TBI: Transient Hierarchical Modeling of Large-Scale Vehicle Sharing Systems

Matthew Sheldon, *Student Member, IEEE,* Daphne Tuncer, Giuliano Casale, *Member, IEEE*

***Abstract*—Due to rush-hour effects, transient analysis may be necessary to model and optimize vehicle sharing systems. However, existing methods for transient modeling face computational challenges at scale. To solve this, we present trajectory-based iteration (TBI), a method for the efficient decomposed modeling of vehicle-sharing systems, and we extend it to a scalable method for non-convex, transient optimization problems. The proposed TBI method works by iteratively solving spatial submodels, then periodically updating the vehicle flows between submodels. We then present experimental results where our method results in a 37% improvement in runtime for solving fluid models for mobility systems. Then, we present a case study in which our optimization method results in an estimated 22% improvement in the objective function for a vehicle-sharing system with offline rebalancing.**

***Index Terms*—Vehicle Sharing, Mobility-on-Demand, Fluid Queues**

I. Introduction

More than half of the world's population lives in urban areas. The trend of urbanization is expected to persist, as the World Bank estimates that this ratio will increase to 7 out of every 10 people by 2050 [1]. The concentration of population in densely-built urban environments raises major challenges with respect to the management of urban infrastructure and functionality. Emerging frameworks for urban development, such as the 15-minute city concept [2], [3], aim to revisit the planning and operations of cities in the light of sustainability and resilience objectives. Transportation and mobility services will play a key role in the future organization of the urban space by guaranteeing the accessibility of people and goods to core city functions. To support these upcoming developments, modeling and simulation can help to improve the efficiency of transportation systems.

Vehicle sharing systems (VSS) are regarded as promising contributors to the mobility of the future. VSS enable the implementation of mobility services that provide short-term rental of vehicles, including cars, bikes, scooters, or small trucks. They facilitate access to vehicle-sharing by supporting different types of service delivery modes (return or one-way), different modes of fleet administration (station-based or free-floating), and different pricing strategies (fixed or dynamic) [4]. VSS have many advantages over traditional mobility systems that intersect with sustainability and development goals. They are capable of filling gaps in existing public transportation systems [5], [6], reducing carbon emissions [5], improving congestion in road networks [7], and reducing personal vehicle ownership and usage [5].

A key motivation for the use of simulation and modeling in VSS is to help operators mitigate demand imbalances between stations, which provide significant operational challenges for one-way vehicle sharing systems [8]. Queueing-theoretic methods have been used extensively to analyze such systems, with the goal of improving operational efficiency [9]–[11]. These may be used to analyze both manual rebalancing by adding in rebalancing trips under control of the system operator [7], [12], and incentive-based rebalancing by varying the arrival rate according to the price [8].

Limitations of existing models include an assumption of steady-state, intractability at scale, and effective modeling of station capacities. Steady-state assumptions may not apply to all mobility systems. Significant changes in demand magnitude or direction due to rush hour effects may prevent the system from reaching stability. Time-inhomogeneous or transient queueing methods may scale poorly, and rarely admit analytical solutions [13]. Part of the motivation of this paper is to provide a method for solving such problems at scale.

System scale is a significant challenge for modeling mobility systems. Solving the initial value problem has a quadratic worst-case time complexity with respect to the number of equations, which implies that it may scale poorly when applied naively to real-world models. This can be improved by using fluid approximation techniques, such as mean-field analysis [14], but it may prove to be quite slow when the number of stations is large (e.g., greater than 100). In addition, the amount of time needed to model the system will vary with the time simulated, which provides challenges for simulating systems for periods longer than a few hours.

Finite-capacity queueing may be necessary to accurately model the operation of mobility systems. A bike rack may have a fixed number of docks, or an electric vehicle station may have a limited number of chargers. This may have a significant impact on the dynamics of the system. Networks of finite-capacity queues present several modeling challenges when compared to networks of queues with infinite-capacity buffers. Product-form models require certain modeling assumptions that are restrictive in practice for their use in blocking models [15].

This paper aims to extend previous analytical work on time-varying and transient systems, with a focus on scalability and extensibility. In order to accomplish this, we propose a method for the efficient parallelization of the initial-value problem (IVP) for vehicle-sharing systems, and then extend it to an optimization problem using simulated annealing. Then, we

Matthew Sheldon and Giuliano Casale are affiliated with Imperial College London

Daphne Tuncer is affiliated with Ecole des Ponts ParisTech

For correspondence: matthew.sheldon20@imperial.ac.uk

present an experiment and case-study showing the accuracy and applicability of these methods.

The rest of the paper is organized as follows. In Section II, we present the background and motivation for this method. In Section III, we describe the queueing aspects of this system. In Section IV, we present trajectory-based iteration, the proposed method for efficient modeling. In Section V, we extend this method to an optimization problem aimed at improving the performance of a vehicle sharing system. Finally, Section VI presents empirical results for both the proposed TBI method and the optimization problem.

II. Background and Motivation

Most commonly used model reduction methods, such as Krylov subspace methods or Singular Value Decomposition (SVD), rely on the system being linear. While the Kolmogorov forward equation describes a system of linear ordinary differential equations (ODEs), the mean-field analysis of closed queueing networks typically features non-linear equations. As the number of equations required for mean-field analysis is already significantly fewer than those from the Kolmogorov equation, it is unlikely that SVD-based methods will be significantly more attractive. Also, principal component analysis depends on finding the eigenvectors, which may have greater time complexity than the IVP.

The leading method for model reduction in nonlinear systems is Proper Orthogonal Decomposition (POD). This requires a snapshot matrix of the model state at various times, which may require existing results from a simulation or another model [16], [17]. Furthermore, the reduced system may not share the dynamics of the full system when a decision variable is changed. Therefore, in order to efficiently analyze the transient behavior of mobility systems, we turn to spatial submodeling. Our method may also complement rather than replace these methods. In other words, these submodels may be further reduced through POD or differential equivalences.

Forward or backward differential equivalences (FDE and BDE, respectively) can be used to simplify fluid models [18]. For vehicle sharing systems, equivalences may be hard to find. When examining models of fluid equations describing randomized mobility networks with $10 - 15$ stations, we observe a 2.12% reduction in the number of equations when using FDE, and no observed backward differential equivalences. However, this reduction algorithm, which is co-NP hard [18], takes 49% longer than numerically solving the full model, which takes 3.8 seconds. This implies that differential equivalence methods cannot alone address scalability issues in mobility models.

The use of queueing models in vehicle sharing systems is well-established [7], [8], [10], [19], and a subset of these have used fluid approximation techniques [9], [10], [14], [20]. The work in [20] describes a fluid model that allows one to analyze the system as a whole with finite-capacities and time-inhomogeneous demands, but does not allow one to differentiate between individual stations or analyze the spatial aspects of the model. In [10], Waserhole *et al.* introduce a continuous quadratic program for optimizing prices in a bikesharing system, but there is a different queueing model in which the throughput is independent of the amount of fluid at each station. In [9], the service depends on whether or not the queue is empty, rather than a function of the queue length ($\min$ or otherwise). Both approaches may prove more inaccurate than fluid models based on mean-field approximation when there is a high proportion of queues with low mean populations. In addition, preservation of mean-field limits [21] may be important. Wollenstein-Betech, *et al.* propose a method where total service rate into a queue is constrained to equal the total rate leaving, while Waserhole, *et al.*, include a very similar method as a heuristic. In both methods, this constraint on service rates is combined with a constraint on the fleet size to ensure that all queues are non-empty. The model in [14] uses mean-field analysis to limit the number of problematic stations in the context of finite-capacity queueing systems, but requires identical service rates and demands, and the fluid limit preservation is only shown for a single station. The framework we present aims to be more directly applicable and efficient than existing models, resorting to numerical analysis to deal with reduced tractability.

III. Vehicle Sharing System Model

We model the vehicle sharing system with vehicle queueing stations and transit delays. Each vehicle queueing station represents a location at which vehicles are parked when they are not in use. This may be a physical location such as a bike rack or an electric vehicle charging station, or it may also be an abstraction of an area at which customers access mobility services. Queueing models at which the queueing stations do not represent a physical location are common in the literature on transportation systems, particularly in mobility-on-demand and ridesharing applications [7], [11].

The length of time each vehicle spends in a queueing station is distributed according to the outbound demand for trips at that station, which is assumed to follow a Poisson process [7], [10], [20], [22]. Transit delays, in which vehicles wait for a stochastic period of time without queueing, represent the transit time incurred by vehicles in motion between one station and another. The service distribution for the transit delays may be fitted from the observed distribution of travel times between each station. These may represent individual roads, or be a larger-scale abstraction of the time it takes to transit between regions. It is important to note that transit delays assume that vehicles in the network do not cause congestion among each other, but congestion from other factors outside the system may be reflected in the service time distribution.

We approximate the time-varying dynamics of these systems by dividing the timeframe of the model into different stages, $e \in 1 \dots E$. In this paper, we focus on sequential and deterministic changes from one stage to the next. Therefore, we will represent each stage by its ending time T_e. Finally, $T_E = T$, where T is the ending time of the model. For convenience, we also use the term $T_0 = 0$. Each transition rate is held constant within stage e, but is allowed to change between stages. Note that these methods can be extended to Markov chain-modulated random environments [23].

Each individual vehicle queueing station is represented as a $-/M/1/k$ or $-/M/1$ queue, where the capacity k is either

a finite integer or is infinite. At these stations, vehicles wait for customers to arrive according to a Poisson process. Once a customer arrives, the vehicle exits the queue. In networks of finite capacity queues, a strategy must be chosen for how entities act when they arrive to a queue at capacity. In this case, we assume that a vehicle leaves and travels to another local station. Jobs in the queueing network correspond to vehicles in the system. Each ordered pair of stations is connected using a $-/Cox/\infty$ delay, which captures the transit time from the first station to the second. The vehicle then exits the delay once the service process has finished. This is appropriate for modeling transit between different stations with arbitrarily-disributed travel times, as long as vehicles within the system do not cause congestion with each other. They may be influenced by congestion from vehicles outside the system, which is reflected in a suitably-fitted service distribution. The service rate in each transit delay is Cox-distributed. A Cox distribution can be represented as a series of phases $a \in 1 \dots K^e_{ij}$, where K^e_{ij} is the number of phases, (i, j) indexes the delay, and e represents the stage the model is in. Phase transition rates μ^e_{ija} represent the probability of transitioning from phase a to phase $a + 1$ within the delay indexed (i, j), while service probabilities ϕ^e_{ija} represent the probability of service upon departing phase a, at delay (i, j). All jobs begin in service phase 1. The advantage of using Cox distributions is that they can be used to efficiently model travel times with any squared coefficient of variation (SCV) [15]. Any acyclic phase-type (APH) distribution can be mapped to a Cox distribution [24]. This includes a number of commonly-used distributions, including the Erlang, hyper-Erlang, exponential, and hyper-exponential distributions.

As the service distribution is exponential at all queueing stations, we represent the demand at station i as its service rate λ^e_i. It is important to note that we use the term "demand" to represent the demand from users for trips from a given station, and not the queueing-theoretic service demand $\frac{e_i}{\mu_i}$. We use the notation λ^e_{ij} to represent the demand for users to transit from station i to station j, and it is equal to $p^e_{ij}\lambda^e_i$, where p^e_{ij} is the routing probability to station j after service at station i.

We use the notation $x_i(t)$ to represent the number of vehicles at station i with respect to time, and $y_{ija}(t)$ to represent the number of vehicles in transit between station i and station j, in phase a, at time t. In order to approximate the transient behavior of the queueing network, we use mean-field approximation, which is a deterministic approximation using ODEs [21], [23]. Using the mean-field approximation of a network of $-/M/1$ and $-/PH/\infty$ queues, we can derive the following system of ODEs, which describe the evolution of the system within the time interval $[T_{e-1}, T_e]$,

$$\dot{x}_i(t) = \sum_{j=1}^{M}\sum_{a=1}^{K^e_{ij}} \mu^e_{jia}\phi^e_{jia}y_{jia}(t) - \lambda^e_i \min(x_i(t), 1) \quad (1)$$

$$\dot{y}_{ij1}(t) = \lambda^e_{ij} \min(x_i(t), 1) - \mu^e_{ij1}y_{ij1}(t) \quad (2)$$

$$\dot{y}_{ija}(t) = \mu^e_{ij,a-1}(1 - \phi^e_{ij,a-1})y_{ij,a-1}(t) - \mu^e_{ija}y_{ija}(t) \quad (3)$$

If we interpret λ^e_i, μ^e_{ija}, and ϕ^e_{ija} as piecewise-constant functions of t, the system can be analyzed on the entire time interval $[0, T]$.

IV. Spatial Submodels for Large-Scale Systems

A. TBI Algorithm Formulation

Solving the initial-value problem has a quadratic time complexity with respect to the number of equations, which can be considered intractable at scale. Many rank-reduction algorithms are inapplicable due to nonlinearities caused by the $\min$ term in (1)-(2). Therefore, we decompose the model into spatial submodels, which we refer to as cells, to improve the runtime. Each cell features a fixed number of stations, and any transit delays between stations within the cell. Transit delays connecting different cells are termed "connecting delays", while those connecting stations within a cell are termed "interior delays".

In order to efficiently solve models such as (1)-(3), we propose trajectory-based iteration (TBI), a method for splitting the ordinary differential equations. In this process, we iteratively solve the IVP for each cell, and then update the trajectory of the connecting delays at each step. Therefore we include any interior delays as part of the model, and we use the prior connecting delays from the previous iteration. Let n be the iteration number. We then represent (1)-(3) as,

$$\begin{aligned}\dot{x}^n_i(t) = \quad & \sum_{j\notin C_k}^{M}\sum_{a=1}^{K^e_{ij}} \mu^e_{jia}\phi^e_{jia}y^{n-1}_{jia}(t) \\ & + \sum_{j\in C_k}^{M}\sum_{a=1}^{K^e_{ij}} \mu^e_{jia}\phi^e_{jia}y^{n}_{jia}(t) \\ & - \lambda^e_i \min(x^n_i(t), 1) \end{aligned} \quad (4)$$

$$\dot{y}^n_{ij1}(t) = \quad \lambda^e_{ij} \min(x^n(t), 1) - \mu^e_{ij1}y^n_{ij1}(t) \quad (5)$$

$$\begin{aligned}\dot{y}^n_{ija}(t) = \quad & \mu^e_{ij,a-1}(1 - \phi^e_{ij,a-1})y^n_{ij,a-1}(t) \\ & - \mu^e_{ija}y^n_{ija}(t) \end{aligned} \quad (6)$$

In this case, $x^n_i(t)$ represents the number of vehicles at station i at time t, during iteration n. It is also important to note that n is simply a variable superscript, and not an exponent. As before, e represents the current stage of the environment. An advantage of decomposition is that the solution of submodels can be efficiently parallelized, which further reduces the runtime. This technique is intended to work best when submodels are loosely coupled, or the number of jobs within connecting delays is expected to be low. A detailed pseudocode of this method is provided in Algorithm 1.

B. Convergence Analysis

Next, we present results showing that TBI provably converges as ε goes to 0. The full proofs have been omitted for the sake of brevity. In addition to the convergence result, we prove a few other useful properties of TBI. Firstly, we establish the unique existence of the solution. Secondly, it is shown that both $x^n_i(t)$ and $y^n_{ij,a}(t)$ are monotonically increasing over n. Thirdly, it can be shown that the total number of jobs in the system is bounded above by the original job population N.

Algorithm 1 Trajectory-Based Iteration

Decompose queueing network into cells $1 \dots K$.
Set initial values $\mathbf{x}_i(0)$
Set tolerance ε
Initialize $\mathbf{y}_{ij}^0(t) = 0$ for all cells.
while $|y_{ij}^n(t) - y_{ij}^{n-1}(t)| > \varepsilon$ for some $i, j = 1 \dots M$ **do**
  $n \leftarrow n + 1$
  for cell $k = 1 \dots K$ **do**
    Formulate IVP for $\mathbf{x}_i^n(t)$ (4), $\mathbf{y}_{ij}^n(t)$ (2)-(3) for all $i \in C_k$, based on prior delay trajectories $\mathbf{y}_{ji}^{n-1}(t)$, $j \notin C_k$
    Solve IVP on time interval $[0, T]$
    Save outbound delay trajectories $\mathbf{y}_{ij}^n(t)$ where $j \notin C_k$
  end for
end while

We begin by presenting the core existence result, in Theorem 1. This applies to both the full model (1)-(3), as well as the decomposed model (4)-(6). Detailed proofs can be found in [25].

Theorem 1. *For all iterations $n \geq 1$, both the full model* (1)-(3)*, and the decomposed model,* (4)(2)-(3)*, are Lipschitz continuous over x_i^n, y_{ij1}^n, and y_{ija}^n, respectively. Furthermore, they have a unique solution on the finite time interval* $[0, T]$*, with constant initial values* $\{\boldsymbol{x}(0), \boldsymbol{y}(0)\}$.

Proof Sketch. The full proof may be found in [25]. The proof of this begins by establishing that all differential equations are both Lipschitz continuous with respect to x_i^n and $y_{ij,a}^n$. Then, it is also established that they are all continuous with respect to t. From these results, the Picard-Lindelöf theorem implies the existence of a unique solution given any initial condition. □

Next, we present two Lemmas that present key characteristics about the algorithm. The first, Lemma 2, establishes that the system is always positive, at any iteration and any time point. The second, Lemma 4, establishes that the total number of jobs in the system at any point in time is bounded above by the initial population of N. Both of these are important results for the convergence proof, but also establish that the system does not enter an invalid state, such as having negative job populations at certain stations or a total number of jobs larger than the initial population N.

Lemma 2. $y_{ija}^n(t) \geq 0$ *and* $x_i^n(t) \geq 0$ *for all iterations* n *and* $t \geq 0$.

Lemma 4. *The total number of jobs in the system at any time* t *is bounded above by the initial number of jobs* N.

Next, we present the main convergence result. In particular, it is established that the model converges as the error tolerance ε becomes small. We can also establish that the system is monotonically increasing over n at every parameter. Therefore, we can measure the accuracy at each iteration, by summing all parameters and subtracting from the original job population.

Theorem 2. *Trajectory-based iteration converges pointwise to the solution of the full model as $\varepsilon \to 0$. Furthermore, $y_{ij,a}^{n+1}(t) > y_{ij,a}^n(t)$, and $x_i^{n+1}(t) > x_i^n(t)$, for all iterations n and times t.*

Proof Sketch. The full proof may be found in [25]. The proof begins by showing that if each variable in the system increases from iteration n to iteration $n+1$, then all variables must also increase from iteration $n+1$ to iteration $n+2$. For the base case, a direct proof is used to establish that the system is increasing from iteration 1 to iteration 2. By induction, this implies that the system is always increasing.

Based on the induction argument, the prior boundedness result, and a few other properties including the boundedness of the derivatives and the finite measure of $[0, T]$, it can be shown that the system must converge to a result by the monotone convergence theorem. The limit of the system of results as $n \to \infty$ is then shown to be equal to that of the full model. Therefore, we have established convergence to the full model as $n \to \infty$.

In order to show convergence as $\varepsilon \to 0$, further results are necessary. We first establish the uniform equicontinuity of both $x_i^n(t)$ and $y_{ij,a}^n(t)$, over the iteration n and time t. The the Arzelà-Ascoli theorem can be used to establish the existence of a uniformly continuous subsequence of iterations. A somewhat technical argument after that, based on the fact that ε bounds the maximum norm, shows that the system converges as $\varepsilon \to 0$. □

V. Pricing and Vehicle Allocation

A. Heuristics for Optimizing Finite-Capacity Queues

When optimizing a vehicle sharing system using a fluid model, it is possible that queue lengths that are too high or too low might not be adequately penalized. This is due to the fact that the throughput is assumed to be equal to the service rate if the mean queue length is above 1 when using the mean-field based approach proposed in this paper. This assumption is justifiable theoretically via Kurtz's theorem, but may overestimate the number of trips taken in real-world scenarios. In addition, the fluid limits we have used do not necessarily apply to $-/M/1/k$ queues with rerouting. To ameliorate these issues, we propose a heuristic to model finite-capacity queues based on the steady-state transition probabilities, and a similar heuristic for estimating the number of lost jobs. The proposed approach is to revise the equations for each single-server queue i, with a heuristic correction for the capacity k_i (4). We start by estimating $\pi_{k_i}(x_i(t))$, or the probability that the queue is full when the mean is equal to $x_i(t)$. In this case, we estimate this value by assuming that it is equal to the steady-state probability, using $x_i(t)$ as the mean queue length. Let μ be the service rate, γ be the arrival rate, and $\rho = \frac{\gamma}{\mu}$ is the traffic intensity. We then state the equation for the mean queue length in steady state for an $M/M/1/k$ queue.

$$Q = \frac{1-\rho}{1-\rho^{k+1}} \sum_{i=0}^{k} i\rho^i \tag{7}$$

For all but the lowest capacities, this equation cannot be inverted efficiently, as it is a high-dimensional rational equation. For transient analysis, we can build a lookup table between

values of ρ and Q, where $\rho(Q)$ is the traffic intensity such that the mean queue length in steady state is Q. As a heuristic method, it is not necessarily true that $\rho(x_i(t))$ is equal to the arrival rate divided by the service rate during a particular stage. Using $\rho(x_i(t))$, we can estimate the probability of finding a specific station i at capacity,

$$\hat{\pi}_{k_i}(x_i(t)) = \frac{1-\rho(x_i(t))}{1-\rho(x_i(t))^{k+1}}\rho(x_i(t))^k \tag{8}$$

Arrivals into a full queue can be routed to another station within the same cell, using a predetermined routing probability $p_{ij}^k(t)$. We also use $\sigma_{k_i}(t) = 1 - \hat{\pi}_{k_i}(x_i(t))$ as the probability of finding station i at a level that is less than its capacity. We can revise (4),

$$\begin{aligned}\dot{x}_i^n(t) = \quad & \sigma_{k_i}(t)\sum_{j\notin C_k}^{M}\sum_{a=1}^{K_{ij}^e}\mu_{jia}^e\phi_{jia}^e y_{jia}^{n-1}(t) \\ & +\sigma_{k_i}(t)\sum_{j\in C_k}^{M}\sum_{a=1}^{K_{ij}^e}\mu_{jia}^e\phi_{jia}^e y_{jia}^{n}(t) \\ & -\lambda_i^e\min(x_i^n(t),1)\end{aligned} \tag{9}$$

We also introduce $p_{ij}^k(t)$, which is the probability that a user will transition to station j after arriving at i at full capacity. We can re-route these trips to another station by revising (2),

$$\begin{aligned}\dot{y}_{ij1}^n(t) = \quad & p_{ij}^k(t)(\hat{\pi}_{k_i}(x_i(t)))\sum_{j\notin C_k}^{M}\sum_{a=1}^{K_{ij}^e}\mu_{jia}^e\phi_{jia}^e y_{jia}^{n-1}(t) \\ & +p_{ij}^k(t)(\hat{\pi}_{k_i}(x_i(t)))\sum_{j\in C_k}^{M}\sum_{a=1}^{K_{ij}^e}\mu_{jia}^e\phi_{jia}^e y_{jia}^{n}(t) \\ & +\lambda_{ij}^e\min(x^n(t),1) - \mu_{ij1}^e y_{ij1}^n(t)\end{aligned} \tag{10}$$

Furthermore, it is useful when optimizing the network to know the rate at which users attempt to access station i, to find it at capacity. We represent an estimate of this value as $\gamma_i^n(t)$. This value can be found by multiplying the effective rate at which users attempt to return to the station by the probability that they find it at capacity,

$$\begin{aligned}\gamma_i^n(t) = \quad & \sigma_{k_i}(t)\sum_{j\notin C_k}^{M}\sum_{a=1}^{K_{ij}^e}\mu_{jia}^e\phi_{jia}^e y_{jia}^{n-1}(t) \\ & +\sigma_{k_i}(t)\sum_{j\in C_k}^{M}\sum_{a=1}^{K_{ij}^e}\mu_{jia}^e\phi_{jia}^e y_{jia}^{n}(t)\end{aligned} \tag{11}$$

In the full model, this can be represented as,

$$\gamma_i(t) = \quad \sigma_{k_i}(t)\sum_{j}^{M}\sum_{a=1}^{K_{ij}^e}\mu_{jia}^e\phi_{jia}^e y_{jia}(t) \tag{12}$$

It is important to note that this may not preserve the fluid limits given by Kurtz's theorem [21], but it has given sufficiently accurate results in our numerical experiments. This should be considered as a heuristic approach to analyze the impact of full station capacities on customer trips. We analogously use the $M/M/1/k$ mean queue-length to produce an estimate for the probability that station i is empty,

$$\hat{\pi}_0(x_i(t)) = \frac{1-\rho(x_i(t))}{1-\rho(x_i(t))^{k+1}} \tag{13}$$

Using this, we estimate the rate at which customers arrive to an empty queue as $\lambda_i\hat{\pi}_0(x_i(t))$.

Algorithm 2 Decomposed Optimization with Simulated Annealing

Set initial temperature F_0, decay factor α, primary tolerance ε_1, secondary tolerance ε_2 and initial system parameters $\mathbf{U}$
for $i = 1 \dots N_{epochs}$ **do**
Reconciliation:
  Determine rewards r and trajectories $y_{ij}^i(t)$, from initial state $\mathbf{x}_0$ and $\mathbf{U}$, set $F_i \leftarrow F$ using Algorithm 1 and primary tolerance ε_1
  Randomly shuffle *cells*
Annealing:
  for cell k in *cells* **do**
    $F_{ik} \leftarrow F_i$
    for $j = 1 \dots N_{substeps}$ **do**
      Find a neighbor of cell-specific control parameters $\mathbf{u}_k'$
      cells $\leftarrow \{k\}$
      while length (*cells*) > 0 **do**
        Solve IVP on time interval $[0, T]$ for each cell, and update reward difference Δp
        Set *cells* to equal any cells that have an inbound trajectory that changes by more than ε_2
      end while
      if $rand(0,1) \leq F_{ik}\exp\Delta p$ **then**
        Update cell parameters to $\mathbf{u}_k'$, as well as all trajectories $y_{ij}(t)$ for visited cells
      end if
      $F_{ik} \leftarrow \alpha F_{ik}$
    end for
  end for
  $F \leftarrow \alpha^{N_{substeps}} F_i$
end for

B. Decentralized Optimization Problem for Joint Pricing and Rebalancing

In this section, we apply trajectory-based iteration to an optimization problem for maximizing the daily profit of a vehicle sharing system with both offline and incentive-based rebalancing. The control variables are the initial levels, $x_i(0)$, and the price for a customer to rent a vehicle from a particular station p_i. The total number of vehicles in the system at time 0 is unconstrained, and there is no price incurred when the system adds another vehicle, other than changes in the daily operational cost of the system.

We introduce two penalties in our optimization problem. The first, c_c, represents the cost incurred when a user attempts to return a vehicle to a full station. This may reflect second-order costs incurred when a user attempts to return a vehicle to

a full station, or may simply be viewed as a penalty to ensure adequate service. The second penalty, c_r, represents the cost to rebalance a vehicle if the ending levels $x_i(T)$ differ from the starting levels $x_i(0)$.

The optimization problem can be formulated as,

$$\max_{p_i, x_i(0)} \quad \frac{1}{T}\sum_{i=1}^{M}\int_0^T (p_i\lambda_i(t,p_i)(1-\hat{\pi}_0(t,p_i))x_i(t)dt - \sum_{j=1}^{M}\int_0^T c_c\gamma_i(t)dt - \frac{c_r}{2}\sum_{i=1}^{M}|x_i(T,p_i)-x_i(0,p_i)| \tag{14}$$

s.t.

$$p_{\min} \le p_i(t) \le p_{\max}, \quad \forall i \in [0,M] \tag{15}$$

$$0 \le x_i(0) \le k_i, \quad \forall i \in [0,M] \tag{16}$$

The goal of this optimization problem is to maximize the total revenue, less any costs from rebalancing or penalties from queues being at capacity. As we assume that no rebalancing takes place during the time period $(0,T]$, this optimization problem can be classified as analyzing static repositioning [26]. In this program, we assume that only one rebalancing stage is used. However, this formulation may be extended to multiple rebalancing stages, by integrating over multiple time intervals $[0,T]$, $[T,2T]$, ..., $[(n-1)T,nT]$, and applying the rebalancing cost between the ending levels at one interval and the starting level at the next interval.

The problem is highly non-linear and non-concave, so we use a variant of simulated annealing [27]. The algorithm we have used is described in Algorithm 2. In this algorithm, one iterates on each submodel separately for a fixed number of steps, and then solves the model as a whole to reconcile all trajectories. Then, this process is repeated. The core assumption is that all cells are loosely coupled, and changes in the optimization problem can be accurately estimated based on modeling only a few cells, in which outbound trajectories change by a significant amount. This algorithm features two error tolerance parameters, ε_1 and ε_2. The first is the tolerance for solving the model as a whole, using the TBI algorithm presented in Algorithm 1. The second one is used to determine when to stop determining the changes in each submodel. In our empirical results, we use a much smaller value of ε_2, compared to ε_1. This is to detect more granular changes on how cell parameters affect the objective function of a submodel, both locally and on its neighbors. Furthermore, a smaller value is necessary to prevent propagation of errors within the annealing step, as trajectories are updated once a step is accepted.

VI. Numerical Results

A. Evaluating Trajectory-Based Iteration

We evaluate the proposed TBI method of Algorithm 1 using randomly generated networks of $-/M/1$ queues. The number of queues in each model is uniform within $[75,100]$, with interstation demands λ_{ij} as uniform within the range $(0,0.05)$. Stations are uniformly placed on a 1×1 grid. To model the travel time between stations, we introduce a normal random variable $X\sim\mathcal{N}(0,0.2)$ as a noise term. Then, the travel time between each stations i and j is set to $\max(d_{ij}+X,0.01)$, where d_{ij} is the Euclidean distance between each station on the unit square. Each test is ran until $t=4$.

TABLE I
TBI Accuracy and Speed with BDF and RK45 ($\varepsilon=2$)

Method	*Time Full Model*	*Time TBI*	*CDE*	*MDE*	*SMAPE*
BDF	232	18.4	$2.02\cdot10^{-3}$	$2.54\cdot10^{-4}$	0.48%
RK45	9.32	5.83	$2.05\cdot10^{-3}$	$2.56\cdot10^{-4}$	0.51%

Three error metrics are used to analyze the results of the experiment. The first is the symmetric mean absolute percentage error (SMAPE). The second, which we call cumulative displacement error (CDE) is the total absolute difference of station trajectories, divided by twice the number of samples S and the total number of jobs N. This metric is only defined for queue lengths, and is intended to measure the percentage of entities that are misplaced in the queueing network.

$$CDE = \frac{1}{2NS}\sum_{i=1}^{M}|\hat{x}_i(t)-x_i(t)| \tag{17}$$

Based on the monotonicity and boundedness results shown in Theorem 2 and Lemma 4, we know that the results are always less than that of the full model. Therefore, the CDE gives an accurate estimation of how close the algorithm is to convergence, as a percentage of the total number of jobs.

The third, which we call maximum displacement error (MDE), is the maximum difference in trajectories at any station, at any time during the run, divided by the total number of jobs N. This is similar to the error metric in [23]. This is intended to provide an upper bound of the error, and to measure how far the system is from convergence.

$$MDE = \frac{1}{N}\max_i\max_t(|\hat{x}_i(t)-x_i(t)|) \tag{18}$$

This experiment is ran on a server with 16 processor cores and 16 GB of RAM, with parallelization of submodels. Results can be found in tables I and II. Table I shows a $16.0-37.4\%$ improvement in runtime under Runge-Kutta 4/5, and up to a 92.1% improvement when using Backward Differentiation Formula. The latter result implies that this method might be especially useful for stiff systems. Table II shows the accuracy with different values of the error tolerance parameter ε, and shows that there can still be significant performance gains with an even more accurate configuration. We recommend setting ε to 2 when time is the most important parameter, but we also note that $\varepsilon=0.5$ shows accurate results with only 21.6% higher runtime. Furthermore, averaging 5 stations per cell gives the best results in terms of runtime, with minimal loss of accuracy.

The CDE results show that the algorithm is very close to convergence, with the total counts being between 0.2% and 0.066% of the jobs at the end of the iteration, depending on ε. This implies not just accuracy of the final result, but also rapid convergence of the TBI algorithm.

TABLE II
TBI ACCURACY AND SPEED WITH DIFFERENT VALUES OF ε (RK45)

ε	*Time Full Model*	*Time TBI*	*CDE*	*MDE*	*SMAPE*
2	9.32	5.83	$2.05 \cdot 10^{-3}$	$2.56 \cdot 10^{-4}$	0.51%
1	9.32	7.07	$7.52 \cdot 10^{-4}$	$5.08 \cdot 10^{-5}$	0.42%
0.5	9.32	7.09	$7.49 \cdot 10^{-4}$	$5.02 \cdot 10^{-5}$	0.42%
0.25	9.32	7.83	$6.60 \cdot 10^{-4}$	$3.51 \cdot 10^{-5}$	0.41%

TABLE III
EXPERIMENT WITH RESAMPLED PARAMETERS

Metric	Average RHL	SMAPE	CDE
Q	10.0%	4.3%	$2.36 \cdot 10^{-2}$
T	13.2%	5.7%	—

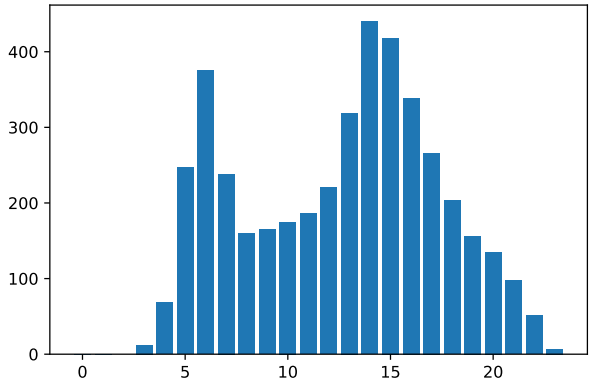


Fig. 1. Average Trip Count by Hour in Oslo (Weekdays)

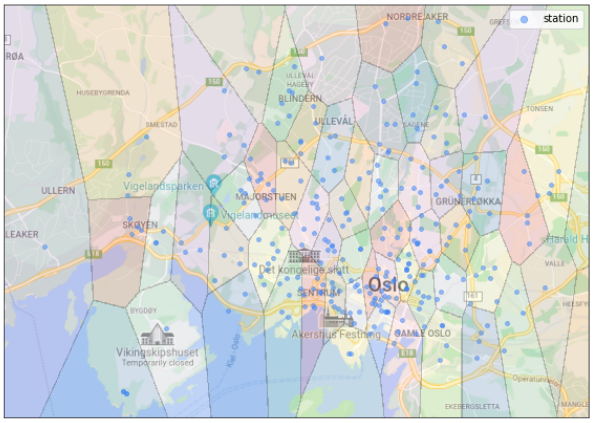


Fig. 2. Map of Clusters in Oslo

B. Modeling the Oslo Bikesharing Network

Next, we introduce our model of the Bysykkel bikesharing system in Oslo. Bysykkel was chosen as it covers a wide range of the Oslo metropolitan area, it provides high quality data, and reflects significant rush-hour effects. Bysykkel includes 285 stations within the city of Oslo and some of the surrounding areas. We use a public dataset[1] of 1,116,344 trips over the course of 2021. We filter the data for trips between 5 am and 9 pm on weekdays. Figure 1 shows the average number of trips, per hour. This shows significant rush-hour effects centered around both 5 am and 3 pm, representing a significant challenge for rebalancing. Figure 2 shows a map of Oslo, with an overlay showing the 57 clusters used in the model.

The routing probabilities and service rates are held constant within each hour, but are allowed to vary between hours. We use the mean hourly realized trips as the value for the trip service rate λ_{ij}^e. Let N_{days} is the number of days in the data, e is the hour, and $\{t_n^{i,j}\}$ is a sequence of starting times for trips from i to j. Then, we estimate λ_{ij}^e as,

$$\hat{\lambda}_{ij}^e = \frac{1}{N_{days}} \sum_n 1_{e \le t_n^{i,j} < (e+1)} \tag{19}$$

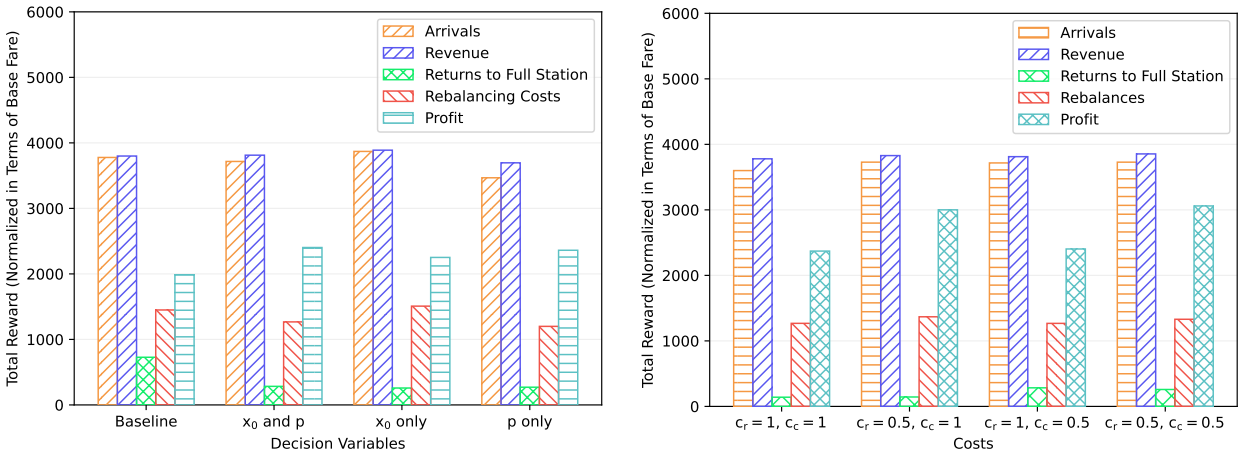
This represents a lower bound for the actual value, and can be viewed as assuming that the data reflects a perfectly rebalanced system. As customers are assumed to arrive according to a Poisson process, this is an accurate estimation of the rate, under the perfect-rebalancing assumption. It is also possible to fit based on the inter-arrival times of customers, but since the time window is small in comparison to the rate at certain stations, it may result in a biased estimation.

In order to parametrize the transit delays, we use the Cox fitting algorithm from the LINE software package [28], fitted based on the the first two moments of the travel duration between any two cells. As a further simplifying assumption, we let the travel time distribution between any pair of cells, (k, l), be shared across all stations. In particular, the travel time distribution is fitted based on the squared coefficient of variation SCV. Where $\delta = 0.001$, if $SCV \le 0.5 + \delta$, an Erlang distribution is fitted. The number of phases in this case set at $\frac{1}{\tau}\lceil\frac{1}{SCV}\rceil$, where τ is the mean travel time in the data. This is by far the most common case. If $0.5 + \delta < SCV < 1 - \delta$, a two-phase hyper-Erlang distribution is fitted. If $1 - \delta \le SCV \le 1 + \delta$, an Exponential distribution is fitted. Finally, if $SCV > 1 + \delta$, then a two-phase hyper-exponential distribution is fitted.

In Table IV, we present our results of a model validation experiment. We compare the results from TBI modeling with a discrete-event simulation, a trace-based simulation, and the raw number of trips. In our full model, we use $\epsilon = 0.5$ to parametrize the TBI algorithm. For the discrete-event simulation, we use the SSA algorithm [29], and perform a total of 128 runs. The discrete-event simulation is a stochastic queueing simulation with identical parameters to the queueing network model. For the trace-based simulation, we iterate through each trip within the hours of 5 am and 9 pm, over each day, keeping track of the queue lengths at each point in time. If a trip would arrive at an empty station, it is discarded. If a trip attempts to return a bike to a full station, it is rerouted to another local station.

In Table IV, "Trips" represents the number of trips in the data, "DES" stands for discrete-event simulation, "TBS" stands for trace-based simulation, and "Fluid" represents the fluid model, using trajectory-based iteration. We analyze both the throughput and queue lengths of each station i, at the beginning of each hour. The results indicate that both the

[1]https://oslobysykkel.no/en/open-data/historical



(a) Num Players vs T

(b) Price vs T

Fig. 3. Change in Objective Values (Using Simulation)

TABLE IV
MODEL VALIDATION

Method A	*Method B*	*Metric*	*SMAPE*	*CDE*
Trips	DES	$T_i(t)$	17.4%	—
Trips	Fluid	$T_i(t)$	12.5%	—
DES	Fluid	$T_i(t)$	10.8%	—
TBS	DES	$T_i(t)$	16.2%	—
TBS	Fluid	$T_i(t)$	16.0%	—
DES	Fluid	$Q_i(t)$	6.57%	$1.53 \cdot 10^{-2}$
TBS	DES	$Q_i(t)$	9.21%	$3.19 \cdot 10^{-2}$
TBS	Fluid	$Q_i(t)$	12.8%	$3.80 \cdot 10^{-2}$

discrete-event simulation, the trace-based simulation, and the TBI-based fluid model are accurate reflections of each other. We continue to use the simulation as a baseline, even if it may provide less accurate results in certain cases, as it accounts for the stochasticity of the system.

Furthermore, we also conduct an experiment to show the robustness of the queueing model against sampling error. We use bootstrap resampling to estimate the mean $\hat{\bar{\lambda}}_i$ and standard deviation $\hat{\sigma}_i$ for the demands λ_i. Then, we analyze 16 different models. We let λ_i be randomly generated according to the normal distribution $\mathcal{N}(\hat{\bar{\lambda}}_i, \hat{\sigma}_i)$. Results, including the mean relative half lengths, the SMAPE, and CDE against the original model, are presented in Table III. The results indicate a 4.3% SMAPE when the queue length Q is considered, and a 5.7% SMAPE when the throughput T is considered.

C. Optimizing the Oslo Bikesharing Network

We now apply our model to an optimization case study. We set values tolerance parameters at $\varepsilon_1 = 0.5$, $\varepsilon_2 = 0.02$. For the algorithm runtime, we use 40 epochs, with 5 steps per epoch, for a total of 120 optimization steps in total. Initially, we set each station to a starting level of $x_i(0) = 0.5k_i$. This results in a total of approximately 3234 bikes, which is near to the true value at around 3000.

Following [9], we use a linear pricing model to estimate the impact of prices on interstation demands $\lambda_{ij}(t, p_i)$. We model the changes in user demand, according to price, using a price elasticity of 1.0. This implies that an estimation of the profit without accounting for lost trips, $\sum_{i,j} \lambda_{ij}(t, p_i)p_i$, is already maximized. Therefore, any change in the objective function can only result from improving the operational efficiency of the system, including reducing the number of lost trips due to an empty rack, or preventing attempted returns to a full station. We set c_c to 0.5 and c_r to 1.0. The value for c_r is based on the estimated rebalancing costs for Paris' Velib bikesharing network, which is equal to \$3.50 [30]. This is close to the non-subscriber fare of €3.10. We currently do not model other operational and fixed costs, which can vary between different systems [30]. This experiment is run on an HP Omen laptop with 32 GB of memory, and 16 processor cores.

Results are shown in Table V, Table VI, and Figure 3a. Table V and Figure 3a demonstrates the cost under different decision variables, and a comparison to the baselines provided. The results suggest a 16.7% to 29.3% improvement in the objective in the fluid model, which reduces to a 14.7% to 22.0% improvement when tested in the simulation, depending on the decision variable(s) chosen. Operational improvements, when both decision variables are considered, reflect a 62.2% reduction in trips in which a customer attempts to return a bike to a full station and a 12.5% reduction in rebalancing cost, when paired with a marginally small increase in revenue. This validates the potential of our framework to improve the operational performance of vehicle sharing systems. In Figure 3, we indicate how the revenue and different cost factors change, when validated with the simulation.

In Table VI and Figure 3b, we present a variation of the system under different values of c_r and c_c, in order to to validate the performance of the optimization method under different operational models. For both values, we vary the cost within $\{0.5, 1.0\}$. The results indicate a $16.4 - 46.7\%$ improvement in the fluid model, and a $13.9 - 49.4\%$ improvement using the simulation. In particular, we note that the model performs better under higher costs, and particularly under higher costs for failed returns in comparison to higher costs for rebalancing.

TABLE V
OSLO PERFORMANCE IMPROVEMENT WITH DIFFERENT DECISION VARIABLES

Method	*Decision Variable(s)*	*Fluid Model Improvement*	*Simulation Improvement*
TBI-SA	p and x_0	29.3%	22.0%
TBI-SA	x_0	16.7%	14.7%
TBI-SA	p	23.6%	20.3%
CQP	p and x_0	—	2.28%
QP	p^e_{ij}	—	7.30%

In all variations, our model outperforms both (QP) and (QCP).

For comparison with other methods, we introduce two baselines. We begin with adapting the continuous quadratic program from [8]. This has been adapted to match the existing problem. In particular, we remove the constraint for conserved parkings and our reward depends on the realized number of trips rather than the number of vehicles in transit at each point in time.

$$\max_{p_i, x_i(0)} \quad \sum_{i=1}^{M} \int_0^T p_i \lambda_i(t, p_i) dt \tag{20}$$

s.t.

$$x_i(t) = x_i(0) + \int_0^t \sum_{j=1}^{M} \lambda_{ji}(t, p_j) dt \quad \forall i \in [1, M], t \in [0, T] \tag{21}$$

$$p_{\min} \leq p_i(t) \leq p_{\max}, \quad \forall i \in [1, M], t \in [0, T] \tag{22}$$

$$0 \geq x_i(t) \leq k_i, \quad \forall i \in [1, M], t \in [0, T] \tag{23}$$

$$x_i(T) = x_i(0), \quad \forall i \in [1, M], t \in [0, T] \tag{24}$$

When the cost function is linear, as we assume in this case study, this becomes a continuous quadratic program. The goal is to find the maximum price, while using constraints to ensure that all queues are in heavy load, remain below capacity, and end with the same population that they begin with. Table V displays results for this approach as "CQP", and shows a 2.28% improvement in the objective value.

We consider another baseline that uses pricing to balance rates, over each time epoch. This program is formulated as,

$$\max_{p_i, x_i(0)} \quad \sum_{i=1}^{M} \sum_{j=1}^{M} \sum_{e=1}^{E} p^e_{ij} \lambda^e_{ij}(p^e_{ij}) \tag{25}$$

s.t.

$$p_{\min} \leq p^e_{ij}(t) \leq p_{\max}, \quad \forall i, j \in [0, M], e \in [0, E] \tag{26}$$

$$\sum_j \lambda^e_{ij}(p^e_{ij}) = \sum_j \lambda^e_{ji}(p^e_{ji}), \quad \forall i, j \in [0, M], e \in [0, E] \tag{27}$$

This program is based on the approach proposed in [7], [9], [12], as well as a heuristic proposed in [10]. In this case, the estimated profit is maximized while rates are constrained such that all rates into a particular queue equal the rates out, during all stages. Results can be found in Table V, labeled as "QP", and shows a 7.3% increase in the objective value. The performance of these baselines suggest that quadratic programming techniques may be insufficient in certain cases, under a transient and time-varying environment.

TABLE VI
OSLO PERFORMANCE IMPROVEMENTS WITH VARYING COST PARAMETERS

Method	c_r	c_c	*Fluid Model Improvement*	*Simulation Improvement*
TBI-SA	1	1	46.7%	49.4%
CQP	1	1	—	25.5%
QP	1	1	—	24.1%
TBI-SA	0.5	1	26.9%	29.8%
CQP	0.5	1	—	1.73%
QP	0.5	1	—	2.49%
TBI-SA	1	0.5	29.3%	22.0%
CQP	1	0.5	—	2.28%
QP	1	0.5	—	7.30%
TBI-SA	0.5	0.5	16.4%	13.9%
CQP	0.5	0.5	—	−11.9%
QP	0.5	0.5	—	−7.50%

VII. CONCLUSION

In conclusion, we have demonstrated Trajectory-Based Iteration (TBI), a method that enables efficient modeling and parallelization of transient and time-varying vehicle-sharing systems at scale. We have then extended this method for optimization with simulated annealing, and demonstrated its use for solving an optimization problem for pricing and offline rebalancing. Using this method, we have shown that our method can enable substantial operational improvements in vehicle sharing systems with simulated annealing. In a case study, we have used this framework to optimize a real-world bikesharing system, and observed a 14.7% to 22.0% improvement in performance, depending on the decision variables.

In future work, the model may be extended to account for stochastically-modulated changes in user behavior with random environments, or to online optimization. Another improvement may be to apply other model reduction methods for non-linear systems, such as proper orthogonal decomposition.

REFERENCES

[1] "Urban Development," World Bank, Tech. Rep., 2023. [Online]. Available: https://www.worldbank.org/en/topic/urbandevelopment/overview

[2] C. Moreno, "La ville du quart d'heure : pour un nouveau chrono-urbanisme," *La Tribune*, 2016. [Online]. Available: https://www.latribune.fr/regions/smart-cities/la-tribune-de-carlos-moreno/la-ville-du-quart-d-heure-pour-un-nouveau-chrono-urbanisme-604358.html

[3] "Driving Urban Transitions – Sustainable future for cities," Urban Europe, Tech. Rep., 2022.

[4] S. Ataç, N. Obrenović, and M. Bierlaire, "Vehicle sharing systems: A review and holistic management framework," *Euro journal on transportation and logistics*, vol. 10, 2021.

[5] D.-Y. Lin and J.-K. Kuo, "The vehicle deployment and relocation problem for electric vehicle sharing systems considering demand and parking space stochasticity," *Transportation Research Part E*, vol. 156, 2021.

[6] R. Nair, E. Miller-Hooks, R. Hampshire, and A. Bušić, "Large-Scale Vehicle Sharing Systems: Analysis of Vélib'," *International Journal of Sustainable Transportation*, vol. 7, pp. 85–106, 2013.

[7] R. Iglesias, F. Rossi, R. Zhang, and M. Pavone, "A BCMP network approach to modeling and controlling autonomous mobility-on-demand systems," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 357–374, 2019.

[8] A. Waserhole and V. Jost, "Pricing in vehicle sharing systems: optimization in queuing networks with product forms," *Euro journal on transportation and logistics*, vol. 5, pp. 293–320, 2014.

[9] S. Wollenstein-Betech, I. C. Paschalidis, and C. G. Cassandras, "Joint Pricing and Rebalancing of Autonomous Mobility-on-Demand Systems," in *2020 59th IEEE Conference on Decision and Control (CDC)*, Juju Island, Republic of Korea, 2020.
[10] A. Waserhole and V. Jost, "Vehicle Sharing System Pricing Regulation: A Fluid Approximation," *HAL Open Science*, 2012, preprint.
[11] G. Zardini, N. Lanzetti, M. Pavone, and E. Frazzoli, "Analysis and control of autonomous mobility-on-demand systems," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 633–658, 2022.
[12] R. Zhang and M. Pavone, "A Queueing Network Approach to the Analysis and Control of Mobility-On-Demand Systems," in *2015 American Control Conference (ACC)*. IEEE, 2015, pp. 4702–4709.
[13] J. A. Schwarz, G. Selinka, and R. Stolletz, "Performance analysis of time-dependent queueing systems: Survey and classification," *Omega*, no. 63, pp. 170–189, 2016.
[14] Q.-L. Li, C. Chen, R.-N. Fan, L. Xu, and J.-Y. Ma, "Queueing Analysis of a Large-Scale Bike Sharing System through Mean-Field Theory," *Queueing Models and Service Management*, vol. 5, no. 1, pp. 45–83, 2022.
[15] Bolch, *Queueing Networks and Markov Chains : Modeling and Performance Evaluation with Computer Science Applications*. Wiley-Interscience, 2006.
[16] L. Cordier and M. Bergmann, "Réduction de dynamique par Décomposition Orthogonale aux Valeurs Propres (POD)," 2006.
[17] J. Roth, "Proper Orthogonal Decomposition for Fluid Mechanics Problems," Ph.D. dissertation, Leibniz University Hanover, 2021.
[18] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin, "Symbolic computation of differential equivalences," in *POPL '16: Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, St Petersburg, Florida, USA, 2016, pp. 137–150.
[19] R. Zhang, F. Rossi, and M. Pavone, "Model predictive control of autonomous mobility-on-demand systems." IEEE, 2016, pp. 1382–1389.
[20] C. Fricker and N. Gast, "Incentives and redistribution in homogeneous bike-sharing systems with stations of finite capacity," *Euro journal on transportation and logistics*, vol. 5, no. 3, pp. 261–291, 2016.
[21] T. G. Kurtz, "Solutions of Ordinary Differential Equations as Limits of Pure Jump Markov Processes," *Journal of Applied Probability*, vol. 7, no. 1, pp. 49–58, 1970.
[22] C. Fricker, N. Gast, and H. Mohamed, "Mean field analysis for inhomogeneous bike sharing systems," in *Discrete Mathematics Theoretical Computer Science Proceedings*, Montreal, Canada, Jan. 2012, pp. 1–12.
[23] G. Casale, M. Tribastone, and P. G. Harrison, "Blending randomness in closed queueing network models," *Performance Evaluation*, vol. 82, pp. 15–38, 2014.
[24] M. Faddy, "On inferring the number of phases in a Coxian phase-type distribution," *Stochastic Models*, vol. 14, no. 1-2, pp. 407–417, 1998.
[25] M. Sheldon, D. Tuncer, and G. Casale, "Addendum: Convergence proof and running example for the tbi algorithm," 2024.
[26] G. Laporte, F. Menuier, and R. W. Calvo, "Shared mobility systems: an updated survey," *Annals of Operations Research*, vol. 271, 2018.
[27] M. Pincus, "A Monte-Carlo Method for the Approximate Solution of Certain Types of Constrained Optimization Problems," *Journal of the Operations Research Society of America*, vol. 18, no. 6, pp. 967–1235, 1970.
[28] J. Pérez and G. Casale, "Line: Evaluating software applications in unreliable environments," *IEEE Transactions on Reliability*, vol. 66, no. 3, pp. 837–853, 2017.
[29] D. Gillespie, "Stochastic Simulation of Chemical Kinetics," *Annual Review of Physical Chemistry*, vol. 58, no. 1, pp. 35–55, 2007.
[30] P. DeMaio, "Bike-Sharing: History, impacts, models of provision, and future," *Journal of Public Transportation*, vol. 12, no. 4, 2009.

Matthew Sheldon received the BS degree in Industrial and Systems Engineering in 2019 and the MSc degree in Computer Science in 2021. He is currently a PhD student in the department of Computing at Imperial College London. His research interests focus on mobility systems, rational queueing, and reinforcement learning.

Daphne Tuncer is currently with Ecole des Ponts ParisTech in France. While her past work mostly focused on understanding how we can better manage networked system infrastructures and resources, she is now working at the intersection between computer networks, data science and digitalized mobility and energy systems. Daphne is the recipient of a 2018 Imperial College Research Fellowship and the 2021 IEEE CNOM Young Professional award. She has a Ph.D. in Electronic and Electrical Engineering from University College London, UK, and a "Diplôme d'ingénieur" from Télécom SudParis, France.

Giuliano Casale joined the Department of Computing at Imperial College London in 2010, where he is currently a Reader. Previously, he worked as a research scientist and consultant in the capacity planning industry. He teaches and does research in performance engineering and cloud computing, topics on which he has published more than 150 refereed papers. He has served on the technical program committee of several conferences in the area of performance and dependability. His research work has received multiple recognitions, recently the best paper award at ACM SIGMETRICS. He serves on the editorial boards of IEEE TNSM and ACM TOMPECS and as current chair of ACM SIGMETRICS.