A Sampling Strategy for High-Dimensional Simulation-Based Transportation Optimization Problems

Working Paper

Timothy Tay*[1] and Carolina Osorio†[1]

[1]Department of Civil and Environmental Engineering, Massachusetts Institute of Technology, Cambridge MA 02139, USA

Abstract

The increasing computational cost of evaluating a traffic simulation has led to the need for more efficient algorithms when tackling high-dimensional simulation-based optimization (SO) problems. An approach to solving SO problems efficiently involves incorporating problem-specific structural information in the optimization algorithm. Alternatively, this problem-specific structural information can also be used for sampling points to be used in the optimization algorithm. This paper proposes an inverse cumulative distribution function (cdf) sampling mechanism to make use of problem-specific structural information in the form of an analytical model to efficiently sample for points with good performance. The resulting sampling distribution assigns greater probability to sampling points with better predicted performance. This allows the inverse cdf sampling mechanism to exploit prior information while still allowing for nonzero probability of sampling any point. We validate the inverse cdf sampling mechanism on a toy network. We then explore whether problem-specific structural information should be used in the sampling mechanism and/or the optimization algorithm when tackling a high-dimensional traffic signal control problem in Midtown Manhattan. The results show that the use of inverse cdf sampling mechanism as part of an optimization framework can help to quickly and efficiently identify solutions with good performance.

1 Introduction

Simulation-based urban traffic and mobility models have been extensively used by stakeholders to evaluate the performance of traffic management strategies (e.g. signal control, congestion pricing), as well as to evaluate new algorithms (e.g. dispatch algorithms in ridesharing) before releasing it to production (Greenhall 2016). These simulators can embed detailed traveler behavior models of departure time choice, mode choice, route choice and response to real-time traffic information. Moreover, stochastic models can account for uncertainty in demand and in supply. For instance, the behavioral models above may be specified as disaggregate probabilistic models, such as discrete choice models. Furthermore, the simulators are able to model the interactions of a large number of travelers to a resolution that analytical models are unable to. A review of traffic simulation models is given in Barceló (2010).

As the resolution, quantity and availability of urban mobility data increases, so does the interest by both public and private stakeholders in developing detailed, high-resolution mobility models. Moreover, with the increase in connectivity and real-time responsiveness of our mobility systems, comes an increase in the spatial dependency of network or link performance. This calls for an increase in the spatial coverage of the network to be studied. Stakeholders are shifting from performing a local (e.g., an arterial, a couple of intersections, a neighborhood) analysis to studying the impact across a full city or metropolitan area. This simultaneous increase in both the model resolution and its spatial coverage brings numerous methodological challenges.

*tayt@mit.edu
†osorioc.edu

In this work, we focus on the use of stochastic simulation-based models for continuous optimization problems. We consider continuous simulation-based optimization (SO) algorithms. Some examples within the transportation field where continuous simulation-based optimization (SO) problems are encountered include traffic signal optimization (Osorio and Bierlaire 2013, Osorio and Nanduri 2015a,b, Osorio and Chong 2015, Chong and Osorio 2018), origin-destination (OD) matrix calibration (Lu et al. 2015, Tympakianaki et al. 2015, Zhang et al. 2017), and congestion pricing (Osorio and Atasoy 2017).

The main challenges in urban mobility SO are twofold. The first challenge is the computational cost of evaluating the simulator. Computational cost increases with both the model resolution and with the spatial network coverage. Hence, the new generation of mobility models is increasingly costly to evaluate. Furthermore, since we are considering stochastic models, multiple simulation replications are required to obtain accurate objective function estimates, leading to even more demanding computation requirements. The increase in computing power has been outpaced by our interest in developing more detailed and larger-scale models. While the amount of available computing power has increased dramatically over the years, it is still insufficient for applying SO to "the control of complex stochastic systems" (Xu et al. 2016). At the same time, while real-time simulation of smart cities may become possible in the near future (Alibaba DAMO Academy 2019), further increases in computing power and improvements in computational efficiency are still required to make it possible to perform SO in real-time.

The second challenge involves dealing with high-dimensional optimization problems. Some problems, such as OD calibration, are naturally high-dimensional, with the number of dimensions (e.g. OD pairs) reaching into the thousands (Lu et al. 2015, Zhang et al. 2017). In other cases, such as urban traffic signal optimization, we are interested in jointly optimizing the network performance across an entire urban network, which leads to a high-dimensional optimization problem (Osorio and Chong 2015). With more variables to optimize, the computational resources required to solve the optimization problem is likely to increase.

This paper focuses on the above 2 challenges, with the goal of contributing to the design of SO algorithms that are both computationally efficient and suitable for high-dimensional SO problems. In this work, the focus is on continuous SO problems.

The computational inefficiency of large-scale high-resolution mobility simulators has limited their use for optimization. Most often, the use of the simulators is limited to what-if analysis (i.e., evaluation of a set of predetermined solutions) (Ben-Akiva et al. 2003). The most common approaches in the transportation SO literature include general-purpose algorithms such as genetic algorithms (Jin et al. 2017, Sebastiani et al. 2016, Paz et al. 2015, Stevanovic et al. 2008, Teklu et al. 2007, Yun and Park 2006) and simultaneous perturbation stochastic approximation (SPSA) (Tympakianaki et al. 2018, 2015, Lu et al. 2015, Balakrishna and Koutsopoulos 2008). The advantage of these algorithms is their broad generality (i.e., they are not limited to a specific type of transportation problems (e.g., congestion pricing)). However, this generality comes with a lack of computational efficiency. The underlying algorithms are designed based on asymptotic properties and exploit little to no problem structure information. Hence, they are not designed to be used within tight computational budgets, yet that is how they are used for any large-scale problem instance or case study.

However, work has also been done to address the inefficiencies of the general-purpose algorithms in transportation SO problems. For instance, Jin et al. (2017) proposed the use of an archive-based genetic algorithm in a stochastic traffic signal control optimization framework. The archive-based genetic algorithm differs from traditional genetic algorithms by storing the globally elite genes in an external archive, thus helping it achieve faster convergence. Several extensions to the original SPSA algorithm have also been proposed to achieve faster convergence (Tympakianaki et al. 2015, Lu et al. 2015).

To design computationally efficient SO algorithms, one recent line of work has been to complement general-purpose SO algorithms with problem-specific structural information. The problem-specific structural information provides prior knowledge of the problem, even before any simulation is done, thus helping to quickly identify some good solutions. This can come in the form of analytical models, which provide an approximation of the objective function (e.g. queueing network model representing a traffic network (Osorio and Bierlaire 2009)).

Metamodel SO algorithms are an example of a class of general-purpose SO algorithms which can be combined with problem-specific structural information. They have been proposed for various classes of continuous transportation SO problems (Chong and Osorio 2018, Osorio and Chong 2015, Osorio and Nanduri 2015a,b, Osorio and Bierlaire 2013) and more recently discrete SO problems (Zhou et al. 2018). A metamodel is an analytical approximation of the unknown SO objective function. The main idea behind this line of work has been to tackle the SO problem by solving a sequence of analytical optimization problems. More specifically, at every iteration of the SO algorithm, the SO objective function is replaced with the analytical metamodel and then an analytical optimization problem, known as metamodel optimization problem, is solved. Computational efficiency is achieved by formulating metamodels that have problem-specific structural information. More specifically, analytical net-

work models are formulated and used to approximate the SO objective function (i.e., to derive the metamodel). The main challenge in this approach is the formulation of an analytical network model that is computationally efficient and scalable while also being a good approximation of the SO objective function.

However, there are some limitations to this approach. First, when considering tight computational budgets, the metamodels used across iterations remain similar, and hence, the sequence of analytical optimization problems solved tend to yield nearby, or even identical, solutions. Second, although analytical models can provide some problem-specific structural information, they are just approximations of the objective function. More specifically, the analytical models do not necessarily reflect the shape of the objective function perfectly across the entire feasible region. This means that the SO algorithm has to be able to account for the inaccuracies in the model.

Alternatively, the problem-specific structural information can be used for sampling for points to be used in the SO algorithm. More specifically, the analytical model, containing the structural information of the underlying problem, can be used to construct the sampling distribution in such a way that points with good predicted performance are sampled with greater probability. This differs from the traditional metamodel SO algorithms (Osorio and Bierlaire 2013), where the analytical model is used during optimization. Given that solving the metamodel optimization problem with the analytical model is a computationally intensive task (see e.g., Figure 7 of Osorio and Chong (2015)), it is useful to see if using the analytical information during sampling instead could help to reduce the overall computational demand.

The analytical model could be used to construct a joint sampling distribution. However, it can be difficult to draw samples from the joint distribution. Monte Carlo methods, including importance sampling and Markov chain Monte Carlo methods, are often used to address this problem. In importance sampling, the goal is to estimate an integral or an expectation value with respect to a target distribution, while reducing the variance on the estimate. This is achieved by drawing samples from a different proposed distribution which overweights the important regions, and weighting each sample by the likelihood ratio when estimating the expectation value (Owen 2013, Chapter 9, pages 3-5). Since the goal of importance sampling is different from the goal of this work and does not sample according to the target distribution, we will not discuss it in detail. For a review on importance sampling, we refer the reader to Tokdar and Kass (2010).

Markov chain Monte Carlo (MCMC) is a class of methods for generating samples from a sampling distribution while exploring the state space using Markov chain mechanism. One of the most popular MCMC methods is the Metropolis-Hastings algorithm (Metropolis et al. 1953, Hastings 1970). The Metropolis-Hastings algorithm involves sampling from a proposal distribution (which is easy to sample from) in order to mimic samples drawn from the target distribution asymptotically. However, the choice of proposal distribution is important in ensuring the success of algorithm (Andrieu et al. 2003, Section 3.1). If the expressions for the full conditionals of the target distribution are known, the Gibbs sampler can then be used to draw samples from the target distribution (Geman and Geman 1984), eliminating the need to specify a proposal distribution. Another drawback of MCMC methods is the need to discard an initial set of samples, in practice, to avoid starting biases (Andrieu et al. 2003, Gelman and Shirley 2011). This is also known as the burn-in period or mixing time for the Markov chain to reach a stationary distribution. To avoid starting biases, Gelman and Shirley (2011) recommends discarding the first half of the samples. As such, the inefficiencies associated with the burn-in suggests that MCMC may not be suitable when computational resources are limited.

This study considers the question of whether problem-specific structural information should be used in the sampling mechanism, in the optimization algorithm, or both for a high-dimensional SO problem. We propose an inverse cumulative distribution function (cdf) sampling mechanism to make use of problem-specific structural information in the form of an analytical model (Osorio and Bierlaire 2009, Osorio and Chong 2015) to efficiently sample for points with good performance in a high-dimensional transportation optimization problem. This provides a positive correlation between the probability of sampling a particular point and the predicted performance of the point. In contrast to uniform sampling, which is traditionally used for picking initial points for optimization, the inverse cdf sampling mechanism attempts to exploit prior information while still allowing for nonzero probability of sampling any point. Thus, the inverse cdf sampling mechanism should be able to pick out better optima compared to a uniform sampling method, especially when working with the constraint of a limited computational budget. In addition, the analytical model gives a global approximation of the objective function without the need for any simulation evaluations, so this design should be able to achieve good short-term performance in a high-dimensional problem when used in an optimization framework.

The proposed inverse cdf sampling mechanism thus contributes in the following ways:

- **Scalability:** The inverse cdf sampling mechanism makes use of an analytical model that consists of a simple system of equations. The model is linear in the number of links in the network, which allows the inverse

cdf sampling mechanism to be scaled to high dimensions.

- **Efficiency:** By exploiting prior information in the form of an analytical model of the traffic network, the inverse cdf sampling mechanism samples for points with good performance with higher probability so that solutions with good performance can be found with just a small number of simulation evaluations.
- **Broad class of optimization problems:** While we demonstrate the use of the inverse cdf sampling mechanism on a transportation optimization problem, the inverse cdf sampling mechanism can be applied to other classes of optimization problems, in particular problems with expensive-to-evaluate black-box functions. This is possible as long as there is an analytical model available to provide an approximation of the objective function.

In the following section, we present the methods used to implement the inverse cdf sampling mechanism. A validation of the inverse cdf sampling mechanism is done in Section 3. We then test the inverse cdf sampling in a case study using a model of Mid-town Manhattan for a fixed time traffic signal control problem. The results of the case study are presented, along with a discussion, in Section 4. Our conclusions are provided in Section 5.

2 Methodology

In this section, we first explain the main idea behind the inverse cdf sampling mechanism in Section 2.1. The optimization problem of interest is formulated in Section 2.2. Then, Section 2.3 formulates the analytical model that is used to derive the analytical approximation of the SO objective function. The expressions for the sampling distributions are then derived from the analytical model. The SO algorithm is summarized in Section 2.4, along with some implementation details in Section 2.5.

2.1 Main idea

To illustrate the main idea of the proposed approach, we consider a one-dimensional minimization problem for which we have an analytical approximate expression, denoted f^A, of the SO objective function. The goal is to use f^A to define a sampling distribution that assigns high sampling probability to regions (of the feasible region) with good predicted performance (i.e., low f^A values) and low sampling probability to regions with bad predicted performance (i.e., high f^A values). The probability density function (pdf) of such a sampling distribution is given by:

$$g(x) = \frac{1}{\kappa_0}(\kappa_1 - f^A(x)), \tag{1}$$

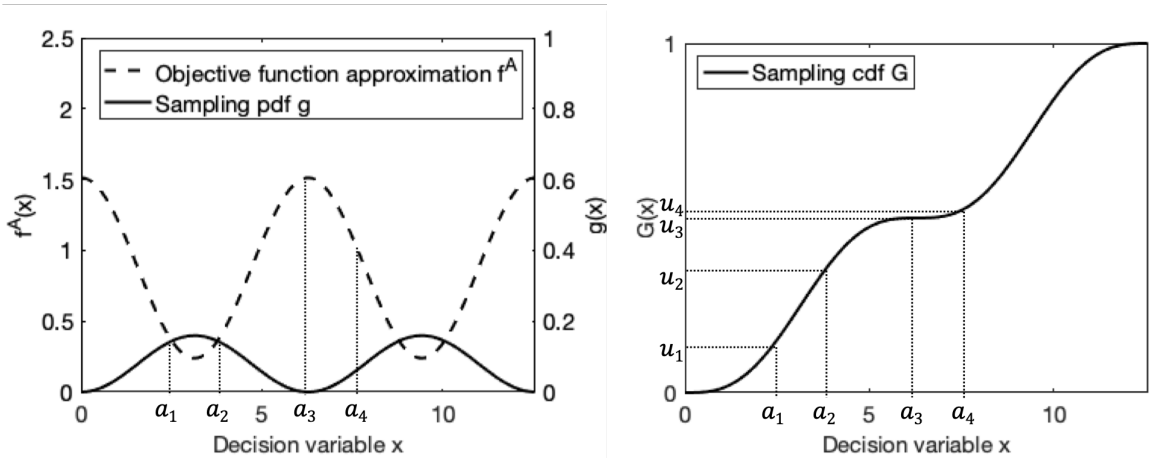
where κ_1 is a scalar upper bound of f^A, which ensures non-negativity of g, and κ_0 is a scalar normalization constant, which ensures that g integrates to 1.

This is graphically illustrated in Figure 1a, where f^A (resp. g) is represented by the dashed (resp. solid) line. In this figure, the magnitude of g is inversely proportional to that of f^A. For instance, the interval $[a_1, a_2]$ has good predicted performance (i.e., low f^A values) and thus high sampling probabilities (high g values). The inverse holds for interval $[a_3, a_4]$, which has bad predicted performance and thus low sampling probabilities.

To sample according to the pdf g, we use the inverse transform method, also known as inversion sampling or Smirnov transform. See for instance Casella and Berger (2002, Section 5.6.1, pages 247-249). The method assumes that the cumulative distribution function (cdf) of g, denoted G, is invertible, i.e., G^{-1} exists. It samples according to $G^{-1}(U)$, where U is a standard uniform random variable (i.e., a uniform distribution with support $[0, 1]$).

Given an analytical objective function, it might be possible that G^{-1} does not exist. Even if it does, G^{-1} might be numerically difficult to compute. This issue may be encountered in some transportation problems. However, a possible workaround could be to use an alternative analytical quantity as f^A (e.g. average delay), that exists and is easy to compute, and is also highly correlated with the objective function (e.g. average travel time). This would still result in higher sampling probabilities in regions with good predicted performance.

Figure 1b depicts the cdf G of the pdf g of Figure 1a. Intuitively, the cdf has steeper gradients in the regions where the magnitude of g is large. The inverse transform method samples independently from a standard uniform distribution, places these realizations along the y-axis of Figure 1b, and projects them according to G^{-1} on the x-axis. Recall the example above where interval $[a_1, a_2]$ has better predicted performance than interval $[a_3, a_4]$.



(a) (b)

Figure 1: One-dimensional illustration of how the objective function (to be minimized) can be used to construct (a) the sampling probability density function (pdf), which can be integrated to form (b) the cumulative distribution function (cdf) so that the probability of sampling is concentrated in regions with better predicted performance.

In Figure 1b sampling from interval $[a_1, a_2]$ (resp. $[a_3, a_4]$) occurs with probability $u_2 - u_1$ (resp. $u_4 - u_3$). Since $(u_2 - u_1) > (u_4 - u_3)$, the probability of sampling points in $[a_1, a_2]$ is greater than that of sampling points in $[a_3, a_4]$. In other words, sampling from the interval with better performance ($[a_1, a_2]$) occurs with a higher probability than sampling from the interval with worse performance ($[a_3, a_4]$).

The main challenge in our proposed approach is to formulate an approximation function f^A that balances providing: (i) a good approximation of the unknown SO objective function, and (ii) a computationally efficient and scalable (i.e., suitable for high-dimensions) way of sampling from the feasible region. In this paper, we consider a specific optimization problem and propose such a formulation.

2.2 Traffic signal control problem

The proposed sampling mechanism is applicable to a broad range of continuous SO problems. To illustrate its performance, we focus in this paper on one specific problem: a large-scale traffic signal control problem. For a review of traffic signal control terminology, see Osorio (2010, Appendix A, pages 119-121). The traffic signal control problem considered here is that of a fixed-time control strategy. A fixed-time signal plan is cyclic (i.e., periodic) for a given time interval and a given intersection. The cycle time is the time duration required to complete one sequence of signals. In this paper, the decision variables are the green splits (i.e., normalized green times) of each signal phase of each intersection in the network. All other control variables (e.g. offsets, stage structure, cycle times) are predetermined and assumed fixed. Hereafter, we define signal phase to be the interval in a cycle assigned to pre-specified traffic movements. The notation used in the formulation of the traffic signal control problem is given in Table 1.

The problem is formulated as follows:

$$\min_{\mathbf{x}} f(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta}) = \mathbb{E}[F(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta})] \tag{2}$$

$$\text{subject to} \sum_{j \in \mathcal{P}_1(\ell)} x_j = \frac{c_\ell - d_\ell}{c_\ell}, \ \forall \ell \in \mathcal{I} \tag{3}$$

$$\mathbf{x} \geq \mathbf{x}^{LB}. \tag{4}$$

The decision vector $\mathbf{x} = (x_1, \ldots, x_n)$ consists of the green splits for each signal phase. The objective function f (Eq. (2)) is taken to be the expected number of vehicles in the road network. The expectation is defined by the stochastic traffic simulator, with F representing the corresponding random variable. This objective function depends on a vector of exogenous simulation parameters $\boldsymbol{\theta}$, which represents, for instance, network topology or fixed lane attributes (e.g., lane length, maximum lane speed). It also depends on a vector of endogenous simulation

Table 1: Traffic Signal Control Problem Notation

f	SO objective function (expected number of vehicles in the road network);
F	random variable denoting the number of vehicles in the road network;
x_j	green split of signal phase j (decision variable);
$\mathbf{x}$	vector of all green splits (decision vector);
$\mathbf{z}$	vector of endogenous simulation variables.
	Exogenous problem parameters:
c_ℓ	cycle time of intersection ℓ;
d_ℓ	fixed cycle time of intersection ℓ;
$\mathbf{x}^{LB}$	vector of minimal green splits;
$\boldsymbol{\theta}$	vector of exogenous simulation parameters;
n	total number of signal phases to be optimized (i.e., dimension of the decision vector $\mathbf{x}$);
$\mathcal{I}$	set of signal controlled intersection indices;
$\mathcal{P}_1(\ell)$	set of signal phase indices of intersection ℓ.

variables $\mathbf{z}$, which represents, for instance, link-level and network-level performance metrics (e.g., expected queue-lengths, speeds, flows, travel times). The right hand side of Eq. (3) represents the proportion of the cycle time that can be optimized (i.e., that is not fixed). In practice, portions of the cycle time are constrained to be fixed typically for safety considerations and to comply with local transportation regulations. These fixed portions can include all-red periods, where the signal is red for all traffic streams at the given intersection, as well as portions where some traffic streams have short green times (e.g., such as to enable a smooth transition between two signal phases). Eq. (3) equates, for a given intersection, the proportion of non-fixed cycle time to the total endogenous green time (i.e., sum of green splits) allocated to that intersection (left hand side). The green splits have lower bounds (Eq. (4)), which are typically determined based on safety considerations following guidelines from local transportation authorities.

2.3 Formulation of the analytical sampling distribution

Recall from Eq. (1) that the sampling distribution (g) is defined based on an analytical approximation (f^A) of the simulation-based objective function. In this section, we derive an expression for f^A. We represent a given road network by a finite (space) capacity queueing network. More specifically, each lane i of the network is represented by a finite (space) capacity queue $M/M/1/k_i$, where k_i denotes the finite space capacity of the underlying lane. By using finite, rather than infinite, space capacity queues, we account for the limited number of cars each lane can accommodate based on the physical length of the lane. This allows us to model vehicular spillback (i.e., when the length of the queue of vehicles exceeds the lane length, and spills back to upstream lanes) through the queueing theoretic concept of blocking. To introduce this model, we define the notation in Table 2. The index i refers to a given queue.

Hereafter the terms queue and lane are used interchangeably. For such a network the expected number of vehicles in the network is defined as:

$$f^A(\mathbf{x}) = \mathbb{E}\Big[\sum_{i=1}^{m} N_i\Big] \tag{5}$$

$$= \sum_{i=1}^{m} \mathbb{E}\big[N_i\big] \tag{6}$$

$$= \sum_{i=1}^{m} \Big(\frac{\rho_i}{1-\rho_i} - \frac{(k_i+1)\rho_i^{k_i+1}}{1-\rho_i^{k_i+1}}\Big). \tag{7}$$

In Eq. (5), N_i depends on ρ_i (as shown in Eq. (7)), and ρ_i depends on $\mathbf{x}$. The relationship between $\boldsymbol{\rho}$ and $\mathbf{x}$ is explained in Eq. (8) and (9). The transition from Eq. (6) to (7) is based on finite capacity queueing theory. See for instance, Bocharov et al. (2004, Chapter 3, pages 96-98). A derivation of this expression is given in Osorio and Chong (2015, Appendix A).

Since the expression of Eq. (7) does not depend explicitly on $\mathbf{x}$, but instead depends on $\boldsymbol{\rho}$ which itself is a function of $\mathbf{x}$, we simplify the notation by using ρ_i to refer to $\rho_i(\mathbf{x})$. We now define the mapping between $\boldsymbol{\rho}$ and

Table 2: Analytical Model Notation

	Endogenous variables of the analytical model:
ρ_i	traffic intensity;
$\boldsymbol{\rho}$	vector of traffic intensities of all queues;
λ_i	arrival (i.e., demand) rate;
N_i	number of vehicles in queue i;
$P(N_i = k_i)$	probability of queue i being full, also known as the blocking or spillback probability;
μ_i	service rate;
	Exogenous parameters of the analytical model:
m	total number of queues in the network;
γ_i	external arrival (i.e., demand) rate;
p_{ij}	probability of turning from queue i to queue j;
k_i	space capacity in terms of number of vehicles;
$\mathcal{M}$	set of all queues;
$\mathcal{U}_i$	set of upstream queues of queue i;
$\mathcal{D}_i$	set of downstream queues of queue i;
$\mathcal{L}$	set of signalized queues;
$\mathcal{P}_2(i)$	set of signal phase indices of signalized queue i;
$\mathcal{P}_3(i)$	index of the intersection that queue i leads to;
$\mathcal{P}_4(i)$	set of signal indices at the intersection that queue i leads to, but not including the indices of signals which are green in favor of vehicles in queue i.

x. To do this, we use the analytical network model of Osorio and Chong (2015, Eq. 6), which is defined as the following system of $3m$ nonlinear equations:

$$\lambda_i = \gamma_i\big(1 - P(N_i = k_i)\big) + \sum_{j \in \mathcal{U}_i} p_{ji}\lambda_j \tag{8a}$$

$$\rho_i = \frac{\lambda_i}{\mu_i} + \Big(\sum_{j \in \mathcal{D}_i} p_{ij} P(N_j = k_j)\Big)\Big(\sum_{j \in \mathcal{D}_i} \rho_j\Big) \tag{8b}$$

$$P(N_i = k_i) = \frac{1 - \rho_i}{1 - \rho_i^{k_i+1}}\rho_i^{k_i}. \tag{8c}$$

Eq. (8a) is a flow conservation equation relating the demand rate of queue i (left hand side of the equality) to the sum of the demand rate of vehicle trips that start in queue i (first term of the right hand side) and of the demand rate of vehicle trips that arise from upstream queues (second term of the right hand side). More specifically, the demand rate for trips that start in queue i is represented by γ_i, and the term $(1 - P(N_i = k_i))$ enforces that trips can only start in queue i if it is not full. The turning probabilities (p_{ij}) are assumed exogenous, thus the impact of the signal plans on the travelers route choices is not accounted for. Eq. (8b) defines the traffic intensity. The first term of the right hand side is the traffic intensity when the queue is not full (i.e., when there is no spillback). The second term accounts for the impact in queue i due to spillbacks from its downstream queues. Eq. (8c) gives the expression of the spillback probability (i.e., the blocking probability) as defined for an $M/M/1/k_i$ queue (see for instance, Bocharov et al. (2004, Chapter 3, pages 96-98)).

In Osorio and Chong (2015, Eq. 6), λ_i (resp. ρ_i) is denoted λ_i^{eff} (resp. ρ_i^{eff}) and is referred to as the effective arrival rate (resp. effective traffic intensity). In Osorio and Chong (2015), the effective traffic intensity is used to approximate the traffic intensity. In this paper, we use the model of Osorio and Chong (2015) and we simplify the terminology by not using the term *effective*. We refer the reader to Osorio and Chong (2015, Section 3.3) for a discussion on how the effective traffic intensity differs from the traffic intensity.

The endogenous variables of the above system of equations are related to the decision vector (the green split

vector $\mathbf{x}$) by the following linear equations:

$$\mu_i = s\left(e_i + \sum_{j \in \mathcal{P}_2(i)} x_j\right) \ \forall i \in \mathcal{L}, \tag{9}$$

where s denotes an exogenous scalar that represents the saturation flow rate (i.e., maximum queue discharge rate), and e_i is an exogenous parameter that represents the ratio of fixed green time to cycle time for signalized queue i.

Eq. (9) states that the service rate of a signalized queue i is given by the saturation rate scaled by the proportion of cycle time the queue has a green phase. This proportion is given by the term in parenthesis, which depends on the fixed (i.e., not optimized) time (term e_i) and the variable time (summation term, which represents the sum of the green splits of the signal phases of queue i).

We combine Eq. (1) and (7) to obtain the joint sampling pdf g, in the $\boldsymbol{\rho}$ space. We use the notation $g_{\rho_1,\ldots,\rho_m}$ to indicate that the joint pdf is with regards to $\rho_1, \ldots, \rho_m$. In other words, we have:

$$g_{\rho_1,\ldots,\rho_m}(\rho_1, \ldots, \rho_m) = \frac{1}{\kappa_0}\big(\kappa_1 - f^A(\boldsymbol{\rho})\big). \tag{10}$$

The computation of the scalars κ_0 and κ_1 is detailed in Section 2.5.3.

Based on Eq. (10) we compute the corresponding joint cdf $G_{\rho_1,\ldots,\rho_m}(\rho_1, \ldots, \rho_m)$ through integration:

$$G_{\rho_1,\ldots,\rho_m}(\rho_1, \ldots, \rho_m) = \frac{1}{\kappa_0}\int_0^{\boldsymbol{\rho}} \kappa_1 - f^A(\tilde{\boldsymbol{\rho}}) d\tilde{\boldsymbol{\rho}}, \tag{11}$$

The inverse transform method is defined for univariate distributions, while G (of Eq. (11)) is an m-variate distribution. We use the method of Rosenblatt (1952) to transform an m-variate distribution into a uniform distribution on the m-dimensional hypercube. This transformation breaks down the m-variate distribution into a set of univariate conditional distributions, allowing the components of ρ_i to be sampled sequentially. The transformation is given by:

$$\rho_1 = G^{-1}_{\rho_1}(u_1) \tag{12}$$

$$\rho_2 = G^{-1}_{\rho_2|\rho_1}(u_2) \tag{13}$$

$$\ldots$$

$$\rho_m = G^{-1}_{\rho_m|\rho_1\ldots\rho_{m-1}}(u_m) \tag{14}$$

where u_j denotes a realization of a univariate standard uniform random variable, $G^{-1}_{\rho_i}$ denotes the inverse of the cdf of the marginal distribution of ρ_i, and $G^{-1}_{\rho_i|\rho_1\ldots\rho_j}$ denotes the inverse of the cdf of ρ_i conditional on $\rho_1, \ldots, \rho_j$. Note that in Eq. (12)-(14), the order of conditioning is defined by the user.

Before introducing the expressions for the marginal and conditional cdf's, we summarize the notation to be used in Eq. (15)-(22) below:

G_{ρ_i}	marginal cdf of ρ_i;
$G_{\rho_i\|\rho_1\ldots\rho_j}$	conditional cdf of ρ_i conditional on $\rho_1, \ldots, \rho_j$;
κ_0	scalar normalization constant;
κ_1	scalar upper bound of f^A;
ρ_i	traffic intensity of queue i;
$\hat{\rho}_i$	upper bound on ρ_i;
k_i	space capacity of queue i in terms of number of vehicles;
u_i	realization of a univariate standard uniform random variable;
m	total number of queues in the network;
$\mathcal{M}$	set of all queues.

By explicitly computing the integral in Eq. (11), we can obtain the analytical expression of the joint cdf. The derivation of the expression for the joint cdf is provided in Appendix B.1. We define $\hat{\boldsymbol{\rho}} = (\hat{\rho}_1, \ldots, \hat{\rho}_m)$ to be the vector containing the upper bounds on the possible values of all ρ_i, such that $\rho_i \in [0, \hat{\rho}_i], \forall i \in \mathcal{M}$. Then,

the analytical expression of the marginal cdf of ρ_1 (Eq. (16)) can be obtained from the joint cdf by letting $\rho_i = \hat{\rho}_i$ for $i = 2, \ldots, m$. In other words, the variables $\rho_i, i = 2, \ldots, m$ are marginalized out of the joint cdf. The conditional cdf can be computed from the joint cdf by using Leibniz's integral rule (Abramowitz and Stegun 1964, Eq. (3.3.7)) to give the expression (18). The variable q_i (Eq. (19)) and constant r_i (Eq. (20)) were defined to simplify the expressions of the marginal cdf (Eq. (16)) and the conditional cdf (Eq. (18)). The variable $h(\rho_i)$ (Eq. (21)) corresponds the integral of the expected number of vehicles in queue i (i.e., the term in parentheses in Eq. (7)) with respect to ρ_i. The derivative $\frac{dh(\rho_i)}{d\rho_i}$ (Eq. (22)) then corresponds to the expected number of vehicles in queue i. Both $h(\rho_i)$ and $\frac{dh(\rho_i)}{d\rho_i}$ are used to simplify the expressions in Eq. (16)-(20). A detailed derivation of the expression of the marginal and conditional cdf is given in Appendices B.2 and B.3 respectively. Eq. (16) and (18) are solved using the *fzero* routine in Matlab to obtain the sampled value of ρ_i.

$$u_1 = G_{\rho_1}(\rho_1) \tag{15}$$

$$= \frac{1}{\kappa_0}\left[\left(\kappa_1\rho_1 - h(\rho_1)\right)\prod_{k=2}^{m}\hat{\rho}_k - r_1\rho_1\right] \tag{16}$$

$$u_i = G_{\rho_i|\rho_1,\ldots,\rho_{i-1}}(\rho_i|\rho_1,\ldots,\rho_{i-1}) \tag{17}$$

$$= \frac{\kappa_1\rho_i\prod_{k=i+1}^{m}\hat{\rho}_k - q_i\rho_i - r_i\rho_i - h(\rho_i)\prod_{k=i+1}^{m}\hat{\rho}_k}{\kappa_1\hat{\rho}_i\prod_{k=i+1}^{m}\hat{\rho}_k - q_i\hat{\rho}_i - r_i\hat{\rho}_i - h(\hat{\rho}_i)\prod_{k=i+1}^{m}\hat{\rho}_k} \text{ for } i = 2, \ldots, m \tag{18}$$

$$\text{where } q_i = \left(\sum_{j=1}^{i-1}\frac{dh(\rho_j)}{d\rho_j}\right)\left(\prod_{k=i+1}^{m}\hat{\rho}_k\right) \text{ for } i = 2, \ldots, m\ , \tag{19}$$

$$r_i = \sum_{j=i+1}^{m}\left(h(\hat{\rho}_j)\prod_{k=i+1,k\neq j}^{m}\hat{\rho}_k\right) \quad \forall i \in \mathcal{M}\ , \tag{20}$$

$$h(\rho_i) = -\rho_i - \log(1-\rho_i) + \rho_i\log(1-\rho_i^{k_i+1}) + \sum_{\alpha=1}^{\infty}\frac{\rho_i^{\alpha k_i+\alpha+1}}{\alpha(\alpha k_i+\alpha+1)} \quad \forall i \in \mathcal{M}\ , \tag{21}$$

$$\frac{dh(\rho_i)}{d\rho_i} = \frac{\rho_i}{1-\rho_i} - \frac{(k_i+1)\rho_i^{k_i+1}}{1-\rho_i^{k_i+1}}. \tag{22}$$

2.4 Algorithm

The inverse cdf sampling algorithm for generating one realization of $\mathbf{x}$ is summarized in Algorithm 1. In brief, a uniform random vector is transformed to $\boldsymbol{\rho}$ using the inverse transform method as specified by Eq. (12)-(14). After sampling each ρ_i, the constraints on ρ_i (see Section 2.5.1 for details) are checked if they are satisfied. If the constraints are not satisfied, ρ_i is resampled based on a newly generated uniform random number according to Eq. (18). After all the constraints have been satisfied, the complete set of values for $\boldsymbol{\rho}$ is transformed into the set of values of $\mathbf{x}$ using a pseudoinverse as described briefly in Section 2.5.1 and in detail in Appendix B.4.

2.5 Implementation details

2.5.1 Constrained sampling

When generating a set of values for $\boldsymbol{\rho}$ using inverse cdf sampling, the constraints given by Eq. (3), (4), (8) and (9) have to be satisfied. Eq. (3) and (4) provide the upper and lower bounds on x_i respectively. Combining Eq. (4) and (9) allows the computation of a lower bound on μ_i, which occurs when all the phases involving queue i are assigned the minimum green split. From Eq. (9), the lower bound μ_i^{LB} is then given by

$$\mu_i^{LB} = \sum_{j\in\mathcal{P}_2(i)} x_j^{LB}s + e_i s \tag{23}$$

where x_j^{LB} is the j^{th} component of $\mathbf{x}^{LB}$. Eq. (3) and (9) together provide an upper bound on μ_i based on the maximum allocable green split for a given intersection (i.e., one phase is allocated all the available green split for that intersection, and the rest of the phases are allocated only the minimum green split). The upper bound μ_i^{UB}

Algorithm 1: Inverse cdf sampling

0. **Initialization**

   (a) Initialize all exogenous parameters
   Initialize $\mathbf{x}^{LB}, s, e_i(\forall i \in \mathcal{L}), p_{ij}(\forall i, j \in \mathcal{M})$ for the underlying network
   Initialize c_ℓ and d_ℓ for all intersections ℓ
   Set $\hat{\rho}_i = 0.999, \forall i \in \mathcal{M}$ (see Section 2.5.2)
   Set κ_1 according to Eq. (27)

   (b) Generate uniform random vector $\mathbf{U} = (u_1, \ldots, u_m) \in [0,1]^m$

   (c) Compute $h(\hat{\rho}_i), \forall i \in \mathcal{M}$ according to Eq. (21)

   (d) Compute $r_i, \forall i \in \mathcal{M}$ according to Eq. (20)

   (e) Compute the normalization constant κ_0 according to Eq. (29)

1. **Compute inverse of marginal cdf for the first component** ρ_1

   (a) Find the root of $u_1 = G_{\rho_1}(\rho_1)$ to obtain ρ_1 according to Eq. (16)

2. **Compute inverse of conditional cdf for the remaining components** ρ_i
   for $i = 2, \ldots, m$, in increasing order of i

      (a) Compute $\frac{\partial h(\rho_{i-1})}{\partial \rho_{i-1}}$ according to Eq. (22)

      (b) Compute q_i according to Eq. (19)

      (c) Conditioning on the sampled components $\rho_1, \ldots, \rho_{i-1}$, find the root of $u_i = G_{\rho_i|\rho_1,\ldots,\rho_{i-1}}(\rho_i|\rho_1, \ldots, \rho_{i-1})$ to obtain ρ_i according to Eq. (18)

      (d) Check constraints on ρ_i if $\rho_j, j \in \mathcal{D}_i$ have been sampled

         i. Check that ρ_i satisfies the constraints (25) and (26)

         ii. **if** constraints are met
            continue to Step 3
         else
            while constraints are not met

               A. Let Ψ denote the set of indices of the components which do not satisfy the constraints (25) and (26)

               B. Generate uniform random vector $\mathbf{U} \in [0,1]^{|\Psi|}$

               C. **for** $i \in \Psi$, in increasing order of i
                  Repeat Steps 2a-2c for component i

               D. Check that ρ_i satisfies the constraints (25) and (26)

Algorithm 2: Inverse cdf sampling (continued)

3. **Transform $\boldsymbol{\rho}$ to x**
   Using the complete set of sampled values of ρ_i,
   (a) Compute $P(N_i = k_i), \forall i \in \mathcal{M}$ according to Eq. (8c)
   (b) Compute $\lambda_i, \forall i \in \mathcal{M}$ simultaneously according to Eq. (8a)
   (c) Compute $\mu_i, \forall i \in \mathcal{M}$ according to Eq. (8b)
   (d) Let $\boldsymbol{\Xi}$ be the matrix with elements $\xi_{ij} = 1$ if $j \in \mathcal{P}_2(i), 0$ otherwise; $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_m)$; and $\mathbf{e} = (e_1, \ldots, e_m)$. Then, using a Moore-Penrose pseudoinverse of $\boldsymbol{\Xi}$, compute $\mathbf{x}$ according to Eq. (9) (see Appendix B.4, Eq. (B.45) for details)

given by

$$\mu_i^{UB} = \Big(\frac{c_{\mathcal{P}_3(i)} - d_{\mathcal{P}_3(i)}}{c_{\mathcal{P}_3(i)}} - \sum_{j \in \mathcal{P}_4(i)} x_j^{LB}\Big)s + e_i s \tag{24}$$

where $\mathcal{P}_3(i)$ refers to the index of the intersection that queue i leads to, and $\mathcal{P}_4(i)$ denotes the set of phase indices at the intersection that queue i leads to, but not including the indices of phases which are green in favor of vehicles in queue i. The term in brackets in Eq. (24) represents the maximum allocable green split, which is the fraction of cycle time left for the phase after assigning the minimum green split to the other phases, and accounting for fixed cycle time. The constraint (24) is not the tightest possible constraint in the sense that, at a given intersection, the sum of service rates of the queues leading to the intersection could be greater than the flow capacity of the intersection. This would manifest itself as the sum of green splits at that intersection not adding up to the cycle time of the intersection when transforming the sampled values of $\boldsymbol{\rho}$ to $\mathbf{x}$ (Step 3 of Algorithm 1). For instance, there could be phases that are simultaneously green during a given period in the cycle, which should not happen in practice. This issue is overcome by scaling the green splits proportionally, as mentioned in Appendix B.4.

The constraints on μ_i can be related to constraints on ρ_i through Eq. (8b), where replacing μ_i with μ_i^{LB} provides the upper bound on ρ_i and vice versa:

$$\rho_i \leq \frac{\lambda_i}{\mu_i^{LB}} + \Big(\sum_{j \in D_i} p_{ij} P(N_j = k_j)\Big)\Big(\sum_{j \in D_i} \rho_j\Big) \tag{25}$$

$$\rho_i \geq \frac{\lambda_i}{\mu_i^{UB}} + \Big(\sum_{j \in D_i} p_{ij} P(N_j = k_j)\Big)\Big(\sum_{j \in D_i} \rho_j\Big) \tag{26}$$

Note that the right hand sides of the bounds (25) and (26) contain endogenous variables (e.g. ρ_j for $j \neq i$), so they do not act as bounds when drawing samples of ρ_i. Instead, when drawing a sample for ρ_i from the conditional cdf (Eq. (18)), the bounds (25) and (26) are only checked if all $\rho_j, j \in \mathcal{D}_i$ have been sampled. If ρ_i does not satisfy the bounds (25) and (26), it is then resampled from the conditional cdf (Eq. (18)) again. This process is repeated for all i until a complete set of values for $\boldsymbol{\rho}$ that satisfy the constraints is obtained. This method of rejection sampling (Robert and Casella 1999, Chapter 2.3) ensures that the sampling of ρ_i is consistent with the sampling distribution defined by the conditional cdf.

After sampling for a complete set of values for $\boldsymbol{\rho}$, we transform it into a set of values of $\mathbf{x}$. This is done by first computing the values of the service rates μ_i (using Eq. (8b)). Note that the external arrival rates γ_i, transition probabilities p_{ij} and queue length upper bounds k_i are exogenous. Given $\boldsymbol{\rho}$, the blocking probabilities $P(N_i = k_i)$ can be computed for all i using Eq. (8c). Given $\boldsymbol{\rho}$ and $P(N_i = k_i)$ for all i, λ_i can be solved for using Eq. (8a). Then, given $\boldsymbol{\rho}$, $P(N_i = k_i)$ and λ_i for all i, μ_i can be computed using Eq. (8b). Having computed μ_i, the values of the green splits x_k can be computed using a Moore-Penrose pseudoinverse (see e.g. Campbell and Meyer (1991, Chapter 1)) of the linear equality constraint in Eq. (9), which relates the green splits of a signal phase to the service rate of the corresponding lane. For more details on the transformation of $\boldsymbol{\rho}$ to $\mathbf{x}$, we refer the reader to Appendix B.4. A pseudoinverse is used because the linear system of equations is overdetermined, since the number of queues leading to a given signalized intersection is greater than the number of signal phases at that intersection. To compute the pseudoinverse, the code of Luong (2009) was used, as it is able to handle

sparse matrices. The resulting green splits $\mathbf{x}$ is unique, since the code of Luong (2009) uses the Moore-Penrose pseudoinverse, which is unique.

2.5.2 Support of the joint cdf

To obtain the marginal cdf of ρ_1 (Eq. (16)), we evaluate the integral in Eq. (11), and marginalized the variables $\rho_i, i = 2, \ldots, m$ out of the joint cdf by letting $\rho_i = \hat{\rho}_i, i = 2, \ldots, m$ (see Appendix B.2 for details). In other words, the support of the joint cdf (Eq. (11)) is taken to be $[\mathbf{0}, \hat{\boldsymbol{\rho}}]$. It should be noted that from Eq. (7), $f^A(\boldsymbol{\rho})$ is undefined for any $\rho_i = 1$. This is true, by extension, for Eq. (21) and (22). As such, for the case studies of this paper, we set $\hat{\rho}_i = 0.999$ for all i. Note that in theory ρ_i can be greater than 1 (when there is hypercongestion). This occurs when the arrival rate to queue i is greater than its service rate (see Eq. (8b)), resulting in a build-up of congestion on queue i. Hence, by assuming that the greatest value that ρ_i can take is $\hat{\rho}_i = 0.999$, the distribution is actually truncated. However, the truncation means there is more sampling probability mass in the region $\rho_i < 1$, where the arrival rate to link i is smaller than its service rate. This means that congestion is less likely to occur in queue i. Thus, this can have the effect of generating ρ_i sample values that result in less congestion in queue i.

2.5.3 Computation of κ_0 and κ_1

Recall from Eq. (1) that κ_1 is a scalar upper bound of f^A, which represents the expected number of vehicles in the network (Eq. (5)). The number of vehicles in each queue is bounded above by the queue's space capacity (k_i). Thus, we set as upper bound:

$$\kappa_1 = \sum_{i=1}^{m} k_i. \tag{27}$$

The normalization constant κ_0 is obtained by explicitly evaluating the following integral:

$$\kappa_0 = \int_0^{\hat{\boldsymbol{\rho}}} \kappa_1 - f^A(\boldsymbol{\rho}) d\boldsymbol{\rho}. \tag{28}$$

The resulting expression used to compute κ_0 is given by:

$$\kappa_0 = \kappa_1 \prod_{i=1}^{m} \hat{\rho}_i - \sum_{i=1}^{m} \left[h(\hat{\rho}_i) \Big(\prod_{j \neq i}^{m} \hat{\rho}_j \Big) \right] \tag{29}$$

The derivation of Eq. (29) is similar to the derivation of the joint cdf, as given in Appendix B.1 (Eq. (B.1) - (B.16)).

2.5.4 Computation of an infinite summation

When computing the value of $h(\rho_i)$ in Eq. (21), we have to evaluate an infinite summation term. This infinite summation term arises from evaluating the integral in Eq. (11) to obtain the analytical expression of the joint cdf. It is obtained from the definite integral $\int_0^{\rho_i} \log(1 - \tilde{\rho}_i^{k_i+1}) d\tilde{\rho}_i$ by integrating the Taylor expansion of the term in the integral (see Appendix B.1, Eq. (B.12) to (B.14) for details). Since we consider $\rho_i < 1$ and the definite integral can be evaluated to be a constant, the infinite summation term converges to a constant. For computational purposes, the infinite summation term is approximated by taking the sum of just the first 100 terms, i.e.,

$$\sum_{\alpha=1}^{\infty} \frac{\rho_i^{\alpha k_i + \alpha + 1}}{\alpha(\alpha k_i + \alpha + 1)} \approx \sum_{\alpha=1}^{100} \frac{\rho_i^{\alpha k_i + \alpha + 1}}{\alpha(\alpha k_i + \alpha + 1)}. \tag{30}$$

3 Validation

To validate the proposed method we consider a synthetic toy network defined in Osorio and Yamani (2017). The network is depicted in Figure 2. It consists of 20 single-lane links and 4 intersections, with a main arterial (horizontal links) and 4 side roads (vertical links). Travel demand is defined such that vehicles only have straight paths (i.e., they do not make any right or left turnings at the intersections). The signal plans of each intersection

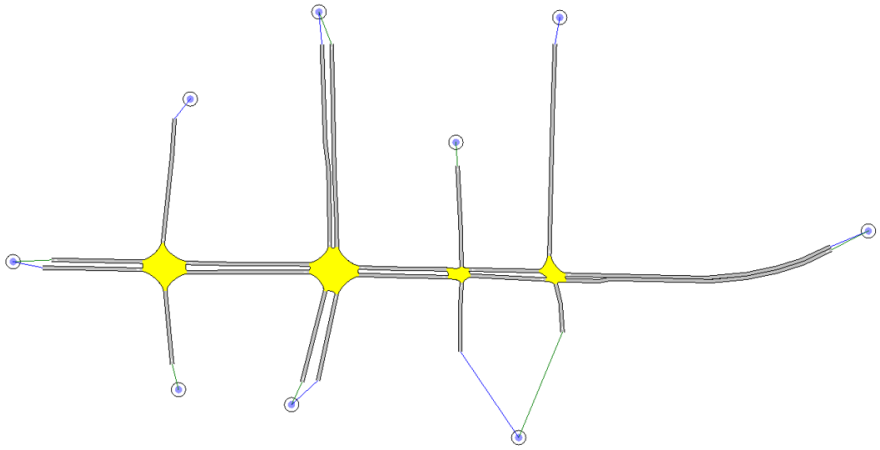


Figure 2: Toy network

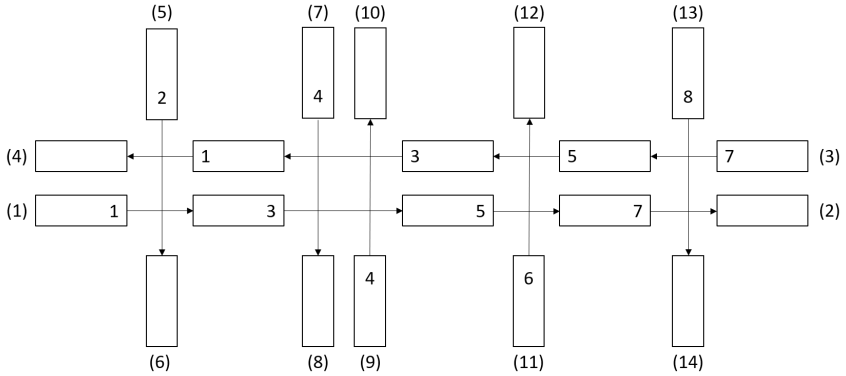


Figure 3: Queueing network representation of the toy network

are thus composed of two signal phases (i.e., there are no signal phases for turning movements). Origins and destinations of trips occur at the boundaries of the network which are represented by the circles in Figure 2. The queueing representation of the network is displayed in Figure 3. Each rectangle represents a queue. The arrows show the possible turns. The numbers in the rectangles represent the signal phase index (i.e., the index of the signal phase for which the underlying traffic movement has green). The numbers in brackets outside the rectangles denote the indices of the origin and destination nodes. Table 3 shows the origin-destination (OD) travel demand of the network. OD pairs which are not included in Table 3 have a demand of zero. Based on this OD travel demand, there is no congestion build-up with a well-chosen set of green splits. On the contrary, a poorly-chosen set of green splits will result in congestion, particularly along the main arterial.

First, we compare an estimate of the simulation-based objective function to its analytical approximation (i.e., we compare f of Eq. (2) to f^A of Eq. (7)). For each intersection the two signal phases are constrained by the linear equality constraint Eq. (3). Thus, there is only one degree of freedom per intersection. This leads to a total of four degrees of freedom. Figure 4 displays contour plots for each of the 2-dimensional projections of the objective function, while keeping the other two signal phases fixed at the histogram bin center value where the

Table 3: Origin-destination travel demand (i.e., external arrival rate) in veh/h

(1) → (2)	(3) → (4)	(5) → (6)	(7) → (8)	(9) → (10)	(11) → (12)	(13) → (14)
700	700	100	600	600	100	100

inverse cdf sampling assigns the highest marginal probability as shown in Figure 5a. Details on how the plots are generated are given in Appendix C. In both Figures 4a and 4b, the first (i.e., top) row of plots display the analytical approximations, while the second (i.e., bottom) row display the simulation-based estimates. For each row of plots, the mapping of colors to numerical values is displayed on the right. Figure 4 indicates that for all pairs of signal phases, the analytical and the simulation-based functions have similar trends, with the gradient of the function having the same sign when comparing a given region of a projection. The contour lines also have similar patterns, indicating that the minimum identified by the analytical model is close to that of the simulator. Note however that the magnitude of the functions differ. Recall from Figure 1 that the inverse cdf sampling is based on a comparison of values of f^A. In other words, f^A is used to identify regions of the feasible region that have better objective function values. Thus, the most important aspect is for f^A to capture the relative trends of f. It is not necessary for it to accurately approximate the magnitude of f.

Let us now compare the marginal sampling distributions obtained from the analytical model to those of the simulator and to those of a uniform sampling strategy. The gray bars of Figure 5a display an estimate of the marginal sampling distribution obtained from the analytical approximation f^A. The estimates are obtained from 1000 realizations (or draws) of the decision vector. The x-axis displays the values of the corresponding decision variable (i.e., the green split of the corresponding signal phase) and the y-axis is the estimated sampling probability. For comparison, we also display an estimate of a uniform sampling strategy (displayed as white bars). To sample uniformly from the feasible region (defined by Eq. (3)-(4)) we use the sampling strategy of Stafford (2006). Figure 5a indicates that the sampling distribution of the proposed approach differs from that of a uniform sampling distribution. Figure 5b displays an estimate of the simulator's marginal sampling distribution. Note that the scale on the y-axis of Figure 5b is different from that of Figure 5a, so as to allow the distribution to be seen more clearly. More details on the experimental design underlying these figures are given in Appendix C.

The gray bars for phases 1, 5 and 7 in Figure 5a show that the proposed method assigns a high probability to high values of the corresponding signal phase, which means that it favors the traffic along the main arterial, as opposed to that of the side roads. This is consistent with the corresponding travel demand (Table 3), which is defined such that the arterial has a higher demand compared to the side roads for these intersections. The arterial has a demand of 700 veh/h, while the side roads have demands of 100 veh/h, thus justifying the higher probability that phases 1, 5 and 7 are assigned values greater than 0.8. For phases 3 and 4, the proposed method assigns similar amounts of green time. Again this is consistent with the underlying demand for these intersections: the main arterial and the side road at the second intersection are almost equal, with demands of 700 veh/h and 600 veh/h respectively.

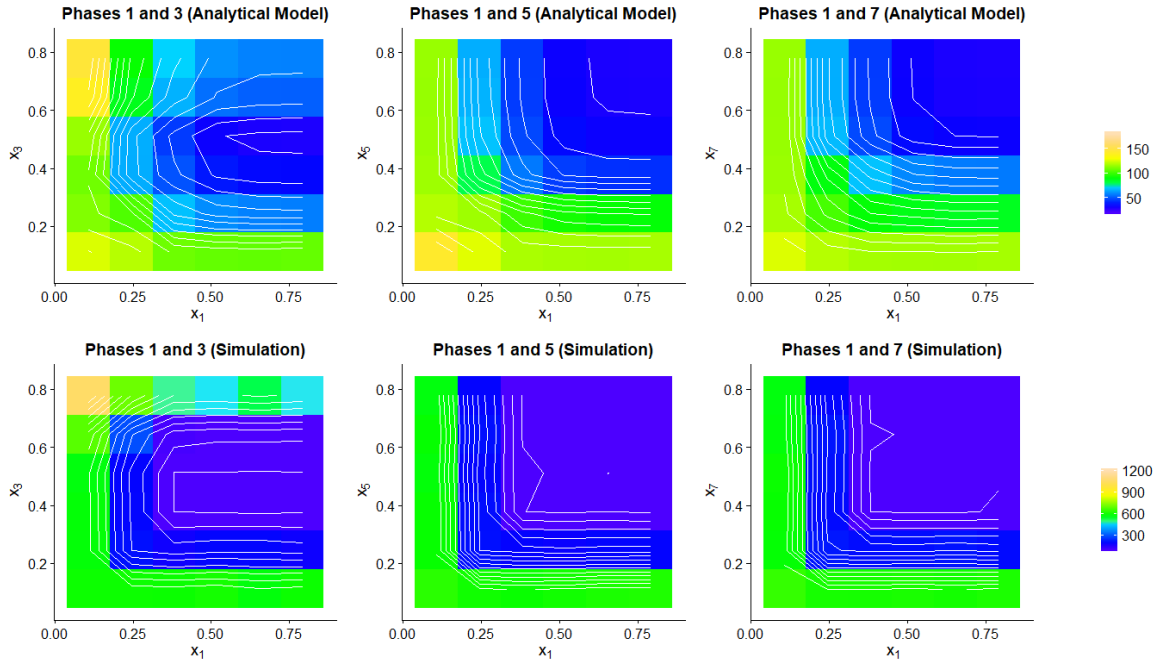
The simulator's distribution (Figure 5b) has the same trends as that of the proposed method. Namely: (i) for phases 1, 5 and 7, a higher sampling probability is assigned to signal plans that favor the traffic along the main arterial compared to the side roads, and (ii) for phases 3 and 4, both roads (arterial and side) are equally favored. Comparing the distribution of the proposed method to that of the simulator, we also observe that the modes of each phase match. This indicates that the proposed method has a higher probability of sampling in the region where the minimum is located than uniform sampling. However, it should be noted that the detailed shape of the distributions differ. The simulator's distributions tend to be flatter. We also note that, due to the linear cycle time constraint (Eq. (3)), the shapes of the distributions for phases 2, 4, 6 and 8 are mirror images of those of phases 1, 3, 5 and 7 respectively.

4 Case Study: Midtown Manhattan signal control

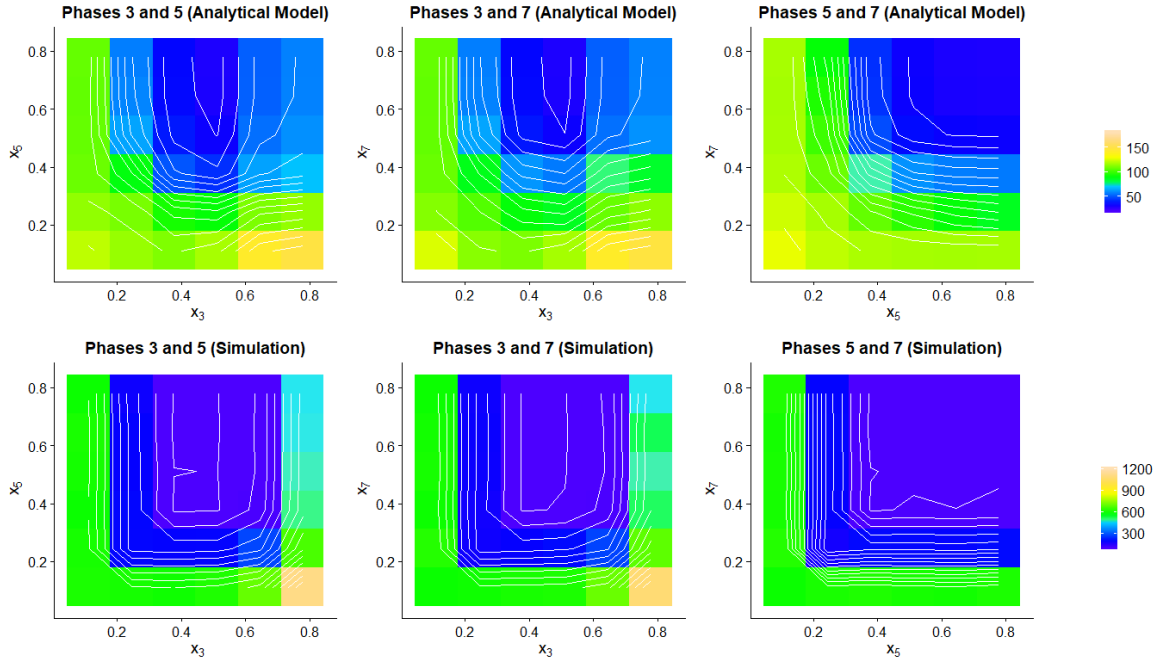
4.1 Experimental design

We now apply our proposed method to a signal control problem for the area of Midtown Manhattan (MTM) in New York City. The area of interest is demarcated by a rectangle in the map of Figure 6. We simulate traffic from 3pm - 6pm and control the signal plans for the peak hour of 5pm-6pm. We control the green splits of 97 intersections for a total of 259 green splits (i.e., decision variables). Osorio and Chong (2015) tackled a problem with 17 intersections and 99 green splits, which is considered to be large-scale in the field of signal control. Furthermore, problems with around 200 dimensions are considered high-dimensional in the field of continuous SO (Wang et al. 2016). Thus, this is considered a high-dimensional signal control SO problem.

The network topology of the simulator is displayed in Figure 7. The MTM simulation model, which is implemented in the Aimsun software (TSS-Transport Simulation Systems 2015), consists of a total of 698 roads, 2756 lanes and 444 intersections. In the simulated hour of 5-6pm, the expected demand is over 21700 trips per

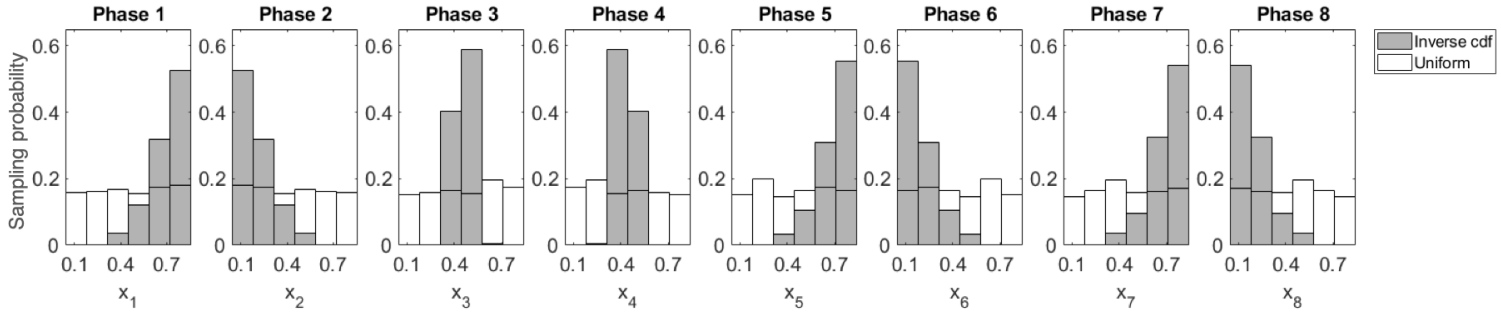


(a)

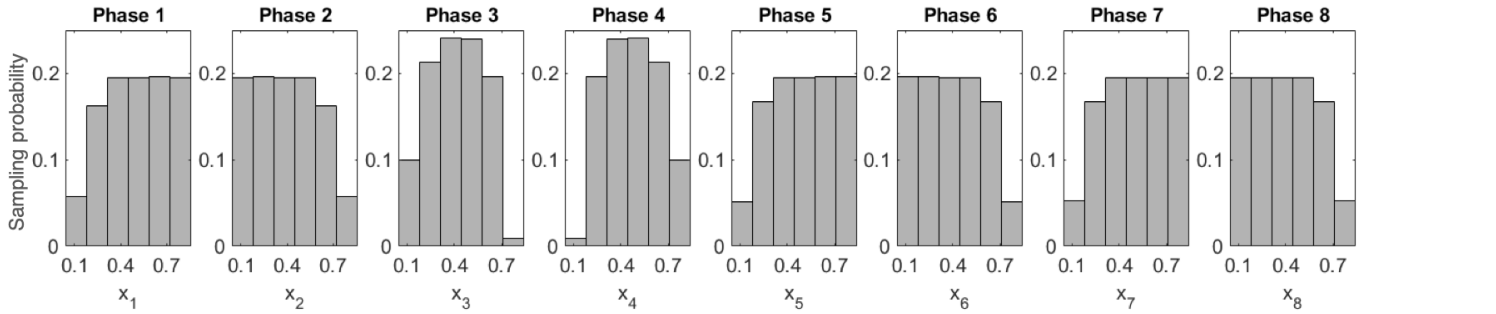


(b)

Figure 4: Two-dimensional projections of the objective function surface as estimated by the analytical model and simulation model.



(a) Distribution based on inverse cdf and uniform sampling



(b) Distribution based on traffic simulation results

Figure 5: Comparison of marginal green split distributions obtained using the (a) randomly generated sample points and (b) traffic simulation results.

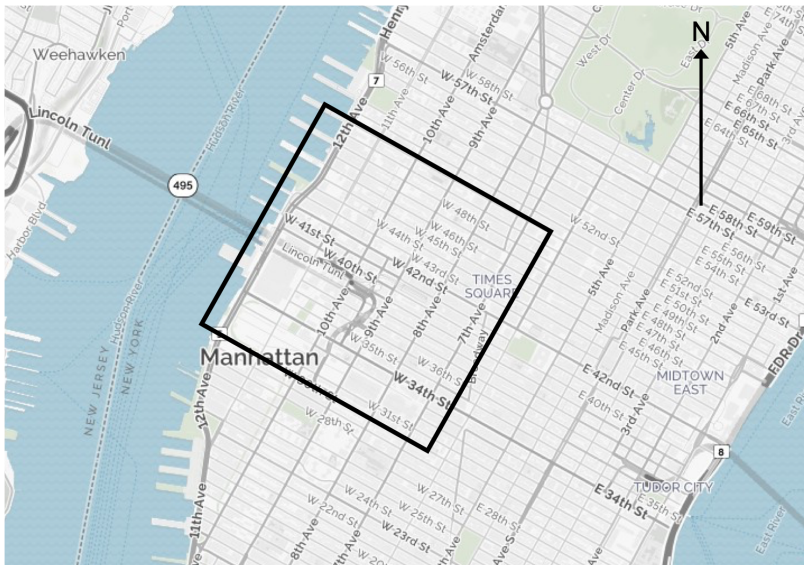


Figure 6: Map of Midtown Manhattan with the area of interest demarcated by a rectangle (MapQuest.com, Inc 2018)

hour, distributed across more than 3500 origin-destination pairs. In this simulation model, all of the green times have a lower bound of 6 seconds. The corresponding components in the vector of minimal green splits $\mathbf{x}^{LB}$ (Eq. (4)) are the ratio of 6 seconds to the cycle time for the intersection which the green split belongs to.

The simulator's objective function (Eq. (2)), which is the expected number of vehicles in the network, accounts for all vehicles that have started their trip. During simulation, when a given queue is full, additional vehicles entering that particular queue are placed in a virtual queue. In order to counter this in our experiments, the total demand of the network (i.e., inflow) was scaled to 75% (i.e. 21700 trips per hour) of the original demand, so that the number of vehicles in the virtual queues is small at any point in time during simulation. However, any vehicles that enter the virtual queues due to congestion are still included in the count of the number of vehicles in the network (i.e. the objective function).

To evaluate the added value of the proposed sampling strategy, we embed it within a simulation-based optimization (SO) algorithm. The SO algorithm used is based on the metamodel SO algorithm of Osorio and Bierlaire (2013). A metamodel is an analytical approximation of the simulation-based objective function. This SO algorithm simulates one of two types of points at each iteration: (i) points defined by a user-defined sampling distribution, (ii) points that are solutions to an analytical optimization problem, which is known as the metamodel optimization problem. For more details on the SO algorithm used here, we refer the reader to Appendix D. We consider 4 instances of this SO algorithm that differ in 2 ways: (i) whether or not the information from the analytical model is used to specify the sampling strategy, and (ii) whether or not the information from the analytical model is used to specify the metamodel. These 4 algorithm instances are summarized in Table 4. The first column defines the name of the algorithm instance. Column 2 indicates whether or not the sampling strategy uses information from the analytical model. If it does, then the proposed inverse cdf sampling strategy is used. If it does not, then a uniform sampling strategy is used. The uniform sampling strategy has been traditionally used as part of the SO algorithm Osorio and Bierlaire (2013). Column 3 indicates whether or not the metamodel uses information from the analytical model. If it does, then the metamodel is denoted m and is defined as the sum of a scaled function of f^A and a quadratic polynomial. If it does not then, the metamodel is denoted ϕ and is a quadratic polynomial. These two metamodel definitions have been traditionally used for metamodel SO (see e.g., Osorio and Bierlaire (2013, Eq. 3 and 4)). In addition, the notation of m and ϕ is consistent with our past works (Osorio and Bierlaire 2013, Osorio and Chong 2015, Osorio and Nanduri 2015a,b, Osorio et al. 2017, Zhang et al. 2017). The method Unif-m is that used in our past signal control SO work (Osorio and Bierlaire 2013, Osorio and Chong 2015, Osorio and Nanduri 2015a,b, Osorio et al. 2017), while the method Unif-ϕ served as the benchmark in those past works.

The algorithms of rows 1 and 2 have a common general-purpose metamodel, which does not use the problem-specific information from the analytical model. Thus, their comparison serves to evaluate the added value of complementing a general-purpose SO algorithm with a problem-specific sampling strategy. The algorithms of

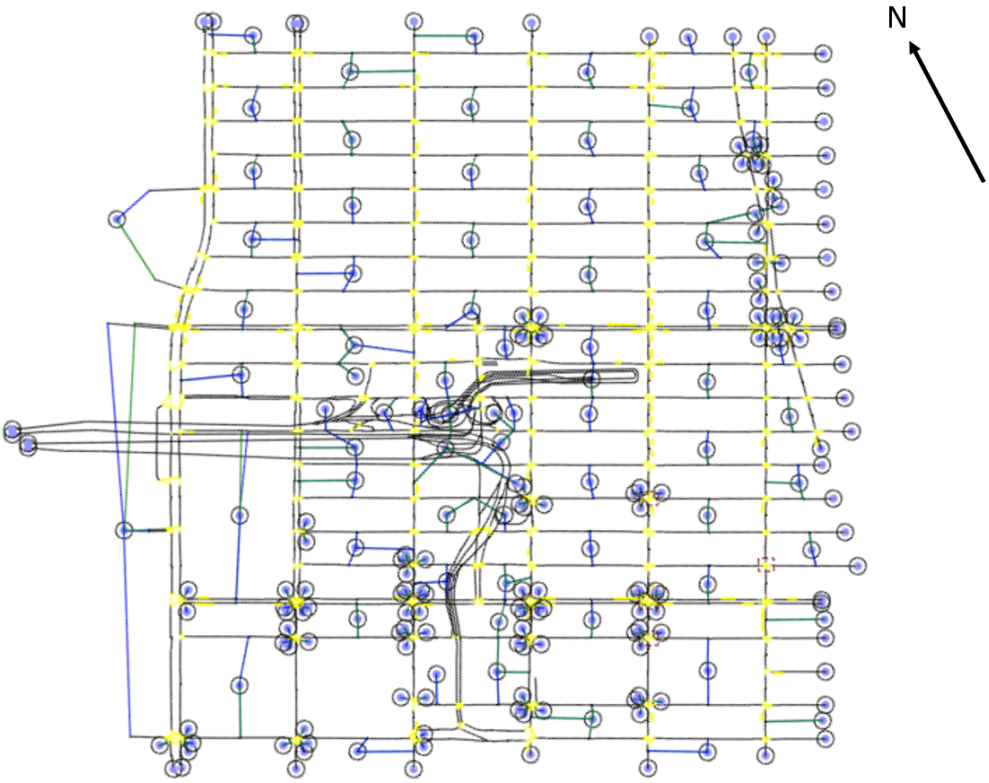


Figure 7: Midtown Manhattan model

rows 3 and 4 have a common metamodel that uses problem-specific information from the analytical model. Thus, their comparison serves to evaluate the added value of using a problem-specific sampling strategy given that the metamodel already has a problem-specific component.

For each algorithm instance, we perform experiments that allocate either 10%, 50% or 100% of the simulation budget to the sampling strategy, while the remaining simulation budget (i.e., 90%, 50% and 0%, respectively) is used to simulate the solutions of the metamodel optimization problems.

We first consider the case of 100%. In this case, there is no metamodel optimization. In other words, the SO algorithm consists only of the sampling strategy. This allows us to directly benchmark the performance of the proposed sampling approach compared to that of a uniform sampling approach. We then consider the cases of 50%, where sampling takes place every other iteration of the SO algorithm, and 10%, where sampling takes place every of 10^{th} iteration. Note that in each iteration, the SO algorithm may only either draw a sample based on the sampling strategy or solve the metamodel optimization problem.

To estimate the objective function, for each signal plan, we take the sample mean from 5 simulation replications. For each replication, we compute the average (over time) number of vehicles in the network during the non-warm-

Table 4: SO Methods

	Sampling distribution based on the analytical model?	Metamodel based on the analytical model?
Unif-ϕ		
Invcdf-ϕ	✓	
Unif-m		✓
Invcdf-m	✓	✓

Table 5: Proportion of Simulation Budget for Sampling

Sampling Proportion	Description
10%	Sampling done in the $(10i-9)^{th}$ iteration, for $i = 1, 2, 3, ...$
50%	Sampling done in the $(2i-1)^{th}$ iteration, for $i = 1, 2, 3, ...$
100%	Only sampling done, no metamodel optimization takes place

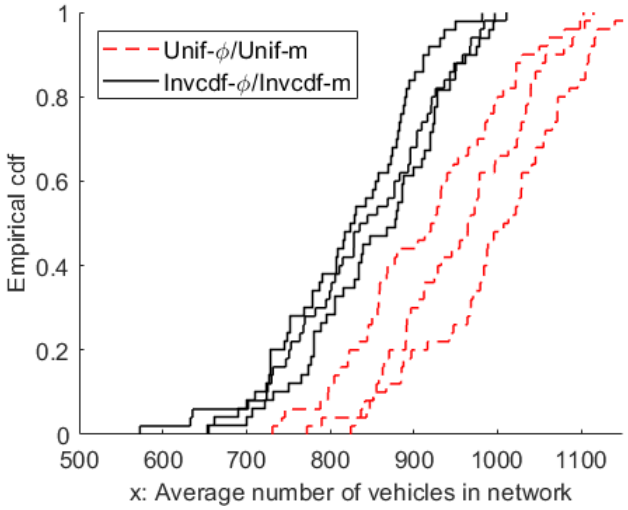


Figure 8: Cdf's of the average number of vehicles in the network achieved by the best signal plans generated by the Unif-ϕ and Invcdf-ϕ methods at 100% sampling.

up period: 5:20pm - 6:00pm. This average is obtained from observations of the number of vehicles in the network collected every minute. We consider a tight computational budget of 250 simulation evaluations, i.e., each algorithm run is terminated once a total of 250 simulation evaluations are carried. Since we use 5 simulation replications per point, this means that the computational budget allows to simulate a total of 50 points. For each experiment (i.e., a given sampling proportion), we consider 3 initial points, which are randomly and uniformly sampled from the feasible region using the code of Stafford (2006). For each initial point, we run each SO algorithm 3 times.

4.2 Numerical results

For each final solution derived by each algorithm run, we evaluate its performance based on 50 simulation replications and then construct the cumulative distribution function (cdf) across these 50 simulation observations. In Figures 8 - 12, the x-axis represents the average number of vehicles in the network, i.e., the simulation-based estimate of the objective function. For a given value along the x-axis, the corresponding value on the y-axis shows the fraction of simulation replications (out of 50) that are smaller than the value on the x-axis. Since the objective is to minimize the expected number of vehicles in the network, the more the cdf of a given signal plan lies to the left of the figure, the better the performance of the underlying signal plan.

Figure 8 shows the cdf's for 100% sampling (i.e., when no metamodel optimization is done). The red dashed line represents the Unif-ϕ method, while the solid line denotes the Invcdf-ϕ method. The Unif-m and Invcdf-m methods are not shown on this plot, because 100% of the simulation budget is allocated to the sampling strategy (i.e., no metamodel optimization is performed). This 100% sampling setting allows us to directly compare between uniform sampling and inverse cdf sampling. In Figure 8, the cdf's of the Invcdf-ϕ method are all to the left of those of Unif-ϕ. This indicates that, for all 3 initial points, Invcdf-ϕ outperforms Unif-ϕ. This shows that the information of the analytical model increases the chance of sampling points with good performance. Thus, there is an added value of using problem-specific analytical information to specify the sampling strategy.

The results of the experiments with sampling proportions of 50% and 10% for the Invcdf-ϕ and Unif-m methods are given in Figure 9. The left column of Figure 9 shows the results for 50% sampling, while the right column displays the results for 10% sampling. Each row of plots considers a given initial point. In Figure 9, Invcdf-ϕ is

Table 6: T-statistics for paired t-test (Invcdf-ϕ vs. Unif-m)

Initial Point	50% Sampling	10% Sampling
1	-0.2412	5.3719
2	**-3.3832**	**-4.0747**
3	**-4.4261**	**-2.3207**

Table 7: T-statistics for paired t-test (Invcdf-m vs. Unif-m)

Initial Point	50% Sampling	10% Sampling
1	-0.3121	**-1.5955**
2	**-2.9563**	**-4.5719**
3	**-2.8764**	**-2.0474**

represented by the black solid lines, and Unif-m is denoted by the blue line with stars.

Figure 9 allows us to compare the performance of the proposed signal plans from an SO method which uses information of the analytical model only in the sampling mechanism (i.e., Invcdf-ϕ) against that which uses the information only in the metamodel optimization (i.e., Unif-m). The plots in Figure 9 show that the signal plans proposed by the Invcdf-ϕ method are similar or better in performance than those proposed by Unif-m.

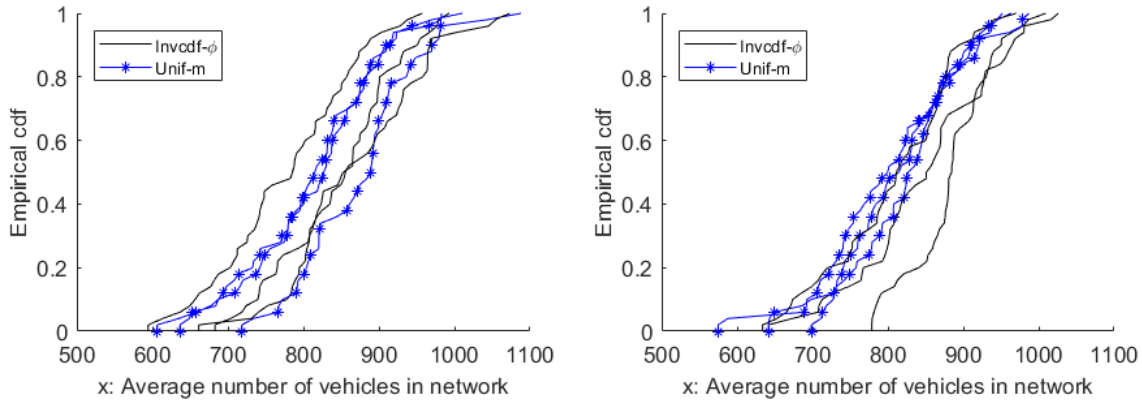
To further determine if the differences in performance of the proposed plans are statistically significant, we performed a one-sided paired t-test based on the average values of the 3 runs for each initial point. Here, the null hypothesis states that the mean of the 50 simulation replications (averaged over the 3 runs for a given initial point) using the signal plans proposed by Invcdf-ϕ is not smaller than that of Unif-m, while the alternative hypothesis states that the mean of the 50 simulation replications using the signal plans proposed by Invcdf-ϕ is smaller than that of Unif-m. The resulting t-statistics are shown in Table 6. Each row considers a given initial point. The second column shows the t-statistics for 50% sampling, while the third column shows the t statistics for 10% sampling. Each t-test is considered at the 10% level of significance, with 49 degrees of freedom, leading to a critical value of -1.299. The t-statistics with values below the critical value are displayed in bold. The results of the t-tests indicate that two of the signal plans proposed by Invcdf-ϕ have better performance than those proposed by Unif-m for both the 50% and 10% sampling settings. This shows that there could be added value to using information from an analytical model in the sampling mechanism, as opposed to the metamodel optimization.

We now consider whether it is beneficial to use the information from the analytical model in both the sampling mechanism and metamodel optimization (i.e., Invcdf-m). We compare the performance of signal plans proposed by Invcdf-m with those proposed by Unif-m and Invcdf-ϕ. For clarity purposes, the results are plotted on two separate figures. Figure 10 compares Invcdf-m and Unif-m, while Figure 11 compares Invcdf-m and Invcdf-ϕ.

The plots in Figure 10 show that the plans proposed by the Invcdf-m method perform similar to or better than those proposed by Unif-m. The difference in performance is more pronounced for the 10% sampling setting. Similar to the comparison between Invcdf-ϕ and Unif-m, we perform one-sided, paired t-tests to determine if the signal plan proposed by Invcdf-m are significantly better than those proposed by Unif-m. The t-statistics are displayed in Table 7. The critical value is -1.299, and t-statistics with values below the critical value are shown in bold. The results in Table 7 show that the plans proposed by Invcdf-m outperform those proposed by Unif-m for two out of three initial points for the 50% sampling setting, and for all initial points for the 10% sampling setting. This shows that there is added value to using the information from the analytical model in the sampling mechanism, in addition to the metamodel optimization.

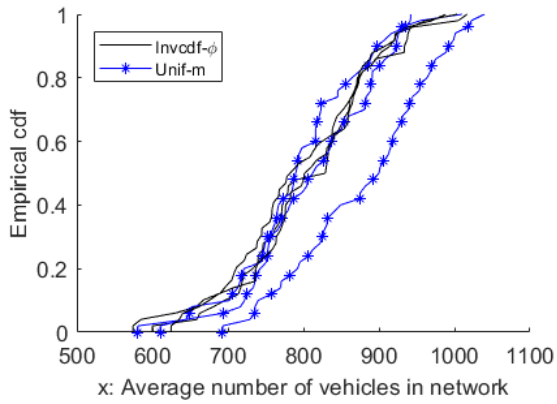
To see if there is added value to using the information from the analytical model in the metamodel optimization, in addition to the sampling mechanism, we consider the plots in Figure 11, and the one-sided, paired t-tests in Table 8. The plots in Figure 11 indicate that the signal plans proposed by Invcdf-m have similar performance to those proposed by Invcdf-ϕ. This is confirmed by the results of the t-tests in Table 8, which shows that there is no significant improvement in performance of the signal plans proposed by Invcdf-m, compared with Invcdf-ϕ, for all the initial points in the 50% sampling setting and two out of three initial points in the 10% sampling setting.

Figure 12 summarizes the information shown in Figures 9 - 11. Each cdf curve is constructed using all the observations across all three initial points (i.e., each curve consists of $50 \times 3 \times 3 = 450$ observations). For the 50% sampling setting, Figure 12a shows that the signal plans proposed by Invcdf-ϕ and Invcdf-m outperform those proposed by Unif-m. The plans proposed by Invcdf-ϕ and Invcdf-m have similar performance. For the 10% sampling setting, Figure 12b shows that the signal plans proposed by Invcdf-m again outperform those

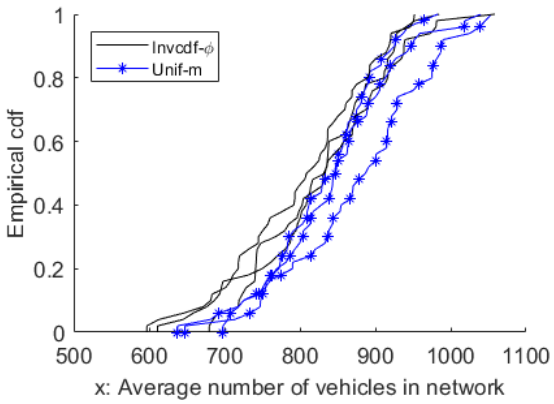


(a) 50% Sampling - Initial Point 1

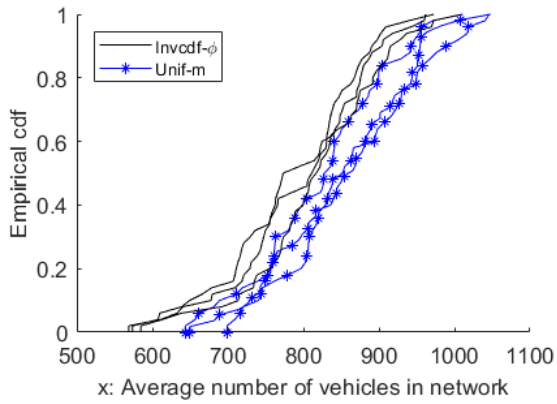
(b) 10% Sampling - Initial Point 1



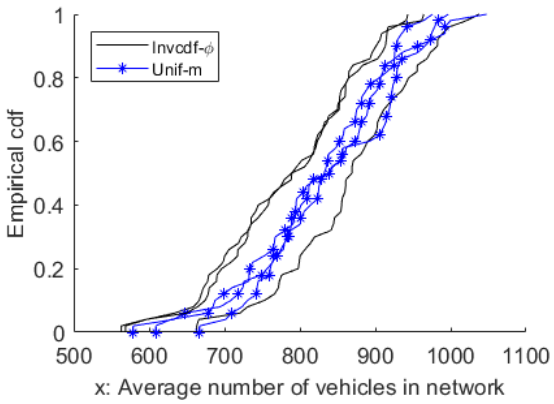
(c) 50% Sampling - Initial Point 2



(d) 10% Sampling - Initial Point 2

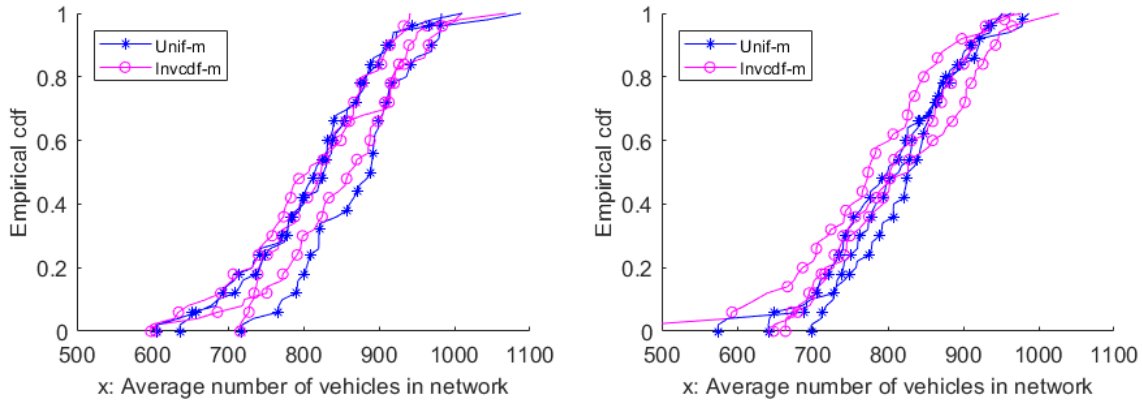


(e) 50% Sampling - Initial Point 3



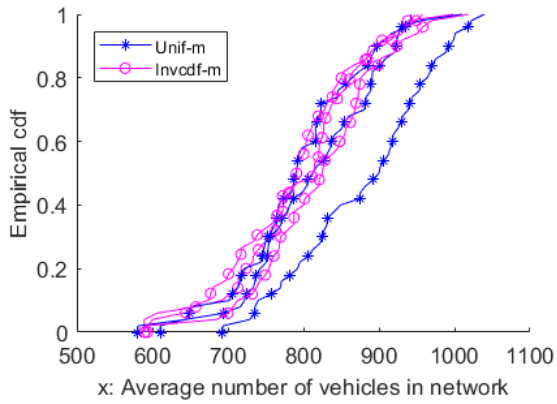
(f) 10% Sampling - Initial Point 3

Figure 9: Cdf's of the average number of vehicles in the network for the Invcdf-ϕ and Unif-m methods using the 50% and 10% sampling settings and considering 3 random initial points.

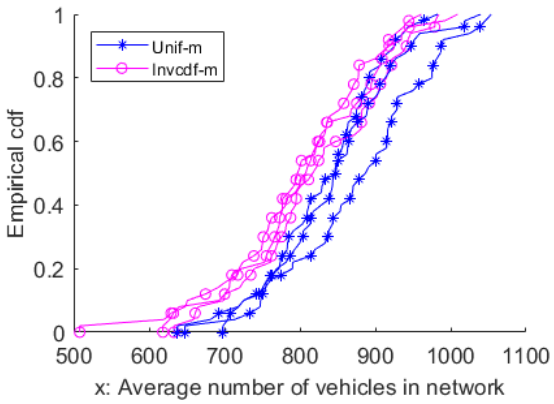


(a) 50% Sampling - Initial Point 1

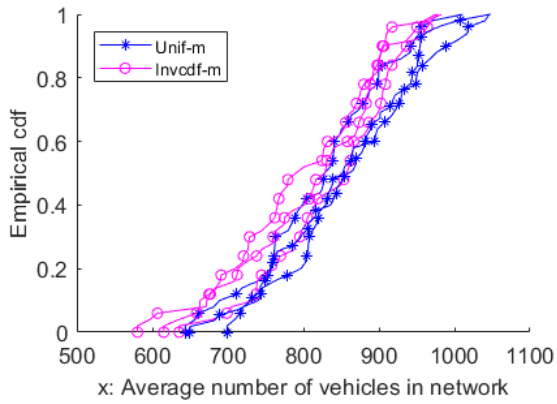
(b) 10% Sampling - Initial Point 1



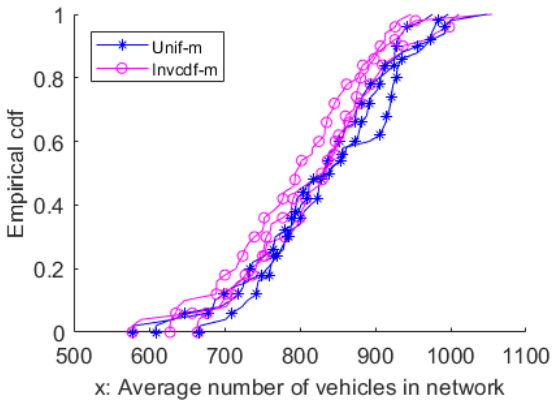
(c) 50% Sampling - Initial Point 2



(d) 10% Sampling - Initial Point 2

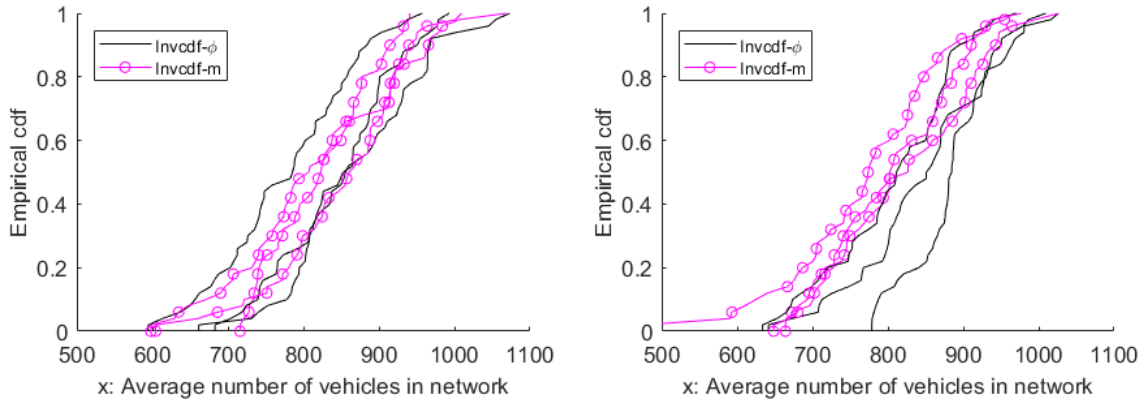


(e) 50% Sampling - Initial Point 3



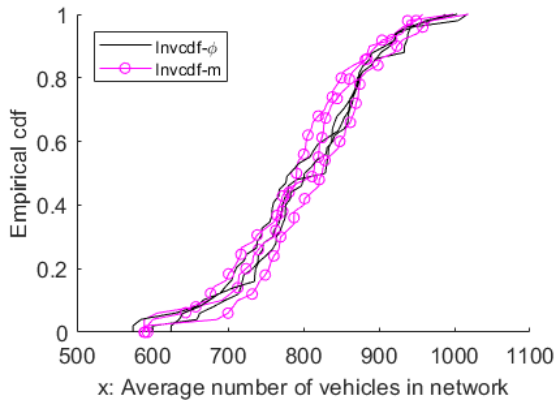
(f) 10% Sampling - Initial Point 3

Figure 10: Cdf's of the average number of vehicles in the network for the Invcdf-m and Unif-m methods using the 50% and 10% sampling settings and considering 3 random initial points.

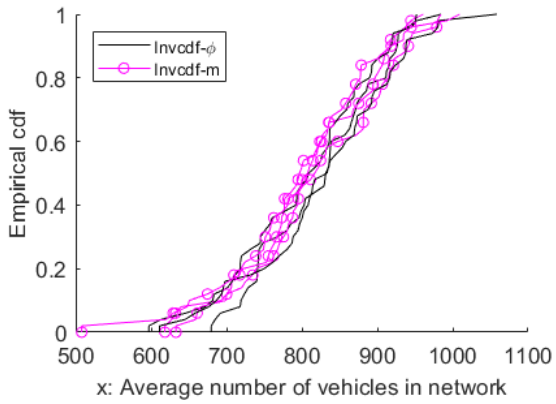


(a) 50% Sampling - Initial Point 1

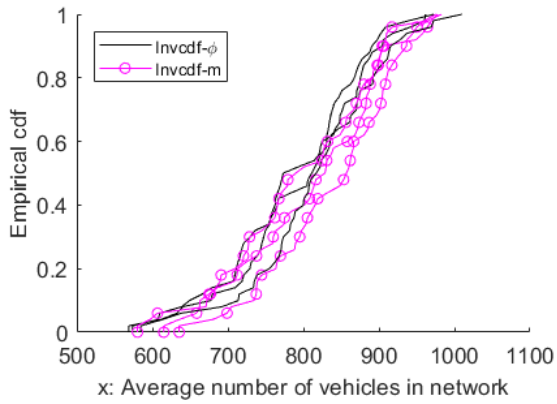
(b) 10% Sampling - Initial Point 1



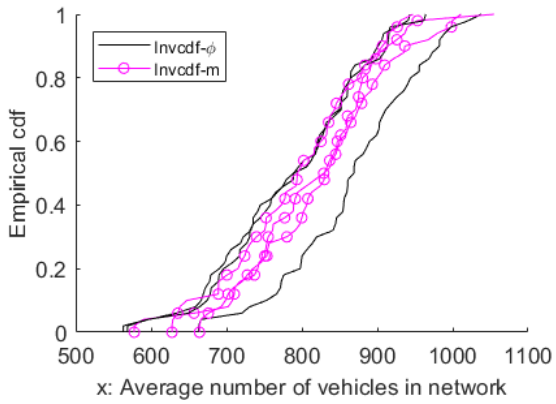
(c) 50% Sampling - Initial Point 2



(d) 10% Sampling - Initial Point 2



(e) 50% Sampling - Initial Point 3

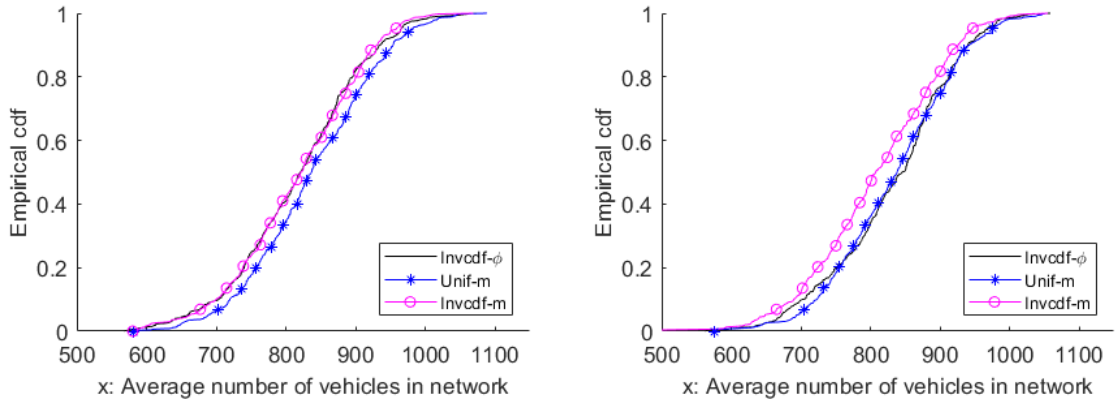


(f) 10% Sampling - Initial Point 3

Figure 11: Cdf's of the average number of vehicles in the network for the Invcdf-m and Invcdf-ϕ methods using the 50% and 10% sampling settings and considering 3 random initial points.

Table 8: T-statistics for paired t-test (Invcdf-m vs. Invcdf-ϕ)

Initial Point	50% Sampling	10% Sampling
1	-0.0902	**-5.8983**
2	-0.2877	-0.8306
3	2.3148	-0.2485



(a) 50% Sampling (b) 10% Sampling

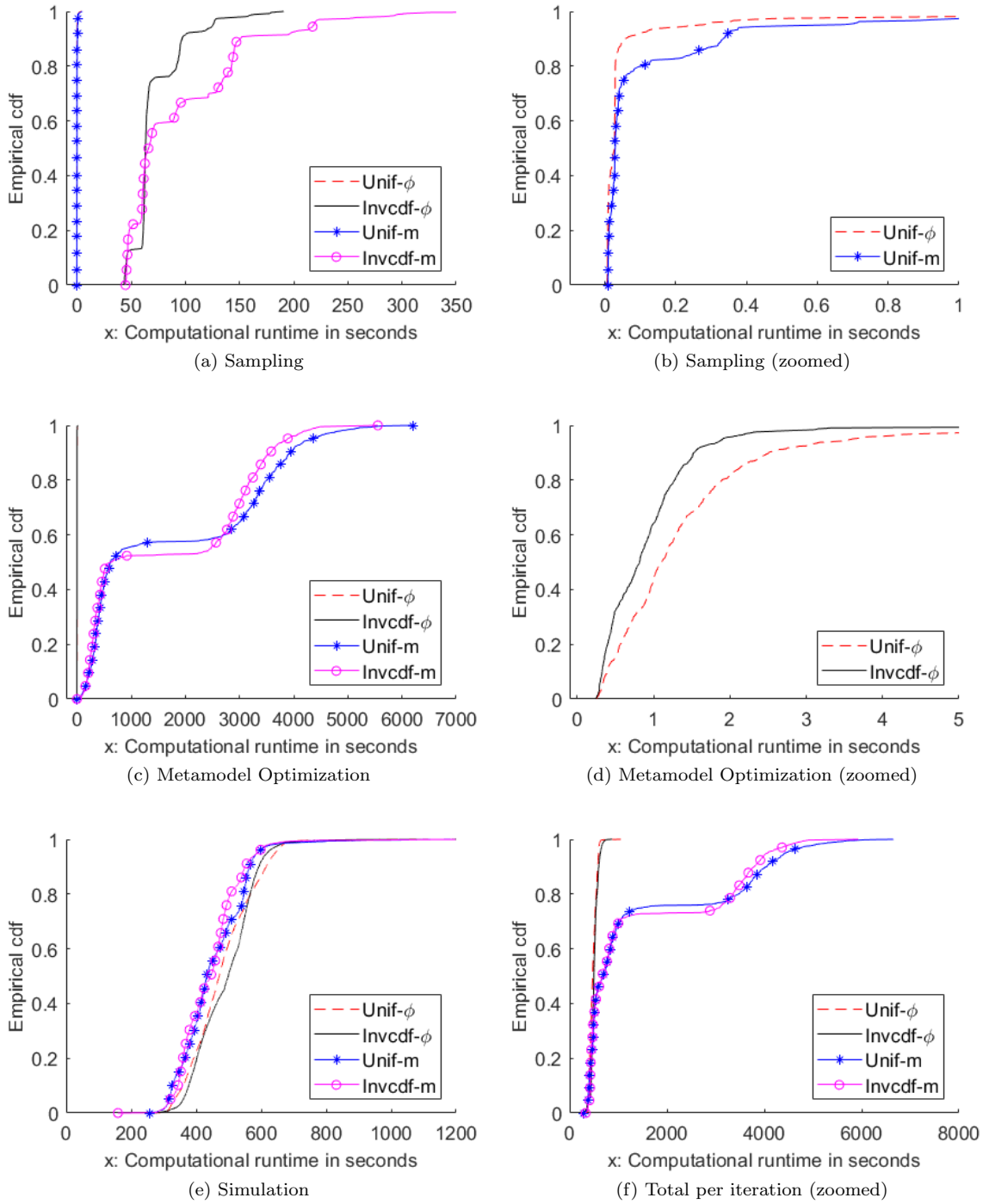
Figure 12: Cdf's of the average number of vehicles in the network after aggregating across initial points.

proposed by Unif-m, as well as those identified by Invcdf-ϕ. Here, the performance of the signal plans proposed by Invcdf-ϕ and Unif-m are similar. Thus, the results suggest that there is limited value in using information from the analytical model in the metamodel optimization, in addition to the sampling mechanism, especially when the sampling proportion of the computational budget is high. When the sampling proportion is low, using the information in the metamodel optimization, in addition to the sampling mechanism, might still be helpful.

4.3 Computational efficiency

Each iteration of the SO algorithm involves three computationally intensive tasks: (i) solving the metamodel optimization problem; (ii) sampling following the sampling strategy; (iii) using the traffic simulator to evaluate the new points derived from (i) or (ii). Here, we compare the computational runtime required for these three tasks, with a total of 270 observations for sampling, 630 observations for metamodel optimization and 4500 observations for simulations collected while running the experiments for Section 4.2. The experiments were run on a laptop computer with an Intel Core i7 (2.30 GHz) CPU and 12 GB of RAM. Figure 13 displays the cdf's for the computational runtimes for the respective tasks. The x-axis represents the computational runtime required for the task in seconds. For a given value along the x-axis, the corresponding value on the y-axis shows the fraction of observations that had runtimes smaller than the value on the x-axis. The red dashed line represents the Unif-ϕ method, the black solid line represents the Invcdf-ϕ method, the blue line with stars denotes the Unif-m method and the magenta line with circles denotes the Invcdf-m method.

Figure 13a shows the runtimes needed for sampling. Figure 13b shows a zoomed-in section of the plot so that the curves for Unif-ϕ and Unif-m can be seen in more detail. The Unif-ϕ (red dashed line) and Unif-m (blue line with stars) methods, which make use of uniform sampling, has an overall mean sampling runtime of 0.1 seconds. On the other hand, the Invcdf-ϕ (black solid line) and Invcdf-m (magenta line with circles) methods use inverse cdf sampling, which has an overall mean runtime of 84 seconds. Thus, drawing a sample using the inverse cdf sampling mechanism takes around 80 seconds longer than drawing a sample using uniform sampling. The step-like appearance of the empirical cdf curves for the Invcdf-ϕ and Invcdf-m methods is the result of having to resample components which do not satisfy the constraints (see Step 2d in Algorithm 1); in some cases, some components have to be resampled multiple times. In Figure 13a, the difference between the curves for Invcdf-ϕ and Invcdf-m can be attributed to the randomness during sampling, since the inverse cdf sampling mechanism used in the two methods are the same, and does not depend on the metamodel in use or previous simulation evaluations.



(a) Sampling

(b) Sampling (zoomed)

(c) Metamodel Optimization

(d) Metamodel Optimization (zoomed)

(e) Simulation

(f) Total per iteration (zoomed)

Figure 13: Computational runtimes for (a, b) sampling (per sample point), (c, d) metamodel optimization, (e) simulation and (f) total computational runtime per iteration.

Figure 13c illustrates the runtimes required to solve the metamodel optimization problem. Figure 13d shows a zoomed-in section of the plot so that the curves for Unif-ϕ and Invcdf-ϕ can be seen in more detail. The Unif-ϕ and Invcdf-ϕ methods, which make use of the ϕ-metamodel for optimization, tend to have very fast mean runtimes of 1.4 seconds and 1.0 seconds respectively. Figure 13d also shows that metamodel optimization for the Invcdf-ϕ method is faster than the Unif-ϕ method. On the other hand, the Unif-m and Invcdf-m methods, which use the m-metamodel for optimization, have mean runtimes of 1764 seconds and 1694 seconds respectively. The mean runtimes for metamodel optimization suggest that the metamodel optimization is faster when paired with inverse cdf sampling, as opposed to when paired with uniform sampling. This is observed for both the ϕ-metamodel and m-metamodel. However, based on Figure 13c, it can also be seen that metamodel optimization for the Unif-m and Invcdf-m methods have runtimes that seem to be distributed in a bimodal fashion. Metamodel optimization steps that took less than 1500 seconds have mean runtimes of 427 seconds and 343 seconds respectively for the Unif-m and Invcdf-m methods, while metamodel optimization steps that took 1500 seconds or longer have mean runtimes of 3570 seconds and 3189 seconds respectively for the Unif-m and Invcdf-m methods.

Figure 13e shows that the runtimes needed for simulation are quite similar for all the SO methods. Each simulation run has a mean runtime of 466 seconds. Finally, Figure 13f shows the total computational runtimes needed per iteration of the SO algorithm. The runtime for a given iteration is computed by summing the mean of the simulation runtimes (since the simulation replications can be run in parallel), and the runtime for sampling or metamodel optimization (depending on whether it is a sampling iteration or metamodel optimization iteration). The mean runtimes per iteration required by the Unif-ϕ and Invcdf-ϕ methods are 481 seconds and 509 seconds respectively, while the mean runtimes of the Unif-m and Invcdf-m methods are much longer, with mean runtimes of 1450 seconds and 1434 seconds respectively. This shows that the Unif-ϕ and Invcdf-ϕ methods are much faster than the Unif-m and Invcdf-m methods. The runtimes of the Unif-m and Invcdf-m methods are again bimodal due to the metamodel optimization step. Iterations which took less than 2000 seconds have a mean runtime of 630 seconds and 601 seconds for the Unif-m and Invcdf-m methods respectively, while iterations which took 2000 seconds or longer have a mean runtime of 4024 seconds and 3690 seconds for the Unif-m and Invcdf-m methods respectively.

Based on the results shown in Figures 13a and 13c, the inverse cdf sampling strategy takes about 80 seconds longer than uniform sampling to generate a sample point, and optimizing the m-metamodel takes up to 3500 seconds longer than optimizing the ϕ-metamodel. Given that the solutions obtained when using information from the analytical model in the sampling mechanism (i.e., Invcdf-ϕ) have similar or better performance (in terms of objective function value) to those obtained using the information in the metamodel optimization (i.e., Unif-m), it would be more beneficial to use the information in the sampling mechanism than in the metamodel optimization since Invcdf-ϕ requires significantly less computational runtime. Figure 13f confirms that the Invcdf-ϕ method requires much less computational runtime (509 seconds per iteration on average) compared to the Unif-m method (1450 seconds per iteration on average). This corresponds to an 65% reduction in computational runtime in some iterations, while achieving similar or better performance when using the Invcdf-ϕ method. Furthermore, there is no clear advantage to using the information from the analytical model in the metamodel optimization, in addition to the sampling mechanism (i.e., Invcdf-m), especially when the sampling proportion of the computational budget is high (i.e., 50% sampling). In the 50% sampling setting, the signal plans proposed by Invcdf-ϕ and Invcdf-m have no significant difference. However, Invcdf-m would require 1778 seconds on average to generate a sample point and optimize once (i.e., 84 seconds to sample by inverse cdf sampling and 1694 seconds to optimize the metamodel), compared to 85 seconds for Invcdf-ϕ (i.e., 84 seconds to sample by inverse cdf sampling and 1 second to optimize the metamodel). This represents a 2100% increase in computational cost.

Thus, when comparing the performance (Figure 13f) for the different methods, we see that using the analytical model in the sampling mechanism (Invcdf-ϕ) leads to solutions similar or better performance than using the analytical model in the metamodel optimization (Unif-m), or both (Invcdf-m). However, using the analytical mode in the sampling mechanism helps to significantly reduce the amount of computational runtime.

5 Conclusion

This paper answers the question of whether problem-specific structural information from analytical models should be used in the sampling mechanism, optimization algorithm, or both, when attempting to solve a transportation SO problem. We also present an alternative sampling strategy for efficient stochastic simulation-based transportation optimization. It puts forward the idea of making use of an analytical model, which contains structural information of the traffic network, in the sampling mechanism of the optimization framework. This increases the

probability of sampling points with good performance, while still having a non-zero chance of sampling in other regions for exploration. Increasing the chances of sampling good points could help to reduce the computational runtime required to obtain a good solution.

Evaluating this approach on a fixed-time signal control problem of Midtown Manhattan, there have been promising results for tackling high-dimensional transportation optimization problems. The results show that placing the analytical model in the sampling mechanism and using an inverse transform method was able to achieve similar or better performance to the traditional method of using the queueing model in the metamodel for optimization, coupled with uniform sampling. This was in addition to reducing the overall amount of computational runtime required. Thus, the inverse cdf sampling strategy can be used as part of an optimization framework to quickly and efficiently identify solutions with good performance, when there exists prior information on the structure of the problem.

While the use of inverse cdf sampling has shown promising results, there are some limitations to its use. First, since the objective function has to be integrated in order for the inverse cdf sampling to work, there is a need to find an objective function whose integral has a closed analytical form. Ideally, the objective function should be integrable with respect to the decision variables. However, as shown in this paper, the objective function can also be integrated with respect to another variable, as long as it can eventually be transformed to the decision variable.

Another area that can be studied more is the order in which the conditionals (as shown in Eq. (12) - (14)) are taken. It is possible that the order of the conditionals have an implication on the outcome of the sampling. Although randomizing the order for sampling with the toy network did not produce any noticeable changes, this cannot be said to be true for all other networks.

Moving forward, we will investigate possible ways to adapt the inverse cdf sampling method into a sequential design. This could be done in a way similar to the expected improvement criterion in the Efficient Global Optimization algorithm (Jones et al. 1998). The idea is to take into account points that have already been evaluated by the simulator, so as to update the analytical model. Based on the updated analytical model, the sampling mechanism would then be able to pick out the points with better performance with greater probability. At the same time, it would also be useful to further incentivize the sampling mechanism to sample points far away from previously evaluated points. This is especially so after finding a good solution, so that there would be a better exploration-exploitation balance, which could lead to even better solutions.

Acknowledgement

The authors would like to thank the New York City Department of Transportation (NYCDOT), in particular Jingqin Gao, Mohamad Talas and Michael Marsico, for providing us with the simulation model for Midtown Manhattan. Timothy Tay would also like to thank the Agency for Science, Technology and Research (A*STAR) Singapore for funding his work.

A List of Notation Used

f	objective function (expected number of vehicles in the road network);
f^A	approximate analytical expression of the objective function;
F	random variable denoting the number of vehicles in the road network;
x_j	green split of signal phase j (decision variable);
$\mathbf{z}$	vector of endogenous simulation variables (e.g, queue-lengths, vehicular speeds per lane, vehicular densities per lane);
	Exogenous problem parameters:
c_ℓ	cycle time of intersection ℓ;
d_ℓ	fixed cycle time of intersection ℓ;
e_i	ratio of fixed green time to cycle time of signalized link i;
$\mathbf{x}^{LB}$	vector of minimal green splits;
$\boldsymbol{\theta}$	vector of exogenous parameters;
n	total number of signal phases to be optimized (i.e., dimension of the decision vector $\mathbf{x}$);
$\mathcal{I}$	set of intersection indices;
$\mathcal{P}_1(\ell)$	set of phase indices of intersection ℓ;
	Exogenous parameters of the analytical model:
γ_i	external arrival rate at queue i;
m	total number of queues in the network;
k_i	space capacity of queue i in terms of number of vehicles;
p_{ij}	transition probability from queue i to queue j;
s	saturation flow rate [veh/h];
$\mathcal{M}$	set of all queues;
$\mathcal{L}$	set of indices of the signalized lanes;
$\mathcal{U}_i$	set of upstream queues of queue i;
$\mathcal{D}_i$	set of downstream queues of queue i;
$\mathcal{P}_2(i)$	set of phase indices of lane i;
$\mathcal{P}_3(i)$	index of intersection that queue i leads to;
$\mathcal{P}_4(i)$	set of phase indices at the intersection that queue i leads to, but not including the indices of phases which are green in favor of vehicles in queue i;
	Endogenous parameters of the analytical model:
ρ_i	traffic intensity;
λ_i	arrival rate;
N_i	total number of vehicles in queue i;
$P(N_i = k_i)$	probability of queue i being full, also known as the blocking or spillback probability;
μ_i	service rate of link i;
	Analytical sampling distribution notation:
$\hat{\rho}_i$	upper bound on the possible values of ρ_i
$g_{\rho_i,\ldots,\rho_j}$	joint pdf of ρ_i to ρ_j;
$G_{\rho_i,\ldots,\rho_j}$	joint cdf of ρ_i to ρ_j;
$G_{\rho_i\|\rho_j}$	conditional cdf of ρ_i conditional on ρ_j;
κ_0	normalization constant to ensure g integrates to 1;
κ_1	upper bound of f^A;
u_i	realization of a univariate standard uniform random variable.

B Derivation of sampling distributions and transformation of $\boldsymbol{\rho}$ to x based on the analytical network model (8)

This appendix details the derivations of the analytical expressions for the marginal and conditional cdf's in Eq. (15)-(22), as well as the transformation of $\boldsymbol{\rho}$ to $\mathbf{x}$ (Section 2.5.1).

B.1 Derivation of the joint cdf $G_{\rho_1,\ldots,\rho_m}(\rho_1,\ldots,\rho_m)$

In order to obtain the analytical expressions of the marginal and conditional cdf's (Eq. (15)-(22)), we first evaluate the integral in Eq. (11) to get the analytical expression of the joint cdf. From Eq. (11), we have:

$$G_{\rho_1,\ldots,\rho_m}(\rho_1,\ldots,\rho_m) = \frac{1}{\kappa_0}\int_0^{\boldsymbol{\rho}} \kappa_1 - f^A(\tilde{\rho}_1,\ldots,\tilde{\rho}_m)d\tilde{\boldsymbol{\rho}} \tag{B.1}$$

$$= \frac{1}{\kappa_0}\Big[\int_0^{\boldsymbol{\rho}} \kappa_1 d\tilde{\boldsymbol{\rho}} - \int_0^{\boldsymbol{\rho}} f^A(\tilde{\rho}_1,\ldots,\tilde{\rho}_m)d\tilde{\boldsymbol{\rho}}\Big] \tag{B.2}$$

Computing the first integral in Eq. (B.2) gives:

$$\int_0^{\boldsymbol{\rho}} \kappa_1 d\tilde{\boldsymbol{\rho}} = \kappa_1\rho_1\ldots\rho_m \tag{B.3}$$

$$= \kappa_1\prod_{i=1}^{m}\rho_i \tag{B.4}$$

Using the expressions for f^A in Eq. (6), the second integral in Eq. (B.2) can be computed as:

$$\int_0^{\boldsymbol{\rho}} f^A(\tilde{\rho}_1,\ldots,\tilde{\rho}_m)d\tilde{\boldsymbol{\rho}} = \int_0^{\boldsymbol{\rho}}\sum_{i=1}^{m}\mathbb{E}\big[N_i(\tilde{\rho}_i)\big]d\tilde{\boldsymbol{\rho}} \tag{B.5}$$

$$= \sum_{i=1}^{m}\int_0^{\boldsymbol{\rho}}\mathbb{E}\big[N_i(\tilde{\rho}_i)\big]d\tilde{\boldsymbol{\rho}} \tag{B.6}$$

$$= \sum_{i=1}^{m}\Big[\int_0^{\rho_i}\mathbb{E}\big[N_i(\tilde{\rho}_i)\big]d\tilde{\rho}_i\Big]\Big(\prod_{j\neq i}\rho_j\Big) \tag{B.7}$$

The transition from Eq. (B.5) to Eq. (B.6) makes use of the fact that $N_i(\rho_i)$ is a function of ρ_i only. Similarly, since $N_i(\rho_i)$ does not depend on $\rho_j, j\neq i$, integrating with respect to $\rho_j, j\neq i$, produces Eq. (B.7). This simplifies the integral to a univariate integral. We define $h(\rho_i)$ to be the integral in Eq. (B.7) and evaluate it:

$$h(\rho_i) = \int_0^{\rho_i}\mathbb{E}\big[N_i(\tilde{\rho}_i)\big]d\tilde{\rho}_i \tag{B.8}$$

$$= \int_0^{\rho_i}\frac{\tilde{\rho}_i}{1-\tilde{\rho}_i} - \frac{(k_i+1)\tilde{\rho}_i^{k_i+1}}{1-\tilde{\rho}_i^{k_i+1}}d\tilde{\rho}_i \tag{B.9}$$

$$= -\rho_i - \log(1-\rho_i) - \int_0^{\rho_i}\frac{(k_i+1)\tilde{\rho}_i^{k_i+1}}{1-\tilde{\rho}_i^{k_i+1}}d\tilde{\rho}_i \tag{B.10}$$

$$= -\rho_i - \log(1-\rho_i) + \rho_i\log\big(1-\rho_i^{k_i+1}\big) - \int_0^{\rho_i}\log\big(1-\tilde{\rho}_i^{k_i+1}\big)d\tilde{\rho}_i \tag{B.11}$$

where Eq. (B.9) uses the expression for f^A in Eq. (7). Integrating the first term in Eq. (B.9) leads to Eq. (B.10). Finally, Eq. (B.11) is obtained by evaluating the integral in Eq. (B.10) by parts. Note that the term in the integral

in Eq. (B.11) can be Taylor expanded before being integrated, giving:

$$\int_0^{\rho_i} \log\left(1-\tilde{\rho}_i^{k_i+1}\right) d\tilde{\rho}_i = \int_0^{\rho_i} \Big[-\sum_{\alpha=1}^{\infty} \frac{\left(\tilde{\rho}_i^{k_i+1}\right)^{\alpha}}{\alpha} \Big] d\tilde{\rho}_i \tag{B.12}$$

$$= -\sum_{\alpha=1}^{\infty} \frac{1}{\alpha} \int_0^{\rho_i} \tilde{\rho}_i^{\alpha k_i+\alpha} d\tilde{\rho}_i \tag{B.13}$$

$$= -\sum_{\alpha=1}^{\infty} \frac{\rho_i^{\alpha k_i+\alpha+1}}{\alpha(\alpha k_i+\alpha+1)} \tag{B.14}$$

where Eq. (B.12) uses the Taylor expansion of the term in the integral. The independence of α and ρ_i allows the order of the summation and integral to be interchanged in Eq. (B.13). Finally, the evaluation of the integral leads to the expression in Eq. (B.14). Inserting Eq. (B.14) into Eq. (B.11) gives the final expression of $h(\rho_i)$ as shown in Eq. (21):

$$h(\rho_i) = -\rho_i - \log(1-\rho_i) + \rho_i \log(1-\rho_i^{k_i+1}) + \sum_{\alpha=1}^{\infty} \frac{\rho_i^{\alpha k_i+\alpha+1}}{\alpha(\alpha k_i+\alpha+1)} \quad \forall i \in \mathcal{M} \tag{B.15}$$

Putting together Eq. (B.2), (B.4), (B.7) and (B.15), the analytical expression for the joint cdf is:

$$G_{\rho_1,\ldots,\rho_m}(\rho_1,\ldots,\rho_m) = \frac{1}{\kappa_0}\Big[\kappa_1 \prod_{i=1}^{m} \rho_i - \sum_{i=1}^{m} h(\rho_i)\Big(\prod_{j\neq i}^{m} \rho_j\Big)\Big] \tag{B.16}$$

B.2 Derivation of the marginal cdf $G_{\rho_1}(\rho_1)$ (Eq. (16))

The marginal cdf $G_{\rho_1}(\rho_1)$ is used in Step 1 of Algorithm 1 to obtain the sample value for ρ_1. It can be derived from the joint cdf (Eq. (B.16)) by marginalizing the variables ρ_i for $i = 2,\ldots,m$ out of the joint cdf. Since the support of the joint cdf is taken to be $[\mathbf{0}, \hat{\boldsymbol{\rho}}]$ (Section 2.5.2), this can be done by setting $\rho_i = \hat{\rho}_i$ for $i = 2,\ldots,m$ (see Evans and Rosenthal (2004, Theorem 2.7.3)). The resulting expression for $G_{\rho_1}(\rho_1)$ is then derived to give Eq. (16), as follows:

$$G_{\rho_1}(\rho_1) = G_{\rho_1,\rho_2,\ldots,\rho_m}(\rho_1,\hat{\rho}_2,\ldots,\hat{\rho}_m) \tag{B.17}$$

$$= \frac{1}{\kappa_0}\Big[\kappa_1\rho_1 \prod_{i=2}^{m} \hat{\rho}_i - h(\rho_1)\prod_{i=2}^{m}\hat{\rho}_i - \sum_{i=2}^{m} h(\hat{\rho}_i)\Big(\rho_1 \prod_{j=2,j\neq i}^{m} \hat{\rho}_j\Big)\Big] \tag{B.18}$$

$$= \frac{1}{\kappa_0}\Big[\big(\kappa_1\rho_1 - h(\rho_1)\big)\prod_{i=2}^{m}\hat{\rho}_i - \rho_1 \sum_{i=2}^{m} h(\hat{\rho}_i)\Big(\prod_{j=2,j\neq i}^{m} \hat{\rho}_j\Big)\Big] \tag{B.19}$$

$$= \frac{1}{\kappa_0}\Big[\big(\kappa_1\rho_1 - h(\rho_1)\big)\prod_{i=2}^{m}\hat{\rho}_i - \rho_1 r_1\Big] \tag{B.20}$$

where Eq. (B.18) is derived from Eq. (B.16) by setting $\rho_i = \hat{\rho}_i$ for $i = 2,\ldots,m$. Eq. (B.19) and (B.20) are then obtained by factorizing and tidying up Eq. (B.18). The constant r_1 is defined in Eq. (20) and (B.32), and helps to simplify the expression (B.20).

B.3 Derivation of conditional cdf $G_{\rho_i|\rho_1,\ldots,\rho_{i-1}}(\rho_i|\rho_1,\ldots,\rho_{i-1})$ (Eq. (18))

The conditional cdf $G_{\rho_i|\rho_1,\ldots,\rho_{i-1}}(\rho_i|\rho_1,\ldots,\rho_{i-1})$ is used in Step 2c in Algorithm 1 to get the sample value for ρ_i, conditioned on the components that have been sampled. The expression for the conditional cdf given by Eq. (18)

can be derived by first noting that

$$
\begin{aligned}
&G_{\rho_i|\rho_1,\ldots,\rho_{i-1}}(\rho_i|\rho_1,\ldots,\rho_{i-1}) \\
&\quad = g_{\rho_i|\rho_1,\ldots,\rho_{i-1}}(P_i \le \rho_i|\rho_1,\ldots,\rho_{i-1}) && \text{(B.21)} \\
&\quad = \frac{g_{\rho_i,\rho_1,\ldots,\rho_{i-1}}(P_i \le \rho_i,\rho_1,\ldots,\rho_{i-1})}{g_{\rho_1,\ldots,\rho_{i-1}}(\rho_1,\ldots,\rho_{i-1})} && \text{(B.22)} \\
&\quad = \frac{1}{g_{\rho_1,\ldots,\rho_{i-1}}(\rho_1,\ldots,\rho_{i-1})}\int_0^{\rho_i} g_{\rho_i,\rho_1,\ldots,\rho_{i-1}}(\tilde{\rho}_i,\rho_1,\ldots,\rho_{i-1})d\tilde{\rho}_i && \text{(B.23)} \\
&\quad = \frac{1}{g_{\rho_1,\ldots,\rho_{i-1}}(\rho_1,\ldots,\rho_{i-1})}\left[\frac{d}{d\rho_{i-1}}\int_0^{\rho_{i-1}}\cdots\right. \\
&\qquad \left.\cdots\frac{d}{d\rho_1}\int_0^{\rho_1}\int_0^{\rho_i} g_{\rho_i,\rho_1,\ldots,\rho_{i-1}}(\tilde{\rho}_i,\tilde{\rho}_1,\ldots,\tilde{\rho}_{i-1})d\tilde{\rho}_i d\tilde{\rho}_1\ldots d\tilde{\rho}_{i-1}\right] && \text{(B.24)} \\
&\quad = \frac{1}{g_{\rho_1,\ldots,\rho_{i-1}}(\rho_1,\ldots,\rho_{i-1})}\left[\frac{d^{i-1}}{d\rho_1\ldots d\rho_{i-1}}\int_0^{\rho_{i-1}}\cdots\right. \\
&\qquad \left.\cdots\int_0^{\rho_1}\int_0^{\rho_i} g_{\rho_i,\rho_1,\ldots,\rho_{i-1}}(\tilde{\rho}_i,\tilde{\rho}_1,\ldots,\tilde{\rho}_{i-1})d\tilde{\rho}_i d\tilde{\rho}_1\ldots d\tilde{\rho}_{i-1}\right] && \text{(B.25)} \\
&\quad = \frac{1}{g_{\rho_1,\ldots,\rho_{i-1}}(\rho_1,\ldots,\rho_{i-1})}\left[\frac{d^{i-1}G_{\rho_i,\rho_1,\ldots,\rho_{i-1}}(\rho_i,\rho_1,\ldots,\rho_{i-1})}{d\rho_1\ldots d\rho_{i-1}}\right] && \text{(B.26)}
\end{aligned}
$$

where P_i is the random variable representing the sampled values of ρ_i. Eq. (B.21) makes use of the definition of cdf's (Evans and Rosenthal 2004, Definition 2.5.1) to represent the conditional cdf in terms of the conditional pdf. The transition from Eq. (B.21) to (B.22) uses the definition of conditional pdf's (Evans and Rosenthal 2004, Definition 2.8.3). The integral in Eq. (B.23) can be integrated with respect to ρ_1 and then differentiated with respect to ρ_1 again with no net effect, because of the Leibniz's integral rule (Abramowitz and Stegun 1964, Eq. (3.3.7)):

$$\frac{d}{dc}\int_{a(c)}^{b(c)} f(x,c)dx = \int_{a(c)}^{b(c)}\frac{\partial}{\partial c}f(x,c)dx + f(b,c)\frac{db}{dc} - f(a,c)\frac{da}{dc} \tag{B.27}$$

The first and third term on the right hand side of Eq. (B.27) would be zero, leaving just the second term. This is repeated with ρ_j for $j = 2,\ldots,i-1$ to give Eq. (B.24). The order of the integrals and derivatives can be switched since $\rho_1,\ldots,\rho_{i-1}$ are assumed independent for the purpose of sampling. Finally, Eq. (B.26) is obtained by making use of the fact that the integral of the joint pdf is the joint cdf of $\rho_1,\ldots,\rho_i$.

Based on the expression (B.26) for the conditional cdf, we require the analytical expressions for the joint pdf $g_{\rho_1,\ldots,\rho_{i-1}}(\rho_1,\ldots,\rho_{i-1})$ and the joint cdf $G_{\rho_i,\rho_1,\ldots,\rho_{i-1}}(\rho_i,\rho_1,\ldots,\rho_{i-1})$. We start with the joint cdf first; similar to deriving the marginal cdf in Section B.2, we marginalize the variables ρ_j for $j = i+1,\ldots,m$ out of the full joint cdf $G_{\rho_1,\ldots,\rho_m}(\rho_i,\rho_1,\ldots,\rho_m)$:

$$
\begin{aligned}
G_{\rho_i,\rho_1,\ldots,\rho_{i-1}}(\rho_i,\rho_1,\ldots,\rho_{i-1}) &= G_{\rho_1,\ldots,\rho_m}(\rho_1,\ldots,\rho_i,\hat{\rho}_{i+1},\ldots,\hat{\rho}_m) && \text{(B.28)} \\
&= \frac{1}{\kappa_0}\left[\kappa_1\prod_{j=1}^{i}\rho_j\prod_{k=i+1}^{m}\hat{\rho}_k - \sum_{j=1}^{i}h(\rho_j)\left(\prod_{k=1,k\neq j}^{i}\rho_k\right)\left(\prod_{k=i+1}^{m}\hat{\rho}_k\right)\right. \\
&\qquad \left. - \sum_{j=i+1}^{m}h(\hat{\rho}_j)\left(\prod_{k=1}^{i}\rho_k\right)\left(\prod_{k=i+1,k\neq j}^{m}\hat{\rho}_k\right)\right] && \text{(B.29)}
\end{aligned}
$$

The derivative in Eq. (B.26) is then given by differentiating Eq. (B.29):

$$
\begin{aligned}
\frac{d^{i-1}G_{\rho_i,\rho_1,\ldots,\rho_{i-1}}(\rho_i,\rho_1,\ldots,\rho_{i-1})}{d\rho_1\ldots d\rho_{i-1}} &= \frac{1}{\kappa_0}\left[\kappa_1\rho_i\prod_{k+1}^{m}\hat{\rho}_k - \rho_i\sum_{j=1}^{i-1}\frac{dh(\rho_j)}{d\rho_j}\left(\prod_{k=i+1}^{m}\hat{\rho}_k\right)\right. \\
&\qquad \left. - h(\rho_i)\prod_{k=i+1}^{m}\hat{\rho}_k - \rho_i\sum_{j=i+1}^{m}h(\hat{\rho}_j)\prod_{k=i+1,k\neq j}^{m}\hat{\rho}_k\right] && \text{(B.30)}
\end{aligned}
$$

In order to simplify the expression in Eq. (B.30), we define the variable q_i and constant r_i, as given in Eq. (19) and (20):

$$q_i = \left(\sum_{j=1}^{i-1} \frac{dh(\rho_j)}{d\rho_j}\right)\left(\prod_{k=i+1}^{m} \hat{\rho}_k\right) \text{ for } i = 2, \ldots, m \ , \tag{B.31}$$

$$r_i = \sum_{j=i+1}^{m} \left(h(\hat{\rho}_j) \prod_{k=i+1, k\neq j}^{m} \hat{\rho}_k\right) \quad \forall i \in \mathcal{M} \tag{B.32}$$

The derivative $\frac{dh(\rho_i)}{d\rho_i}$ in Eq. (B.31) can be evaluated as such:

$$\frac{dh(\rho_i)}{d\rho_i} = \frac{d}{d\rho_i} \int_0^{\rho_i} \mathbb{E}\left[N_i(\tilde{\rho}_i)\right] d\tilde{\rho}_i = \mathbb{E}\left[N_i(\rho_i)\right] \tag{B.33}$$

Inserting Eq. (B.31) and (B.32) into Eq. (B.30),

$$\frac{d^{i-1} G_{\rho_i, \rho_1, \ldots, \rho_{i-1}}(\rho_i, \rho_1, \ldots, \rho_{i-1})}{d\rho_1 \ldots d\rho_{i-1}} = \frac{1}{\kappa_0}\left[\kappa_1 \rho_i \prod_{k+1}^{m} \hat{\rho}_k - q_i \rho_i - r_i \rho_i - h(\rho_i) \prod_{k=i+1}^{m} \hat{\rho}_k\right] \tag{B.34}$$

We now consider the joint pdf $g_{\rho_1, \ldots, \rho_{i-1}}(\rho_1, \ldots, \rho_{i-1})$. The joint pdf can be represented as the derivative of the joint cdf (Evans and Rosenthal 2004, Corollary 2.5.1). Then, by observing that Eq. (B.35) is the same as setting $\rho_i = \hat{\rho}_i$ in Eq. (B.34), we get the expression for the joint pdf in Eq. (B.37):

$$g_{\rho_1, \ldots, \rho_{i-1}}(\rho_1, \ldots, \rho_{i-1}) = \frac{d^{i-1} G_{\rho_1, \ldots, \rho_{i-1}}(\rho_1, \ldots, \rho_{i-1})}{d\rho_1 \ldots d\rho_{i-1}} \tag{B.35}$$

$$= \frac{d^{i-1} G_{\rho_i, \rho_1, \ldots, \rho_{i-1}}(\hat{\rho}_i, \rho_1, \ldots, \rho_{i-1})}{d\rho_1 \ldots d\rho_{i-1}} \tag{B.36}$$

$$= \frac{1}{\kappa_0}\left[\kappa_1 \hat{\rho}_i \prod_{k+1}^{m} \hat{\rho}_k - q_i \hat{\rho}_i - r_i \hat{\rho}_i - h(\hat{\rho}_i) \prod_{k=i+1}^{m} \hat{\rho}_k\right] \tag{B.37}$$

By inserting Eq. (B.34) and (B.37) into Eq. (B.26), we get the expression for the conditional cdf as given in Eq. (18):

$$G_{\rho_i | \rho_1, \ldots, \rho_{i-1}}(\rho_i | \rho_1, \ldots, \rho_{i-1}) = \frac{\kappa_1 \rho_i \prod_{k=i+1}^{m} \hat{\rho}_k - q_i \rho_i - r_i \rho_i - h(\rho_i) \prod_{k=i+1}^{m} \hat{\rho}_k}{\kappa_1 \hat{\rho}_i \prod_{k=i+1}^{m} \hat{\rho}_k - q_i \hat{\rho}_i - r_i \hat{\rho}_i - h(\hat{\rho}_i) \prod_{k=i+1}^{m} \hat{\rho}_k} \text{ for } i = 2, \ldots, m \tag{B.38}$$

B.4 Transformation from $\boldsymbol{\rho}$ to $\mathbf{x}$

Given the sampled values $\boldsymbol{\rho}$, we first transform them to $\boldsymbol{\mu}$, and subsequently to $\mathbf{x}$. The transformation of $\boldsymbol{\rho}$ to $\boldsymbol{\mu}$ involves using the analytical model (8). Rearranging Eq. (8b), we get an expression for μ_i as a function of the sampled ρ_i:

$$\mu_i = \lambda_i \Big[\rho_i - \Big(\sum_{j \in \mathcal{D}_i} p_{ij} P(N_j = k_j)\Big)\Big(\sum_{j \in \mathcal{D}_i} \rho_j\Big)\Big]^{-1} \tag{B.39}$$

From the values of μ_i, a set of values for the decision variables x_j can be obtained. From Eq. (9), we have

$$\mu_i = s\left(e_i + \sum_{j \in \mathcal{P}_2(i)} x_j\right) \ \forall i \in \mathcal{L} \tag{B.40}$$

$$\implies \sum_{j \in \mathcal{P}_2(i)} x_j = \frac{\mu_i}{s} - e_i \tag{B.41}$$

We can replace $\mathcal{P}_2(i)$ with the matrix elements

$$\xi_{ij} = \begin{cases} 1 & \text{if } j \in \mathcal{P}_2(i) \\ 0 & \text{otherwise} \end{cases} \tag{B.42}$$

Rewriting Eq. (B.41) with ξ_{ij},

$$\sum_j \xi_{ij} x_j = \frac{\mu_i}{s} - e_i \quad \forall i \in \mathcal{L} \tag{B.43}$$

In matrix form, this can be written as

$$\boldsymbol{\Xi}\mathbf{x} = \frac{1}{s}\boldsymbol{\mu} - \frac{1}{c}\mathbf{e} \tag{B.44}$$

$$\mathbf{x} = \boldsymbol{\Xi}^{-1}\big[\frac{1}{s}\boldsymbol{\mu} - \frac{1}{c}\mathbf{e}\big] \tag{B.45}$$

where $\boldsymbol{\Xi} \in \{0,1\}^{|\mathcal{L}|\times|\mathbf{x}|}$ is made up of the elements ξ_{ij}, and is known from the road network. As mentioned in Section 2.5.1, the system of linear equations can be solved using a Moore-Penrose pseudoinverse of $\boldsymbol{\Xi}$ to obtain the sample point in terms of $\mathbf{x}$. In cases where the sum of the green splits for a given intersection does not add up to the cycle time of that intersection, the green splits are scaled proportionally so that they add up correctly.

C Details on validation plot generation

In each plot of Figure 4, the phases that are not plotted have green split values that are fixed at the histogram bin center value where the inverse cdf sampling assigns the highest marginal probability as shown in Figure 5a. As an example, for the top left plot, x_5 and x_7 were both fixed at 0.777778. For the plot based on the queueing model, the plots were generated by computing the expected number of vehicles in the network using the expression in Eq. (7), with the green splits for Phases 1, 3, 5 and 7 used to compute the corresponding $\boldsymbol{\rho}$ values using Eq. (8) and (9). For the plots based on traffic simulation results, 10 traffic simulation replications were used to compute the average objective function estimates at each point.

To generate the marginal distributions in Figure 5b, we first note that the toy network has 4 degrees of freedom due to the linear cycle time constraint on the green split at each intersection (Eq. (3)). The joint green split distribution based on traffic simulation estimates can be represented by

$$g_{x_1,x_3,x_5,x_7}(x_1,x_3,x_5,x_7) = \frac{1}{\kappa_0}\big(\kappa_1' - \mathbb{E}[F(x_1,x_3,x_5,x_7)]\big) \tag{C.1}$$

where $g_{x_1,x_3,x_5,x_7}(x_1,x_3,x_5,x_7)$ is the joint green split distribution, κ_0 is the normalization constant as computed according to Eq. (28), and κ_1' is a constant satisfying the constraint $\kappa_1' \geq \mathbb{E}[F(x_1,x_3,x_5,x_7)]$ to ensure non-negativity. Here, the phases are numbered such that Phases 1 and 2 correspond to the first intersection on the left, as displayed in Figure 2, while Phases 3 and 4 correspond to the second intersection from the left and so on. The odd numbered phases are the phases that are green in favor of the main arterial (i.e., OD pairs (1) $\rightarrow$ (2) and (3) $\rightarrow$ (4) of Figure 3), while the even numbered phases are green in favor of the side roads (i.e., OD pairs (5) $\rightarrow$ (6), (7) $\rightarrow$ (8), (9) $\rightarrow$ (10), (11) $\rightarrow$ (12) and (13) $\rightarrow$ (14) of Figure 3). Then, treating $g_{x_1,x_3,x_5,x_7}(x_1,x_3,x_5,x_7)$ like a joint pdf, we compute the marginal distributions by summing out all but the relevant variable. For instance, to obtain the marginal distribution for x_1, we sum out x_3, x_5 and x_7 (see Eq. (C.2)). We first identify 6 green splits that are uniformly distributed in the feasible range of green split values for each phase, sort them in increasing order, and represent them by the indices i,j,k,l for Phase 1, 3, 5 and 7 respectively (i.e., $(i,j,k,l) \in [1,2,3,4,5,6]^4$). The choice of 6 green splits was used as a compromise between having as many green splits as possible and computational runtime. This leads to $6^4 = 1296$ possible combinations of green splits for the 4 intersections. Each of these combinations are simulated 10 times to obtain an average of the objective function estimate. The marginal distributions are then approximated using a discrete sum. Eq. (C.2) shows the case for calculating the marginal distribution for Phase 1, with the rest of the phases (i.e., Phases 3, 5 and 7) taking on a similar form:

$$g_{x_1}(x_{1,i}) \approx \frac{1}{\kappa_0}\sum_{j=1}^{6}\sum_{k=1}^{6}\sum_{l=1}^{6}\big[\kappa_1' - f(x_{ijkl})\big]\Delta x_3 \Delta x_5 \Delta x_7 \tag{C.2}$$

where $x_{1,i}$ denotes the i^{th} green split value for Phase 1. x_{ijkl} refers to the green split combination with i,j,k,l representing the index of the green split of Phase 1, 3, 5 and 7 respectively. Each of Δx_p is the interval between each successive green split for phase p.

D Simulation-based optimization algorithm

The simulation optimization (SO) framework used for this study is based on the metamodel-based optimization method proposed by Osorio and Bierlaire (2013). A metamodel is an analytical function which attempts to approximate the underlying objective function. The notation to be used in the algorithm is defined at a given iteration k as such:

$\mathbf{x}_k$	current iterate
Δ_k	trust region radius
m_k/ϕ_k	metamodel
$\nu_k = (\alpha_k, \boldsymbol{\beta}_k)$	vector of metamodel parameters
n_k	total number of simulation runs conducted up to and including iteration k
u_k	total number of successive trial points rejected
$\hat{f}$	simulation estimate of objective function
τ_k	number of optimization iterations since the last sampling iteration
$\bar{\tau}$	number of optimization iterations to run before switching to a sampling iteration, as defined by the simulation budget for sampling (see Table 5)
n_{max}	total number of simulation run permitted
$\bar{r}$	number of simulation replications per iteration
d	number of decision variables
z	endogenous variables

The metamodels used in Section **??** are denoted m and ϕ (see Table 4). The m metamodel makes use of the analytical network model estimate of the objective function (Eq. (7)), in addition to a functional component, as shown in Eq. (D.1). The ϕ metamodel is a general-purpose metamodel, consisting of just a quadratic polynomial, as shown in Eq. (D.2).

$$m_k(\mathbf{x}_k, z; \alpha_k, \boldsymbol{\beta}_k, p) = \alpha_k f^A(\mathbf{x}_k, z; p) + \phi_k(\mathbf{x}_k; \boldsymbol{\beta}_k) \tag{D.1}$$

$$\phi_k(\mathbf{x}_k; \boldsymbol{\beta}_k) = \beta_{k,1} + \sum_{j=1}^{d} \beta_{k,j+1} x_{k,j} + \sum_{j=1}^{d} \beta_{k,j+d+1} x_{k,j}^2 \tag{D.2}$$

Before running the SO algorithm (Algorithm 3), the SO method (Table 4), along with the proportion of simulation budget to be used for sampling (Table 5), have to first be selected. The initial point $\mathbf{x}_0$ is generated using uniform sampling, and evaluated by simulation to obtain the simulation estimate $\hat{f}(\mathbf{x}_0)$. The simulation estimate is then used to fit the metamodel parameters by solving a least squares problem (Osorio and Bierlaire 2013, Eq. 5). The initialization step (Step 0) is the same as Step 0 in the algorithm proposed by Osorio and Bierlaire (2013, Section 4.2). The notation used is the same, except for $\bar{\gamma}$ and $\bar{r}$, which are represented by γ and r in Osorio and Bierlaire (2013).

Subsequently, at each iteration, the algorithm either (i) samples a new point to evaluate (Step 1), or (ii) it optimizes the fitted metamodel (Steps 2-4) based on the proportion of simulation budget allocated to sampling. In the sampling step, a new point is generated by the chosen sampling strategy, and evaluated by simulation. The simulation observation is then used to update the metamodel parameters. The optimization steps (Steps 2-4) are the same as Steps 2, 3 and 5 in Osorio and Bierlaire (2013, Section 4.2). The notation used is the same, except for ψ_k, which is represented by ρ_k in Osorio and Bierlaire (2013). The main difference between Algorithm 3 and the algorithm proposed in Osorio and Bierlaire (2013) is that the model improvement step (i.e., Step 3 in Osorio and Bierlaire (2013)) is replaced by the sampling step (Step 1 in Algorithm 3). The sampling step of Algorithm 3 is triggered after a deterministic number of optimization iterations, whereas the model improvement step of Osorio and Bierlaire (2013) is triggered based on the change in metamodel parameters. Also, the model improvement step of Osorio and Bierlaire (2013) uses uniform sampling to sample for a new point. For additional details about the metamodel-based optimization steps (Steps 2-4 in Algorithm 3), we refer the reader to Osorio and Bierlaire (2013, Section 4.2, Steps 2, 3 and 5)

Algorithm 3: Simulation-based optimization algorithm

Let $\eta_1, \bar{\gamma}, \gamma_{inc}, \bar{d}, \bar{u}, \Delta_{max}$ be constants which satisfy the following constraints: $0 < \eta_1 < 1$, $0 < \bar{\gamma} < 1 < \gamma_{inc}$, $0 < \bar{d} < \Delta_{max}$, $\bar{u} \in \mathbb{N}^*$ (see Osorio and Bierlaire (2013, Section 4.3) for values used)

0. **Initialization**

   (a) Select metamodel for use: m (Eq. (D.1)) or ϕ (Eq. (D.2))

   (b) Select sampling strategy: uniform sampling (i.e., Unif) or inverse cdf sampling (i.e., Invcdf; see Algorithm 1)

   (c) Select proportion of simulation budget for sampling and set $\bar{\tau}$: 10% ($\bar{\tau} = 9$), 50% ($\bar{\tau} = 1$) or 100% ($\bar{\tau} = 0$)

   (d) Set $k = 0$, $n_0 = 0$, $u_0 = 0$, $\tau_0 = 0$

   (e) Sample an initial point $\mathbf{x}_0$ using uniform sampling

   (f) Determine Δ_0 ($\Delta_0 \in (0, \Delta_{max})$) (see Osorio and Bierlaire (2013, Section 4.3) for values used)

   (g) Run $\bar{r}$ simulations to get $\hat{f}(\mathbf{x}_0)$ and set $n_0 = n_0 + \bar{r}$

   (h) Compute ν_0 according to Osorio and Bierlaire (2013, Eq. (5)) and fit an initial metamodel

   (i) Set $n_{k+1} = n_k$, $k = k + 1$

1. **Sampling**
   if $\tau_k = \bar{\tau}$ **do**

   (a) Sample for a new point $\mathbf{x}_k$ using the chosen sampling strategy

   (b) Run $\bar{r}$ simulations to get $\hat{f}(\mathbf{x}_k)$ and set $n_k = n_k + \bar{r}$

   (c) Include the new observation in the set of point evaluated by simulation.
   Compute ν_k according to Osorio and Bierlaire (2013, Eq. (5)) and fit the new metamodel m_{k+1} or ϕ_{k+1}.

   (d) Set $n_{k+1} = n_k$, $\tau_{k+1} = 0$, $k = k + 1$

   (e) **if** $n_k < n_{max}$ **do** repeat Step 1, **else** stop

   else
   Go to Step 2
   end

2. **Optimization: step calculation**
   Compute a step $\mathbf{s}_k$ that minimizes the metamodel m_k or ϕ_k, such that the trial point $\mathbf{x}_k + \mathbf{s}_k$ is within the trust region.

Algorithm 4: Simulation-based optimization algorithm (continued)

4. **Optimization: acceptance or rejection of the trial point**

   (a) Run $\bar{r}$ simulations to get $\hat{f}(\mathbf{x}_k + \mathbf{s}_k)$ and set $n_k = n_k + \bar{r}$

   (b) Compute the ratio $\psi_k = \frac{\hat{f}(\mathbf{x}_k) - \hat{f}(\mathbf{x}_k + \mathbf{s}_k)}{m_k(\mathbf{x}_k) - m_k(\mathbf{x}_k + \mathbf{s}_k)}$ or $\psi_k = \frac{\hat{f}(\mathbf{x}_k) - \hat{f}(\mathbf{x}_k + \mathbf{s}_k)}{\phi_k(\mathbf{x}_k) - \phi_k(\mathbf{x}_k + \mathbf{s}_k)}$ depending on the chosen metamodel

   (c) **if** $\psi_k \geq \eta_1$ **do**
   Accept the trial point: $x_{k+1} = \mathbf{x}_k + \mathbf{s}_k$, $u_k = 0$, $\tau_k = \tau_k + 1$
   else
   Reject the trial point: $x_{k+1} = \mathbf{x}_k$, $u_k = u_k + 1$, $\tau_k = \tau_k + 1$
   end

   (d) Include the new observation in the set of points evaluated by simulation. Compute ν_k according to Osorio and Bierlaire (2013, Eq. (5)) and fit the new metamodel m_{k+1} or ϕ_{k+1}

5. **Optimization: trust region radius update**

$$\Delta_{k+1} = \begin{cases} \min\{\gamma_{inc}\Delta_k, \Delta_{max}\} & \text{if } \psi_k > \eta_1 \\ \max\{\bar{\gamma}\Delta_k, \bar{d}\} & \text{if } \psi_k \leq \eta_1 \text{ and } u_k \geq \bar{u} \\ \Delta_k & \text{otherwise} \end{cases}$$

   (a) **if** $\psi_k \leq \eta_1$ and $u_k \geq \bar{u}$, **do** set $u_k = 0$

   (b) Set $n_{k+1} = n_k$, $u_{k+1} = u_k$, $k = k + 1$

   (c) **if** $n_k < n_{max}$ **do** go to Step 1, **else** stop

References

Abramowitz, Milton, Irene A Stegun. 1964. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*, vol. 55. Dover Publications, New York, 11.

Alibaba DAMO Academy. 2019. Top 10 technology trends in 2019 predicted. URL `https://damo.alibaba.com/events/51`. Accessed: 2019-01-21.

Andrieu, Christophe, Nando De Freitas, Arnaud Doucet, Michael I Jordan. 2003. An introduction to mcmc for machine learning. *Machine learning* **50**(1-2) 5–43.

Balakrishna, Ramachandran, Haris Koutsopoulos. 2008. Incorporating within-day transitions in simultaneous offline estimation of dynamic origin-destination flows without assignment matrices. *Transportation Research Record* **2085**(1) 31–38.

Barceló, Jaume. 2010. *Fundamentals of traffic simulation*, *International Series in Operations Research & Management Science*, vol. 145. Springer, New York.

Ben-Akiva, Moshe, David Cuneo, Masroor Hasan, Mithilesh Jha, Qi Yang. 2003. Evaluation of freeway control using a microscopic simulation laboratory. *Transportation Research Part C: Emerging Technologies* **11**(1) 29–50.

Bocharov, P, C D'Apice, A Pechenkin, S Salerno. 2004. *Modern probability and statistics: Queueing theory*, chap. 3. Brill Academic Publishers, Zeist, Netherlands, 96–98.

Campbell, Stephen L, Carl D Meyer. 1991. *Generalized inverses of linear transformations*, *Classics in Applied Mathematics*, vol. 56. SIAM.

Casella, George, Roger L. Berger. 2002. *Statistical Inference*. 2nd ed. Cengage Learning.

Chong, Linsen, Carolina Osorio. 2018. A simulation-based optimization algorithm for dynamic large-scale urban transportation problems. *Transportation Science* **52**(3) 637–656.

Evans, Michael J, Jeffrey S Rosenthal. 2004. *Probability and statistics: The science of uncertainty*. Macmillan.

Gelman, Andrew, Kenneth Shirley. 2011. Inference from simulations and monitoring convergence. *Handbook of Markov chain Monte Carlo*, chap. 6. CRC Press, Boca Raton, FL, 163–174.

Geman, Stuart, Donald Geman. 1984. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-6**(6) 721–741.

Greenhall, Adam. 2016. Experimentation in a ridesharing marketplace: Simulating a ridesharing marketplace. URL `https://eng.lyft.com/https-medium-com-adamgreenhall-simulating-a-ridesharing-marketplace-36007a8a31f2`. Accessed: 2019-01-29.

Hastings, W.K. 1970. Monte carlo sampling methods using markov chains and their applications. *Biometrika* **57**(1) 97–109.

Jin, Junchen, Xiaoliang Ma, Iisakki Kosonen. 2017. A stochastic optimization framework for road traffic controls based on evolutionary algorithms and traffic simulation. *Advances in Engineering Software* **114** 348–360.

Jones, DR, M Schonlau, WJ Welch. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization* **13** 455–492.

Lu, Lu, Yan Xu, Constantinos Antoniou, Moshe Ben-Akiva. 2015. An enhanced SPSA algorithm for the calibration of dynamic traffic assignment models. *Transportation Research Part C: Emerging Technologies* **51** 149–166.

Luong, Bruno. 2009. Pseudo-inverse. URL `https://www.mathworks.com/matlabcentral/fileexchange/25453-pseudo-inverse`. Accessed: 2018-10-17.

MapQuest.com, Inc. 2018. New York City, NY, scale undetermined; generated by Timothy Tay using "MapQuest.com, Inc". URL `http://www.mapquest.com`. Accessed: 2018-07-01.

Metropolis, Nicholas, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, Edward Teller. 1953. Equation of state calculations by fast computing machines. *The journal of chemical physics* **21**(6) 1087–1092.

Osorio, C, M Bierlaire. 2009. An analytic finite capacity queueing network model capturing the propagation of congestion and blocking. *European Journal of Operational Research* **196**(3) 996–1007.

Osorio, C, L Chong. 2015. A computationally efficient simulation-based optimization algorithm for large-scale urban transportation problems. *Transportation Science* **49**(3) 623–636.

Osorio, Carolina. 2010. Mitigating network congestion: analytical models, optimization methods and their applications. Ph.D. thesis, École Polytechnique Fédérale de Lausanne.

Osorio, Carolina, Bilge Atasoy. 2017. Efficient simulation-based toll optimization for large-scale networks. Technical report, Massachusetts Institute of Technology. Under review. Available at: http://web.mit.edu/osorioc/www/papers/osoAtaTollSO.pdf.

Osorio, Carolina, Michel Bierlaire. 2013. A simulation-based optimization framework for urban transportation problems. *Operations Research* **61**(6) 1333–1345.

Osorio, Carolina, Kanchana Nanduri. 2015a. Energy-efficient urban traffic management: a microscopic simulation-based approach. *Transportation Science* **49**(3) 637–651.

Osorio, Carolina, Kanchana Nanduri. 2015b. Urban transportation emissions mitigation: Coupling high-resolution vehicular emissions and traffic models for traffic signal optimization. *Transportation Research Part B: Methodological* **81** 520–538.

Osorio, Carolina, Chen Xiao, Bruno Filipe Santos. 2017. Simulation-based travel time reliable signal control. *Transportation Science* Forthcoming. Available at: http://web.mit.edu/osorioc/www/papers/osoCheSanReliableSO.pdf.

Osorio, Carolina, Jana Yamani. 2017. Analytical and scalable analysis of transient tandem Markovian finite capacity queueing networks. *Transportation Science* **51**(3) 823–840.

Owen, Art B. 2013. *Monte Carlo theory, methods and examples*. Available at: https://statweb.stanford.edu/%7Eowen/mc/.

Paz, Alexander, Victor Molano, Ember Martinez, Carlos Gaviria, Cristian Arteaga. 2015. Calibration of traffic flow models using a memetic algorithm. *Transportation Research Part C: Emerging Technologies* **55**(4) 432–443.

Robert, Christian, George Casella. 1999. *Monte Carlo statistical methods*. Springer, New York.

Rosenblatt, M. 1952. Remarks on a multivariate transformation. *Annals of Mathematical Statistics* **23**(3) 470–472.

Sebastiani, Mariana Teixeira, Ricardo Luders, Keiko Veronica Ono Fonseca. 2016. Evaluating electric bus operation for a real-world brt public transportation using simulation optimization. *IEEE Transactions on Intelligent Transportation Systems* **17**(10) 2777–2786.

Stafford, R. 2006. The theory behind the "randfixedsum" function. URL `http://www.mathworks.com/matlabcentral/fileexchange/9700-random-vectors-with-fixed-sum`. Accessed: 2017-11-25.

Stevanovic, Jelka, Aleksandar Stevanovic, Peter T Martin, Thomas Bauer. 2008. Stochastic optimization of traffic control and transit priority settings in vissim. *Transportation Research Part C: Emerging Technologies* **16**(3) 332–349.

Teklu, Fitsum, Agachai Sumalee, David Watling. 2007. A genetic algorithm approach for optimizing traffic control signals considering routing. *Computer-Aided Civil and Infrastructure Engineering* **22**(1) 31–43.

Tokdar, Surya T, Robert E Kass. 2010. Importance sampling: a review. *Wiley Interdisciplinary Reviews: Computational Statistics* **2**(1) 54–60.

TSS-Transport Simulation Systems. 2015. Aimsun 8 users' manual. Version 8.0. (Barcelona, Spain).

Tympakianaki, Athina, Haris N. Koutsopoulos, Erik Jenelius. 2015. c-SPSA: Cluster-wise simultaneous perturbation stochastic approximation algorithm and its application to dynamic origin-destination matrix estimation. *Transportation Research Part C: Emerging Technologies* **55** 231–245.

Tympakianaki, Athina, Haris N. Koutsopoulos, Erik Jenelius. 2018. Robust SPSA algorithms for dynamic od matrix estimation. *Procedia Computer Science* **130** 57–64.

Wang, Wenyu, Hong Wan, Kuo-Hao Chang. 2016. Randomized block coordinate descendant strong for large-scale stochastic optimization. *2016 Winter Simulation Conference (WSC)*. IEEE, 614–625.

Xu, Jie, Edward Huang, Liam Hsieh, Loo Hay Lee, Qing Shan Jia, Chun Hung Chen. 2016. Simulation optimization in the era of Industrial 4.0 and the Industrial Internet. *Journal of Simulation* **10**(4) 310–320.

Yun, Ilsoo, Byungkyu Brian Park. 2006. Application of stochastic optimization method for an urban corridor. *Proceedings of the 38th conference on Winter simulation*. Winter Simulation Conference, 1493–1499.

Zhang, Chao, Carolina Osorio, Gunnar Flötteröd. 2017. Efficient calibration techniques for large-scale traffic simulators. *Transportation Research Part B: Methodological* **97** 214–239.

Zhou, Tianli, Carolina Osorio, Evan Fields. 2018. A data-driven discrete simulation-based optimization algorithm for large-scale two-way car-sharing network design. *Transportation Science* Under review. Available at: http://web.mit.edu/osorioc/www/papers/zhoOsoFieCarSharing.pdf.