathrm{d}}_{\kappa})$; `/* Run the supply model */`

6 **until** $\mathbf{t}^{d}_{\kappa} \neq \mathbf{t}^{d}_{\kappa-1}$ **and** $T_{\kappa} \neq T_{\kappa-1}$;

Output: $\mathbf{t}^{\mathrm{d}}_{\kappa}$, T_{κ}

Algorithm 1 is "naive" in the sense that, at each iteration, the travel-time function used in the demand model is equal to the travel-time function simulated during the previous iteration. However, this naive algorithm will usually never converges, i.e., it will never reach a point where the departure times and travel-time function do not vary from one iteration to another (see Figure 19 in Appendix A for a numerical illustration).

Instead, METROPOLIS2 introduces a learning model so that the expected travel-time function used to simulate the departure times is a weighted average of the simulated travel-time functions from the previous iterations. Algorithm 2 describes the procedure used in METROPOLIS2, with the changes compared to the naive algorithm highlighted in blue.

Algorithm 2 introduces a new variable for each iteration κ: the expected travel-time function, $\hat{T}_{\kappa}$, used in the demand model instead of the previously simulated travel-time function, $T_{\kappa-1}$ (line 4). In the added learning model (line 6), the next expected travel-time function, $\hat{T}_{\kappa+1}$, is computed based on the simulated travel-time function, T_{κ}, and the current expected travel-time function, $\hat{T}_{\kappa}$. The stopping criteria (line 7) are updated compared to

Algorithm 2: Pseudo-code of the METROPOLIS2 algorithm, adapted to the bottleneck model

Input: $\hat{T}_1$

$\kappa \leftarrow 0$;

repeat

  $\kappa \leftarrow \kappa + 1$; /* Update the iteration counter */

  $\mathbf{t}_\kappa^{\mathrm{d}} = f^{\mathrm{demand}}(\hat{T}_\kappa)$; /* Run the demand model */

  $T_\kappa = f^{\mathrm{supply}}(\mathbf{t}_\kappa^{\mathrm{d}})$; /* Run the supply model */

  $\hat{T}_{\kappa+1} = f^{\mathrm{learning}}(\kappa, T_\kappa, \hat{T}_\kappa)$; /* Run the learning model */

until $\mathbf{t}_\kappa^d \neq \mathbf{t}_{\kappa-1}^d$ **and** $T_\kappa \neq \hat{T}_\kappa$;

Output: $\mathbf{t}_\kappa^{\mathrm{d}}$, T_κ, $\hat{T}_\kappa$

the naive algorithm so that the current simulated travel-time function, T_κ, is compared to the current expected travel-time function, $\hat{T}_\kappa$.[5]

Learning model The learning model can be interpreted as a representation of how agents update their expectations regarding the travel time they anticipate based on the observed travel times in the past, although, it does not aim to be behaviorally accurate in modeling how agents form their anticipations. It is assumed that both the simulated and expected travel-time functions, T_κ and $\hat{T}_\kappa$ respectively, are common knowledge among all agents. The selection of the learning model, specifically the choice of the function f^{learning}, plays a crucial role in the convergence of the algorithm toward an equilibrium. In the numerical applications, we consider the following learning function corresponding to the exponential smoothing method:

$$f^{\mathrm{learning}}(\kappa, T_\kappa, \hat{T}_\kappa) = \frac{\lambda}{a_{\kappa+1}} T_\kappa + (1-\lambda)\frac{a_\kappa}{a_{\kappa+1}} \hat{T}_\kappa, \tag{8}$$

[5]This change is required to prevent the algorithm from converging to a non-equilibrium. This would be the case whenever $f^{\mathrm{learning}}(\kappa, T_\kappa, \hat{T}_\kappa) \equiv \hat{T}_\kappa$, i.e., the expected travel-time function is never updated, which would imply the chosen departure times and thus the simulated travel-time function to be the same from one iteration to another, despite equations (6) and (7) not being necessarily satisfied.

where $\lambda \in (0, 1]$ is the smoothing factor, κ is the iteration counter and $a_\kappa = 1 - (1 - \lambda)^\kappa$ is a normalization term to correct for the weight of the initial value $\hat{T}_1$, such that, by recurrence,[6]

$$\hat{T}_{\kappa+1} = \frac{1}{a_\kappa} \lambda \sum_{i=0}^{\kappa} (1 - \lambda)^i T_\kappa,$$

with the convention that $T_0 = \hat{T}_1$.

With $\lambda = 1$, the learning function is $f^{\text{learning}}(\kappa, T_\kappa, \hat{T}_\kappa) = T_\kappa$ and we are back to the naive algorithm (Algorithm 1). With $\lambda \to 0$, the learning function goes to $f^{\text{learning}}(\kappa, T_\kappa, \hat{T}_\kappa) = 1/(\kappa+1) \sum_{i=0}^{\kappa} T_\kappa$ and the expected value is simply an arithmetic mean of the past simulated values.

Numerical applications In Appendix A, we present the results of various simulations of METROPOLIS2 for the bottleneck model considered in this section. The results show that, when the smoothing factor λ is not too large, the METROPOLIS2 algorithm is "almost" converging to an equilibrium. The simulations replicate the analytical results from de Palma et al. (1983) very accurately, suggesting that the discretization of the model does not impact its results.

We also perform many sensitivity analysis. We find that the simulations converge even with a small number of agents and a small number of breakpoints in the travel-time function. We find that when μ is too small (the model is close to the deterministic model of Arnott et al., 1990), the simulation does not converge to an equilibrium.

4 Simulator Overview

The previous section described the METROPOLIS2 simulator on the simple single-road bottleneck model. In this section, we present an overview of METROPOLIS2 in the general

[6] In the standard exponential smoothing method, the weight of the initial value, $\hat{T}_1$, can get abnormally high, compared to the other values, $\hat{T}_\kappa$, $\kappa > 1$, for a small λ value.

case, which can be obtained by extending the bottleneck model with an arbitrary road network, heterogeneous agents, various modes, different vehicle types, etc.

Algorithm Algorithm 3 defines a pseudo-code of the process followed by the METROPOLIS2 simulator. It is a generalization of Algorithm 2 used for the bottleneck model (Section 3). The simulator starts with some input data $\boldsymbol{D}$ and initial expected network conditions $\hat{\boldsymbol{T}}_1$ (by default equal to free-flow conditions). The *input data* $\boldsymbol{D}$ encompasses a population (agents with their preferences and assigned trips), a road network (roads and their attributes), vehicle types and technical parameters. Further details on the input data of METROPOLIS2 are provided in Appendix B. The term *network conditions* denotes a data structure representing the road-level time-dependent travel times, which are common to all agents sharing the same vehicle type. See Appendix D for more details on these network conditions.

At each iteration, the simulator executes three models: the demand model, the supply model and the learning model. Termination of the simulator occurs upon meeting one of the stopping criteria, which are discussed at the end of this section. Subsequently, the equilibrium travel decisions, along with the expected and simulated network condition, are returned. *Travel decisions*, denoted by $\mathbf{z} \equiv \{z_n\}_{1 \leq n \leq N}$, represent the mode, departure time and route chosen by agents for each of their trips. For further details regarding the output data of METROPOLIS2, refer to Appendix C. Table 5 summarizes the primary notations used in this section.

Figure 5 illustrates the fundamental operational flow of METROPOLIS2. We now provide a brief description of the three models. More details on the demand and supply models are provided in Appendix E.

Demand model The demand model simulates the travel decisions made by all agents, given the expected network conditions for the current iteration. Initially, agents choose between various *travel alternatives*. In its simplest form, a travel alternative represents a

Table 5: Main notations used in METROPOLIS2

Name	Notation
Input data (population, network, parameters)	$\boldsymbol{D}$
Iteration counter	κ
Simulated network conditions at iteration κ (vector of road-level travel-time functions)	$\boldsymbol{T}_\kappa$
Expected network conditions for iteration κ (vector of road-level travel-time functions)	$\hat{\boldsymbol{T}}_\kappa$
Travel decisions of agent n at iteration κ	$z_{n,\kappa}$
Travel decisions at iteration κ (vector of agent-level travel decisions)	$\mathbf{z}_\kappa$

Algorithm 3: Pseudo-code of the METROPOLIS2 algorithm

Input: $\boldsymbol{D}$, $\hat{\boldsymbol{T}}_1$

$\kappa \leftarrow 0$;

repeat

  $\kappa \leftarrow \kappa + 1$; `/* Update the iteration counter */`

  $\mathbf{z}_\kappa = f^{\text{demand}}(\hat{\boldsymbol{T}}_\kappa; \boldsymbol{D})$; `/* Run the demand model */`

  $\boldsymbol{T}_\kappa = f^{\text{supply}}(\mathbf{z}_\kappa; \boldsymbol{D})$; `/* Run the supply model */`

  $\hat{\boldsymbol{T}}_{\kappa+1} = f^{\text{learning}}(\kappa, \boldsymbol{T}_\kappa, \hat{\boldsymbol{T}}_\kappa; \boldsymbol{D})$; `/* Run the learning model */`

until *One of the stopping criteria is met*;

Output: $\mathbf{z}_\kappa$, $\boldsymbol{T}_\kappa$, $\hat{\boldsymbol{T}}_\kappa$

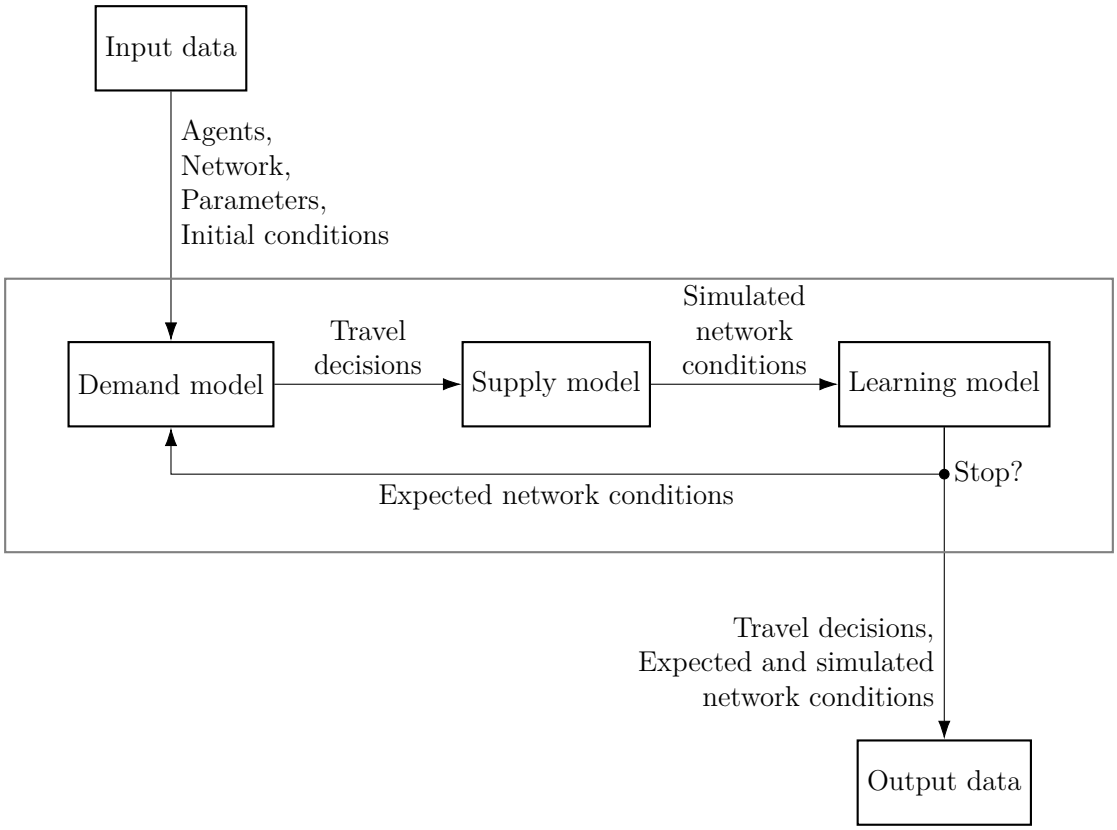

Figure 5: Overview of METROPOLIS2 process

single trip with a given origin, destination and mode. However, it can also encompass chains of trips (e.g., a car trip from home to grocery store, followed by 15 minutes of shopping, followed by another car trip from grocery store to workplace). In most cases, the choice of a travel alternative is akin to a mode choice, although it can also represent a destination choice. Following the selection of a travel alternative, agents make decisions regarding their departure time and the route, if applicable. The agents' choices (travel alternative, departure time and route) are referred to as the travel decisions of the agents. The choices are taken based on utility maximization and the expected network conditions are used to predict the trips' travel time. In the general case, utility is expressed as a polynomial function of travel time, with a linear penalty for early or late arrivals at destinations and for early or late departures from origins (see Appendix E.1).

METROPOLIS2 follows a specific choice order, coherent with the decision-making process of agents in real world scenarios: agents select their travel alternative, then their departure time, and finally, their route. By relying on backward induction, METROPOLIS2 ensures that the optimal travel alternative is chosen knowing that the optimal departure time and route for that alternative will subsequently be determined. The chosen route is the fastest route at the time of departure. Departure-time choice can be represented by a Multinomial Logit model or a Continuous Logit model. Mode choice can be represented by a Multinomial Logit model or by a deterministic choice.[7] The output of the demand model are the travel decisions that the agents are planning to do during the simulated period.

In METROPOLIS2, a routing algorithm determines the route chosen by each agent based on their departure time from origin and the expected network conditions. The algorithm selects the fastest available route for the entire network. The underlying assumption mirrors real-life behavior, where individuals rely on car navigation system to plan their itinerary. This approach differs from many other simulators, which often constrain agents to choose their route from a limited choice set.

[7]Any discrete-choice models can be used for departure-time choice and mode choice, such as Probit model, as long as the random perturbations can be simulated.

When using discrete-choice models, the expected maximum utility of the travel decision is computed by METROPOLIS2 using the logsum formula (See Appendix E.1 for a formal definition). This value reflects the range of choices available to the agent in the complete choice hierarchy. It can be interpreted as the agent's surplus for cost-benefit analysis.

The demand model can be represented mathematically by a function f^{demand}, taking as an argument the expected network conditions $\hat{\boldsymbol{T}}$ and the input data $\boldsymbol{D}$, and returning the travel decisions of all agents $\mathbf{z} = \{z_n\}_{1 \leq n \leq N}$, i.e.,

$$f^{\text{demand}} : (\hat{\boldsymbol{T}}, \boldsymbol{D}) \mapsto f^{\text{demand}}(\hat{\boldsymbol{T}}; \boldsymbol{D}) = \mathbf{z}.$$

Supply model The supply model simulates the movements of vehicles on the road network.[8] These movements are a direct outcome of the travel decisions made by agents in the demand model. The simulation operates in continuous time using an event-based model, where events can represent a vehicle departing from its origin or a vehicle reaching the end of its current link, for example. These events are executed in chronological order.

Congestion is simulated using bottleneck queues and speed-density functions. Bottleneck queues limit the inflow and outflow of vehicles at the link-level, based on the link's capacity. The simulation of bottleneck queues is described in details in Section 3.3. With speed-density function, the speed at which a vehicle will travel on a link is determined by the density of vehicles currently on that link when the vehicle enters. The density on a link is computed as the sum of the headway of vehicles currently on the link, divided by the total length of the link (product of its length and its number of lanes). Additionally, when queue propagation is enabled, vehicles are constrained from entering a link unless enough space is available for their headway length. Following a vehicle's departure from a link, the liberated space it creates propagates backward through the link at a predetermined speed until it reaches the

[8] In the current version of METROPOLIS2, there exists a one-to-one correspondence between agents and vehicles, allowing these terms to be used interchangeably. However, it is worth noting that in future versions, this relationship might no longer hold, with the inclusion of features such as ride-sharing or public transit, where multiple agents might share a single vehicle.

link's origin. Only when this free space has reached the beginning of the link can entering vehicles utilize it.[9] At the end of the supply model, the simulated travel-time functions are computed at the link-level, based on the process described in Section 3.3 and Figure 4. These travel-time functions form the *simulated network conditions*.

The supply model can be represented mathematically by a function f^{supply}, taking as an argument the travel decisions of all agents $\mathbf{z}$ and the input data $\boldsymbol{D}$, and returning the simulated network conditions $\boldsymbol{T}$, i.e.,

$$f^{\text{supply}} : (\mathbf{z}; \boldsymbol{D}) \mapsto f^{\text{supply}}(\mathbf{z}; \boldsymbol{D}) = \boldsymbol{T}.$$

Learning model The learning model computes the expected network conditions to be used in the next iteration by combining the current expected network conditions with the simulated network conditions obtained from the supply model. These expected network conditions are then used in the demand model's routing algorithm to predict travel times. The learning model mirrors how car navigation systems leverage historical data from all users to forecast travel times. Different methods can be employed for the learning model, such as the exponential smoothing method, presented in Section 3.3.

The learning model can be represented mathematically by a function f^{learning}, taking as arguments the iteration counter κ, the simulated network conditions for the current iteration $\boldsymbol{T}_\kappa$, the expected network conditions for the current iteration $\hat{\boldsymbol{T}}_\kappa$ and the input data $\boldsymbol{D}$, and returning the expected network conditions for the next iteration $\hat{\boldsymbol{T}}_{\kappa+1}$, i.e.,

$$f^{\text{learning}} : (\kappa, \boldsymbol{T}_\kappa, \hat{\boldsymbol{T}}_\kappa; \boldsymbol{D}) \mapsto f^{\text{learning}}(\kappa, \boldsymbol{T}_\kappa, \hat{\boldsymbol{T}}_\kappa; \boldsymbol{D}) = \hat{\boldsymbol{T}}_{\kappa+1}.$$

[9] When queue propagation is enabled, gridlocks can appear. A gridlock is a situation in which traffic comes to a complete standstill because no vehicle can move until the vehicle in front of it moves. To address these gridlocks, METROPOLIS2 employs a mechanism where a vehicle is forced to move to the next link if it remains in a pending state for a specific duration (configurable). This approach is similar to the one taken in MATSim, while, in METROPOLIS1, the route choice at each intersection prevent gridlocks from occurring.

Nash equilibrium The goal of METROPOLIS2 is to find a Nash equilibrium of the simulation, defined by the input data $\boldsymbol{D}$. Here, a Nash equilibrium is defined as a state where no agent can improve their utility by unilaterally changing their travel decisions, considering the travel decisions made by the other agents. In the context of transport simulations, such a Nash equilibrium is also called an *agent-based User Equilibrium* (Nagel and Flötteröd, 2016). More formally, a Nash equilibrium can be defined as the following fixed-point problem. We want to find equilibrium travel decisions $\bar{\mathbf{z}}$ such that

$$f^{\text{demand}}(f^{\text{supply}}(\bar{\mathbf{z}}; \boldsymbol{D}); \boldsymbol{D}) = \bar{\mathbf{z}}.$$

or, alternatively, we want to find equilibrium network conditions $\bar{\boldsymbol{T}}$ such that

$$f^{\text{supply}}(f^{\text{demand}}(\bar{\boldsymbol{T}}; \boldsymbol{D}); \boldsymbol{D}) = \bar{\boldsymbol{T}},$$

Intuitively, these equations mean that the expected network conditions used by the agents when taking their decisions are identical to the simulated network conditions resulting from these agents' decisions. The definition of an equilibrium with the bottleneck model, in Section 3, equations (6) and (7), is a special case with departure time as the only decision variable and with a single travel-time function (corresponding to the single road).

Properties of the Nash equilibrium In dynamic traffic assignment models, there is no proof of the existence, uniqueness and stability of a Dynamic User Equilibrium in the general case (see Iryo 2013). A Nash equilibrium in METROPOLIS2 is a generalization of a standard Dynamic User Equilibrium (with the addition of mode choice, departure-time choice, trip chaining, etc.) so the existence, uniqueness and stability of a Nash equilibrium in METROPOLIS2 cannot be proven either. Algorithm 3 is a heuristic method to approximate a Nash equilibrium. Analytical applications (see Section 6) reveal that the algorithm does not perfectly converge to a Nash equilibrium. Therefore, it becomes essential to asses the

quality of the solution by measuring its distance to a Nash equilibrium. One approach to evaluate the solution's quality is by calculating the magnitude of change in travel decisions or network conditions relative to the previous iteration. Another option is to compute the distance between the simulated and expected network conditions. See Section 6 for example of indicators computing distance to a Nash equilibrium.

These indicators of a distance to a Nash equilibrium can be used as stopping criteria to ensure that the simulator only stop when the quality of the solution is good enough. Alternatively, one can fix the number of iterations to run and check a posteriori the quality of the solution.

5 Implementation

METROPOLIS2 is implemented as a Rust program. Rust was chosen for its excellent performance and safety guarantees. These safety guarantees ensure that code parallelization can be utilized with minimal risk, enhancing the efficiency and reliability of the program.

To enhance its performance, METROPOLIS2 employs time-dependent contraction hierarchies, a state-of-the-art routing algorithm (Batz et al., 2013). The simulator utilizes two types of routing queries: (i) profile queries, which determine the fastest travel time for any departure time, given an origin-destination pair (without returning corresponding routes), and (ii) earliest-arrival queries, which provide the earliest arrival time along with the corresponding route, for a given origin-destination pair and departure time from the origin. In the demand model, profile queries efficiently retrieve the travel-time function for all trips, required to compute the utility function used for departure-time choice. Earliest-arrival queries are used to retrieve routes once agents have selected their mode and departure time. The adoption of contraction hierarchies involves a preprocessing phase executed during each iteration of METROPOLIS2, enabling the acceleration of subsequent queries. Moreover, code parallelization is integrated into the routing algorithm to speed up the processes, such

as retrieving routes for all agents. Additionally, code parallelization is also employed in the demand model, where agents' decisions can be computed independently.

The choice of CSV and Parquet formats for input and output files in METROPOLIS2 offers a combination of usability, flexibility, and performance. CSV offers simplicity and compatibility with a wide range of tools and platforms. Parquet, on the other hand, is a columnar storage format that is optimized for performance and efficiency, making it suitable for handling large datasets efficiently. This combination of formats satisfies different use cases and preferences, ensuring that users can interact with METROPOLIS2 data in a way that best suits their needs.

Given the complexity of the simulator, issues or unexpected behaviors are likely to appear. To mitigate this risk, we have implemented a rigorous testing framework. The tests evaluate individual components (unit tests) as well as the interactions between various modules (integration tests). The example applications below (Section 6) also serve as validation tests for the correctness of the source code.

6 Applications

6.1 Comparison with METROPOLIS1

As discussed in Section 2, METROPOLIS2 replicates and improves most of the methodology used in METROPOLIS1. In this section, we compare the results of the two simulators when running the same simulation. We rely on the work of Saifuzzaman et al. (2012) which propose a calibrated simulation of METROPOLIS1 for Paris' urban area.[10]

Demand input The demand side of the simulation is composed of 477 067 agents, all performing a single trip between 4:00 a.m. and 1:00 p.m. There is no mode choice, car is

[10]Because we are missing some data, we were not able to replicate exactly their results. However, we did run the two simulators with the exact same input (same population and preference parameters, same road network).

the only available mode. The agents are divided in five categories. In each category, the agents have the same parameters α, β, γ, Δ and μ and the desired arrival times t^* are independent and identically distributed with a Normal or Uniform distribution. The studied area is divided in 1326 zones and there is one origin-destination matrix for each category.

Supply input The road network of the simulation is limited to the main roads of the area, which represent 17 987 nodes and 39 395 links. In addition, there are 4462 connectors (virtual links used to connect the zones' starting and ending nodes to the road network).

In METROPOLIS1, there is no explicit bottlenecks. Instead, the simulator uses the following speed-density function, imitating a road bottleneck:[11]

$$\text{speed(density)} = \min\left(\text{free-flow speed}, \frac{\text{vehicle headway} \times \text{capacity}}{\text{density}}\right). \tag{9}$$

Figure 6 proposes a representation of this speed-density function. This speed-density function is used in both METROPOLIS1 and METROPOLIS2, using the same road capacities. Queue propagation is enabled, with a vehicle headway of 8 meters. With this configuration, the congestion model of METROPOLIS2 is an exact replication of the congestion model of METROPOLIS1, with one difference: in METROPOLIS1, link travel times are truncated, while, in METROPOLIS2, time is continuous (there is no rounding).

[11] To understand equation (9), observe that, using

$$\text{density} = \frac{\text{vehicle headway} \times \text{vehicle count}}{\text{length} \times \text{lanes}},$$

the road travel time can be written as

$$\text{travel time(vehicle count)} = \frac{\text{length}}{\text{speed(density)}} = \min\left(\frac{\text{length}}{\text{free-flow speed}}, \frac{\text{vehicle count}}{\text{capacity} \times \text{lanes}}\right).$$

Assume that there are n vehicles on a road with a bottleneck of capacity s. Because of the bottleneck, there must be at most s vehicles traveling through the road during one hour. This implies that the n vehicles on the road cannot all cross the bottleneck in more than n/s hours. Therefore, the road travel-time is upper bounded by n/s, which corresponds to the second term in the "min" operator of the previous equation.

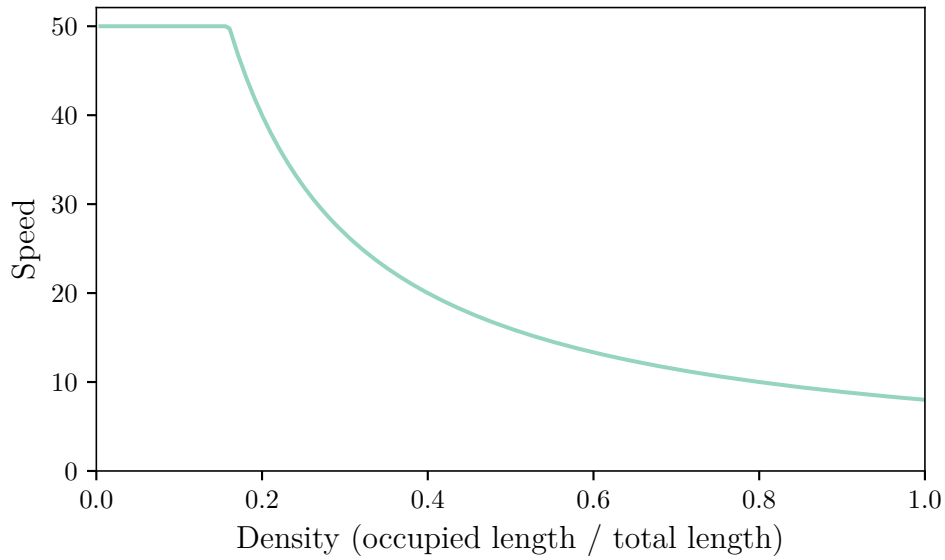

Figure 6: Bottleneck speed-density function on an example road (200 meters, free-flow speed of 50 km/h, 1 lane, capacity of 1000 vehicles per hour)

Technical parameters With both METROPOLIS1 and METROPOLIS2, we run 200 iterations using a linear learning model (the expected travel time for the next iteration is the arithmetic mean of the simulated travel time of all the previous iterations). Following Saifuzzaman et al. (2012), the interval length between two breakpoints is set to 20 minutes.

Output indicators To measure the convergence of the simulations, we use four different indicators. The two first indicators are used to measure by how much the choices of the agents (departure time and route) change from one iteration to another. The change in departure time is measured by the root of the mean squared difference between the departure time chosen in the previous and current iteration, for each agent. This indicator is denoted by $\mathrm{RMSE}_{\kappa}^{\mathrm{dep}}$, for iteration $\kappa > 1$, and is defined as

$$\mathrm{RMSE}_{\kappa}^{\mathrm{dep}} = \sqrt{\frac{1}{N}\sum_{n}(t_{n,\kappa}^{\mathrm{d}} - t_{n,\kappa-1}^{\mathrm{d}})^2},$$

where $t_{n,\kappa}^{\mathrm{d}}$ is the departure time chosen by agent n at iteration κ. The indicator is measured in seconds, so a value $\mathrm{RMSE}_{\kappa}^{\mathrm{dep}} = 5$, for example, means that, on average, the agents shifted

their departure time by plus or minus 5 seconds from iteration $\kappa - 1$ to iteration κ (note that, because the differences are squared, the larger shifts contribute more to the RMSE than the smaller shifts). The second indicator measures how much the routes taken by the agents change from one iteration to another. Let's denote by $\rho_{n,\kappa}$ the share of the length of the route taken by agent n during iteration κ that was not taken during iteration $\kappa - 1$, i.e., a value $\rho_{n,\kappa} = 5\,\%$, for example, means that $5\,\%$ of the length traveled by agent n during iteration κ was made on roads that were not taken during iteration $\kappa - 1$. The indicator $\mathrm{RMSE}_\kappa^{\mathrm{route}}$ is defined as

$$\mathrm{RMSE}_\kappa^{\mathrm{route}} = \sqrt{\frac{1}{N}\sum_n (\rho_{n,\kappa})^2}.$$

The third indicator measures how the simulated travel times (i.e., the simulated network conditions) differ from the expected travel times (i.e., the expected network conditions). To do so, we use the root of the mean squared difference between the simulated and expected travel time for iteration κ, for each link of the road network and for each departure time. This indicator is denoted by RMSE_κ^T and is defined as

$$\mathrm{RMSE}_\kappa^T = \sqrt{\frac{1}{R}\sum_r \frac{1}{t^1 - t^0}\int_{t^0}^{t^1} [T_{r,\kappa}(t) - \hat{T}_{r,\kappa-1}(t)]^2 \,\mathrm{d}\,t},$$

where $T_{r,\kappa}(t)$ (resp. $\hat{T}_{r,\kappa}(t)$) is the simulated (resp. expected) travel time on link r at time t during iteration κ and $[t^0, t^1] = [4\text{ a.m.}, 1\text{ p.m.}]$ is the simulated period.

The fourth indicator measures how well the agents predict the travel time that they will face. It is based on the travel-time expectation error: the difference between the actual and expected travel time. The indicator is denoted by $\mathrm{RMSE}_\kappa^{\mathrm{expect}}$ and is defined as

$$\mathrm{RMSE}_\kappa^{\mathrm{expect}} = \sqrt{\frac{1}{N}\sum_n (\mathrm{tt}_{n,\kappa} - \hat{\mathrm{tt}}_{n,\kappa})^2},$$

where $\mathrm{tt}_{n,\kappa}$ is the travel time of agent n for iteration κ and $\hat{\mathrm{tt}}_{n,\kappa}$ is the expected travel time of agent n for iteration κ.

Output comparison A comparison of the results of the simulations with METROPOLIS1 and METROPOLIS2 is provided in Table 6.[12] The technical results show that METROPOLIS2 runs slightly faster than METROPOLIS1 when using a single thread. However, as METROPOLIS2 can make use of parallel computing, the running time is around 9 times faster when using 16 threads compared to using a single thread (or around 10 times faster compared to the single-threaded METROPOLIS1). METROPOLIS2 peak memory use is almost 30 % larger than METROPOLIS1'. One reason for this difference is that METROPOLIS1 uses single-precision floating-point format (32 bits), while METROPOLIS2 uses double-precision (64 bits).

Table 6: Comparison of the aggregate results for a Paris simulation with METROPOLIS1 and METROPOLIS2

	METROPOLIS1	METROPOLIS2
Technical output		
Running time (1 thread)	18h 29m 12s	16h 12m 48s
Running time (16 threads)	18h 29m 12s	1h 49m 17s
Peak memory use	2.74 GiB	3.52 GiB
Convergence output		
$\text{RMSE}_{200}^{\text{dep}}$	2m 5s	5s
$\text{RMSE}_{200}^{\text{route}}$	18.07 %	5.49 %
RMSE_{200}^{T}	1m 40s	46s
$\text{RMSE}_{200}^{\text{expect}}$	10m 28s	6m 05s
Traffic-related output		
Average utility	−5.58 €	−5.32 €
Average departure time	09:10:44	09:11:58
Average arrival time	09:27:07	09:27:31
Average travel time	16m 23s	15m 33s
Average free-flow travel time	13m 24s	13m 1s
Average route length	15.51 km	15.05 km
Average number of links taken	37.28	35.33

The output related to the convergence of the simulations show that METROPOLIS2

[12] All simulations are run on a machine with a 8-core Intel Xeon CPU (3.20GHz) and with 128GiB of RAM. The source code to run the simulations and generate the graphs is available at `https://github.com/LucasJavaudin/Metropolis1-2Comparison/`. The simulations are run with version 1.0.0 of METROPOLIS2.

achieves a solution that is more stable, in all aspects. This reveals that METROPOLIS2's solution is closer to a Nash equilibrium than METROPOLIS1' solution.[13]

Figure 7 compares the convergence of the $\text{RMSE}_{\kappa}^{\text{dep}}$ metric across iterations, between METROPOLIS1 and METROPOLIS2. Initially, both simulators exhibit similar convergence patterns for the first 15 to 20 iterations. Then, METROPOLIS2 continues to steadily decrease, while METROPOLIS1 appears to reach a plateau.

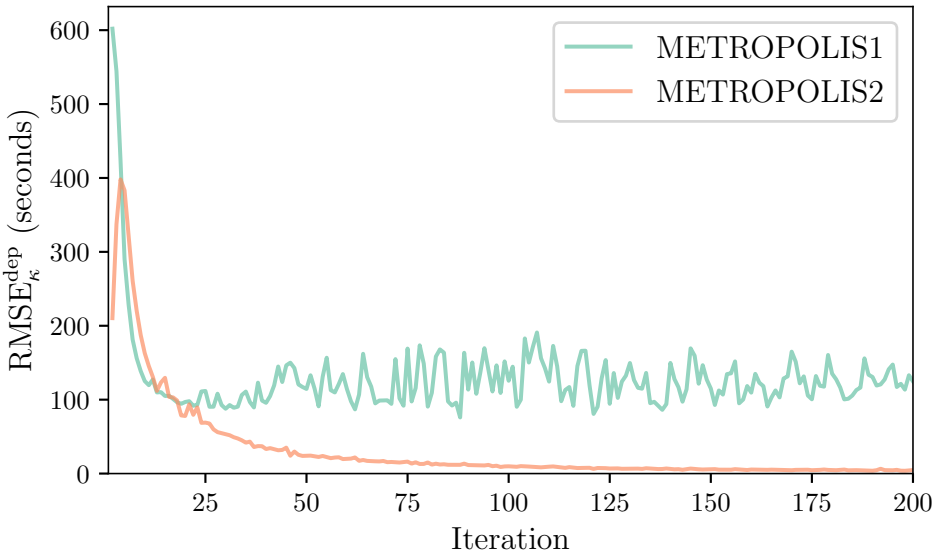

Figure 7: Comparison of the convergence of departure times over iterations, between METROPOLIS1 and METROPOLIS2

Figure 8 compares the convergence of the $\text{RMSE}_{\kappa}^{\text{expect}}$ metric across iterations, between METROPOLIS1 and METROPOLIS2. Initially, METROPOLIS2 shows higher levels of $\text{RMSE}_{\kappa}^{\text{expect}}$. As iterations progress, METROPOLIS2 achieves lower levels of the metric compared to METROPOLIS1. On the other hand, METROPOLIS1 reaches a plateau after about 50 iterations but exhibits less variation of the metric from one iteration to another. This difference in behavior can be attributed to the adaptability of the agents in METROPO-

[13]Observe that the $\text{RMSE}_{200}^{\text{expect}}$ is still quite large in both METROPOLIS1 and METROPOLIS2. This is due to threshold effects that can appear with the speed-density function, where the road travel time can jump from a few seconds to a few minutes with just one more car being on the road segment. These threshold effects are less prominent when using a larger population size (the simulation presented uses 10 % of total population), a smaller breakpoint interval and explicit bottleneck queues instead of speed-density functions.

LIS1: they can adjust their routes during the trip when encountering congestion, thereby avoiding extreme travel times; while, in METROPOLIS2, the route is chosen before leaving origin (in the demand model) and cannot be changed afterward. However, this adaptability results in more frequent route changes between iterations, leading to less predictable travel times overall.

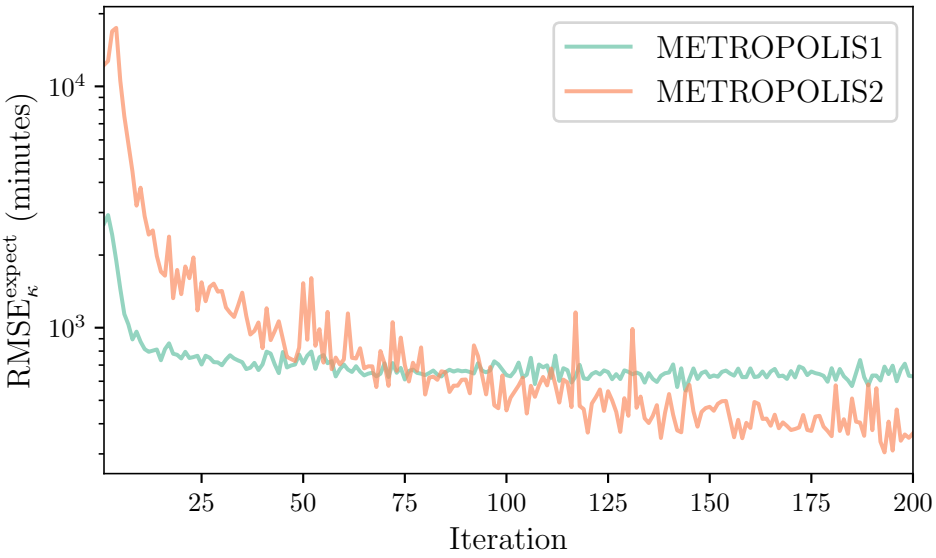

Figure 8: Comparison of the convergence of travel-time expectation error over iterations, between METROPOLIS1 and METROPOLIS2

The traffic-related output on Table 6 shows that METROPOLIS2 reaches a situation were agents are slightly better-off compared to the situation returned by METROPOLIS1 (utility is larger by 26 cents on average, travel time is smaller by 50 seconds on average, etc.). As we mentioned previously, the congestion is modeled in the same way in the two simulations, which means that these differences are explained by the choices of the agents only.

Figure 9 shows that the departure-time distributions are very similar for the two simulations. The peak around 8:00 a.m. is slightly more pronounced in METROPOLIS2. Figure 10 shows that the travel-time distributions are also very similar. Figure 11 shows that the link-level flows are about the same between the two simulations: the correlation coefficient

between link-level flows in METROPOLIS1 and METROPOLIS2 is 95.37 %.

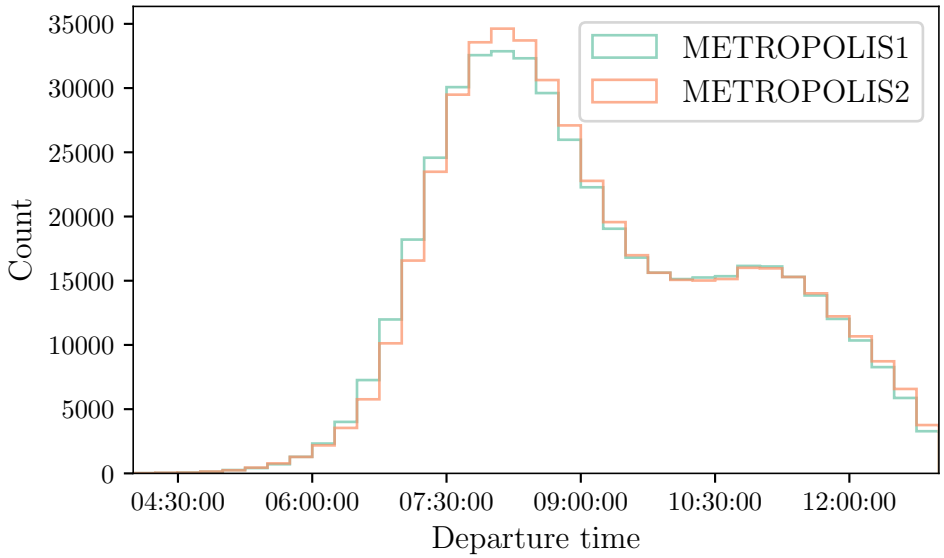

Figure 9: Comparison of the departure-time distribution in the Paris simulation with METROPOLIS1 and METROPOLIS2

In conclusion, the results from METROPOLIS1 and METROPOLIS2 are similar but the latter runs faster and provides a closer approximation to equilibrium compared to METROPOLIS1. Further improvements in convergence could be achieved in METROPOLIS2 by using explicit bottleneck queues, defined in Section 3.3.[14] However, the use of bottleneck queues leads to a distinct congestion model,[15] complicating comparison between

[14]The use of speed-density functions can lead to a simulation that exhibits less stability compared to simulations using explicit bottleneck queues. This instability comes from a discontinuity in the measure of density. Specifically, when a vehicle enters a road at approximately the same time another vehicle exits the same road, the entering vehicle's travel time can experience a rapid jump from small to large value, conditional on whether the exiting vehicle left just before or just after the entry. This discontinuity is more prominent on shorter roads with smaller capacities. Conversely, with explicit bottleneck queues, there is no such discontinuities. The waiting time for a vehicle in the queue decreases linearly with the time that elapsed since the preceding vehicle exited, eventually reaching zero.

[15]Consider a road with a free-flow travel time of 10 seconds and a bottleneck capacity of 300 vehicles per hour (i.e., a bottleneck closing time of 12 seconds). If there is a vehicle entering the road every 11 seconds, the congestion differs significantly between the bottleneck speed-density function and the bottleneck queue. When using the bottleneck speed-density function, equation (9), there is no congestion. This is because the density remains null when a new vehicle enters the road, as the preceding vehicle has already left. However, when using bottleneck queues, congestion does appear. Despite the first vehicle exiting the road before the second one enters, the second vehicle is forced to wait until the bottleneck closing time elapses. Consequently, a queue forms with an increasing waiting time as subsequent vehicles are arriving at a faster rate than the bottleneck's output rate.

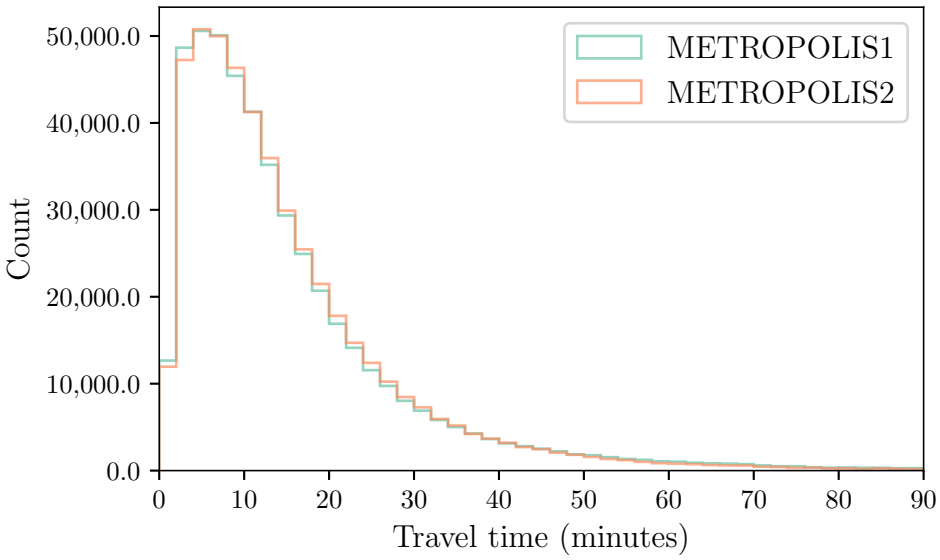

Figure 10: Comparison of the travel-time distribution in the Paris simulation with METROPOLIS1 and METROPOLIS2

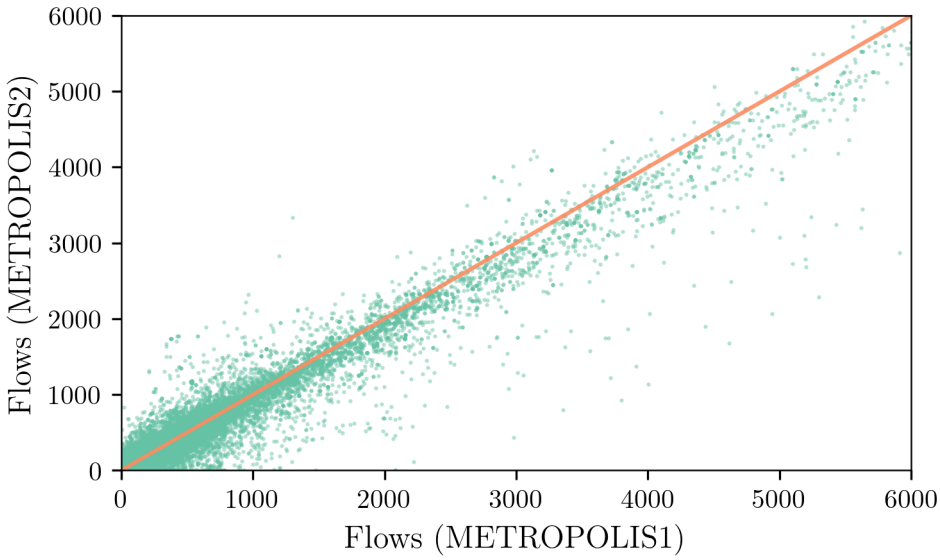

Figure 11: Comparison of the link-level flows for a Paris simulation with METROPOLIS1 and METROPOLIS2

Note: Each dot represents a link of the road network, with the observed flows in the METROPOLIS1 simulation on the x-axis and the observed flows in the METROPOLIS2 simulation on the y-axis.

convergence with speed-density functions and bottleneck queues.

7 Conclusion

This paper has detailed the methodology and implementation details of the METROPOLIS2 simulator. Beginning with the foundational single-road bottleneck model, we provided insights into METROPOLIS2's functionality and demonstrated its ability to replicate key results from the analytical model. We then proposed a comprehensive overview of METROPOLIS2's capabilities in general scenarios, discussed some implementation details and conducted a comparative analysis with the METROPOLIS1 simulator using a Paris case study.

METROPOLIS2 exhibits promising potential for efficiently and accurately simulating traffic equilibrium in large-scale scenarios, making it well-suited for cost-benefit analysis. However, the discussion on generating a synthetic population and calibrating the simulator for real-world scenarios, though crucial, remains unexplored in this paper, leaving avenues for future research.

Furthermore, future work may explore additional features for the simulator. For example, integrating a public-transit model could enhance METROPOLIS2's capabilities, enabling consideration of in-vehicle congestion in route choices for public-transit trips.

Acknowledgments

The authors are grateful for the pioneering contributions of Fabrice Marchal and Yurii Nesterov to the METROPOLIS1 simulator, which laid robust foundations facilitating the development of METROPOLIS2. Additionally, the authors wish to thank Moshe Ben-Akiva, Michel Bierlaire, Nicolas Coulombel, Nikolas Geroliminis, Samarth Ghoslya, Moez Kilani, Romuald Le Frioux, and Robin Lindsey for providing valuable insights and feedback during the development stages of METROPOLIS2.

References

Adnan, M., F. C. Pereira, C. M. L. Azevedo, K. Basak, M. Lovric, S. Raveau, Y. Zhu, J. Ferreira, C. Zegras, and M. Ben-Akiva (2016). Simmobility: A multi-scale integrated agent-based simulation platform. In *95th Annual Meeting of the Transportation Research Board Forthcoming in Transportation Research Record*, Volume 2. The National Academies of Sciences, Engineering, and Medicine Washington, DC.

Arnott, R., A. de Palma, and R. Lindsey (1990, January). Economics of a bottleneck. *Journal of Urban Economics 27*(1), 111–130.

Arnott, R., A. de Palma, and R. Lindsey (1993). A Structural Model of Peak-Period Congestion: A Traffic Bottleneck with Elastic Demand. *The American Economic Review 83*(1), 161–179.

Basu, R., A. Araldo, A. P. Akkinepally, B. H. Nahmias Biran, K. Basak, R. Seshadri, N. Deshmukh, N. Kumar, C. L. Azevedo, and M. Ben-Akiva (2018, December). Automated Mobility-on-Demand vs. Mass Transit: A Multi-Modal Activity-Driven Agent-Based Simulation Approach. *Transportation Research Record: Journal of the Transportation Research Board 2672*(8), 608–618.

Batz, G. V., R. Geisberger, P. Sanders, and C. Vetter (2013, December). Minimum time-dependent travel times with contraction hierarchies. *ACM Journal of Experimental Algorithmics 18*.

Ben-Akiva, M., N. Litinas, and K. Tsunokawa (1985, March). Continuous spatial choice: The continuous logit model and distributions of trips and urban densities. *Transportation Research Part A: General 19*(2), 119–154.

Ben-Akiva, M. and T. Watanatada (1981). Application of a continuous spatial choice logit model. *Structural analysis of discrete data with econometric applications*, 320–343.

Charypar, D. and K. Nagel (2005, July). Generating complete all-day activity plans with genetic algorithms. *Transportation 32*(4), 369–397.

de Palma, A., M. Ben-Akiva, C. Lefèvre, and N. Litinas (1983, November). Stochastic Equilibrium Model of Peak Period Traffic Congestion. *Transportation Science 17*(4), 430–453.

de Palma, A., L. Javaudin, P. Stokkink, and L. Tarpin-Pitre (2022). Ride-sharing with inflexible drivers in the Paris metropolitan area. *Transportation*, 1–24.

de Palma, A., M. Kilani, and R. Lindsey (2005, August). Congestion pricing on a road network: A study using the dynamic equilibrium simulator METROPOLIS. *Transportation Research Part A: Policy and Practice 39*(7-9), 588–611.

de Palma, A. and F. Marchal (1999, January). Analysis of Travel Cost Components Using Large-Scale, Dynamic Traffic Models. *Transportation Research Record: Journal of the Transportation Research Board 1676*(1), 177–183.

de Palma, A. and F. Marchal (2002, July). Implementation of a Dynamic Traffic Simulator to the Paris Area. In *Traffic And Transportation Studies (2002)*, Guilin, China, pp. 1216–1223. American Society of Civil Engineers.

de Palma, A., F. Marchal, and Y. Nesterov (1997, January). METROPOLIS: Modular System for Dynamic Traffic Simulation. *Transportation Research Record: Journal of the Transportation Research Board 1607*(1), 178–184.

Flötteröd, G. (2016, August). *MATSim as a Monte-Carlo Engine*, pp. 327–336. Ubiquity

Press.

Geisberger, R. and P. Sanders (2010). Engineering Time-Dependent Many-to-Many Shortest Paths Computation. pp. 14 pages, 487697 bytes.

Guo, R.-Y., H. Yang, and H.-J. Huang (2018, June). Are We Really Solving the Dynamic Traffic Equilibrium Problem with a Departure Time Choice? *Transportation Science 52*(3), 603–620.

Hörl, S., M. Balac, and K. W. Axhausen (2018). A first look at bridging discrete choice modeling and agent-based microsimulation in MATSim. *Procedia Computer Science 130*, 900–907.

Hörl, S., M. Balać, and K. W. Axhausen (2019, January). Pairing discrete mode choice models and agent-based transport simulation with MATSim. pp. 19 p.

Horni, A., K. Nagel, and K. W. Axhausen (2011, August). High-resolution destination choice in agent-based demand models. pp. 23 p.

Horni, A., K. Nagel, and K. W. Axhausen (Eds.) (2016). *The Multi-Agent Transport Simulation MATSim*. London: Ubiquity Press.

Iryo, T. (2013, April). Properties of dynamic user equilibrium solution: Existence, uniqueness, stability, and robust solution methodology. *Transportmetrica B: Transport Dynamics 1*(1), 52–67.

Kickhöfer, B. and K. Nagel (2016, August). *Microeconomic Interpretation of MATSim for Benefit-Cost Analysis*, pp. 353–364. Ubiquity Press.

Li, Z.-C., H.-J. Huang, and H. Yang (2020, September). Fifty years of the bottleneck model: A bibliometric review and future research directions. *Transportation Research Part B: Methodological 139*, 311–342.

Lopez, P. A., E. Wiessner, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flotterod, R. Hilbrich, L. Lucken, J. Rummel, and P. Wagner (2018, November). Microscopic Traffic Simulation using SUMO. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, HI, pp. 2575–2582. IEEE.

Lu, Y., K. Basak, C. Carrion, H. Loganathan, M. Adnan, F. C. Pereira, V. H. Saber, and M. Ben-Akiva (2015). SimMobility mid-term simulator: A state of the art integrated agent based demand and supply model. *Transportation Research Board 94th Annual Meeting-Transportation Research Board* (15-3937).

Madow, W. G. and L. H. Madow (1944). On the Theory of Systematic Sampling, I. *The Annals of Mathematical Statistics 15*(1), 1–24.

McNally, M. G. (2007, January). The Four-Step Model. In D. A. Hensher and K. J. Button (Eds.), *Handbook of Transport Modelling*, Volume 1, pp. 35–53. Emerald Group Publishing Limited.

Nagel, K. (1996, May). Particle hopping models and traffic flow theory. *Physical Review E 53*(5), 4655–4672.

Nagel, K. and G. Flötteröd (2016, August). *Agent-Based Traffic Assignment*, pp. 315–326. Ubiquity Press.

Nguyen, J., S. T. Powers, N. Urquhart, T. Farrenkopf, and M. Guckert (2021, December). An overview of agent-based traffic simulators. *Transportation Research Interdisciplinary Perspectives 12*, 100486.

Oh, S., R. Seshadri, C. L. Azevedo, N. Kumar, K. Basak, and M. Ben-Akiva (2020, August). Assessing the impacts of automated mobility-on-demand through agent-based simulation:

A study of Singapore. *Transportation Research Part A: Policy and Practice 138*, 367–388.
Otsubo, H. and A. Rapoport (2008, December). Vickrey's model of traffic congestion discretized. *Transportation Research Part B: Methodological 42*(10), 873–889.
Saifuzzaman, M., A. de Palma, and K. Motamedi (2012). Calibration of METROPOLIS for ile-de-france.
Vickrey, W. S. (1969). Congestion Theory and Transport Investment. *The American Economic Review 59*(2), 251–260.
Vosough, S., A. de Palma, and R. Lindsey (2022, July). Pricing vehicle emissions and congestion externalities using a dynamic traffic network simulator. *Transportation Research Part A: Policy and Practice 161*, 1–24.

A Bottleneck Model: Numerical Application

This appendix presents the results of simulations of METROPOLIS2 for the bottleneck model presented in Section 3. It contains some definitions (Section A.1), a comparison to the analytical results from de Palma et al. (1983) (Section A.2) and some sensitivity analysis (Section A.3).

A.1 Definitions and Notations

Figure 7 shows the default values considered for the various parameters of the bottleneck model investigated. We consider a population of 100 000 agents who can choose a departure time during the time period 7:00 a.m. to 8:00 a.m. The desired arrival time of the agents at destination is $t^* =$ 7:30 a.m. The travel time from origin to destination when the bottleneck is not congested is $t^{\mathrm{f}} = 30$ seconds. The bottleneck capacity is set to $s = 1.5 \times N$ cars per hour, which means that all agents can cross the bottleneck in 40 minutes. In the simulations, the interval between two breakpoints for the piecewise-linear travel-time functions is set to $\delta = 1$ minute. The initial value for travel-time function, $\hat{T}_1$, is set to the no-congestion travel-time function (a constant function equal to the free-flow travel-time t^{f}).

Table 7: Default values for the parameters of the model

Parameter	Notation	Value
Number of agents	N	100 000
Time period	$[t^0, t^1]$	[7:00 a.m., 8:00 a.m.]
Desired arrival time at destination	t^*	7:30 a.m.
Marginal disutility of travel time	α	10 \$/h
Marginal disutility of early arrivals	β	5 \$/h
Marginal disutility of late arrivals	γ	7 \$/h
Scale of the utility's random component	μ	1
Free-flow origin-to-destination travel time	t^{f}	30 seconds
Bottleneck flow rate	s	$1.5 \times N$ cars / h
Travel-time function breakpoints interval	δ	1 minute

Measuring convergence of the simulation When running the simulations, the stopping criteria, as defined in Algorithm 2, line 7, are never satisfied. Instead, we run the simulations for a fixed number of iterations and check how far the results are from an equilibrium using the two indicators presented below.

First, we use the root of the mean squared difference between the departure time chosen in the previous and current iteration, for each agent. This indicator is denoted by $\mathrm{RMSE}^{\mathrm{dep}}_{\kappa}$,

for iteration $\kappa > 1$, and is defined as

$$\text{RMSE}_{\kappa}^{\text{dep}} = \sqrt{\frac{1}{N}\sum_{n}(t_{n,\kappa}^{\text{d}} - t_{n,\kappa-1}^{\text{d}})^2},$$

where $t_{n,\kappa}^{\text{d}}$ is the departure time chosen by agent n at iteration κ. Second, we use the root of the mean squared difference between the expected travel-time function and the simulated travel-time function, for each departure time. This indicator is denoted by RMSE_{κ}^{T}, for iteration $\kappa > 1$, and is defined as

$$\text{RMSE}_{\kappa}^{T} = \sqrt{\frac{1}{t^1 - t^0}\int_{t^0}^{t^1}[T_{\kappa}(t) - \hat{T}_{\kappa}(t)]^2 \,\text{d}\, t}.$$

For an iteration κ, $\text{RMSE}_{\kappa}^{\text{dep}} \approx 0$ implies that the agents did not change much their departure time compared to the previous iteration and $\text{RMSE}_{\kappa}^{T} \approx 0$ implies that the travel-time function was correctly anticipated. Therefore, these indicators can be interpreted as a distance to an equilibrium.

Measuring distance to the analytical results The analytical results of the bottleneck model we consider are demonstrated in de Palma et al. (1983). From their findings, we compute the equilibrium rate of departures from origin, $\overline{r}^{\text{d}}(t)$ (see Figure 12), and the equilibrium travel-time function, $\overline{T}(t)$ (see Figure 13).

To compare the simulation results with the analytical results, we use the following distance:[16]

$$D = \max_{t}\left|R^{\text{d}}(t) - \overline{R}^{\text{d}}(t)\right|,$$

with

$$\overline{R}^{\text{d}}(t) = \frac{1}{N}\int_{t^0}^{t}\overline{r}^{\text{d}}(t)\,\text{d}\, t$$

the cumulative rate of departures from the analytical results and

$$R^{\text{d}}(t) = \frac{1}{N}\sum_{n}\mathbf{1}_{t_n^{\text{d}} \leq t}$$

the cumulative rate of departures from the simulation.

Learning model For the learning model, we use the exponential smoothing defined by equation 8. Further research is required to determine the optimal value of the smoothing

[16]Observe that this measure corresponds to the Kolmogorov-Smirnov statistic for comparing a sample of values to a probability distribution, which justify using it here.

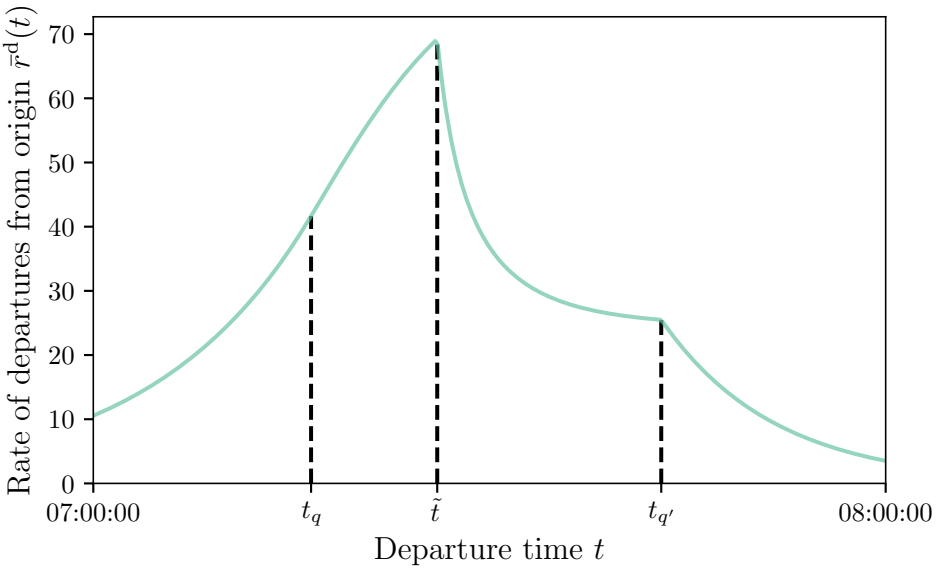

Figure 12: Analytical result for the rate of departures from origin

Note: t_q is the time at which congestion starts to appear, $\tilde{t}$ is the departure time such that the individual arrives on time, $t_{q'}$ is the time at which congestion ends.

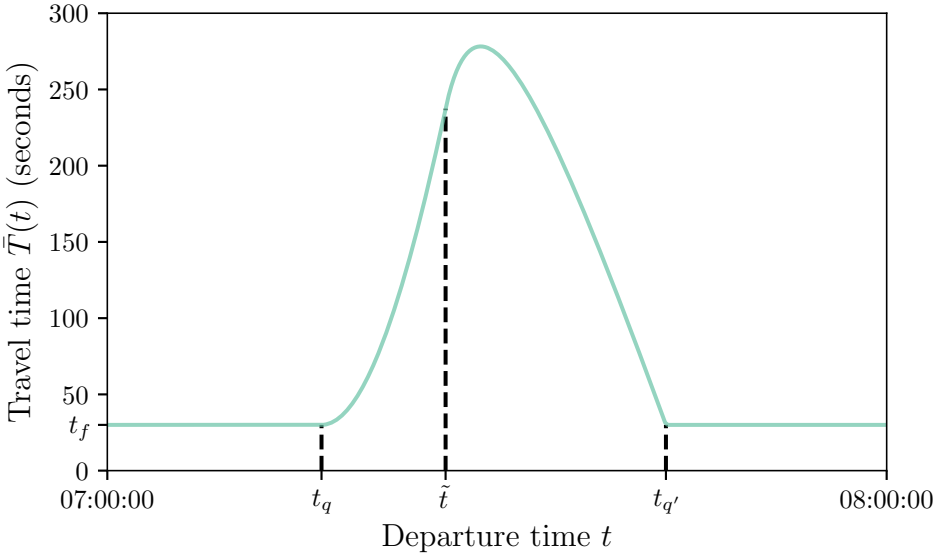

Figure 13: Analytical result for the travel-time function

Note: t_q, $\tilde{t}$, $t_{q'}$ are defined as in Figure 12.

factor λ to use. We give here some guidance for the choice of λ resulting from our experimentations. With smaller values of λ, a smaller weight is put on the simulated travel-time function for the current iteration so the model learns slower but the convergence is more stable (in the sense that it is less sensitive to what happened during a single iteration). Conversely, with larger values of λ, convergence is usually faster but less stable.

In the simulations presented below, we use a value of $\lambda = 0.4$, unless specified otherwise. The impact of λ on the convergence of the simulations is further discussed in Section A.3, using numerical results.

A.2 Comparison to the Analytical Results

We simulate 200 iterations of METROPOLIS2 using the parameters defined in Table 7.[17] Some results for the simulation are presented in Table 8. First, observe that, the simulation is essentially reaching an equilibrium as, at the last iteration, both $\text{RMSE}_\kappa^{\text{dep}}$ and RMSE_κ^T are practically zero (3×10^{-12} and 2×10^{-12} seconds respectively). This means that, from one iteration to another, the chosen departure times stay almost identical and that the simulated travel-time function in the supply model is almost identical to the expected travel-time function in the demand model. One reason the simulation is not converging to exactly zero might be related to the approximation errors of floating-point arithmetic. The maximum distance between the cumulative rate of departures in the analytical results and in the simulation results is $D = 0.19\,\%$, which indicates that the simulation replicates quite well the analytical results. We also find that the expected maximum utility, computed using the logsum formula (3), is at 7.188, which is close to the equilibrium value of 7.276 from the analytical results.

Table 8: Results of running METROPOLIS2 to simulate the bottleneck model

Variable	Value
Running time	3m 24s
Expected max. utility	7.188
Average travel time	1m 57s
Distance to analytical results D	$0.19\,\%$
$\text{RMSE}_{200}^{\text{dep}}$ (seconds)	3×10^{-12}
RMSE_{200}^{T} (seconds)	2×10^{-12}

Figures 14 and 15 show a comparison of the rate of departures from origin and the travel-

[17] All simulations are run on a machine with a 8-core Intel Xeon CPU (3.20GHz) and with 128GiB of RAM. The source code to run the simulations and generate the graphs is available at `https://github.com/LucasJavaudin/MetropolisBottleneck/`. The simulations are run with version 1.0.0 of METROPOLIS2.

time function between the simulation and the analytical results. For the rate of departures (Figure 14), we observe some noise in the curve corresponding to the simulation, due to the discretization of the agents, but the two curves are always close to each other. For the travel-time function (Figure 15), there is less noise and the curves follows each other very closely. The maximum travel time is about 3 seconds larger in the simulation results than in the analytical results.

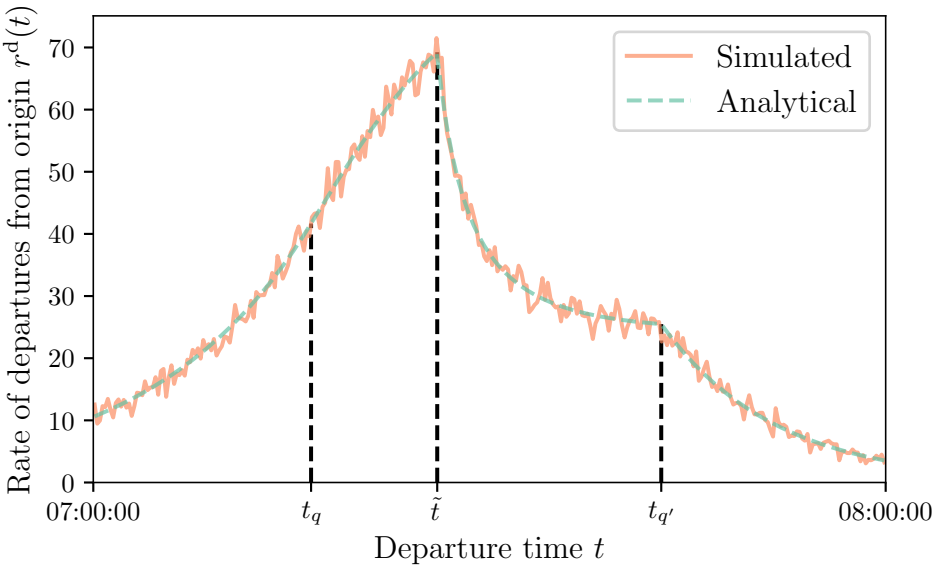

Figure 14: Comparison of the rate of departures from origin in the simulation and in the analytical results

Note: t_q, $\tilde{t}$, $t_{q'}$ are defined as in Figure 12.

Figure 16 shows the convergence of $\mathrm{RMSE}_{\kappa}^{\mathrm{dep}}$ and $\mathrm{RMSE}_{\kappa}^{T}$ from iteration to iteration. Both indicators steadily decrease for the first 120 iterations. Then, the indicators stay stable for the remaining iterations.

Running with different random seeds The only randomness in the simulation is due to the draw of the uniform random numbers u_n to simulate the departure time of each agent using inverse sampling. We run 10 simulations with different random seeds to investigate how the values of the draws affect the results. Table 9 shows how the results vary from one simulation to another. The results are roughly similar for all 10 simulations. For example, the distance to analytical results D varies between 0.19 % and 0.38 %.

When averaging the departure rates for the 10 simulations, we find a distance to the analytical results of $D = 0.10\,\%$, which is almost twice smaller than the smallest distance for any individual simulation. The reason is that averaging over multiple simulations is

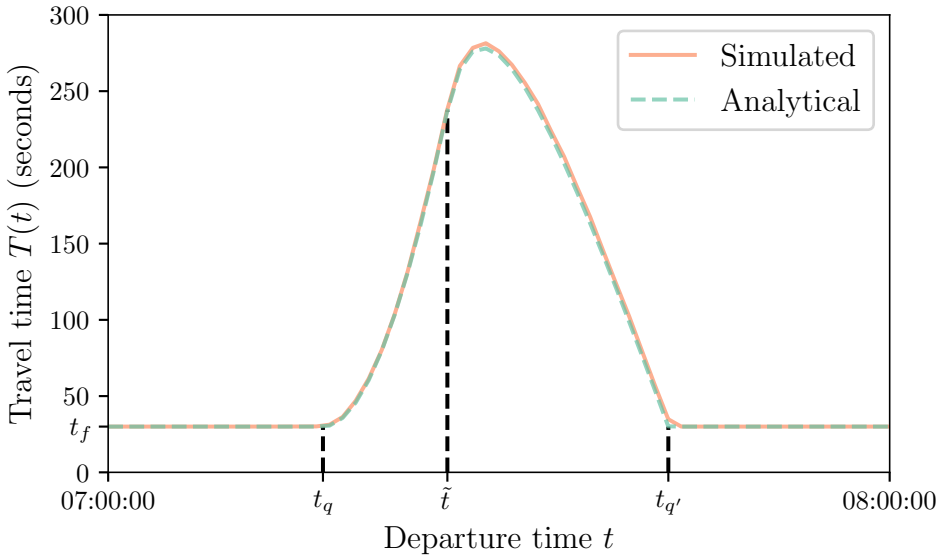

Figure 15: Comparison of the travel-time function from origin to destination in the simulation and in the analytical results

Note: t_q, $\tilde{t}$, $t_{q'}$ are defined as in Figure 12.

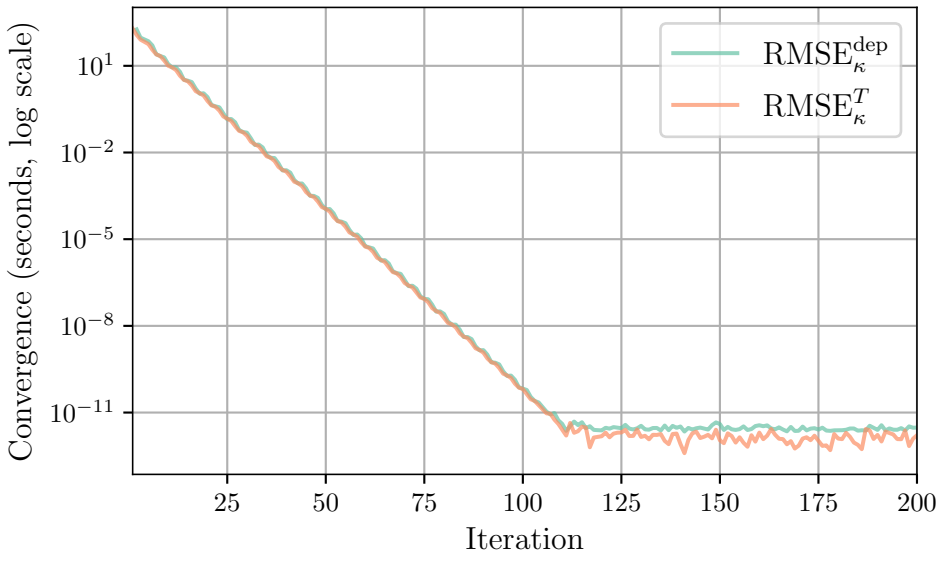

Figure 16: Convergence of $\text{RMSE}_{\kappa}^{\text{dep}}$ and RMSE_{κ}^{T}

Table 9: Average, minimum and maximum values for the results with 10 different random seeds

Variable	Average	Minimum	Maximum
Running time	3m 33s	3m 30s	3m 42s
Expected max. utility	7.190	7.187	7.194
Average travel time	1m 56s	1m 55s	1m 57s
Dist. analytical res. D	0.26 %	0.19 %	0.38 %
$\text{RMSE}^{\text{dep}}_{200}$ (seconds)	3×10^{-12}	2×10^{-12}	3×10^{-12}
RMSE^{T}_{200} (seconds)	1×10^{-12}	8×10^{-13}	2×10^{-12}

akin to simulating more individuals, which reduces the noise created by drawing random values.[18] Figure 17 shows a comparison of the analytical departure rate and the departure rate resulting from the average of the 10 simulations. Compared to Figure 14, we observe much less noise for the simulated departure rate.

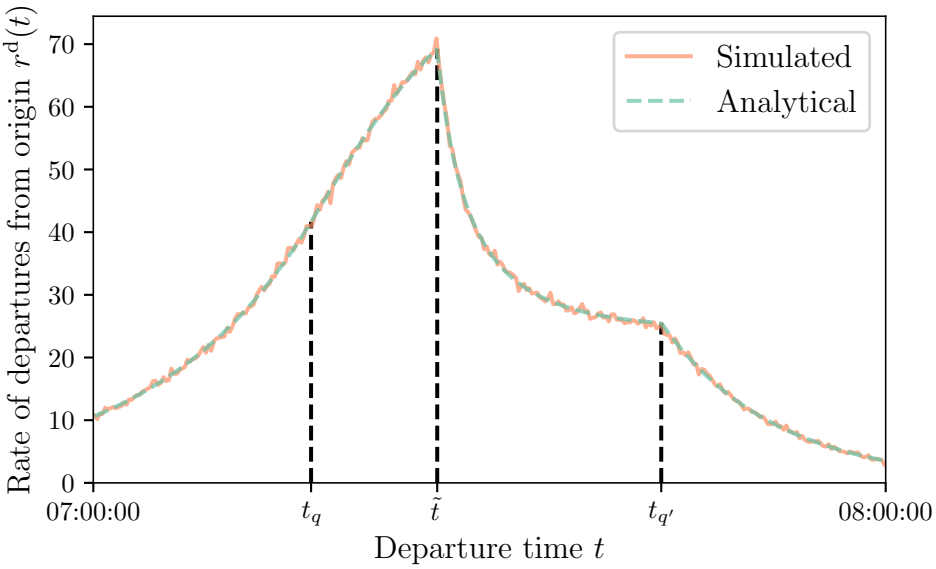

Figure 17: Comparison of the rate of departures when averaging over 10 simulations
Note: t_q, $\tilde{t}$, $t_{q'}$ are defined as in Figure 12.

Using systematic sampling We now try to use systematic sampling (Madow and Madow, 1944) to draw the uniform random numbers u_n. Concretely, the u_n values are set to be evenly spaced between 0 and 1, with an interval of $1/N$ between two values. Figure 18 compares

[18] As shown in Section A.3, the distance to the analytical results gets smaller as the number of agents simulated increases.

the simulated and analytical equilibrium rate of departures, when running 200 iterations of METROPOLIS2. With systematic sampling, we reach a smaller distance to the analytical results of $D = 0.06\,\%$. The reason is that, by using evenly spaced values of u_n, we reduce greatly the noise induced by drawing random values.

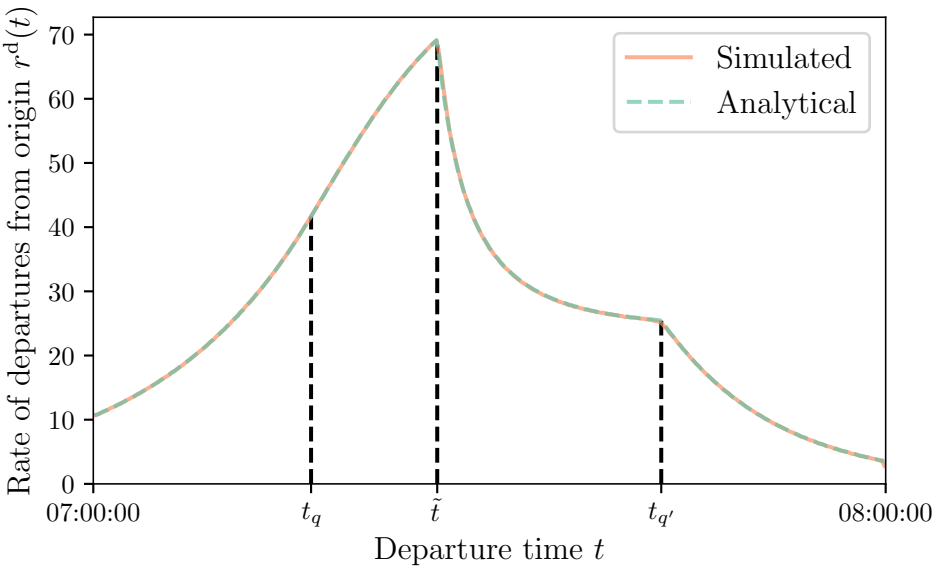

Figure 18: Comparison of the rate of departures when using systematic sampling
Note: t_q, $\tilde{t}$, $t_{q'}$ are defined as in Figure 12.

A.3 Sensitivity Analysis

Impact of the smoothing factor The smoothing factor λ is used in the exponential smoothing, equation 8. It represents the weight of the simulated travel-time function of the current iteration over the weight of the expected travel-time function of the previous iteration. We run a simulation for 1000 iterations with 6 different values of λ: 0.05, 0.2, 0.4, 0.6, 0.8 and 1. Table 10 shows the main results of the simulations for the different values of λ. Figure 19 shows the convergence of $\mathrm{RMSE}_{\kappa}^{\mathrm{dep}}$ for the different values of λ (we do not show here the convergence of $\mathrm{RMSE}_{\kappa}^{T}$ as it is roughly the same). For a value of $\lambda = 1$, there is no convergence: $\mathrm{RMSE}_{\kappa}^{\mathrm{dep}}$ stays at high values. For the other values of λ, the simulation always converge to a very low value of $\mathrm{RMSE}_{\kappa}^{\mathrm{dep}}$ and $\mathrm{RMSE}_{\kappa}^{T}$ and the distance to the analytical results is the same. The convergence is the fastest with $\lambda = 0.4$, which justify using this value in the previous section. We do not observe any major difference with regards to the running time of the simulation between the different values of λ.

Table 10: Sensitivity of the results to the smoothing factor

λ	0.05	0.2	0.4	0.6	0.8	1
Running time	17m 30s	17m 35s	17m 38s	17m 35s	17m 33s	18m
Expected max. surplus	7.188	7.188	7.188	7.188	7.188	6.921
Average travel time	1m 57s	1m 57s	1m 57s	1m 57s	1m 57s	2m 13s
Dist. analytical res. D	0.19 %	0.19 %	0.19 %	0.19 %	0.19 %	13.67 %
$\text{RMSE}^{\text{dep}}_{200}$	2×10^{-12}	4×10^{-12}	3×10^{-12}	3×10^{-12}	7×10^{-12}	2×10^{2}
RMSE^{T}_{200}	2×10^{-12}	1×10^{-12}	3×10^{-12}	2×10^{-12}	4×10^{-12}	2×10^{2}

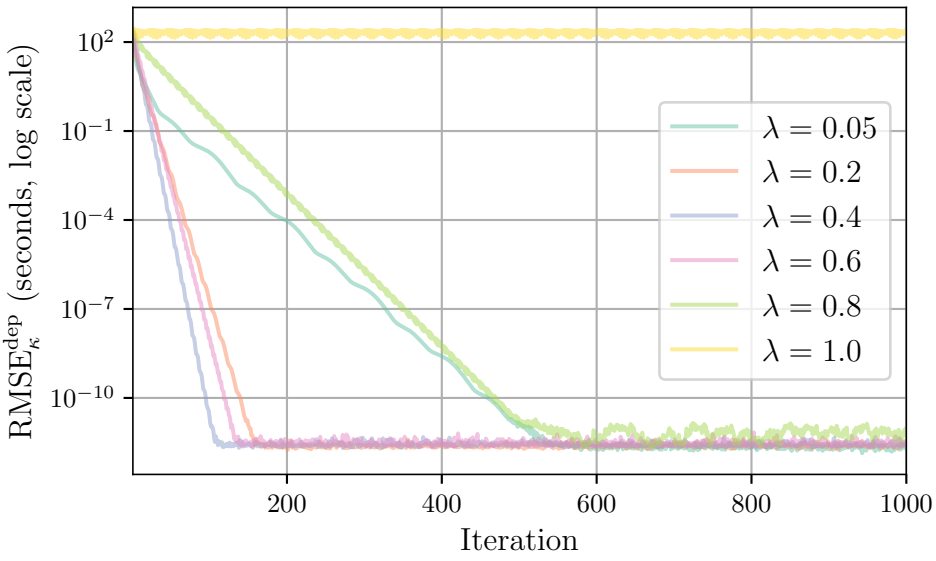

Figure 19: Impact of the smoothing factor on $\text{RMSE}^{\text{dep}}_{\kappa}$ convergence

Impact of the number of agents We run a simulation for 200 iterations with 3 different values of the number of agents N: 1000, 10 000 and 100 000.[19] Note that, since the bottleneck capacity s is proportional to N, the equilibrium travel-time function from the analytical results is not impacted by a variation in N and the equilibrium rate of departures is proportional to N. The results are shown in Table 11. First, we observe that the running time of the simulation is roughly linear in the number of agents. Then, we observe that the simulation results are closer to the analytical results as the number of agents increases, as shown by the distance to the analytical results D. However, the convergence of $\text{RMSE}_\kappa^{\text{dep}}$ is roughly the same for all values of N, as shown on Figure 20. We conclude that, for smaller values of N, the discretization of the agents has a larger impact which explains why the comparison to the analytical results gets worse, but this discretization does not prevent the simulations from converging.

Table 11: Sensitivity of the results to the number of agents

N	1000	10 000	100 000
Running time	1.70s	16.53s	3m 28s
Expected max. utility	7.147	7.176	7.188
Average travel time	2m 13s	2m 1s	1m 57s
Dist. analytical res. D	3.37 %	0.83 %	0.19 %
$\text{RMSE}_{200}^{\text{dep}}$	2×10^{-12}	3×10^{-12}	3×10^{-12}
RMSE_{200}^{T}	8×10^{-13}	3×10^{-12}	3×10^{-12}

Impact of the travel-time function breakpoints interval The parameter δ represents the time interval between two breakpoints in the travel-time function. We run a simulation for 200 iterations with 4 different values of δ: 1 min, 5 min, 10 min and 15 min. Note that, as $\delta \to 0$, the travel-time function converges to a continuous function. The results are shown in Table 12 and Figure 21. We observe that the running time slightly decreases as δ increases. The simulations converges to an equilibrium for all values of δ, convergence is faster for larger values of δ. The convergence of the simulation is satisfactory even with $\delta = 15$ min, i.e., with only 5 breakpoints in total (1 each 15 minutes). Still, the distance to the analytical results D is smaller for smaller values of δ because agents can better predict the travel time that they will face for any departure time.

Figure 22 shows the simulated travel-time function at the last iteration for the different values of δ. With larger values of δ, the simulated travel-time function cannot match well the

[19] We also run simulations with fewer agents but these simulations dependent heavily on the draws of the random numbers so we decided not to report their results.

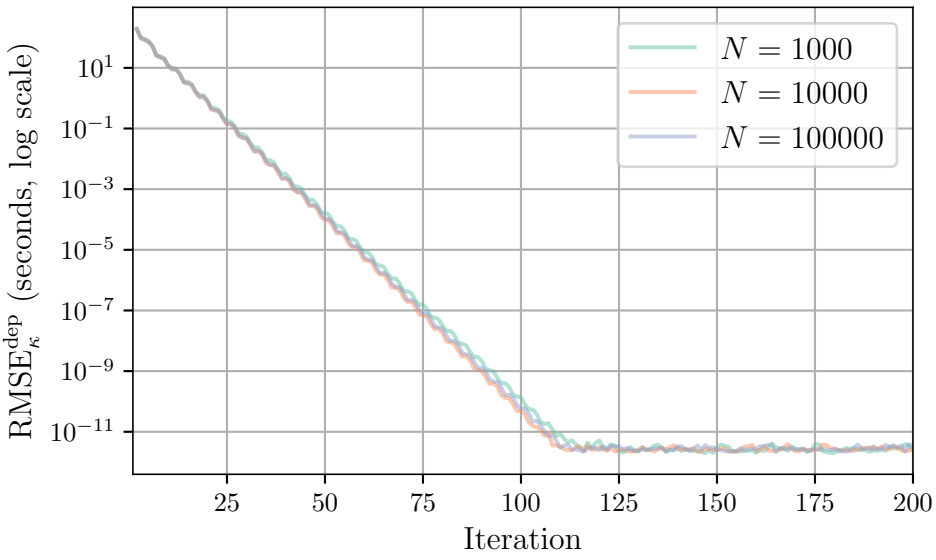

Figure 20: Impact of the number of agents on $\mathrm{RMSE}_{\kappa}^{\mathrm{dep}}$ convergence

Table 12: Sensitivity of the results to the breakpoint interval δ

δ	1 min	5 min	10 min	15 min
Running time	3m 24s	3m 34s	3m 16s	3m 11s
Expected max. utility	7.188	7.181	7.160	7.176
Average travel time	1m 57s	2m 2s	2m 14s	2m 16s
Dist. analytical res. D	0.19 %	1.14 %	2.39 %	3.08 %
$\mathrm{RMSE}_{200}^{\mathrm{dep}}$	3×10^{-12}	3×10^{-12}	2×10^{-12}	2×10^{-12}
RMSE_{200}^{T}	2×10^{-12}	2×10^{-12}	4×10^{-13}	3×10^{-13}

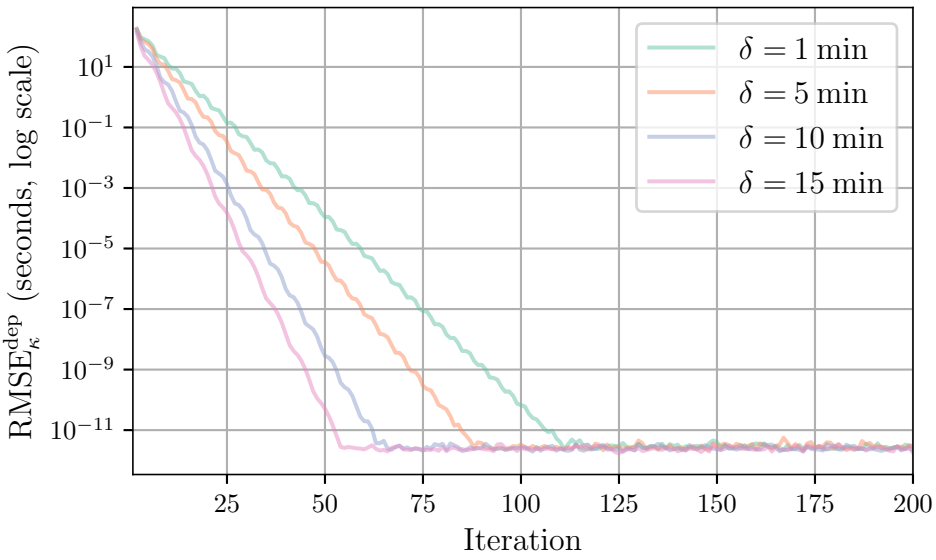

Figure 21: Impact of the breakpoint interval on $\mathrm{RMSE}_{\kappa}^{\mathrm{dep}}$ convergence

analytical results because there is not enough breakpoints. Note that using a larger value for δ is akin to assuming that agents have limited information on the travel-time function. This might give more realistic results than the analytical model which assumes perfect information on the travel times.

Impact of the scale of the utility's random component The parameter μ represents the scale of the utility's random component. Figure 23 shows how the probability distribution of the departure-time choice varies with μ. As $\mu \to 0$, the probability distribution converges to a distribution with a probability 1 to choose the best departure time (i.e., $t^* - t^{\mathrm{f}}$ under free-flow conditions) and the model converges to the deterministic bottleneck model from Arnott et al. (1990). As $\mu \to +\infty$, the probability distribution converges to a uniform distribution.

We run 1000 iterations of METROPOLIS2 with 5 different values of μ.[20] The main results are shown in Table 13 and Figure 24. From the figure, observe that as μ decreases, convergence is slower. With $\mu = 0.2$, the simulation does not converge ($\mathrm{RMSE}_{1000}^{\mathrm{dep}} = 30$, $\mathrm{RMSE}_{1000}^{T} = 100$) and the distance to the analytical results is large ($D = 20.10\,\%$). The reason for that might be that congestion is larger for smaller μ, as shown by the average travel time which ranges from 30s for $\mu = 5$ to 6m 7s for $\mu = 0.2$. When congestion is larger,

[20]For each value of μ, we choose the value of the smoothing factor λ which gives the best convergence. The values used are: $\lambda = 0.01$ for $\mu = 0.2$, $\lambda = 0.1$ for $\mu = 0.5$, $\lambda = 0.4$ for $\mu = 1.0$, $\lambda = 0.8$ for $\mu = 2.0$ and $\lambda = 1.0$ for $\mu = 5.0$. Observe that for smaller values of μ, the convergence is better with a smaller value of λ.

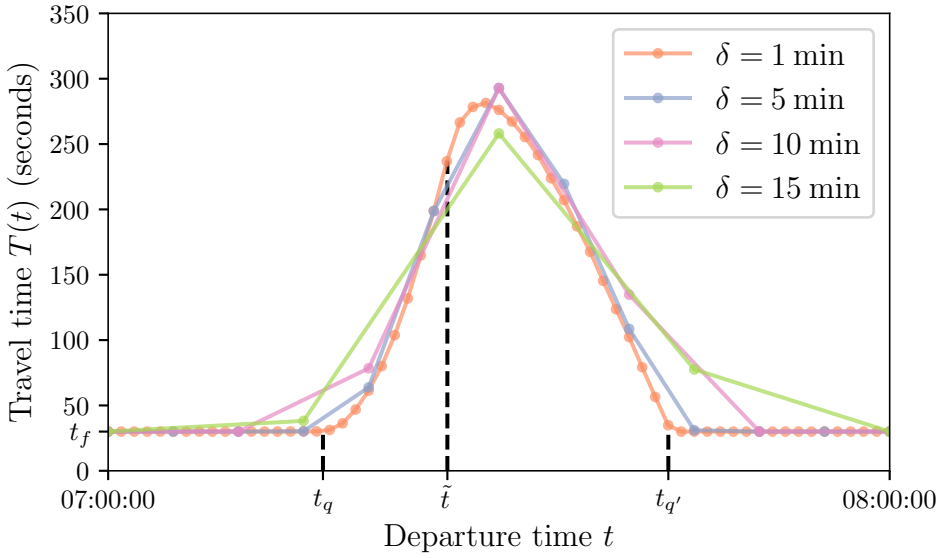

Figure 22: Travel-time function for different values of the breakpoint interval
Note: t_q, $\tilde{t}$, $t_{q'}$ are defined as in Figure 12.

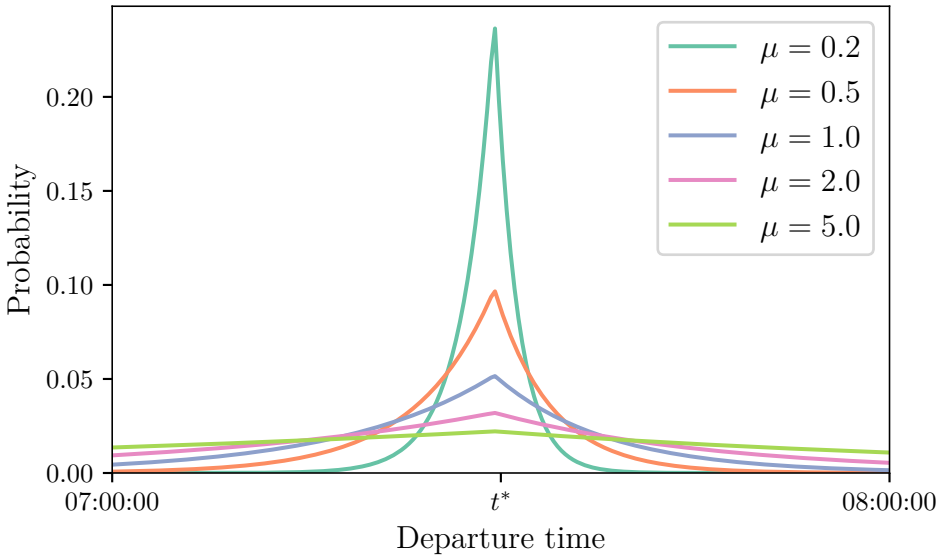

Figure 23: Probability distribution of the departure-time choice for different values of the scale of the utility's random component (assuming a constant travel time)

this means that there are more interactions between agents which might impede convergence. For the simulations which converge ($\mu \geq 0.5$), the distance to the analytical results is similar (from 0.15 % to 0.34 %). The running times are also very similar for all values of μ.

Table 13: Sensitivity of the results to the scale of the utility's random component

μ	0.2	0.5	1	2	5
Running time	17m 40s	17m 48s	17m 32s	17m 14s	17m 1s
Average travel time	6m 7s	3m 30s	1m 57s	44s	30s
Dist. analytical res. D	20.10 %	0.15 %	0.19 %	0.30 %	0.34 %
$\text{RMSE}^{\text{dep}}_{1000}$ (seconds)	3×10^{1}	3×10^{-11}	3×10^{-12}	2×10^{-12}	3×10^{-12}
RMSE^{T}_{1000} (seconds)	1×10^{2}	3×10^{-11}	1×10^{-12}	2×10^{-12}	1×10^{-13}

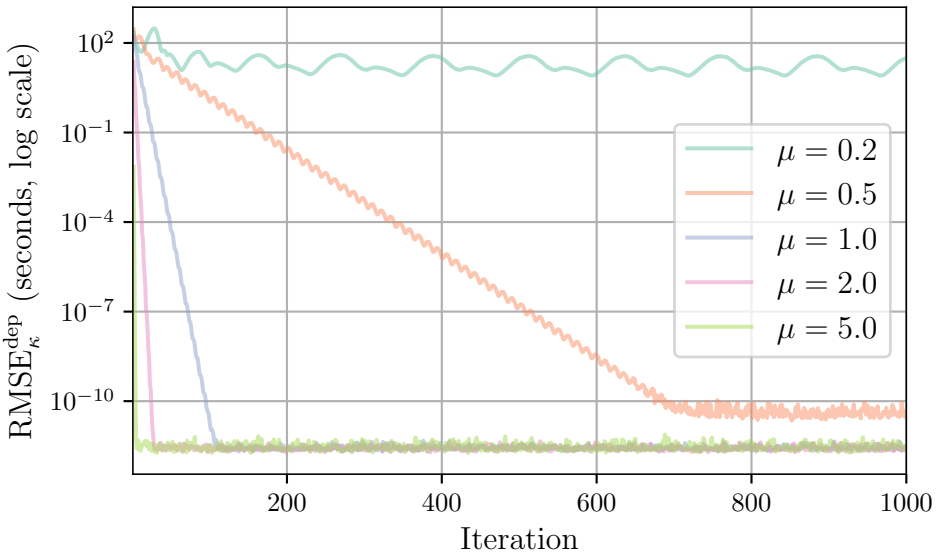

Figure 24: Impact of the scale of the utility's random component on $\text{RMSE}^{\text{dep}}_{\kappa}$ convergence

A.4 Considering Heterogeneity

So far, we have considered a population of homogeneous agents. The only difference between agents, and the reason their departure times differed, was the different values of u_n, which we interpret as unobserved heterogeneity. In this section, we include some observed heterogeneity to the model by assuming that the t^* values are distributed. More specifically, we assume that the desired arrival times of the agents, $\{t^*_n\}_{1 \leq n \leq N}$, are independent and identically distributed with

$$t^*_n \sim \mathcal{N}(\bar{t}^*, \sigma_{t^*}),$$

where $\overline{t}^* = 7{:}30$ a.m. is the center of the desired-arrival-time distribution and $\sigma_{t^*} = 5\,\text{min}$ is the variance of the distribution. The distribution of the t_n^* values drawn is presented on Figure 25.

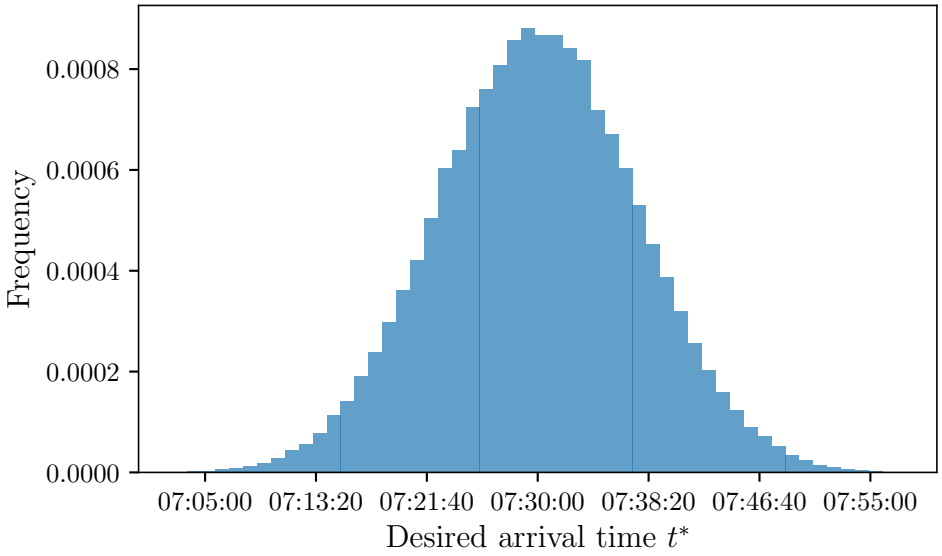

Figure 25: Distribution of the t_n^* values for the simulations with observed heterogeneity

We run two different simulations, with $\mu = 1$ and $\mu = 0.6$. The main results are shown in Table 14. Observe that the running time of METROPOLIS2 is similar between the three scenarios. This is because agents are always simulated as separate entities so there is no speed gains when simulating homogeneous agents.

In the heterogeneous case, the desired arrival times are spread over the simulated period so the distribution of departure times is more spread than in the homogeneous case, as shown on Figure 26. This explains why, with $\mu = 1$, the average travel time is smaller in the heterogeneous case (1m 8s) than in the homogeneous case (1m 57s). In the heterogeneous case, when μ decreases from 1 to 0.6, the average travel time increases from 1m 8s to 2m 1s (consistently with the results from the sensitivity analysis to μ, in Section A.3). Then the homogeneous simulation with $\mu = 1$ and the heterogeneous simulation with $\mu = 0.6$ result in a similar travel time. Finally, in the heterogeneous simulations, $\text{RMSE}_{200}^{\text{dep}}$ and RMSE_{200}^{T} are larger than in the homogeneous simulation but the values are still practically zero.

Table 14: Comparison between the homogeneous and heterogeneous bottleneck models

Model Parameters	Homogeneous $\mu = 1$, $\sigma_{t^*} = 0$	Heterogeneous A $\mu = 1$, $\sigma_{t^*} = 5\,\text{min}$	Heterogeneous B $\mu = 0.6$, $\sigma_{t^*} = 5\,\text{min}$
Running time	3m 24s	3m 25s	3m 27s
Average travel time	1m 57s	1m 8s	2m 1s
$\text{RMSE}^{\text{dep}}_{200}$ (seconds)	3×10^{-12}	7×10^{-6}	6×10^{-4}
RMSE^{T}_{200} (seconds)	2×10^{-12}	2×10^{-5}	4×10^{-4}

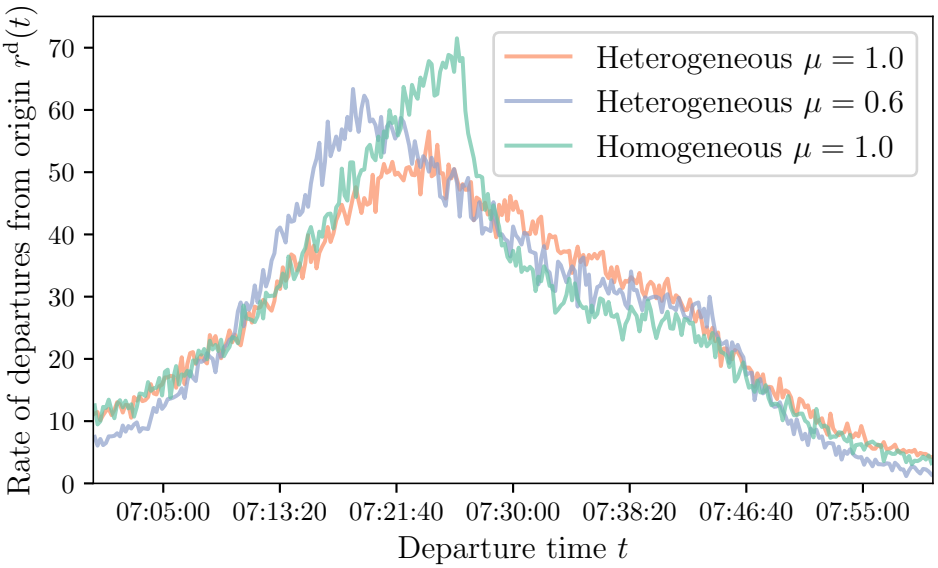

Figure 26: Comparison of the rate of departures between homogeneous and heterogeneous simulations

B Input Data

The input data of METROPOLIS2 is composed of the following categories:

- The *road network*, which defines the infrastructure through which the road trips are performed (see Section B.1).
- The *agents*, which are the entities whose travel behaviors are simulated (see Section B.2).
- The *parameters*, which govern various technical aspects of the simulation, such as stopping criteria (see Section B.3).
- The *initial network conditions*, which are the expected conditions used in the first iteration of the simulator (see Section D). When omitted the free-flow conditions are used.

B.1 Road-Network Input

In METROPOLIS2, a road network is defined by a set of edges and a set of vehicle types.

Each edge is characterized by its fundamental attributes, including its source node, target node, base speed, length, number of lanes, bottleneck capacity and speed-density function parameters.

Each vehicle type is defined by the following characteristics:

- headway: typical distance between two vehicles, measured from tip-to-tip;
- passenger car equivalent: a measure of the congestion impact of the vehicle type, in comparison to a standard passenger car;
- speed restrictions: speed limitations that apply to this vehicle type.

Additionally, it is possible to specifies road restrictions, i.e., edges which are forbidden to some vehicle types.

B.2 Agents

METROPOLIS2 can simulate an arbitrary number of agents. Each agent chooses between various *travel alternatives*, where a travel alternative can represent a single trip (with given mode, origin and destination), a chain of trips, or no trip (e.g., work-at-home). Figure 27 represents the structure of the data for a single agent. An agent is defined by the travel

alternatives from which they can choose and the choice model defining how the chosen travel alternative is determined (e.g., Multinomial Logit model, deterministic model).

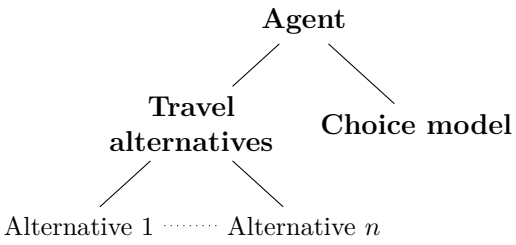

Figure 27: Structure of an agent

A travel alternative can be either a constant utility amount (e.g., to represent the utility of work-at-home) or the description of a trip chain. The structure of trip chains is represented in Figure 28. A single trip can be represented as a special case of a trip chain.

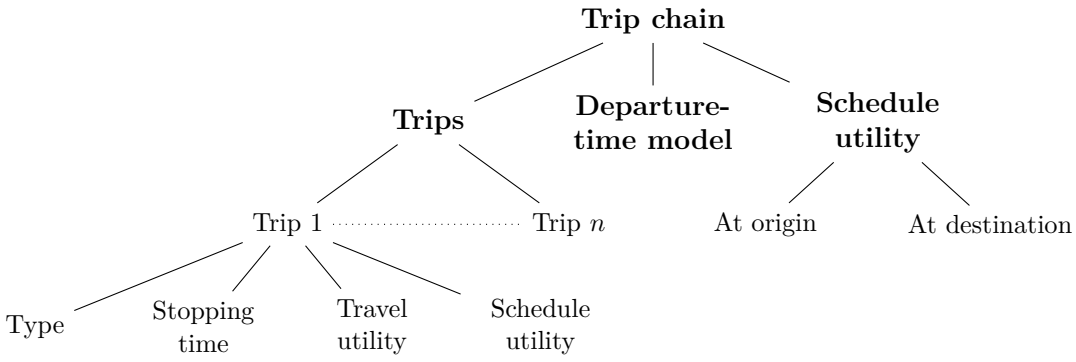

Figure 28: Structure of a trip chain

A trip chain is constructed from an arbitrary number of trips, each of which can belong to one of two distinct types, defined below.

- *Road trip:* a trip defined by an origin and destination node on the road network with a specific vehicle type.
- *Virtual trip:* a trip without any interaction with the network infrastructure (e.g., walk trip, which neither impacts nor is impacted by congestion), defined by either a constant travel time or a travel-time function.

These types can be combined within the same trip chain to represent inter-modal journeys.

Apart from the trip type and its associated data, each trip is characterized by the following elements.

- Stopping time: waiting time at the end of the trip before the start of the next trip, or, equivalently, duration of the activity performed at the trip's destination.

- Travel utility: utility as a function of the trip's travel time.
- Schedule utility: utility as a function of the arrival time.

The sequence followed by an agent n with a chain of multiple trips works as follows. Let $t^{\Delta}_{n,k}$ denote the stopping time at the end of the k-th trip of agent n. The agent's departure time, t^{d}_n, coincides with the start of their first trip, $t^{\mathrm{d}}_{n,1}$. Subsequently, the agent reaches the destination of their first trip upon the completion of the trip's travel time, $\mathrm{tt}_{n,1}$, resulting in their arrival time at the destination of the first trip as $t^{\mathrm{a}}_{n,1} \equiv t^{\mathrm{d}}_{n,1} + \mathrm{tt}_{n,1}$. The second trip starts after the end of the first trip's stopping time, $t^{\Delta}_{n,1}$, occurring at time $t^{\mathrm{d}}_{n,2} \equiv t^{\mathrm{a}}_{n,1} + t^{\Delta}_{n,1}$. This alternation between trips and stopping times continues, until the agent arrives, upon the conclusion of the stopping time of their last trip, at time t^{a}_n. Figure 29 illustrate the timing dynamics of a two-trip chain.

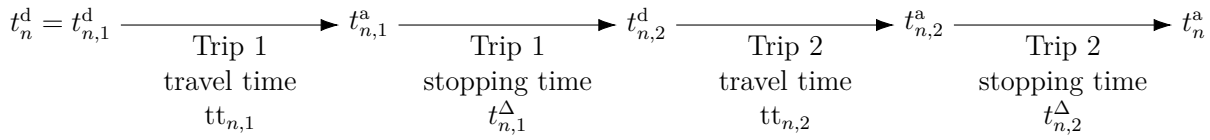

Figure 29: Timing dynamics of a trip chain with two trips

Finally, a trip chain is characterized by a *departure-time model* indicating how the departure time from origin is computed (e.g., Continuous Logit, Multinomial Logit), and a *schedule utility* at origin and at destination (a function of t^{d}_n and t^{a}_n).

B.3 Parameters

The parameters of METROPOLIS2 include:

- the time window within which the trips take place,
- the learning model used to update the expected network conditions at each iteration,
- stopping criteria defining when the simulation should stop.

C Output Data

Output data for METROPOLIS2 can be divided in three categories: agent-specific output (Section C.1), output specific to the road network (Section C.2) and aggregate results (Section C.3).

C.1 Agent Output

For each simulated agent, the output of METROPOLIS2 includes

- the travel alternative chosen;
- the expected utility (pre-trip) and the actual utility (post-trip);
- the departure time, arrival time and travel time of all the trips taken;
- the route taken for all the trips (if applicable), with the detailed timings.

These values correspond to the choices and actions taken for the last simulated iteration.

C.2 Road-Network Output

METROPOLIS2 returns the current value of the simulated and expected network conditions at the end of the simulation. These network conditions contain, for each edge of the road network, the expected travel time as a function of time of day. The simulated network conditions represent the simulated travel time in the supply model, for the last iteration. The expected network conditions represent an average of the simulated travel time over the past iterations, as per the learning model of the simulation.

These values can be analyzed to identify the main congested edges of the network. They can also be used to compute the time-dependent travel time from any origin to any destination of the network.

C.3 Aggregated Results

The aggregated results of METROPOLIS2 include:

- aggregate results specific to the agents (e.g., mean departure time, arrival time, travel time, expected utility, actual utility);
- aggregate results specific to road trips (e.g., mean travel time, mean route length, mean uncongested travel time);
- aggregate results related to convergence (e.g., departure-time and route variation compared to the previous iteration, difference between actual and expected travel times).

These values are returned at the end of each iteration of the simulation.

D Network Conditions

Network conditions are a critical component of METROPOLIS2. They represent the perceptions of agents on the travel time that they will face during their trips. These network conditions are common knowledge, i.e., all agents have the same perceptions. The network conditions are updated at each iteration of the simulator based on the simulated travel times on the network in the supply model.

The road-network conditions are composed of a travel-time function for each edge of the road network. These travel-time functions represent the expected time required to traverse the edge for any time of the day. They are used in the routing algorithm during the demand model to compute the fastest route from origin to destination.

In METROPOLIS2, the travel-time functions are represented as piecewise linear functions so that they can simply be stored as a list of breakpoints. If the simulation contains multiple vehicle types, one travel-time function is saved for each vehicle type because these vehicle types can experience different speed limits.

In the supply model, the occupied length and bottleneck queue are recorded for each edge at any time. These values allows to compute the simulated travel time for any vehicle type and any edge, i.e., the simulated road-network conditions. Then, the road-network conditions are used in the learning model to update the expected conditions for the next iteration.

E Description of the Models

E.1 Demand Model

The demand model simulates the travel decisions of all the agents, given the expected network conditions (which are common knowledge). The travel decisions consist of choosing one of the available travel alternatives and, if applicable, a departure time or route. These choices are based on utility maximization. The network conditions are used to compute the expected travel time, for road trips.

Backward induction is used to select the travel alternative, departure time and route in this order. The demand model thus works as follows for each agent:

1. The selected route is computed for each travel alternative and each possible departure time.

2. Base on the routes chosen in Step 1, the departure time maximizing utility and the expected value of the departure-time choice are computed for each travel alternative.

3. A travel alternative is chosen based on its expected value, which is given by the departure-time choice from Step 2.

The remainder of this section is organized as follows. We provide an explanation of how utility is defined in METROPOLIS2. Then, the route, departure time and travel alternative choices are presented, in this order.

Utility The utility of trip k of agent n is defined by

$$V^{\text{trip}}_{n,k}(t^{\text{d}}_{n,k}, t^{\text{a}}_{n,k}) = V^{\text{so}}_{n,k}(t^{\text{d}}_{n,k}) + V^{\text{tt}}_{n,k}(t^{\text{a}}_{n,k} - t^{\text{d}}_{n,k}) + V^{\text{sd}}_{n,k}(t^{\text{a}}_{n,k}),$$

where $t^{\text{d}}_{n,k}$ is the trip's departure time, $t^{\text{a}}_{n,k}$ is the trip's arrival time, $V^{\text{so}}_{n,k}$ is the schedule utility at origin, $V^{\text{sd}}_{n,k}$ is the schedule utility at destination and $V^{\text{tt}}_{n,k}$ is the travel-time utility. In the current version of METROPOLIS2, $V^{\text{so}}_{n,k}$ is limited to a linear schedule-delay penalty:

$$V^{\text{so}}_{n,k}(t^{\text{d}}_{n,k}) = \beta_{n,k} \cdot [t^{*}_{n,k} - \Delta_{n,k} - t^{\text{d}}_{n,k}]_{+} + \gamma_{n,k} \cdot [t^{\text{d}}_{n,k} - t^{*}_{n,k} - \Delta_{n,k}]_{+},$$

with $\beta_{n,k}$ (resp. $\gamma_{n,k}$) the penalty for early (resp. late) departures, $t^{*}_{n,k}$ the desired departure time, $\Delta_{n,k}$ the length of the desired departure time period and $[x]_{+} = \max(x, 0)$. The same specification can be used for $V^{\text{sd}}_{n,k}$, with (desired) arrival time instead of (desired) departure time. Any other parametric specifications could be implemented easily. The travel-time utility $V^{\text{tt}}_{n,k}$ is limited to a polynomial function of travel time, with up to degree 4, including a constant. Again, any other parametric specification could be implemented easily. To evaluate numerically the utility function, METROPOLIS2 relies on linear approximation.

Route choice In the case of road trips,[21] given a travel alternative and a departure time, agents choose the route that minimizes travel time.[22] These fastest routes are computed using a time-dependent routing algorithm directly in the demand model (there is no en-route choice in METROPOLIS2). They are computed using the expected network conditions which give the time-dependent travel time for each edge of the road network. The relevant information for departure-time choice are the travel time of these fastest routes, for all possible departure times of each travel alternative.

[21] There is no route to choose in the case of virtual trips as travel time is exogenous.

[22] The assumption that agents always take the fastest route allows the use of very efficient routing algorithm. In some special cases, however, the fastest route is not actually the one which maximizes utility. For example, in the α-β-γ model, agents can increase their utility by taking a longer route whenever $\beta > \alpha$, i.e., the value of travel-time savings is smaller than the penalty for being early at destination. In such rare cases, the travel decisions returned by the demand model do not strictly adhere to the utility-maximization principle.

Departure-time choice Departure-time choice is continuous for trip chains composed of road or virtual trips because utility is a continuous function of time. In the case of multiple trips, only the departure time for the first trip is chosen. The departure times for the subsequent trips are then derived based on the departure time of this first trip, along with the corresponding travel times and stopping times for each trip within the chain.

The way the departure time is selected depends on the departure-time choice model of the agent given as input. For example, when the departure-time choice model is a Continuous Logit model, the probability to choose departure time t is given by:

$$p^{\mathrm{d}}_{n,j}(t) = \frac{e^{V_{n,j}(t)}}{\int_{t^0}^{t^1} e^{V_{n,j}(\tau)}\,\mathrm{d}\,\tau},$$

where $V_{n,j}(t)$ is the utility as a function of departure time for travel alternative j of agent n and t^0 (resp. t^1) is the earliest (resp. latest) possible departure time. Inverse transform sampling is used to draw the selected departure time from the probability distribution given above. Then, the expected maximum utility of the travel alternative is given by the logsum formula:

$$V_{n,j} = \ln \int_{t^0}^{t^1} e^{V_{n,j}(\tau)}\,\mathrm{d}\,\tau.$$

Consider the case of an agent n with a chain of K trips, $1, \ldots, k, \ldots, K$. From route choice, METROPOLIS2 gets the expected travel-time function $T_{n,k}$ from origin to destination for each trip k of agent n. Then, for any departure time t^{d} from the first origin, the departure times and arrival times of any trip k can be computed recursively from these expected travel-time functions $\{T_{n,k}\}_{1\le k\le K}$ and the trips' stopping times $\{t^{\Delta}_{n,k}\}_{1\le k\le K}$. For example, the arrival time at destination of trip 1, is $t^{\mathrm{a}}_{n,1} \equiv t^{\mathrm{d}}_{n} + T_{n,1}(t^{\mathrm{d}}_{n})$ the departure time from origin of trip 2 is $t^{\mathrm{d}}_{n,2} \equiv t^{\mathrm{a}}_{n,1} + t^{\Delta}_{n,1}$, and the arrival time at destination of trip 2 is $t^{\mathrm{a}}_{n,2} \equiv t^{\mathrm{d}}_{n,2} + T_{n,2}(t^{\mathrm{d}}_{n,2})$. Therefore, from these travel-time functions and the stopping times, it is possible to compute the utility of the trip chain for any departure time from the first origin.

Choice of a travel alternative The choice between the travel alternatives available to an agent is based on the *choice model* of the agent. The choice model specifies how the travel alternative is selected given the *expected maximum utility* of all the travel alternatives, determined by their departure-time choice. For example, with a deterministic choice model, the travel alternative with the largest expected utility is chosen.

With a Multinomial Logit choice model, the probability that agent n chooses travel

alternative j is given by the Logit formula:

$$p_{n,j} = \frac{e^{V_{n,j}}}{\sum_{j'} e^{V_{n,j'}}},$$

where $V_{n,j}$ is the expected maximum utility of travel alternative j of agent n. Like with the departure-time choice model, the travel alternative can be drawn from such a probability distribution using inverse transform sampling. The expected maximum utility, or surplus, that the agent can get from their available travel alternatives is computed as the logsum formula:

$$V_n = \ln \sum_j e^{V_{n,j}}.$$

E.2 Supply Model

The supply model of METROPOLIS2 uses an event-based model to simulate the movements of agents and vehicles on the network, based on the travel decisions given by the demand model. An event is defined as an action taken by an agent or a vehicle (e.g., a car reaches an intersection, an agent reaches the destination of a virtual trip), together with the time at which the action happens (its *execution time*). The events are stored in a priority queue, based on their execution time, where the earliest events to be executed are on top of the queue. The supply models work as follows:

- When the supply model starts, the event queue is loaded with the initial event of the agents' trips and public-transit trips, which corresponds to the departure from origin.
- The supply model then proceeds in executing all events in chronological order.
- Events can have an impact on the network-infrastructure state when being executed. For example, when a car enters the next edge on its route, vehicle density is reduced on the previous edge and increased on the new one.
- Events can push new events to the priority queue while being executed. For example, when a car enters a edge, an event is created to be executed at the time the car will reach the end of the edge (where the travel time on the edge depends on the current density on the edge).
- The supply model stops when there is no more event to execute in the priority queue.

While the events are being executed, the simulated network conditions are recorded to be used in the learning model.