Analysing Traffic through Emulation

Matteo Bettini
mb2389
Sidney Sussex College

Nicholas Kastanos
nk569
Queens' College

Harry Songhurst
hs778
St Edmund's College

Abstract

Urban planning and traffic management is a growing issue. In order to plan for rapidly urbanising populations, government agencies and local councils need to consider multiple factors, including traffic flow and air quality. Traditional large-scale traffic simulations are resource and time-intensive, especially when simulating microscopic details. Gaussian Processes fitted to simulation data can be used to model the input-output relationships of the system without the complexity of the simulator. In this project, the SUMO traffic simulator is used to create traffic flow and carbon dioxide emulation models for city-centre-like grid road networks through Experimental Design. Once the models have been developed, Bayesian Optimisation techniques are used to locate the optimal configuration of each model to reduce time lost by vehicles, and CO_2 emissions. The sensitivity of the models to each input parameter is then analysed. It is shown that the optimal operating point has low maximum vehicle speeds and accelerations, with long road lengths. An important trade-off exists between the two outputs with respect to the city size. The most significant input is the maximum vehicle speed.

1 Introduction and Motivation

Since 1950, the proportion of the world's population living in urban areas has increased from 30% to approximately 55%. This trend is projected to continue, with an expected 60% of the world's population living in cities by 2030 [1]. With denser concentrations of people comes denser concentrations of cars and other vehicles, and thus the need for better urban planning, traffic management, and, where necessary, laws restricting the usage of motor-vehicles all together.

Government agencies and local councils often confront these challenges with multiple objectives in mind; improving air quality, improving traffic flow, and improving road safety, to name a few. To understand the effect that various confounding factors have on these measures, urban planners have many tools at their disposal such as air quality surveying, traffic monitoring, and perhaps most importantly, running vast and detailed simulations of cars moving through virtual reconstructions of the area they wish to understand.

For simulations to be representative and reliable, they must look at large areas and simulate the movement of tens of thousands of vehicles. This can be *extremely* computationally expensive, especially if we care about collecting highly granular data at every step of the simulation.

This project explores how emulation can be employed to build models of traffic simulators. Specifically, we consider the problems of predicting the average amount of CO_2 a particular simulation setup will produce, and the average amount of time cars will loose whilst stuck in traffic. Our learned emulators are much faster to run than their underlying simulators. They also allow us to analyse the relative importance of input factors affecting any particular result. Taken to production quality, such an emulator could inform time-sensitive decisions that traffic-management agencies must take to improve congestion, and urban planners must take when designing cities of the future.

The report is structured in the following way: Section 2 provides background information on traffic simulators and emulation. Section 3 introduces the traffic simulator and its parameters. Section 4 discusses the inputs and outputs of the designed emulators. Section 5 discusses the experimental design and results of the created emulators. Section 6 investigates travel time and emission optimisation using Bayesian Optimisation. Finally, Section 7, analyses the sensitivity of the emulators with respect to each input parameter. The code used to create and evaluate the emulators can be found at `https://github.com/Nicholas-Kastanos/TrafficEmu`

2 Background

2.1 Traffic Simulation

The problem of traffic congestion has been studied for a long time in civil engineering [2, 3]. Traffic congestion affects users' speed and, thus, travel time and pollution. The dependency between travel time and traffic density (flow) was observed in early studies [4] and more recent works [5, 6].

There are various approaches to model traffic congestion. In optimisation-based models, [7] traffic is represented as flow in a capacitated road network. They formalise the trade-off between user optimal assignment (minimising travel time) and system optimal assignment (avoiding congestion). Other approaches include analytic queuing models [8], and simulation-based models [9, 10, 11].

Traffic simulation models can be divided into three main categories [12]:

- **Macroscopic traffic modeling** [13], where traffic is simulated at a system level with traffic density represented as continuous flows in the road network.
- **Microscopic traffic modeling** [14] simulates traffic at a vehicle level. Each single vehicle can have different characteristics and parameters. This allows for detailed analyses on the interaction between different types of vehicles.
- **Mesoscopic traffic modeling** [15] bridges the two approaches by providing a trade-off between simulation complexity and level of detail.

In this work we will look into microscopic traffic simulation as this type of simulators are the most computationally demanding having to simulate traffic at a vehicles' level.

After reviewing the available state of the microscopic traffic simulators [16, 17, 10] as well as literature surveys on the topic [18], we chose to use SUMO [10].

2.2 Gaussian processes as surrogate models

Gaussian Processes (GPs) are non-parametric models which encode an infinite dimensional distribution over the space of functions. Not relying on parameters, they assign non-zero probability to every continuous function. They rely on a kernel function to encode the similarity between two points. The kernel function used in this work is the Radial Basis Function [19], seen in Equation 1. It encodes the similarity between two input vectors $\mathbf{x}$ and $\mathbf{x}'$ with σ being a free parameter.

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2\sigma^2}\right) \tag{1}$$

Gaussian Processes are commonly used in regression models [20]. By fitting the GP to data gathered from a simulator, a low complexity surrogate model of the computationally complex simulation can be created [21]. This surrogate model is referred to as an *emulator*. The emulator learns the input-output relationship of the original model, however due to the difficulty of gathering data from computationally expensive simulators, the emulator has to be modelled using limited data.

2.2.1 Experimental Design

In machine learning, supervised problems are those for which we learn an input-output mapping by fitting some model to a dataset of input-output examples [22]. In many cases, we do not have access to such a dataset, however, we may have the ability to collect input-output pairs on-the-fly. This

is the setting where active learning, and indeed experimental design, excel. They use heuristics to guide their choice of input points to evaluate, in the hopes of learning as much as possible about the underlying function, thus enabling us to create an accurate surrogate model of said function. In the context of experimental design, these heuristics are known as *acquisitions functions*, and in the case where the underlying function is a simulator, we say that the learned surrogate model is an *emulator* if this model also reports how confident it is in its prediction.

Since the purpose of experimental design is to learn as much as possible about the underlying function, we would like an acquisition function that picks points that tell us as much as possible about the underlying function. The two most popular choices are "uncertainty sampling" and "integrated variance reduction". Both methods rely on being able to evaluate the predictive variance of our model with respect to any particular input; since Gaussian Processes report this with each evaluation, they are a natural model choice for experimental design. Both methods rely on some form of iterative optimization, such as gradient descent, to find the next point in the domain which is predicted to either maximised (in the case of uncertainty sampling) or minimized (in the case of integrated variance reduction) [23] the function.

Uncertainty sampling [24], selects selects at which the emulator has high variance, with the rationale being that points with high variance are 'interesting' and have more to be learned from than points we're already confident about. More formally, we wish to find points $\mathbf{x}$ at which the predictive variance $\sigma^2(\mathbf{x})$ of our model is maximal. In theory, high-variance regions of your learned function will be sampled frequently and therefore have their variance reduced. This should drive us to explore different points in the input domain which we know less about.

Integrated variance reduction (IVR) lets us finds points in the input domain which reduce the total variance of the model. We use an optimization procedure such as gradient descent to find a point $\mathbf{x}$ which minimises a cost. This cost is the Monte-Carlo approximation of the total variance of the function, and is computed by randomly sampling some fixed number of points $\mathbb{X}$, then taking the summative difference in variance had $\mathbf{x}$ been observed versus if it hadn't. Formally, the Monte-Carlo approximation takes following form [23, 25]:

$$a_{IVR} = \quad \approx \frac{1}{\text{samples}} \sum_{i}^{\text{samples}} \left[\sigma^2\left(\mathbf{x}_i\right) - \sigma^2\left(\mathbf{x}_i; \mathbf{x}\right)\right] \tag{2}$$

2.2.2 Bayesian Optimization

Bayesian Optimisation (BO) [26] is a technique that allows for optimization of an explicitly unknown function while minimising the number of evaluations. It works by iteratively querying the underlying function with the objective of discovering new extreme points. By using a GP to quantify our beliefs in an underlying function, we can utilise the uncertainty in these beliefs to guide the search process. A major focus of this search is the balance between *exploitation* and *exploration*. Where *exploitation* pushes towards exploring near the already known extreme point and *exploration* incentives the investigation of areas with a high predictive variance, as they could uncover a better extreme point. For this purpose, an acquisition function is used. The acquisition function models the utility of evaluating a particular point for our task.

The most commonly used acquisition function is Expected Improvement [26]. The utility $u(x)$ of sampling one point using expected improvement is defined relative to how much lower we expect the function value at this point to be compared to the current best estimate.

$$u(x) = \max(0, f(x_*) - f(x)) \tag{3}$$

The expected improvement is then computed by taking the expectation of the utility function over our current gaussian distribution of functions.

2.3 Sensitivity Analysis

One-factor-at-a-time analysis involves iteratively generating outputs from an emulator while moving a single input through its input range [27, 28, 29]. The remaining parameters are kept at a constant value. This is completed for each input variable. While this analysis is cheap to compute and provides

easily understandable results, it only provides insight into a single output at a time. Additionally, the factor under investigation may have significantly different behaviour while at a different operating point.

As opposed to OFAT, global sensitivity analysis aims to take all inputs across its entire domain into consideration [28, 30, 31]. The Hoeffding-Sobol decomposition allows Sobol indices to be calculated for any combination of input parameters. The first-order Sobol indices present a similar measure to OFAT, where the output variance is measured while a single input variable is changed. The Total Effects present the variance of the output attributed by the input variable and all of its interactions. This allows the full impact of the parameter to be observed.

3 Simulation

3.1 SUMO

SUMO (Simulation Of Urban Mobility) is a free and open-source traffic microsimulator. It provides interactive graphical tools for both visualising simulations and creating road networks. Real-world networks can also be imported from Open Street Map [32]. SUMO allows intermodal traffic simulation by modelling pedestrians, public transport and cyclists.

In [18] the authors show that SUMO begins to suffer from performance degradation when simulations exceed 10,000 road network edges, while macrosimulators can cope with very large networks without any performance issues. Our goal is to show that emulation using Gaussian Processes (GPs) can enable users to obtain the detailed output of microsimulations by querying an easy-to-compute surrogate model, preventing them from falling back to less detailed macrosimulation models due to scalability issues.

3.2 Map Creation

We study the traffic congestion problem in the context of cities. In order to be able to tune different road network parameters programmatically, custom networks are generated with a lattice grid topology as seen in Figure 1. These networks represent a portion of a city or a district. Grids have been chosen since ancient Roman times as the main topology for urban environments and, still today, characterise the outline of many major cities in the world (e.g., New York).

The grids created for our simulation present the parameters shown in Table 1. Intersections are regulated by actuated traffic lights[1]. Grid size (N_G), edge length (L_E), and number of lanes (N_L) are the parameters that we are interested in varying throughout our simulations.

[1]Green phases may be prolonged depending on traffic measurements from automatically added induction loops

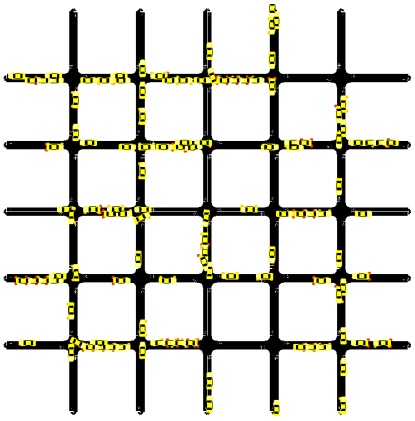


Figure 1: Custom generated 5×5 *grid network*

Table 1: Tunable parameters of the custom grid network

Parameter name	Description
Grid Size (N_G)	Number of intersections along a single axis of the grid
Junction Type	How traffic is regulated at intersections
Edge Length (L_E)	The distance between two intersections in meters
Number of Lanes (N_L)	The number of lanes of each road
Edge Max Speed (v_{max})	The speed limit in the network

3.3 Route Planning

In order to simulate traffic flow in the generated network, vehicles' trips that resemble real urban traffic were generated. For this purpose, we used the `randomTrips` tool provided by SUMO, which is able to generate random trips given an input road network, vehicle specification and route properties. This tool automatically ensures that routes and networks are valid.

Table 2: Tunable parameters of the vehicle generator

Parameter name	Description
Vehicle Class	Vehicle type eg. passenger, emergency, bus
Emissions Class	Vehicle emissions as per the HBEFA v3.1 standard
Acceleration (α_V)	Maximum vehicle acceleration
Deceleration	Maximum vehicle deceleration
Max Speed	Maximum vehicle speed
Speed Factor	Expected multiplier for lane speed limits
Speed Deviation	The deviation of the speed factor

The generator creates vehicles with the properties seen in Table 2. This allows many different vehicle types to be explored, however, to reduce emulator complexity, only passenger vehicles are generated. Every vehicle has the emission class of a gasoline-driven passenger car (Euro norm 4 HBEFA v3.1-based). Additionally, the maximum deceleration remains constant at $4.5\ m/s^2$, the speed factor equals 1 with a 0.1 random deviation.

Table 3: Tunable parameters of the trip generator

Parameter name	Description
Begin Time (t_b)	Trip departure begin time
End Time (t_e)	Trip departure end time
Period (Δt_V)	Vehicle departure period
Fringe Factor	Probability multiplier for trips to start on the edge of the network

Once the vehicle has been created, trips are generated on the road grid. These trips can be tuned using the parameters seen in Table 3. Trips depart throughout the simulation time, which has a duration of $300\ s$. In order to simulate a grid which is part of a larger network, vehicles are 10 times more likely to start or end their trip on the edge of the network. This is set by the fringe factor.

3.4 Tuning congestion

The shape of the simulated networks varies greatly between different simulations. This is because it is highly dependent on the N_G and L_E. Therefore, if we generate the same number of vehicles N_V for every network we will obtain unbalanced traffic congestion throughout the simulations. Small grids are highly congested while large ones have free-flowing traffic.

In order to keep traffic flow constant between different simulations, we start by defining the total number of roads in the grid N_{roads} and the total number of road meters in the grid L_{roads}.

$$N_{roads} = 2N_G(N_G - 1) + 4N_G \tag{4}$$

$$L_{roads} = N_{roads} L_E \tag{5}$$

The maximum number of vehicles $N_{V_{max}}$ that is possible to fit in the simulation will then be:

$$N_{V_{max}} = \frac{L_{roads}}{\text{Vehicle Length}} \tag{6}$$

Finally, to achieve constant traffic density, in each simulation we generate only a percentage of this number by multiplying it by a scaling factor β. We choose a value of 0.05 for β which allows a moderate degree of traffic congestion while avoiding grid-lock.

$$N_V = \beta N_{V_{max}} \tag{7}$$

Equation 8 is used to set the vehicle generation period of the simulation.

$$\Delta t_V = \frac{t_e - t_b}{N_V} \tag{8}$$

4 Experimental Setup

Each simulation we ran relied on SUMO, and an interface we wrote allowing us to dynamically vary properties of vehicles, maps and aspects of the environment such as traffic light timing. This allowed us to analyse the effect of various input parameters on various output parameters. For this project, we cared about the following SUMO-generated outputs:

- average CO_2 released per car, per second.
- average time lost per second due to driving slower than the ideal speed.

We analysed how these outputs were affected by the following inputs:

- Edge Max Speed, $8 \leq v_{max} \leq 19$
- Edge Length, $30 \leq L_E \leq 70$
- Number of Lanes, $N_L \, \epsilon \, \{1, 2, 3\}$
- Grid Size, $N_G \, \epsilon \, \{3, 4, \ldots, 20\}$
- Acceleration, $1.5 \leq \alpha_V \leq 5$

It was important for us to normalize our outputs by a metric that controlled for the size of the grid, because larger grids result in longer travel times, and thus more emissions and lost time. We chose to normalize our outputs by the SUMO-generated “duration" statistic - the average number of seconds cars were in transit for per simulation. This allowed us to explore more accurately the causal relationship between the above inputs and outputs.

All experiments were conducted using Emukit [33] - a Python framework for fitting emulators to simulators. Whilst Emukit is agnostic to the underlying statistical model, it has native integration with GPy [34], a Gaussian Process framework for Python. Emukit allowed us to conduct Experimental Design, Bayesian Optimisation, and Sensitivity Analysis.

5 Experimental Design

We performed experimental design with both uncertainty sampling (aka model variance) and integrated variance reduction. We found that uncertainty sampling was prone to a sort of ‘reward hacking’ [35]. Our optimization continually collapsed to finding a configuration of 4/5 input parameters that produced extremely high variance when varying the remaining 1/5 parameter (‘edgeLength’, the length of roads between intersections, was this parameter). This meant that it failed to learn an accurate model of the underlying function, instead learning only about that one parameter. We originally attributed these highly variable outputs to a bug in our underlying simulation, which the

experimental design process had learned to exploit. However, even after fixing this, uncertainty sampling still favoured extreme points in the input domain. This makes sense since the point of uncertainty sampling is to find points with high variance, however, this was not ideal for us; we wanted confident predictions across the full input domain, not just at the extremes.

Integrated variance reduction (IVR) [25] proved to be more effective for our purposes. We found that IVR was less prone to 'reward hacking', and explored the parameter space more completely - this is reflected in our results.

5.1 Experimental Design Results

To assess the efficacy of our learned emulators, and experimental design in general, we performed t-tests in which we compared emulators learned using principled acquisition functions (uncertainty sampling and integrated variance reduction) to emulators learned using randomly sampled points. To do this, we first collected two sets of randomly sampled points using Emukit's 'RandomDesign' utility; an initialization set of 20 points to initialize each GP with, and a test set of 100 points to evaluate the mean squared error of our models against. Then, we trained three emulators; one using uncertainty sampling (model_variance) as our acquisition function, one using integrated variance reduction, and one using randomly sampled points. Every 50 iterations we evaluated each model on our test set - these results are plotted in Figure 2, please note that y-axes are on different scales.

After all models had 'seen' 500 points, we computed for each a final root mean squared error. Our results are summarised thus:

- Integrated Variance Reduction was 13.98% better than training on randomly sampled points when modelling CO2 per second.
- Integrated Variance Reduction was 7.63% better than training on randomly sampled points when modelling time loss per second.
- Uncertainty Sampling was -8.25% worse than training on randomly sampled points when modelling CO2 per second.
- Uncertainty Sampling was 2.58% better than training on randomly sampled points when modelling time loss per second.

These results show that acquisition through integrated variance analysis is beneficial for our particular problem, however, acquisition through uncertainty sampling is less effective. The advantage of both acquisition functions diminishes relative to random sampling as more and more points are collected, implying that these techniques excel in settings where we have few input points. We ran experimental design 5 times for both CO2 and TimeLoss, then chose the lowest RMSE emulators to pass onto Bayesian Optimization.

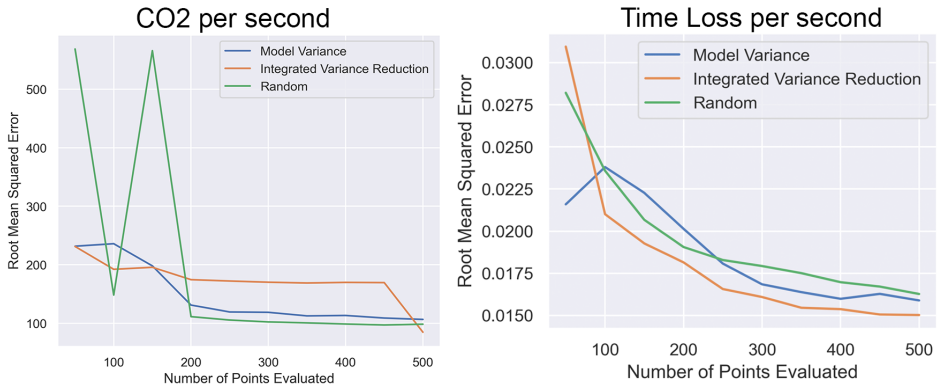


Figure 2: Experimental Design results - root mean square error for models trained on intervals of 50 points acquired through uncertainty sampling (model variance), integrated variance reduction, and random sampling.

6 Bayesian Optimization

How can we reduce traffic congestion? Can we structure our road networks so that navigation is both fast and the impact on the environment is reduced? In this section, we investigate these questions trying to provide answers based on our traffic emulator.

We are interested in finding the optimal parameters' configuration in order to minimise our outputs. However, the function we aim to optimise is explicitly unknown and querying it for a single data point takes a few seconds. For this reason, we use Bayesian Optimisation (BO), already introduced in section 2.2.2. In this work, we choose Expected Improvement as our acquisition function. Optimisation is ran on the models acquired from experimental design (5).

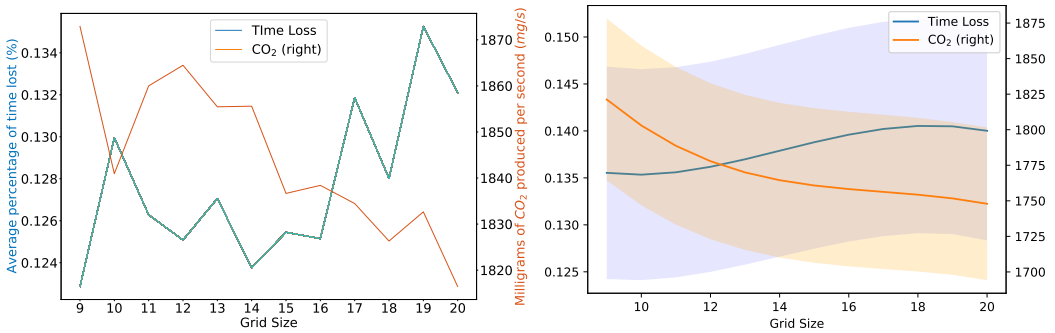
The resulting optimal points can be seen in Table 4. We observe that for both models the lowest output is found with minimum speed and maximum road length. Intuitively, navigating networks with shorter roads implies spending more time traversing intersections, which are a major source of delay and pollution in cities. The low speed limit benefits emissions as it implies that less trip time will be spent accelerating. We also see that more lanes benefit time loss while fewer lanes are better to optimise emissions. This is intuitive as lane changes are known to increase pollution [36]. Finally, the optimal acceleration to optimise emissions may be an unexpected result, but we know that a vehicle's emission efficiency curve is never zero. Thus, a lengthy, low acceleration trip may emit more CO_2 than a short, erratic one.

Table 4: Optimal parameters' configurations found with Bayesian Optimisation

Emulator	N_G	v_{max}	L_E	N_L	α_V	Output
Time loss	10	8	70	2	1.86	12.37
Emissions	20	8	61.36	1	2.96	1721.50
Average	15	8	65.68	1	2.28	

We investigate more deeply the grid size. We observe that the two models pick very different grid size values, this suggests that there could be a trade-off related to the grid size and that optimising this size for both objectives is pareto-optimal. For this purpose, we analyse the simulator and the emulators in the average optimal point shown in Table 4. In Figure 3 we observe that indeed this trade-off exists in the simulator and our emulators are able to capture it. This gives insights on the fact that when planning urban areas the size of the network is a crucial factor. Choices that benefit travel time may pose a danger for the environment.

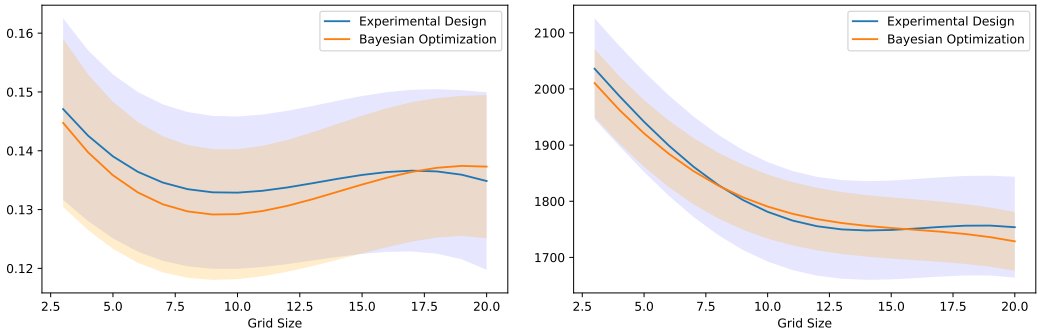
Finally, as we run Bayesian optimisation on the experimentally designed emulators, we aim to compare the difference between the emulators before and after the optimisation phase. In Figure 4 we see that for grid size the predictive variance decreases significantly and also the prediction mean is subject to modification. For further comparisons please refer to Appendix A.



(a) Simulator

(b) Emulator

Figure 3: Trade-off between CO_2 and time loss when varying the grid size in the average optimal point



(a) Time loss *(b) CO_2*

Figure 4: Experimental design and Bayesian Optimisation model comparison for grid size in the average optimal point

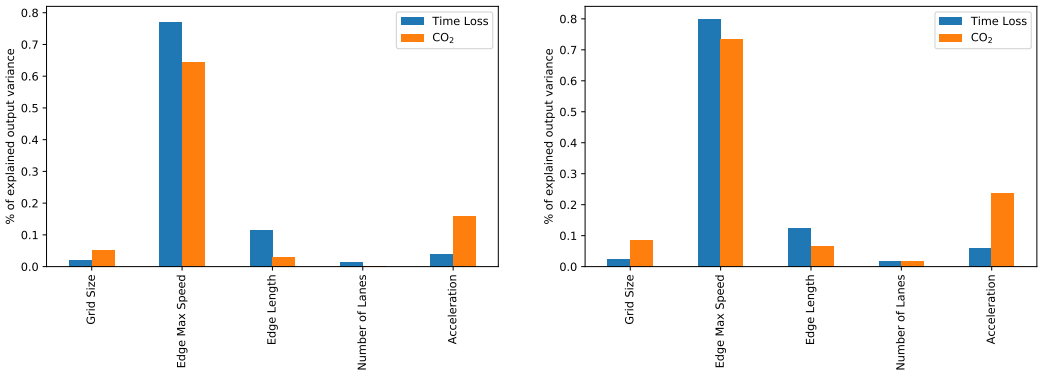
7 Sensitivity Analysis

In order to determine the significant inputs, sensitivity analysis is completed on the optimised emulators. Analyses are completed independently for time-loss and CO_2 emissions. This allows the important factors for each output to be discovered. OFAT and Global Sensitivity analysis are completed on each emulator.

The first-order indices and total effects of each input variable is calculated for both emulators. This is completed using a model-based Monte-Carlo approach with 500000 iterations. The OFAT analysis is conducted by randomly sampling 500 data-points for each input from the parameter space. All other parameters remain constant and are set to the average optimal operating point seen in Table 4.

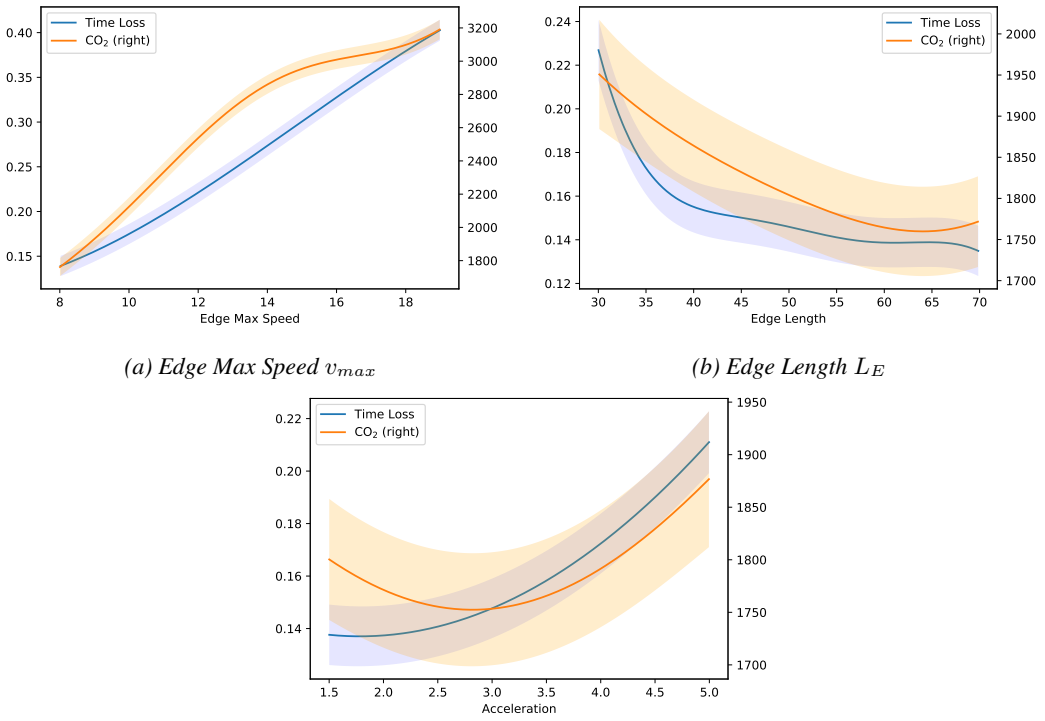
Figure 5 shows that edge max speed is the most significant input parameter to the system, as it contributes to more than half of the output variance. The other input parameters have very little impact on the system. Two exceptions are edge length in the time loss emulator, and acceleration in the CO_2 emulator. Figure 6 shows the OFAT analysis of these input parameters, in which it can be seen that time loss is proportional to emissions.

These results allow us to make interesting observations. High v_{max} in an urban environment results in both high time losses and emissions. Time loss is increased because vehicles reach road junctions faster, and thus have to wait longer to proceed on their journey, which emissions are increased due to extended acceleration periods. Rapid acceleration and deceleration is considered a poor and dangerous driving practice. Figure 6c shows that this behaviour results in more time loss than if one were driving moderately.



(a) Main Effects *(b) Total Effects*

Figure 5: Comparison of Sobol indices obtained from the time loss and CO_2 emulators



(a) Edge Max Speed v_{max}

(b) Edge Length L_E

(c) Maximum Acceleration α_V

Figure 6: OFAT comparison of the emulators at the average operating point. Filled areas show one standard deviation of model variance.

8 Future Work

Due to resource constraints, the emulators were developed using a limited number of initialisation and iteration datapoints. By increasing the number of datapoints used, the fidelity of the emulators can be increased.

Many cities are extremely large and dense, which can be simulated by larger grid sizes. Further research in this field can take these larger grids into account. Larger cities also incorporate multi-modal traffic. Future investigations in this area can include other vehicles such as public transport, pedestrians, and emergency vehicles.

To further test the advantages of microscopic traffic emulation, comparisons between mesoscopic traffic models and microscopic emulators could be done.

9 Conclusion

Two traffic emulators were designed for use in urban traffic management and road planning. These emulators are designed to model the time loss due to driving below the ideal speed, and CO_2 emissions. The emulators use gaussian processes as a surrogate model of a microscopic traffic simulator SUMO, developed using experimental design techniques. It was found that the Integrated Variance Reduction produced the models with the smallest RMSE using 20 randomly initialised points, followed by the experimental design loop with 500 iterations.

In order to find the minimum-output operating point for each emulator, Bayesian Optimisation with Expected Improvement point acquisition was used. The minimum operating points were very similar. The operating points are investigated using sensitivity analysis. It was found that the most significant parameter is the edge maximum speed, followed by edge length and maximum acceleration for time loss and CO_2 emissions respectively.

The analysis shows that low vehicle speeds in combination with long edges give ideal time loss and emission performance. Additionally, by increasing the number of road lanes, there is less time loss in the network. An important trade-off between time loss and emissions is also discovered when varying the network size. This pareto-optimality can lead to useful insights for urban planners.

References

[1] Jorge Bravo. Sustainable cities, human mobility and international migration, 2018.

[2] John Glen Wardrop. Road paper. some theoretical aspects of road traffic research. *Proceedings of the institution of civil engineers*, 1(3):325–362, 1952.

[3] Michael James Lighthill and Gerald Beresford Whitham. On kinematic waves ii. a theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 229(1178):317–345, 1955.

[4] BD Greenshields, JR Bibbins, WS Channing, and HH Miller. A study of traffic capacity. In *Highway research board proceedings*, volume 1935. National Research Council (USA), Highway Research Board, 1935.

[5] Florian Siebel and Wolfram Mauser. On the fundamental diagram of traffic flow. *SIAM Journal on Applied Mathematics*, 66(4):1150–1162, 2006.

[6] David James Wilkie, Jur Van den Berg, Ming Lin, and Dinesh Manocha. Self-aware traffic route planning. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[7] Srinivas Peeta and Hani S Mahmassani. System optimal and user equilibrium time-dependent traffic assignment in congested networks. *Annals of Operations Research*, 60(1):81–113, 1995.

[8] Carolina Osorio and Michel Bierlaire. An analytic finite capacity queueing network model capturing the propagation of congestion and blocking. *European Journal of Operational Research*, 196(3):996–1007, 2009.

[9] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Microscopic simulation of congested traffic. In *Traffic and granular flow'99*, pages 365–376. Springer, 2000.

[10] Daniel Krajzewicz, Georg Hertkorn, Christian Rössel, and Peter Wagner. Sumo (simulation of urban mobility)-an open-source traffic simulation. In *Proceedings of the 4th middle East Symposium on Simulation and Modelling (MESM20002)*, pages 183–187, 2002.

[11] Michael Balmer, Marcel Rieser, Konrad Meister, David Charypar, Nicolas Lefebvre, and Kai Nagel. Matsim-t: Architecture and simulation times. In *Multi-agent systems for traffic and transportation engineering*, pages 57–78. IGI Global, 2009.

[12] Jaume Barceló et al. *Fundamentals of traffic simulation*, volume 145. Springer, 2010.

[13] Harold J Payne. Freflo: A macroscopic simulation model of freeway traffic. *Transportation Research Record*, (722), 1979.

[14] Jaume Barceló, Esteve Codina, Jordi Casas, Jaume L Ferrer, and David García. Microscopic traffic simulation: A tool for the design, analysis and evaluation of intelligent transport systems. *Journal of intelligent and robotic systems*, 41(2-3):173–203, 2005.

[15] Wilco Burghout, Haris N Koutsopoulos, and Ingmar Andreasson. A discrete-event mesoscopic traffic simulation model for hybrid traffic simulation. In *2006 IEEE Intelligent Transportation Systems Conference*, pages 1102–1107. IEEE, 2006.

[16] Kay W Axhausen, Andreas Horni, and Kai Nagel. *The multi-agent transport simulation MATSim*. Ubiquity Press, 2016.

[17] Jaime Barceló and Jordi Casas. Dynamic network simulation with aimsun. In *Simulation approaches in transportation analysis*, pages 57–98. Springer, 2005.

[18] G Kotusevski and KA Hawick. A review of traffic simulation software. 2009.

[19] Wen Chen. New rbf collocation methods and kernel rbf with applications. In *Meshfree methods for partial differential equations*, pages 75–86. Springer, 2003.

[20] Robert B. Gramacy. *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences*. Chapman Hall/CRC, Boca Raton, Florida, 2020. `http://bobby.gramacy.com/surrogates/`.

[21] Bruno Sudret, Stefano Marelli, and Joe Wiart. Surrogate models for uncertainty quantification: An overview. In *2017 11th European conference on antennas and propagation (EUCAP)*, pages 793–797. IEEE, 2017.

[22] Supervised learning, December 2020. Page Version ID: 994773800.

[23] Emukit and Experimental Design, October 2020.

[24] David D. Lewis and Jason Catlett. Heterogeneous Uncertainty Sampling for Supervised Learning. In *Machine Learning Proceedings 1994*, pages 148–156. Elsevier, 1994.

[25] Jerome Sacks, William J. Welch, Toby J. Mitchell, and Henry P. Wynn. Design and Analysis of Computer Experiments. *Statistical Science*, 4(4):409–423, November 1989. Publisher: Institute of Mathematical Statistics.

[26] Jonas Močkus. On bayesian methods for seeking the extremum. In *Optimization techniques IFIP technical conference*, pages 400–404. Springer, 1975.

[27] Saman Razavi and Hoshin V. Gupta. What do we mean by sensitivity analysis? the need for comprehensive characterization of "global" sensitivity in earth and environmental systems models. *Water Resources Research*, 51(5):3070–3092, 2015.

[28] Navid Delgarm, Behrang Sajadi, Khadijeh Azarbad, and Saeed Delgarm. Sensitivity analysis of building energy performance: A simulation-based approach using OFAT and variance-based sensitivity analysis methods. *Journal of Building Engineering*, 15:181 – 193, 2018.

[29] Andrea Saltelli and Paola Annoni. How to avoid a perfunctory sensitivity analysis. *Environmental Modelling Software*, 25(12):1508 – 1517, 2010.

[30] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

[31] Ilya M. Sobol. Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. *Mathematics and Computers in Simulation*, 55(1):271 – 280, 2001. The Second IMACS Seminar on Monte Carlo Methods.

[32] Mordechai Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

[33] Andrei Paleyes, Mark Pullin, Maren Mahsereci, Neil Lawrence, and Javier González. Emulation of physical processes with emukit. In *Second Workshop on Machine Learning and the Physical Sciences, NeurIPS*, 2019.

[34] GPy. GPy: A gaussian process framework in python. `http://github.com/SheffieldML/GPy`, since 2012.

[35] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart Russell, and Anca Dragan. Inverse Reward Design. *arXiv:1711.02827 [cs]*, October 2020. arXiv: 1711.02827.

[36] Hesham Rakha, Michel Van Aerde, Kyoungho Ahn, and Antonio Trani. Requirements for evaluating traffic signal control impacts on energy and emissions based on instantaneous speed and acceleration measurements. *Transportation Research Record*, 1738(1):56–67, 2000.

Appendix A Comparing OFAT between experimental design and Bayesian Optimisation

In this appendix we report the comparison between the emulators before and after Bayesian Optimisation. On the left side time loss is shown while on the right sides CO_2 emission is shown.

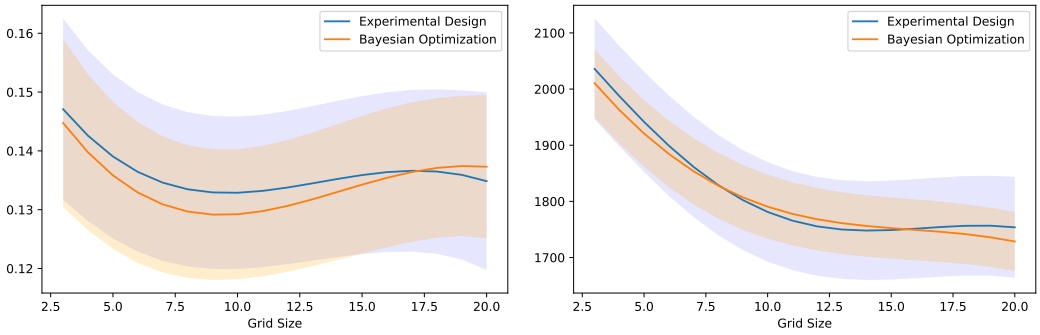


Figure 7: Grid size

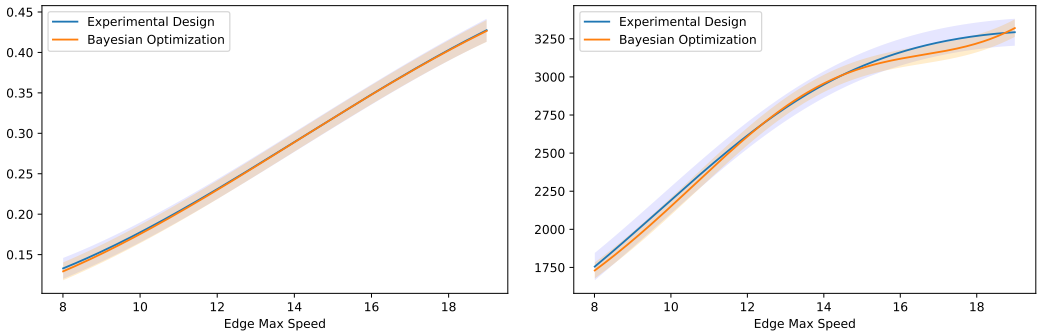


Figure 8: Road max speed

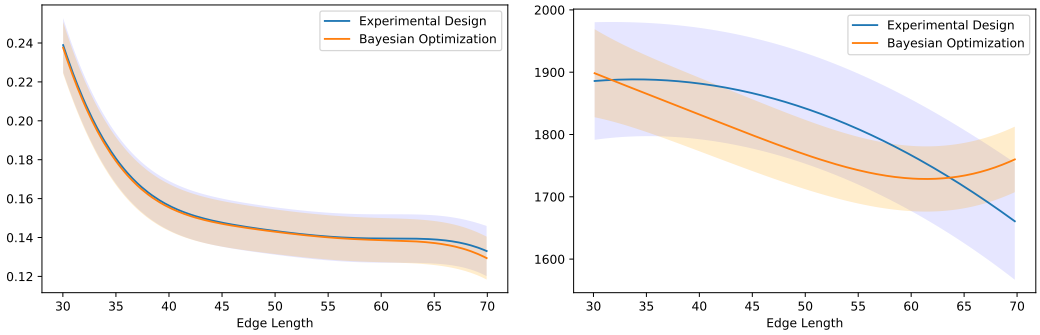


Figure 9: Road length

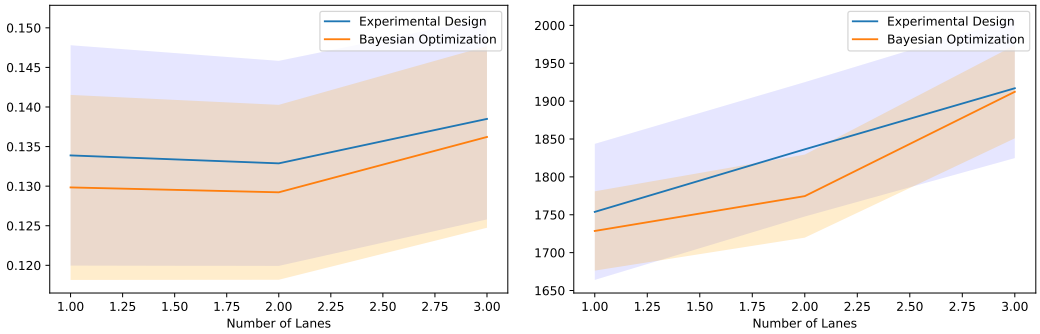


Figure 10: Number of lanes

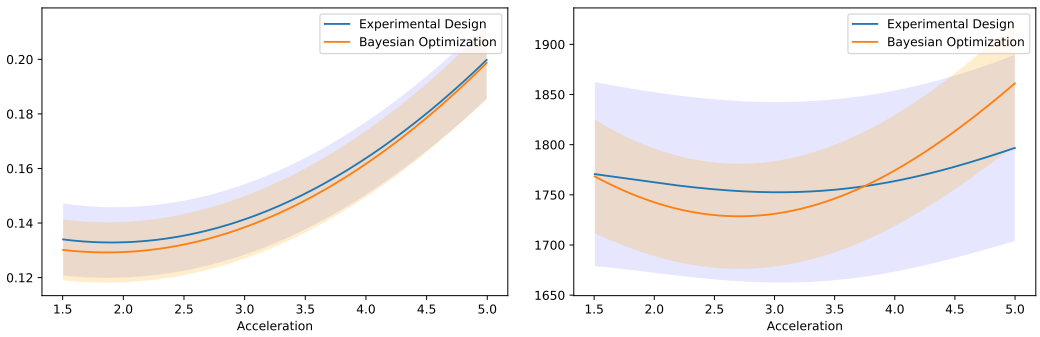


Figure 11: Acceleration