Simulating the Impact of Dynamic Rerouting on Metropolitan-scale Traffic Systems

CY CHAN, Lawrence Berkeley National Laboratory
ANU KUNCHERIA and JANE MACFARLANE, University of California, Berkeley

The rapid introduction of mobile navigation aides that use real-time road network information to suggest alternate routes to drivers is making it more difficult for researchers and government transportation agencies to understand and predict the dynamics of congested transportation systems. Computer simulation is a key capability for these organizations to analyze hypothetical scenarios; however, the complexity of transportation systems makes it challenging for them to simulate very large geographical regions, such as multi-city metropolitan areas. In this article, we describe enhancements to the Mobiliti parallel traffic simulator to model dynamic rerouting behavior with the addition of vehicle controller actors and vehicle-to-controller reroute requests. The simulator is designed to support distributed-memory parallel execution using discrete event simulation and be scalable on high-performance computing platforms. We demonstrate the potential of the simulator by analyzing the impact of varying the population penetration rate of dynamic rerouting on the San Francisco Bay Area road network. Using high-performance parallel computing, we can simulate a day in the San Francisco Bay Area with 19 million vehicle trips with 50 percent dynamic rerouting penetration over a road network with 0.5 million nodes and 1 million links in less than three minutes. We present a sensitivity study on the dynamic rerouting parameters, discuss the simulator's parallel scalability, and analyze system-level impacts of changing the dynamic rerouting penetration. Furthermore, we examine the varying effects on different functional classes and geographical regions and present a validation of the simulation results compared to real-world data.

CCS Concepts: • **Computing methodologies** → **Discrete-event simulation**; **Massively parallel and high-performance simulations**; **Agent/discrete models;**

Additional Key Words and Phrases: Large-scale transportation simulation, dynamic vehicle rerouting, high-performance computing, parallel discrete event simulation, actor-based modeling

ACM Reference format:
Cy Chan, Anu Kuncheria, and Jane Macfarlane. 2023. Simulating the Impact of Dynamic Rerouting on Metropolitan-scale Traffic Systems. *ACM Trans. Model. Comput. Simul.* 33, 1-2, Article 7 (February 2023), 29 pages.
https://doi.org/10.1145/3579842

This report and the work described were sponsored by the U.S. **Department of Energy** (**DOE**) **Vehicle Technologies Office** (**VTO**) under the Big Data Solutions for Mobility Program, an initiative of the **Energy Efficient Mobility Systems** (**EEMS**) Program. The following DOE Office of **Energy Efficiency and Renewable Energy** (**EERE**) managers played important roles in establishing the project concept, advancing implementation, and providing ongoing guidance: David Anderson and Prasad Gupta. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.
Authors' addresses: C. Chan, Lawrence Berkeley National Laboratory, 1 Cyclotron Rd., Berkeley, California 94122; email: cychan@lbl.gov; A. Kuncheria and J. Macfarlane, University of California, Berkeley, 109 McLaughlin Hall, Berkeley, California 94122; emails: {anu_kuncheria, janemacfarlane}@berkeley.edu.

1049-3301/2023/02-ART7 $15.00
https://doi.org/10.1145/3579842

1 INTRODUCTION

Active traffic management strategies using changeable message signs or broadcast media are regularly used by traffic management centers to present alternate routes to drivers when abnormal events occur on the roadways. Shifting traffic onto alternate routes maximizes the efficiency and capacity of the network and increases safety by reducing secondary vehicle accidents due to unexpected congestion. With highways, parallel corridors that handle the rerouted traffic must be available with adequate capacity for rerouting to be successful. This requires traffic centers, with humans in the loop, to assess the situation, have knowledge of the parallel routes via predefined control strategies, and implement an active control plan. In the past five years, smartphone navigation apps have gained popularity and serve a similar role. These applications use their current understanding of congestion to compute new routes in which the travel time is shorter for the user of the device. The congestion information is generated by aggregating the speeds and locations of all of the devices that the app is monitoring. A road congestion estimate is created using road network information and other traffic information, such as historical traffic information, and combined with the current user's destination. If the projected congestion of the user's current route significantly impacts their expected travel time, the app may suggest a new route to the user with a shorter travel time. As such, these navigation applications are also serving as active traffic managers.

The introduction of these new active traffic managers makes it more difficult for researchers and government transportation agencies to understand and predict the dynamics of congested transportation systems. Traffic simulation is a key capability for these organizations to analyze different scenarios and predict the potential impacts of different infrastructure changes, policies, or control strategies. However, the complexity of transportation systems makes them extremely difficult to simulate at the urban scale. As such, little is known about how to control and actively manage large-scale road networks in the presence of significant congestion, particularly with the active management now being implemented by different agents, such as smartphone apps and traffic centers. We do know that providing information about congested states and having a variety of sources provide new travel routes will likely create unpredictable patterns and dynamic variation. Control strategies implemented through traffic management can include infrastructure modifications such as signal-phase-timing adjustments of signals to redirect traffic and expedite congestion mitigation. These control decisions are determined by estimating the impact of the rerouting action. To-date, these do not include rerouting that occurs as a result of drivers following an app's routing information. This was made evident when during an evacuation event, drivers were venturing into dangerous situations with routes that were not informed by road closures associated with the event [11].

In the last two decades, transportation network simulation has increased in popularity for emulating driving behaviors, predicting traffic dynamics, and predicting impacts of control strategies. Applied models can broadly be categorized into both equilibrium models, often referred to as **traffic assignment** models, and non-equilibrium models, often referred to as **simulation models** [9, 43]. The simulation models complement traffic assignment, which is one of the essential tools that planners use for estimating congestion. With a reliable origin-destination demand matrix, an accurate traffic assignment generates optimized traffic-state predictions [43]. While equilibrium models have been very popular in the past due to mathematical clarity, their main drawback is the inability to capture the congestion-dependent evolution of a driver's route [30]. This limitation is overcome with non-equilibrium models that allow for modeling route guidance dynamics and unexpected events such as incidents or evacuation strategies [9, 22].

In order to predict the emergent traffic dynamics that result from congestion-dependent rerouting and unexpected events, we have extended our parallel discrete event simulation of large-scale

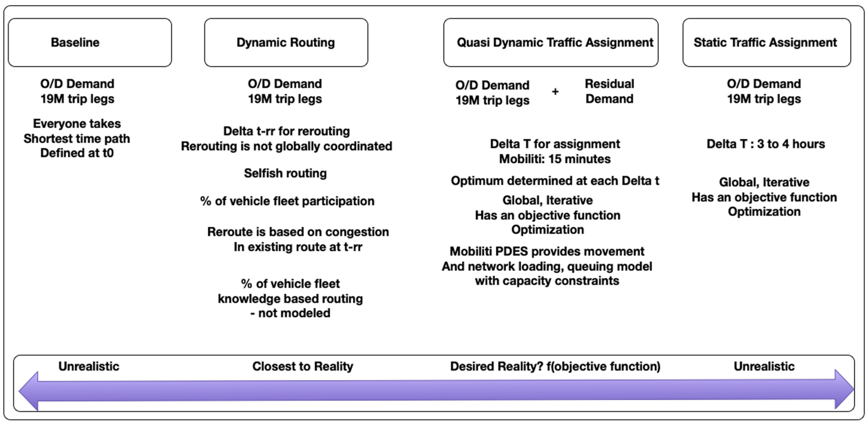


Fig. 1. Four traffic models for large-scale network modeling that can be employed for a wide array of transportation planning projects. While the first two employ non-equilibrium traffic assignment modeling, the last two use equilibrium based traffic assignment for route choice decisions.

traffic dynamics [14] with new dynamic vehicle rerouting capabilities. To understand the context of this article, Figure 1 describes four different traffic models we have developed for large-scale network modeling, each with a different set of algorithms. Details of the baseline model can be found in our previous article [14] and the quasi-dynamic traffic assignment model will be described in a forthcoming publication [15]. In this article, we focus on the second model with dynamic routing which we believe best captures realistic emergent traffic dynamics. Note that we acknowledge that there are a percentage of routes that are neither shortest free speed routes nor dynamically routed in the real-world. We denote them as *knowledge-based* routes, representing routes that are arbitrarily chosen by a user, such as a route chosen to include a café stop en route to work. At this time, we do not model these types of routes; thus, all 19 M routes are either shortest free speed routes or dynamically routed.

The ultimate goal of active traffic management is to drive the system towards an optimized state, which is the focus of most traffic assignment algorithms. But in reality, reaching equilibrium is highly unlikely regardless of the attempted active management strategies. As such, a mechanism for simulating and analyzing hypothetical scenarios that will reveal emergent dynamics with prescribed dynamic routing algorithms would be extremely valuable for those designing active traffic management systems. With increased urbanization in cities today, understanding how to actively manage the dynamics on the road network is becoming increasingly urgent, not only from a loss in productivity perspective but from a fuel consumption perspective. Our research focus is to breakdown the computational barriers of simulating large-scale road network dynamics in order to investigate the impacts of active traffic management strategies and reduce the multifaceted cost of the increasing congestion in our cities.

This article presents a methodology for representing the transportation system with dynamic rerouting capabilities in a scalable **parallel discrete event simulation** (**PDES**) formalism. We include a description of the core link actor model and its constituent components that calculate vehicle traversal times, timing constraints, and storage capacity constraints. We also introduce vehicle controller actors that monitor congestion within the traffic system and handle dynamic reroute requests from simulated vehicles. We demonstrate that the resulting mesoscopic model achieves scalable computational performance for a large system (19 million vehicles over a network with 1 million links representing the San Francisco Bay Area), simulating a day of traffic with 50 percent dynamic rerouting penetration in three minutes on 256 cores of the NERSC Cori computer at Lawrence Berkeley National Laboratory [39]. We provide a parallel performance and scalability analysis, a rerouting parameter sensitivity analysis, a discussion of the simulated impacts of varying the dynamic rerouting penetration rate on various transportation system metrics, and finally, present a validation of the simulation results compared to real-world data sources.

2 RELATED WORK

There has been much previous work in the area of transportation system modeling and simulation. Previous simulators can be broadly classified as macroscopic, mesoscopic, and microscopic based on the level of detail of the behavior of individual agents simulated. The level of behavior detail in the model typically also correlates to the size of the area under simulation—larger area simulations tend to use macroscopic models, while simulations focusing on small areas use more detailed microscopic models. **Dynamic traffic assignment** (**DTA**) solvers and dynamic simulators such as MATSim [38], POLARIS [3], DTALite [58], DynaMIT [10], DynaSmart [35], Aimsun [13], SUMO [34], INTEGRATION [42], BEAM [46], Ugirumurera et al. [51], MANTA [56], and many others fall along the spectrum of different modeling approaches, target problem scales, modeling fidelities, and output (e.g., traffic assignment versus simulation). The level of fidelity targeted by our Mobiliti simulator is mesoscopic since it does not resolve lane-changing or vehicle following behavior, although it does resolve individual vehicle movements and rerouting decisions.

There is a large body of previous work in the area of **parallel discrete event simulation** (**PDES**, see [23] for an overview). In particular, seminal work by Chandy and Misra [16], Bryant [1], and Jefferson [28] laid the groundwork for parallel discrete event simulation, which was shown to run efficiently on supercomputers [7, 8]. Previous traffic simulators that utilize PDES include those by Perumalla [40] and Yoginath [57], who used optimistic PDES for modeling traffic grids, though they evaluated their system on smaller-scale synthetic grid networks and used a different mechanism to mediate congestion among vehicles. Thulasidasan et al. [48] also modeled road networks using queue-based parallel discrete event simulation, but they did not include adaptive vehicle *rerouting* behavior in their model, where vehicles can respond to congestion by rerouting *in the middle* of (rather than just at the beginning of) their trip. Furthermore, our article includes a description of the design and implementation of a *dynamic vehicle rerouting system* with a parameter sensitivity analysis for vehicles and controllers at a large-scale.

In the area of dynamic re-routing, research by Liang and Wakahara [33] showed how proactive dynamic re-routing could reduce average travel time for congested road networks using the SUMO micro simulator on a medium-sized area of London (3,002 links and 332 nodes with 954 vehicles). Zhao et al. [58] gave a theoretical analysis of the dynamics and stability of equilibrium with travelers that reroute depending on cost difference, and demonstrated their results on some small networks (up to 31 nodes and 40 links with three OD pairs). Similarly, [52] provides a theoretical treatment of distributed multi-agent route selection problem with incentives, with numerical examples of their approach on the Sioux Falls road network (24 nodes). In [49], the authors utilize SUMO and OMNeT++ to model an area of Brooklyn with 380 nodes and 474 links, with a traffic demand of 2,500 vehicles. Their approach is similar to ours (albeit on a smaller network) in that they utilize a distributed cloud-based vehicle control system, where sensors detect congestion, and routes are suggested to avoid the congestion. In [31], the authors utilize a combination of SUMO, Veins, and OMNeT++ tools to analyze the impact of rerouting on a 2×2 grid map (9 nodes, 24 links). In [41], the authors describe a vanpool scheduler architecture for dynamic rerouting. They evaluate their approach on a network of the Munich city center, where the focus is more on responding to stochastic vanpool requests rather than dynamic congestion effects. In [32], they investigate the information comply model to simulate how drivers can react to information about an event and evaluated their approach on a network with approximately 1,000 links. There has also been related work in the area of route selection in the context of dynamic traffic assignment (e.g., [4]). Dynamic traffic assignment approaches typically model converged dynamic driver behavior adapted to daily congestion patterns, rather than more reactive dynamic rerouting scenarios that explore how traffic can respond to unexpected events. Our approach differs from these previous works in that we leverage high-performance computing techniques to simulate the effect of dynamic rerouting on much larger road networks (on the order of hundreds of thousands to millions of links) with tens of millions of vehicle trips per day to resolve system-wide impacts on large urban regions.

Other prior work in utilizing HPC for simulating traffic systems have focused on various aspects of parallelization and performance optimization. In one early article on this subject [12], they describe two parallel implementations for microscopic simulation on a data-parallel (vector) architecture and a message passing architecture, simulating a road network with up to 20,000 miles and 200,000 vehicles up to 2.7× real-time speeds. An improved geographic recursive bisection algorithm was presented in [53] for computing the parallel domain decomposition so that the workload is better balanced across processors. They evaluated their approach on a small network with 232 roads and 670 connections between roads, and observed a parallel speedup of up to 4.1× using 336 CPUs. In [54], they focus on evaluating the impact of a dynamic load balancing strategy for reducing computational load imbalance and improving parallel scalability, showing that parallel speedup improved from 3.1× to 4.2× for a network with 80,000 links and 1.2 million trips using 32 compute cores. Following that work, they introduced a method for reducing the overheads of conservative synchronization in a nanoscopic simulation to improve performance by relaxing the causality of simulated events [55]. In that article, they used a similar sized network of 80,000 links on a compute cluster consisting of three compute nodes, each with 16 cores, and evaluated the impact of relaxing causality on the accuracy of the simulation.

In Reference [24], a parallel simulation with a dynamic route solution module is presented for a test network of 62 links, showing scaling performance up to 6 processors with a parallel speedup of up to 2.3× versus serial. Their parallel dynamic rerouting methodology, where vehicle routes are iteratively updated in a synchronous fashion, does not simulate the individual request/response communication mechanism that occurs between the vehicles (or smart navigation devices) and the vehicle controller actors, which serves to aggregate network congestion data and handle rerouting requests. Since the routes are solved iteratively and synchronously, their methodology is more appropriate for solving the traffic assignment problem, rather than studying the sensitivities in a dynamic rerouting mechanism (e.g., frequency of reroute checks, thresholds for rerouting). Furthermore, they evaluated their method on a network with 62 links for up to one hour of simulated time, which they reported takes over 15 minutes of computational time and reaches its maximum parallel speedup with 4 single-core processors (the execution time actually *increases* when more than 4 processors are used). This is quite different in scale compared to the simulated San Francisco Bay Area system we examine in this article, which contains over one million links with 19 million vehicles (each with its own specific origin and destination nodes), and scales to much higher core counts, achieving a 146× parallel speedup on 256 compute cores.

In Reference [6], the authors study the impact of dynamic rerouting (re-planning) in the context of VANets using SUMO and evaluate the impacts of dynamic rerouting on both mobility and ad-hoc network connectivity. In contrast to our work, their analysis did not formulate the dynamic rerouting mechanism as a distributed process that involves coordination between link and controller actors on different compute nodes in a parallel computing environment. SUMO uses a serial simulation engine, so all data is generated and available to the one thread executing at all times, whereas in a distributed memory platform, messages must be sent between compute nodes to share their simulation state. Moreover, that work did not investigate the sensitivity of additional parameters such as setting a minimum time savings threshold on the new route before taking a reroute action.

Our article goes beyond these prior works by (1) focusing on the design, implementation, and evaluation of dynamic vehicle rerouting in the context of a distributed-memory traffic simulation (which requires passing messages to share state between compute nodes), and (2) evaluation, performance analysis, and parameter sensitivity study of a large-scale HPC simulation with millions of road network links and tens of millions of vehicle trips. The problem of modeling dynamic vehicle rerouting behavior adds considerable computational load and model complexity to the simulation to coordinate link cost updates and compute new vehicle routes. Traffic simulation systems use a variety of models and algorithms based on the problem that is being addressed. Geographical scale, modeling fidelity, time horizon, and desired output data all lead to a varied landscape of simulation tools. However, to the best of our knowledge, this article presents the first simulation framework to model the impact of dynamically rerouting vehicles in a large urban region at scale with millions of nodes and links and millions of vehicles in short execution times, as well as present a rerouting parameter sensitivity discussion and detailed model validation analysis.

3 MOBILITI SIMULATION

Mobiliti is a distributed-memory, parallel simulation framework that runs on high-performance computing platforms. It uses optimistic PDES based on the Time Warp [28] protocol to model the traffic congestion in the network and the rerouting behavior of vehicles in the system. The optimistic nature of the simulator allows events to be speculatively executed and then rolled back if needed to maintain causality in the simulation. In order to avoid excessive synchronization overheads and enable a high degree of parallelism, our optimistic PDES simulator utilizes a computational design similar to the **actor model** of concurrent computation [2, 5]. In the actor model, the system is composed of a set of actors, which are entities that can execute concurrently with respect to one another and can only obtain information from each other through message passing. Moreover, with the arrival of each message, an actor may do computation that modifies its *local* state and may send new messages to other actors. There is no shared *global* state, thus avoiding the need for expensive locks, mutexes, or atomics and enabling distributed-memory parallelism.

In our optimistic PDES simulator, the entire state of the simulated system is divided among entities (called **logical processes** in the PDES literature [23]), which correspond to the actors in the actor model. The logical processes send events to one another, which correspond to the messages in the actor model. As with the actor model, each logical process may only do computation that updates its local partition of the system state (again, no shared global state) and send new events to other logical processes. A key distinguishing feature is that our simulator also utilizes a synchronization protocol (Time Warp) to enforce the correct execution ordering of events, even if the events arrive out of order over the network. For the remainder of this article, we will refer to the logical process entities as *actors* since that term is more familiar to researchers outside the PDES community.

The overall workflow of the simulator is presented in Figure 2, which shows a concise subset of the inputs, stages of initialization and simulation, and outputs produced by the simulator. During the simulation phase, the links are modeled as actors, and the vehicles are represented as events that are passed from link to link as they travel through the road network. Figure 3 illustrates the representation of network links as actors passing vehicle events between them.

For background on the simulator's implementation, please see our previous article [14]. This section describes relevant updates to the simulation that improve the accuracy of the link model and enable vehicle dynamic rerouting capabilities. Specifically, we have added mechanisms inside the link actor to enforce more accurate link timing constraints and storage capacity constraints,

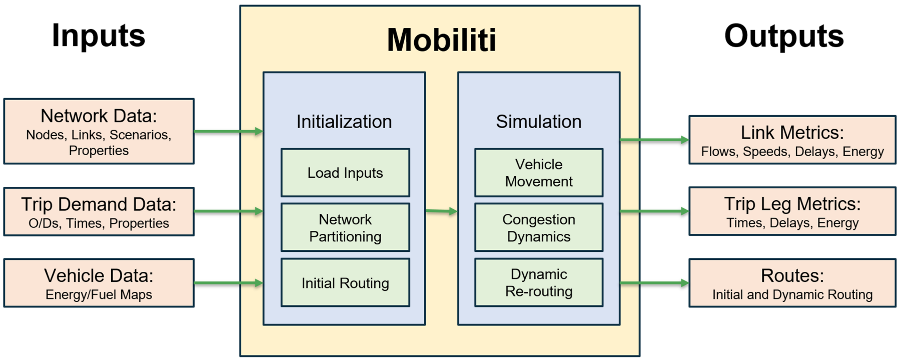


Fig. 2. The Mobiliti simulator workflow. A subset of the inputs and outputs are shown along with a subset of the software stages and modules contained within Mobiliti.

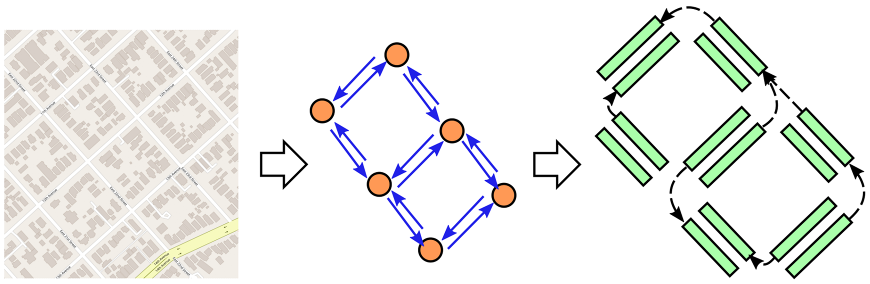


Fig. 3. The Mobiliti simulator represents the road network as a collection of link actors that send events to each other representing individual vehicles traversing the network. Vehicles are initialized at their origin nodes and propagated from link to link until they reach their destination. Links are responsible for computing the vehicles' traversal times based on the link's congestion model.

and we have added a new set of vehicle controller actors that can be queried by vehicles to compute better routes using the current congested state of the network.

3.1 Computational Link Model and Representation

When a vehicle event arrives at a link actor at time T_0, the link actor is responsible for determining the simulated time that the vehicle transitions to the next link on its route. As shown in Figure 4, the link actor utilizes three sub-models to determine the vehicle's transition time: a flow-based congestion delay model, a link timing model, and a storage constraint model that limits the occupancy on each link based on physical dimensions.

The flow-based congestion delay model (Figure 5) uses characteristics of the link (such as the designated flow capacity) and the current activity on the link to estimate the time ΔT_1 it takes for vehicles to traverse from the beginning to the end of the link. The resulting time $T_1 = T_0 + \Delta T_1$ represents when the vehicle reaches the end of the link; however, the vehicle may be delayed further before actually transitioning to the next link in its route due to additional timing and storage capacity constraints.

The link timing model computes the times at which vehicles may legally traverse from the current link to the next. Separate internal queues are maintained for each maneuver through the intersection (i.e., each downstream link sequence) so that vehicles utilizing different signal phases

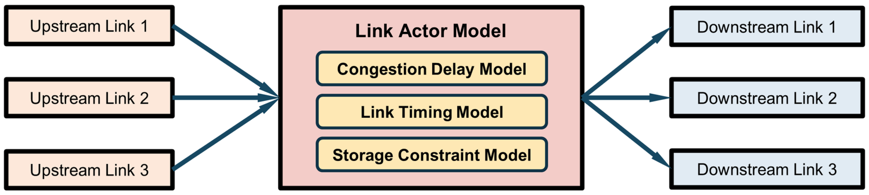


Fig. 4. Each link in the simulation is modeled with a Link Actor object that utilizes three sub-models to resolve the various dynamics of the road network.

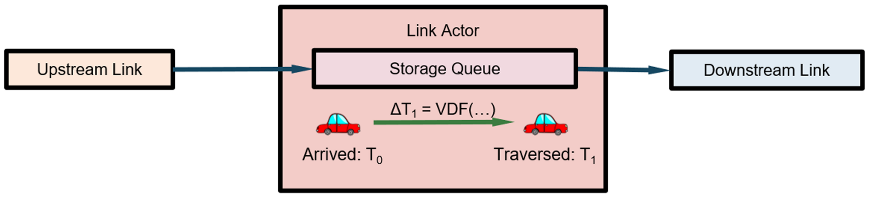


Fig. 5. The Congestion Delay Model consists of a **vehicle delay function** (**VDF**) that uses link characteristics (such as designated flow capacity) and current activity (such as flow rate) to compute estimated traversal times ΔT_1.

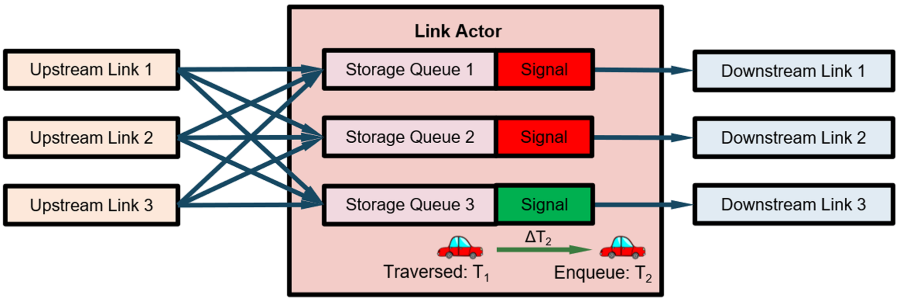


Fig. 6. The Link Timing Model assigns transition request times based on minimum vehicle spacing requirements and signal phase timing information.

can be handled independently. Figure 6 shows a diagram of the internal queue data structures each link actor utilizes to track the vehicles currently occupying it. The timing model for each link is fully parameterized so that different, coordinated signal timing plans can be used at multiple intersections in the simulation.

The link timing model actually ensures two things: (1) that vehicles do not transition from link to link at a rate that is faster than physically allowable (determined by the product of vehicle speed, density, and a number of lanes), and (2) that vehicles obey the traffic signals (if there is one at this link). For each queue within the link actor, it keeps track of the previous vehicle's transition time so that the next vehicle traversal maintains a minimum time spacing between transitions to the downstream link. Furthermore, if the intersection has a traffic signal, vehicles may only transition to the downstream link when the corresponding signal phase is green. When combined with the minimum spacing requirement, each green phase is modeled as a time interval with a fixed number of consecutive vehicle *slots* during which no more than that fixed number of vehicles may transition to the downstream link. For each vehicle, the timing model assigns a time

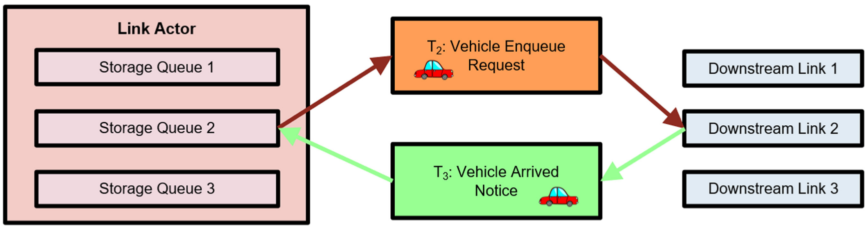


Fig. 7. The Storage Capacity Constraint Model ensures that vehicles only transition from the upstream link to a downstream link if the downstream link has sufficient physical storage capacity to accept the vehicle.

Table 1. Tunable Dynamic Rerouting Parameters

Parameter	Description	Nominal Value
t_{lsu}	Absolute link status update threshold	60 s
r_{lsu}	Relative link status update threshold	1.0
t_{check}	Reroute check interval	300 s
t_{delay}	Absolute reroute threshold	120 s
r_{delay}	Relative reroute threshold	0.2

$T_2 = T_1 + \Delta T_2$ that obeys the above constraints. For our current experiments, we utilized the timing model only to enforce the maximum rate for vehicle arrivals at each link since we do not currently have comprehensive signal timing and phase information for the San Francisco Bay Area network. However, the link capacity parameters used in the vehicle delay functions in the simulator should reflect the lower capacity constraints for links that contain signals.

Finally, the storage capacity constraint ensures that links do not allow more vehicles to occupy the link than there is physical space. The storage capacity constraint is mediated by a system of `EnqueueRequest` and `ArrivedNotice` events between the upstream and downstream links. After a vehicle's preliminary link traversal time T_2 is calculated using the congestion delay and timing models, the link actor sends a vehicle enqueue request event to the corresponding downstream link actor at its requested transition time T_2. However, the vehicle remains in the upstream link's storage queues until a response is received. Only when there is sufficient capacity on the downstream link to accept the incoming vehicle does the downstream link send an `ArrivedNotice` event to the upstream link (at time $T_3 = T_2 + \Delta T_3$) to notify the upstream link that the vehicle has made the transition. At that time, the upstream link may remove the vehicle from its storage queues, potentially freeing up space to send another `ArrivedNotice` event further upstream. Figure 7 illustrates the described protocol that coordinates the transition of vehicles between connected link actors. The final traversal time for the vehicle over the link is $\Delta T = \Delta T_1 + \Delta T_2 + \Delta T_3$.

3.2 Dynamic Rerouting Design

This section details the simulation mechanism that controls how vehicles dynamically reroute in response to system congestion. The subset of vehicles that have dynamic rerouting enabled (e.g., those with navigation devices) is specified at program initialization, determining the population rerouting penetration rate. In order to simulate dynamic rerouting behavior in the system, we implemented an additional set of actors and events that are responsible for coordinating when and how vehicles reroute. These include `VehicleController` actors, `LinkStatusUpdate` events, and vehicle `RerouteCheck` events. These new actors and events are described in the following sections, and relevant parameters are summarized in Table 1.

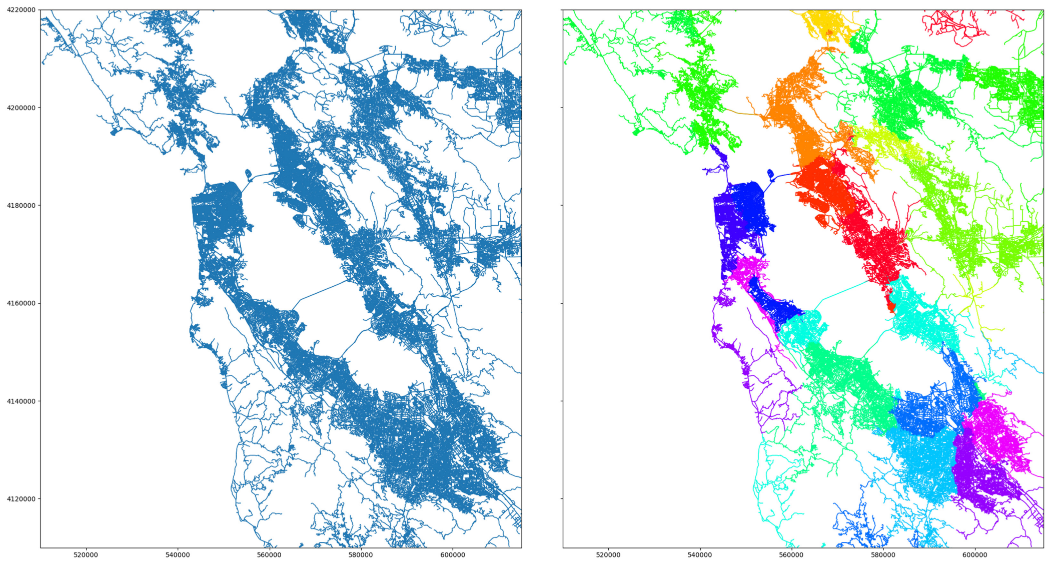
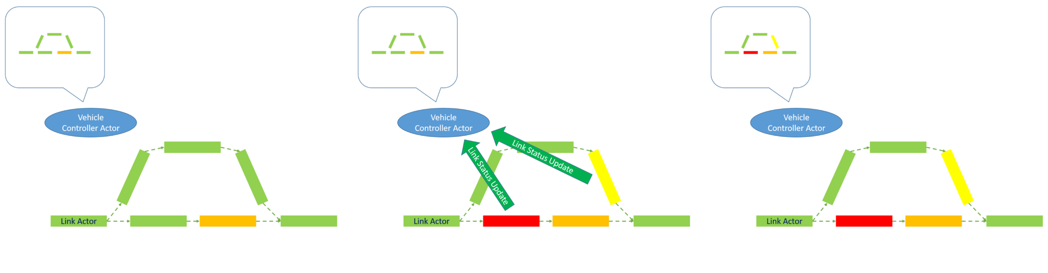


Fig. 8. An example partitioning of the SF Bay Area road network into sixteen subgraphs. Mobiliti partitions the road network into subgraphs and assign one to each computational rank in the simulator. The partitions are load balanced to enable parallel scalability, and each partition is assigned its own `VehicleController` actor to service rerouting requests from vehicles currently in the partition.

3.2.1 `VehicleController` *Actors.* `VehicleControllers` are a new class of actors that can be queried by vehicles to check whether the vehicle should change its route based on current congestion conditions to get to its destination more quickly. There are multiple controllers instantiated throughout the road network, and since we already partition the road network across simulator ranks (threads) for parallel execution, we use the same partitioning for the `VehicleControllers` in our experiments. The network partitioning is computed using the METIS graph partitioner [29] by constructing a *line graph* (or *edge graph*) [26] representation of the road network, where the nodes of the graph are the road links, and the edges of the graph are link-to-link connections. The weights of the nodes are based on the corresponding link actor's compute load (i.e., the number of events the actor must compute), and the weights of the edges are based on the communication load between the corresponding link actors (i.e., the number of events sent between the actors). The METIS partitioner uses heuristic techniques to simultaneously balance the total node weight (i.e., compute load) assigned to each partition and minimize the total weight of the edges cut by the partition boundaries (i.e., communication load).

While each individual controller is only responsible for servicing requests originating from vehicles *within* its partition, it maintains current congestion data across the *entire* network, so newly computed routes take the state of the whole system into account. Figure 8 shows an example of how the San Francisco Bay Area road network could be partitioned into sub-networks. A distinct `VehicleController` actor is assigned to each sub-network shown in different colors. When vehicles check whether to reroute, they contact the `VehicleController` assigned to the vehicle's current road partition. This method preserves geospatial locality within the model and also has good computational behavior since the `VehicleControllers` responsible for routing calculations are parallelized across compute resources in a similar manner that link actors are parallelized.

3.2.2 `LinkStatusUpdate` *Events.* In order for the `VehicleControllers` to have an up-to-date picture of current congestion conditions to make rerouting decisions, each link has a new sub-



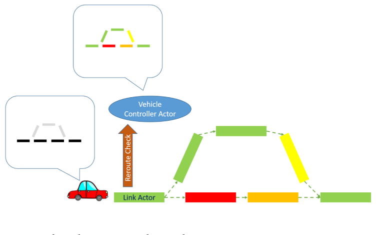
(a) Initial controller and link states **(b)** `LinkStatusUpdate` events **(c)** Updated controller state

Fig. 9. In (a), the `VehicleController` actor's knowledge of the congestion on the links is up to date. When the congestion on a link changes during simulation, the link actor sends `LinkStatusUpdate` events to the `VehicleController` actors to notify them of its current condition. Figure (b) illustrates an example where two links experience increased congestion and send updates to the `VehicleController` actor. The controller then updates its knowledge of the graph (c) and uses this updated information to calculate routes for rerouting vehicles.

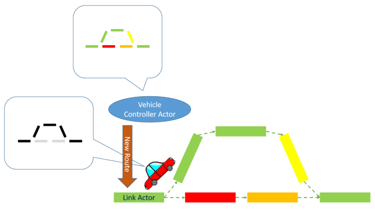
component that sends updates about its current congestion status to the `VehicleControllers`. Figure 9 illustrates an example where two link actors send status updates to a `VehicleController`, which then updates its knowledge of the road network's congested link traversal times.

Every time a vehicle departs a link l, the link actor broadcasts an update to **all** `VehicleControllers` with its new congested traversal time if it differs from the previously sent traversal time by at least $\min(t_{\text{lsu}}, r_{\text{lsu}} \cdot t_f(l))$, where t_{lsu} is an absolute threshold, r_{lsu} is a relative threshold ratio and $t_f(l)$ is the link's freespeed traversal time. By allowing some deviation, the thresholds prevent links from sending excessive status updates while ensuring that the values used by the `VehicleControllers` for rerouting are still close to the current value experienced on the link. By making the thresholds tighter, we can increase the frequency that `LinkStatusUpdate` events are sent to the `VehicleControllers`, potentially improving rerouting accuracy at the cost of processing more update events. The sensitivity to these threshold parameters is explored in Section 3.3. Finally, each link actor **that has active traffic** periodically sends a *heartbeat* update to the vehicle controllers to indicate that it is still servicing traffic at a given speed. When a vehicle controller has not heard from a link in more than the heartbeat period, it knows the link has not had any traffic, and thus it can purge stale congestion data about the link from its database.

3.2.3 `RerouteCheck` *Events.* When a vehicle arrives at a link, a vehicle can query the local `VehicleController` with a message that contains the vehicle's current path p (sequence of links) from its present location to its destination (see Figure 10). The controller estimates the total congested time $t_c(p)$ on the vehicle's path to its destination using its knowledge of current network congestion. If the delay on its path $d(p) = t_c(p) - t_f(p)$ exceeds a threshold $\max(t_{\text{delay}}, r_{\text{delay}} \cdot t_f(p))$, where t_{delay} is an absolute threshold, r_{delay} is a relative threshold ratio, and $t_f(p)$ is the freespeed traversal time on path p, then the controller calculates a new shortest path p' from the vehicle's current location to its destination. If the new path time $t_c(p')$ improves the vehicle's expected arrival time by more than $\max(t_{\text{delay}}, r_{\text{delay}} \cdot t_c(p)$, then the vehicle accepts the new path p'; otherwise, it continues on its existing path p. We assume a 100 percent compliance rate with the route suggestion given by the `VehicleController` (with no delay) since our simulator does not include human behavior models at this time. A lower compliance rate may be approximated by adjusting the population rerouting penetration rate accordingly. Finally, in order to prevent excessive reroute queries, the vehicle only sends a `RerouteCheck` event if it has been more than t_{check} seconds since its last check. For our experiments, the `VehicleController` actors are homogeneous,



(a) Vehicle arrival and RerouteCheck query



(b) RerouteCheck response with new route

Fig. 10. Example RerouteCheck event for dynamic rerouting a vehicle based on current congestion conditions. Since the vehicle's original route is congested, the controller sends an alternate route around the congestion, which the vehicle then takes.

simulating a unified network control agency. They could also be configured heterogeneously to simulate the impact of having multiple companies with different (imperfect) network information servicing vehicle reroute requests. This is a subject for future investigation.

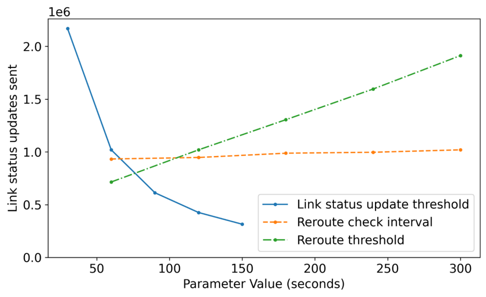
3.3 Model Parameter Selection and Sensitivity

The various parameters described above and summarized in Table 1 control various aspects of the system's rerouting behavior and can be tuned to explore how the system would hypothetically behave with different parameterizations. The parameters can be grouped into three categories: (1) t_{lsu} and r_{lsu} control the frequency of link status updates, where lower values result in more update messages and provide the controllers with more accurate information; (2) t_{check} controls the frequency that vehicles contact the controllers to see if they should reroute, where lower values result in more requests; and (3) t_{delay} and r_{delay} control how aggressive the controllers are at rerouting vehicles, where lower values result in additional reroutes in situations with smaller time saving benefits.

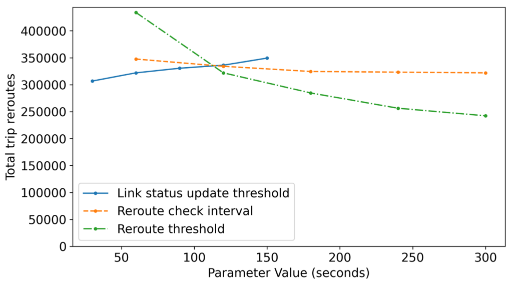
In order to understand the sensitivity to these parameters, we conducted parameter sweeps for each of these three groups, where we simultaneously varied the values of the parameters in each group, while keeping the other parameters constant to isolate the impact of each group of parameters. Table 1 shows the baseline parameter values. In the first parameter sweep shown in Figure 11, we varied the link status update thresholds: t_{lsu} (on the x-axis) from 30 seconds to 150 seconds in increments of 30, and set $r_{\text{lsu}} = t_{\text{lsu}}/60$. As expected, the number of link status updates sent (a) decreases significantly as the thresholds increase; however, the impact on the number of trip leg reroutes (b) and on the total system delay (c) is relatively small, indicating that there is little benefit from sending very frequent updates, and that the vehicle controllers do a satisfactory job even with coarse information.

In the second parameter sweep shown in Figure 11, we varied the minimum time interval between a vehicle's reroute check queries: t_{check} (on the x-axis) from 60 seconds to 300 seconds in increments of 60. As t_{check} increases, vehicles check whether they should reroute *less* frequently, but we observe that the total number of trip reroutes (b) declines only modestly. This is because the vast majority of vehicles that reroute only have to check and switch to a better route **once**, and then it typically sticks to the new route (e.g., with 60% rerouting penetration, 98.2% of rerouted trips only reroute once during its journey). Thus, increasing the time interval between checks mostly results in a small delay in the timing of a switch to a better route. As a result, the impacts on the number of link status updates (a) and total system delay (c) are also relatively minor.

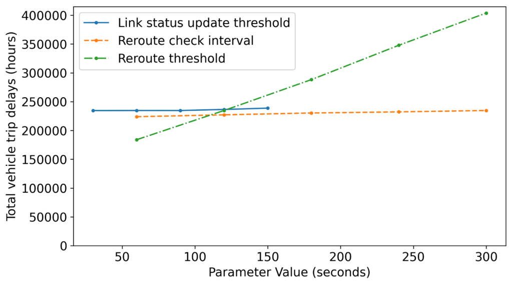
In the third parameter sweep shown in Figure 11, we varied the reroute delay thresholds for when a vehicle should reroute: t_{delay} (on the x-axis) from 60 seconds to 300 seconds in increments



(a) Link Status Updates Sent



(b) Trip Leg Reroutes



(c) Total (System) Vehicle Delay

Fig. 11. Parameter sensitivity sweeps for link status update thresholds, rerouting check frequency, and rerouting delay thresholds. The X-axis is the value (in seconds) of the parameter being varied: link status update threshold (t_{lsu}), reroute check interval (t_{check}), and reroute threshold (t_{delay}). The Y-axes shows the impact of varying the parameters on three system metrics: (a) link status updates sent, (b) trip leg reroutes, and (c) total (system) vehicle delay.

of 60, and set $r_{\text{delay}} = t_{\text{delay}}/600$. As t_{delay} increases, the vehicle controllers are less aggressively rerouting vehicles, keeping them on their original paths until their current path congestion is higher and their best alternative route saves them more time. This is directly seen in (b) as the total number of trip reroute requests decreases significantly as the thresholds increase. The number of link status updates (a) increases due to more dynamic variation in congestion, as the controllers are less effective at balancing traffic among available alternatives. The total system delay (c) increases since more vehicles are staying on less optimal routes, thus increasing their delay. While the smallest threshold resulted in the best system efficiency, it is debatable whether using a very low delay threshold in the real-world with human drivers is desirable because of the cognitive cost in asking a driver to alter their route.

For the experiments in the next section, we used the parameter values in Table 1, which provide a good balance between reality and improvement in system congestion. The parameters with the largest effect on total system delay were the reroute threshold parameters (t_{delay} and r_{delay}). For these, we selected t_{delay} equal to 2 minutes and r_{delay} equal to 0.2 as a compromise between system efficiency and excessive rerouting.

4 EXPERIMENTS

4.1 Experimental Methodology Description

In order to demonstrate and evaluate our Mobiliti simulator, we followed the following steps to model and analyze road network traffic in the San Francisco Bay Area (the steps would be similar for other areas). First, we obtained the necessary network, trip demand, and vehicle data inputs shown in Figure 2, left. After obtaining the inputs, we run the Mobiliti simulator (Figure 2, center) for a variety of parameter configurations, including sweeping the dynamic rerouting penetration rate, the link status update threshold, the reroute check interval, and the reroute threshold. The simulator outputs include link-level metrics such as vehicle counts, average speeds, and congestion delay series; leg-level metrics such as trip times and delays; and rerouting information such as when and where rerouting occurs. We then process the simulator logs and outputs (Figure 2, right) using a combination of software tools (such as Python) to conduct the various analyses presented in the following subsections.

The road network model was derived from a HERE Technologies [27] map consisting of 454,651 nodes and 1,008,959 links spanning from Santa Rosa, Napa, and Vacaville to the north, San Jose to the south, and Oakland, Hayward, Fremont, and Livermore to the east (see Figure 8). While the link actor model currently supports the signal timing mechanism described in Section 3.1, the following experiments were run without detailed signal behavior due to a lack of comprehensive, accurate location and timing data for all of the signals in the Bay Area. However, the link capacity properties in the input road network already take into account the presence of signals and are used in each link actor's Congestion Delay Model (see Section 3.1) to compute vehicle traversal times, thus the simulator slows vehicles according to those flow capacity values compared to having no signals.

The trip demand is initialized from an input file with 19 million trip legs (origin/destination pairs) based on disaggregate simulated trip records from the **San Francisco County Transportation Authority** (**SFCTA**) SF CHAMP 6.1 model [45]. Each trip leg is specified with origin and destination travel analysis zones (chosen from 40,000 micro-analysis zones) and a start time. Since our simulator models each individual vehicle traversing from link to link at discrete times, we chose specific origin and destination nodes within the given **Traffic Analysis Zones** (**TAZs**). We weighted each node by its nearby population density derived from the **Global Human Settlement** (**GHS**) database [21] to avoid choosing nodes that are in very sparsely populated regions of the map, which would unrealistically send traffic to remote areas. We also avoided selecting freeway or ramp nodes as origins or destinations. Figure 12 shows the population density map we used for initializing our simulated trip legs. Note that the coarse granularity of the heat map is a result of the resolution of the provided dataset (250 meters).

Figure 13(a) shows the temporal profile of the SFCTA demand model's trip legs during the simulated model day. Our simulation runs a single model day, and the figure shows on the y-axis the number of trips that start in each hour of the day. The number of trip legs per hour varies from very low in the early morning hours to very high during the late afternoon rush hour. Since trip lengths can vary considerably, by weighing each trip by its length, Figure 13(b) shows the total ***vehicle miles traveled*** (**VMT**) by start time, which is defined as the sum of the total trip distance over all trips that start in each hour of the day. This figure illustrates the time-varying magnitude of the total load on the road network.

4.2 Computational Performance

This section details the design and analysis of the computational performance of the Mobiliti simulator. As mentioned in Section 3, Mobiliti utilizes an optimistic parallel discrete event simulation protocol that allows for speculative event execution and rollback to maintain causality (see [23] for details). For our experiments, we ran Mobiliti on the Cori supercomputer [39]. Adding dynamic rerouting adds significant computational cost to the simulation compared to the original statically routed case. This is mainly due to the addition of shortest path route calculations by VehicleController actors during RerouteCheck events. We mitigate the added cost by using an efficient routing

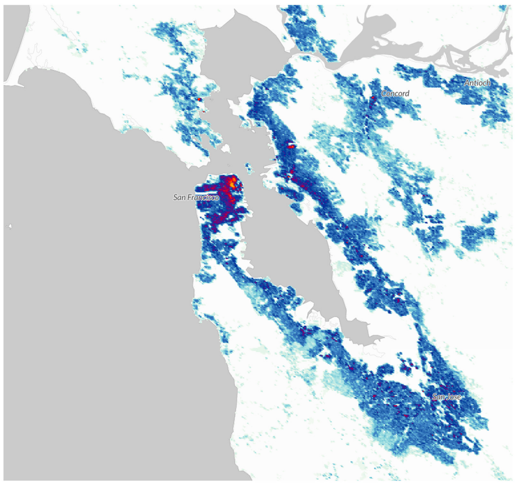
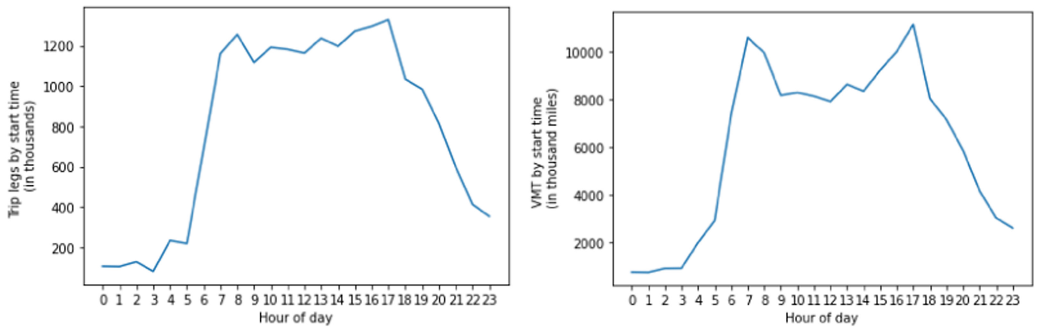


Fig. 12. GHS density map [21] showing where people live as a function of geographical location. Mobiliti uses this information coupled with a traffic analysis zone-based demand model to compute specific node origins and node destinations for each trip leg. The raster resolution of the GHS data is 250 meters by 250 meters. Red shows high population density, followed by a dark blue with medium density and lighter blues with low density.

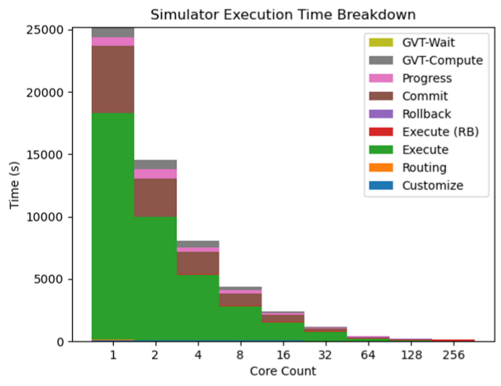


(a) Total trip legs by start time

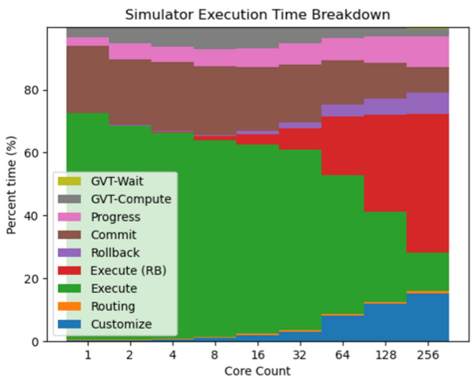
(b) Total vehicle miles traveled (VMT) by start time

Fig. 13. These figures give a temporal profile of how demand evolves through the simulated day. Figure (a) shows the total number of trip legs starting at different times of the day. Figure (b) shows the total VMT summed over trips that start at different times of day.

algorithm (Customizable Contraction Hierarchies [17, 25]) via the RoutingKit [44] library. RoutingKit's Customizable Contraction Hierarchies library employs relatively costly preprocessing steps (i.e., customizations) to enable very efficient subsequent routing queries. This strategy is ideal for computing many routes in a network with static weights, since the cost of the preprocessing step can be amortized across all of the subsequent queries. However, in a dynamic rerouting simulation the link weights are constantly changing due to congestion varying over the course of the simulated day. In order to support dynamically changing link weights, we designed the VehicleController to keep track of which links have updated their travel time since the previous customization, and then only re-customize the contraction hierarchy when a RerouteCheck query is received. This method of batching LinkStatusUpdates avoids excessive re-customizations every time an update



(a) Absolute execution time



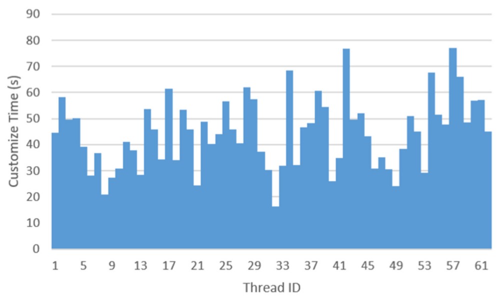
(b) Category percentages

Fig. 14. Parallel Scaling Performance. (a) Breakdown of Mobiliti simulation times (excluding program initialization) by category as we vary the core count from 1 to 256 cores of the Cori-Haswell computer at NERSC [39]. All times shown are for simulating 19 million trip legs with 50 percent dynamic rerouting penetration on the whole San Francisco Bay Area road network with 0.5 million nodes and 1 million links. When simulating a normal model day, execution time is decreased from over seven hours on one core to less than three minutes on 256 cores running in parallel. (b) Breakdown of execution by category as a percent of total execution time as the core count is varied from 1 to 256 cores. See the text for a description of categories.

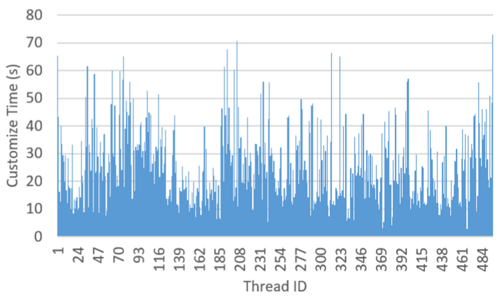
is received. Furthermore, if no congestion update is received during a sequence of RerouteCheck queries, they can all use the same customization and thus be serviced very efficiently.

Figure 14(a) shows the execution time of Mobiliti when simulating a full normal model day with 19 million trip legs over the San Francisco Bay 0.5 million nodes and 1 million links with 50 percent dynamic rerouting penetration. As we increase the core count from 1 to 256, the simulation execution time (excluding program initialization) is reduced from more than seven hours (25,000 seconds) to less than three minutes (171 seconds), corresponding to a 146× parallel speed up. For the shared memory runs (16 cores or fewer), we use a single process, multi-threaded configuration (utilizing one thread per core). Using shared memory avoids the overheads of inter-process communication, but limits execution to a single node. For the distributed memory runs (32 cores or more), we use a multi-process, multi-threaded configuration with hyper-threading (two threads per core), which enables executing on multiple nodes, better data locality in the memory subsystem due to non-uniform memory access [36], and higher core utilization. Furthermore, strong scaling (increasing core count for a fixed problem size) yields the benefit of reducing the active working set sizes for each core, enabling better data re-use and on-chip cache utilization.

Figure 14(b) shows the percent of total time spent doing various tasks, including customization of the Contraction Hierarchy router (blue), running Contraction Hierarchy routing queries (orange), executing events (green and red), rolling back events (purple), committing/logging events (brown), runtime messaging overhead (pink), and **global virtual time** (**GVT**) overheads (grey and yellow). Note that **Execute** time (green) refers to the execution time for events that are eventually committed, while **Execute (RB)** time (red) refers to the *forward* execution time for events that are speculatively executed and eventually rolled back. Furthermore, **Rollback** time (purple) refers to the time to roll back (*reverse*) the events that were mis-speculated. For the single core run, there is no misspeculation because events are trivially executed in increasing timestamp order, so no time is spent rolling back events. As the number of cores increases, the percentage of total time spent doing router customization (blue) and executing and rolling back mis-speculated



(a) 64 thread (32 core) customization load distribution



(b) 512 thread (256 core) customization load distribution

Fig. 15. Time spent in Contraction Hierarchy router customization across threads for the 64 thread (32 core) and 512 thread (256 core) cases. Note that the maximum customization load remains roughly the same between the two cases, indicating that the customization load imbalance becomes a scalability bottleneck for higher core counts.

events (red and purple) increases, which reduces the effective parallel speedup. The reason that the router customization percentage increases is that as the simulation is scaled onto more cores, we correspondingly increase the number of vehicle controllers so that each road partition has a local vehicle controller to service its vehicles' reroute queries, thus parallelizing the reroute *query* workload across cores and avoiding core-to-core round trip message overheads to service each rerouting request. However, since each of the controllers must still do router *customization* computations, the *per-core* customization cost makes up an increasing percentage of total execution time (as the other parts of the simulation scale more effectively).

There are two main reasons for the increasing mis-speculation and rollback (red and purple) costs. First, as core count increases, the number of discrete events that cross core boundaries increases, leading to an increase in the number of events that trigger a rollback at the destination. This would occur due to core-to-core messaging latencies even if the simulation were perfectly load balanced across cores. Second, when the simulation compute load is *not* balanced, the load imbalance manifests as mis-speculation and rollback in our metrics since the underloaded cores mis-speculate as they simulate faster than the overloaded cores. We observed that while the normal vehicle transition events stay load balanced with higher parallelism, the router customization load imbalance increases with core count since reroute requests can be highly localized to where congestion occurs in the network, making it more difficult to compute an effective parallel partitioning compared to when there is no dynamic rerouting. Figure 15 shows the distribution of time spent in customization computations across all threads in the 32 core and 256 core simulations. We observe that the most overloaded thread spends over 70 seconds in customization computations (in both cases), which is quite significant compared to the 171 second total simulation time in the 256 core experiment. We suspect the scalability could be further improved for even higher core counts by tuning the parallel distribution of the dynamic re-routing workload to refine the computational load balance thereby reducing time spent mis-speculating. A more in-depth parallel performance and load balancing study of the simulator will be explored in future work.

4.3 Impact of Dynamic Rerouting on System Metrics

To understand the effect of rerouting penetration, we enabled dynamic rerouting for varying percentages of vehicle trip legs for the entire Bay Area. We chose to study a range of penetration rates from 0% to 100% at 10% increments. Figure 16 illustrates the impact of enabling dynamic rerouting

Table 2. Share of Rerouted Trip Legs

Penetration rate (%)	**Trip legs reroutable** (in thousands)	**Trip legs rerouted** (in thousands)	**Percentage rerouted** (%)
10	1,900	130	7
20	3,800	221	6
30	5,700	274	5
40	7,600	291	4
50	9,500	301	3
60	11,400	317	3
70	13,300	329	2
80	15,200	340	2
90	17,100	355	2
100	19,000	374	2

for 100% of vehicle trips for the entire metropolitan-scale system simulated. The difference between baseline and 100% rerouting case is that the former uses static shortest path routes based on free speed traversal times, whereas the latter uses dynamic routes computed by the vehicle controllers based on their knowledge of the current traffic congestion patterns. In Figure 16, blue links handle a lesser number of vehicle traversals when 100% dynamic rerouting is enabled, while the red links handle a greater number. The figure shows how the traffic is rerouted away from certain links to reduce congestion (blue), while other links end up handling higher traffic (red).

Figure 17 shows how the reroutes are temporally distributed throughout the simulation day, illustrating that almost 80% of the rerouting occurs during the morning and evening rush hours when the demand is the highest. The distribution of reroutes is heavily influenced by the temporal distribution of trip legs in the demand model input (Figure 13). As can be seen in Figure 13(b), there is a peak in the total VMT in the morning (7 am to 10 am) and the evening rush hours (3:30 pm to 6:30 pm). Because the level of demand during rush hours is the highest, we see corresponding peaks in the number of reroutes. Further, it must be noted that penetration rate indicates the number of trip legs allowed to reroute, but not all reroutable trip legs actually reroute. Table 2 indicates the percentage of trip legs rerouted as a percentage of allowed reroutable legs. At higher penetration rates, the percentage decreases even though the actual number of reroutes is higher, since not many trip legs engage in any relevant congestion and hence do not reroute. Furthermore, among the trips that do reroute, the number of *reroutes per trip* remains small, with 99.9% of trips rerouting three times or fewer in the 100% penetration scenario.

Using the delay and fuel model described in our previous work [14], we can make impact estimates for dynamic rerouting penetration rates. Figure 18 exhibits the system level **vehicle hours of delay** (**VHD**) and the number of reroutes for different penetration levels, and Table 3 shows the system level VMT and fuel consumption resulting from the dynamic rerouting. As the penetration rate increases, the delay reduces without any significant change in VMT. The minimum system delay is incurred with 100% penetration rate. However, if we look at the "knee of the curve" for VHD, the return starts diminishing after 70% penetration. On average, a rerouted trip saves 16 minutes in travel time with no additional trip distance with 100% penetration rate compared to baseline as shown in Figure 19(a).

We observe that dynamic rerouting effectively rearranges the vehicle flows from high-utilized highways and arterials to low-utilized neighborhood links to reduce the overall system delay. We analyze these effects by investigating the rearrangement of the traffic flow by functional classification of links. We maintain the definitions of functional class roads as defined by HERE Technologies [27]. Specifically, functional classes are hierarchical classifications of roads according to speed, importance, and connectivity. A road can be one of five functional classes defined in Table 4.

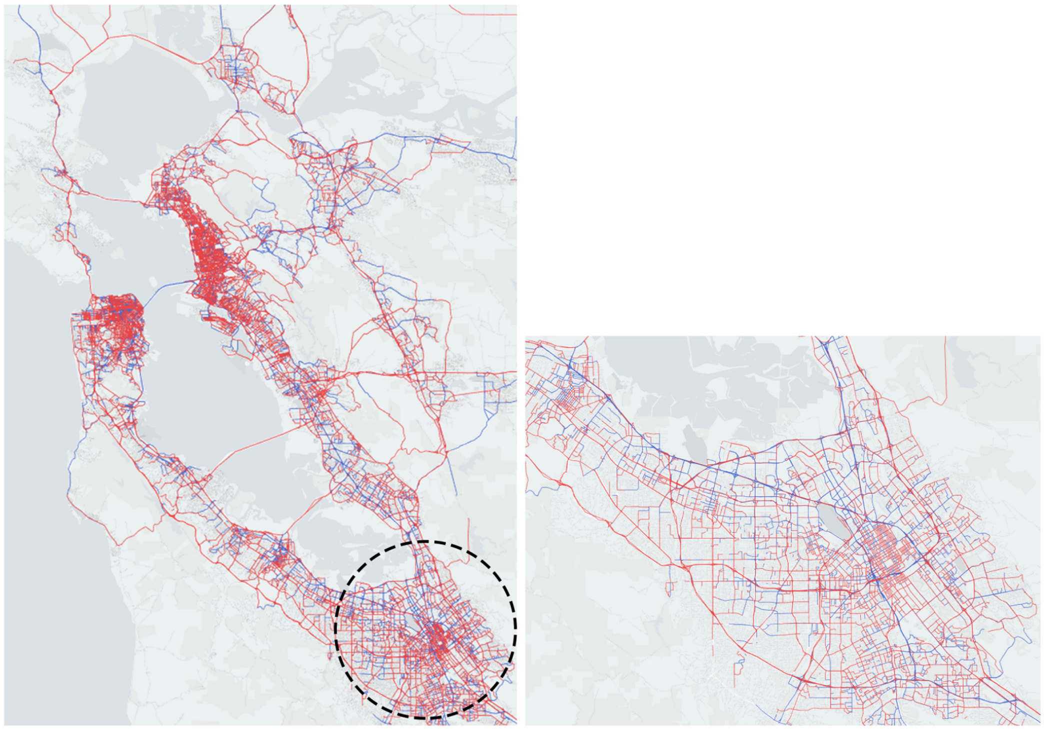


Fig. 16. The figures show the system-wide impact on the number of vehicle link traversals from enabling dynamic rerouting (baseline versus 100% penetration) for Bay Area (left) and San Jose city (right). Red or blue represents an increase or decrease in vehicle traversal count for a day.

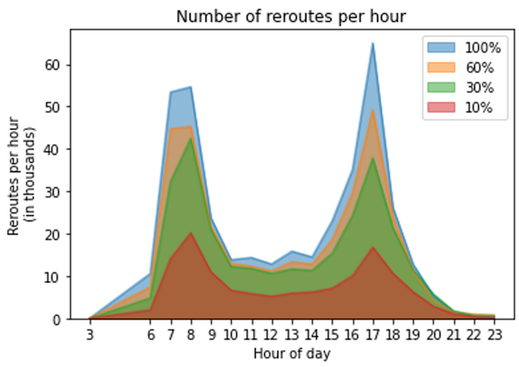


Fig. 17. The figure gives a temporal profile of the rate of dynamic vehicle reroutes for a day. Sharp increases in reroutes can be seen in the morning and evening peak hours.

By examining the traffic flow by ***functional class*** (**FC**) with 100% dynamic rerouting penetration, we observe that traffic shifts from FC 2 and 3 to FC 4 and 5. This significantly reduces highway delays while increasing traffic on FC 5 in the morning and evening peaks. It is also interesting to note that the increased traffic volume on FC 5 does not always cause congestion in those links, as many links do not reach congested levels with the increased flow. Specifically, 7000 kilometers

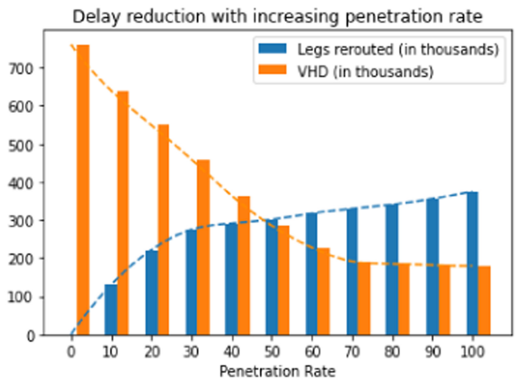


Fig. 18. The figure shows the system wide VHD and the number of trip legs rerouted from varying dynamic rerouting penetration rates. As the penetration rate increases, the overall system delay reduces. If we look at the elbow of the VHD curve, after 70% penetration rate the returns start diminishing.

Table 3. VMT and Fuel Consumption

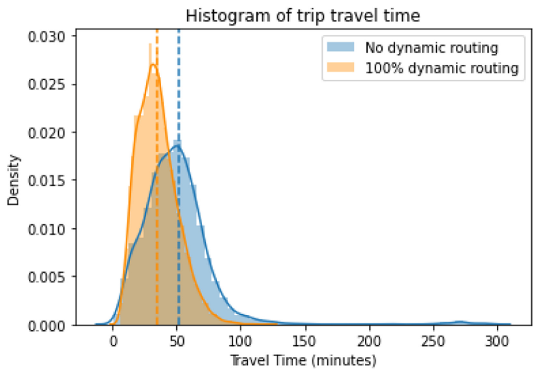
Penetration rate	**Vehicle Miles Traveled** (in thousand miles)	**Fuel** (in thousand gallons)
0%	146,847	5,906
10%	146,783	5,903
20%	146,707	5,899
30%	146,652	5,894
40%	146,605	5,891
50%	146,572	5,888
60%	146,546	5,886
70%	146,537	5,884
80%	146,517	5,883
90%	146,505	5,882
100%	146,490	5,882

Table 4. Functional Road Classes

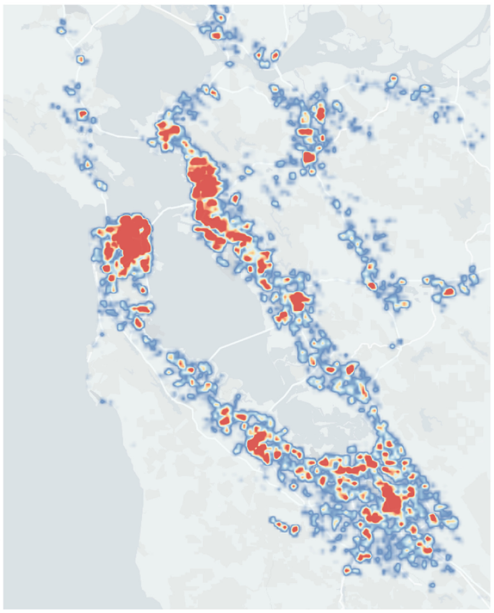
Functional class	**Definition**
1	Allows high volume, maximum speed traffic movement
2	Allows high volume, high speed traffic movement
3	Provides high volume of traffic movement
4	Provides high volume of traffic movement at moderate speeds between neighborhoods
5	Local roads with volume and traffic movement below the level of any functional class

of FC 5 links received additional traffic flow with 100% dynamic rerouting, of which 75% received fewer than six additional vehicles during the morning peak.

Of the FC 5 links with increased traffic volume, 440 kilometers are congested with a volume-over-capacity ratio higher than 0.75. Spatial analysis of the congestion shows that the cities of San Francisco, San Jose, Berkeley, Oakland, and Fremont are the most affected by the increased traffic on the local roads (Figure 19(b)). The local roads (FC 5) in these cities have a mean increase of 70 vehicles during the morning peak. Finally, of the 440 kilometers of congested road network in the morning peak, 110 kilometers reflects *new* congestion created due to dynamic rerouting on the



(a) The figure shows the distribution of the travel times for trip legs. The mean travel time per trip leg with 100% dynamic routing is 35 minutes, and without dynamic routing is 51 minutes, as shown by the corresponding orange and blue vertical lines.



(b) The figure shows the heat map of areas with the highest congestion (volume over capacity more than 0.75) with increased vehicle counts due to dynamic rerouting in the morning rush hour for functional class 5 links. Red indicates high congestion, and blue represents low congestion.

Fig. 19. These figures illustrate the distributions of trip travel times and spatial congestion.

Table 5. User Equilibrium Traffic Assignment: System Metrics

	Vehicle Miles Traveled	**Vehicle Hours of Delay**	**Fuel**
	(in thousand miles)	(in thousand hours)	(in thousand gallons)
User Equilibrium	146,051	90	5,915

local roads. These roads would arguably be some of the most negatively impacted areas by high dynamic rerouting penetrations.

Finally, we present the results of **user equilibrium** (UE) traffic assignment in Table 5 for comparison using the methods we describe in [15]. Comparing with the system metrics for dynamic rerouting with 100% penetration rate, we can see that VHD is nearly half, fuel consumption is slightly higher, and VMT slightly lower in user equilibrium. Since the user equilibrium is a steady-state solution computed through iterative optimization, it results in routes with lower delays than the more reactive dynamic rerouting approach. However, in reality, the user equilibrium state is never actually achieved, and hence congestion is underestimated in the user equilibrium traffic assignment.

4.4 Validation

Validation was performed for the simulation runs with different penetration rates to test the effectiveness of representing the real-world traffic environment. We conducted a three-stage validation procedure using multiple data sources. Our results show that the simulation with 60% dynamic rerouting is the closest to representing real-world traffic. We have only included the validation for

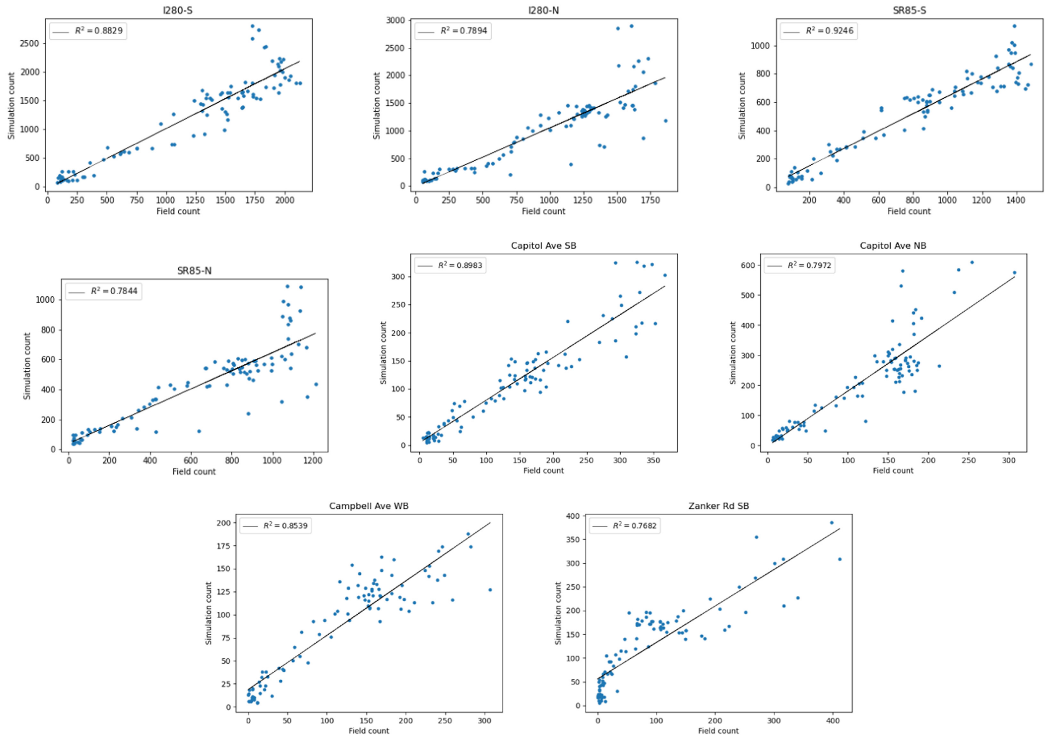


Fig. 20. The figure shows the validation of traffic counts in 15-minute increments for links in different functional classes. All links have satisfactory R^2 of greater than 0.7.

this penetration rate here for brevity. Our results are consistent with multiple surveys stating that the percentage of Americans having smartphones who uses online maps or navigation services daily ranges from 55% to 65% [20, 37, 47].

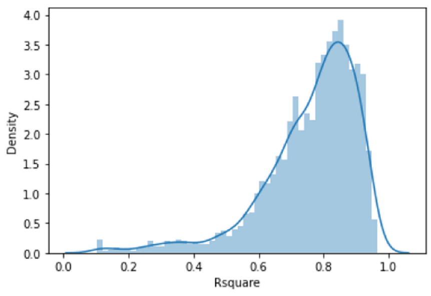
Stage 1 of the validation procedure involves comparing the traffic flows or counts between the simulation and real-world data. This includes checking (a) traffic counts for eight corridor links in San Jose city, (b) **average daily traffic** (**ADT**) counts for four main bridges in the Bay Area, and (c) traffic flows for all major highways in the Bay area. The traffic count for each link was compared against the field data for the entire day in 15 minutes increments. The field data for city roads and highways were collected from the city of San Jose, and the Caltrans **Performance Measurement System** (**PeMS**) website [18] respectively for the year 2019. Each corridor provided information regarding traffic volumes by time of the day and direction. Since PeMS data is prone to measurement error, data from multiple weekdays in April and May 2019 were averaged to get a typical day value. A coefficient of determination **R-squared** (**R^2**) of 0.7, which is typically used as a satisfactory criterion for link count checks, is used as the threshold. Figure 20 shows R^2 values for the eight corridors under consideration. The modeled corridors are closely matched with the field data, with the lowest R^2 value observed being 0.76 for Zanker road.

Additionally, ADT for four main bridges in the Bay Area in both directions is shown in Table 6. The field data were obtained from the Caltrans website [18] for the year 2019. The target error was ±25%, and seven of the eight links met this criterion. We believe that the high relative error for the Golden Gate Bridge **northbound** (**NB**) count is due to the Caltrans field count not accurately representing the actual bridge count. The discrepancy is due to the sensor placement **after** a major exit, which results in a significant percentage of bridge traffic not being counted by the sensor.

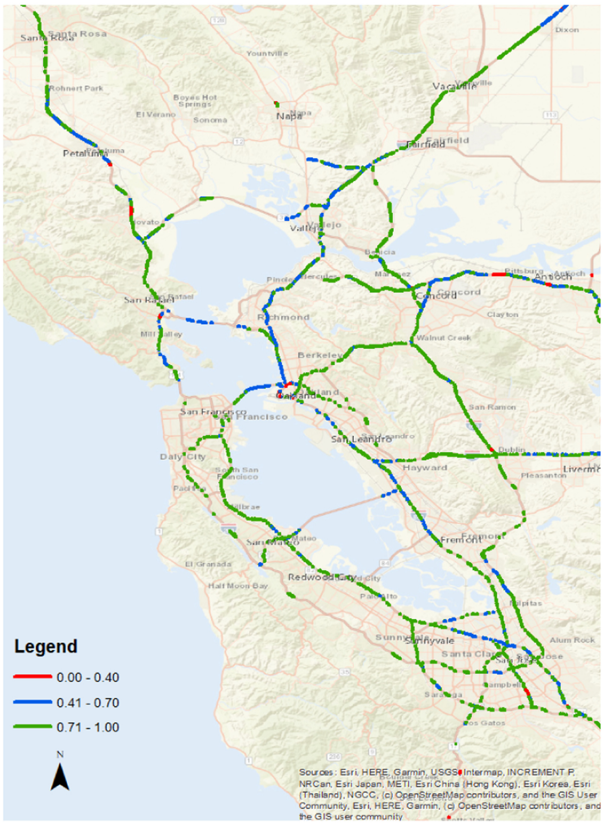
Table 6. Average Daily Traffic (ADT) Comparison

Sl No	Bridge	Field Count	Simulation Count	Relative Error (%)
1	I-580 Richmond San Rafael Bridge EB	56,182	51,551	−8%
2	I-580 Richmond San Rafael Bridge WB	41,597	52,131	25%
3	I-80 Bay Bridge EB	132,000	148,105	12%
4	I-80 Bay Bridge WB	131,861	139,993	6%
5	US-101 Golden Gate Bridge NB	32,212*	63,730	98%
6	US-101 Golden Gate Bridge SB	74,526	70,020	−6%
7	CA-92 San Mateo Bridge EB	56,510	53,684	−5%
8	CA-92 San Mateo Bridge WB	62,597	50,199	−20%

* Golden Gate Bridge northbound (NB) link's closest PeMS sensor is located after an off ramp and hence the field count does not reflect the full bridge traffic count.



(a) Distribution of R^2 values for the highway links is shown in the figure. It follows a left skewed normal distribution with mean 0.75 and standard deviation 0.15


Legend
0.00 - 0.40
0.41 - 0.70
0.71 - 1.00
N
Santa Rosa
Napa
Vacaville
Fairfield
Petaluma
Novato
Vallejo
San Rafael
Richmond
Berkeley
Oakland
Concord
Walnut Creek
San Francisco
San Leandro
Hayward
Dublin
Livermore
Daly City
San Mateo
Fremont
Redwood City
Sunnyvale
Santa Clara
San Jose
Sources: Esri, HERE, Garmin, USGS, Intermap, INCREMENT P, NRCan, Esri Japan, METI, Esri China (Hong Kong), Esri Korea, Esri (Thailand), NGCC, (c) OpenStreetMap contributors, and the GIS User Community, Esri, HERE, Garmin, (c) OpenStreetMap contributors, and the GIS user community

(b) The map shows the spatial distribution of R^2 values of flows for the links.

Fig. 21. Validation metrics for highway link flows compared to PeMS sensors are shown.

Next, we evaluated R^2 for all links with a corresponding PeMS sensor in Bay Area. We were able to match 2061 links with mainline sensors and the resulting R^2 distribution is shown in Figure 21(a). Of the total matched links, 72% have R^2 greater than 0.7, and 5% have lower than 0.4 (Figure 21(b)).

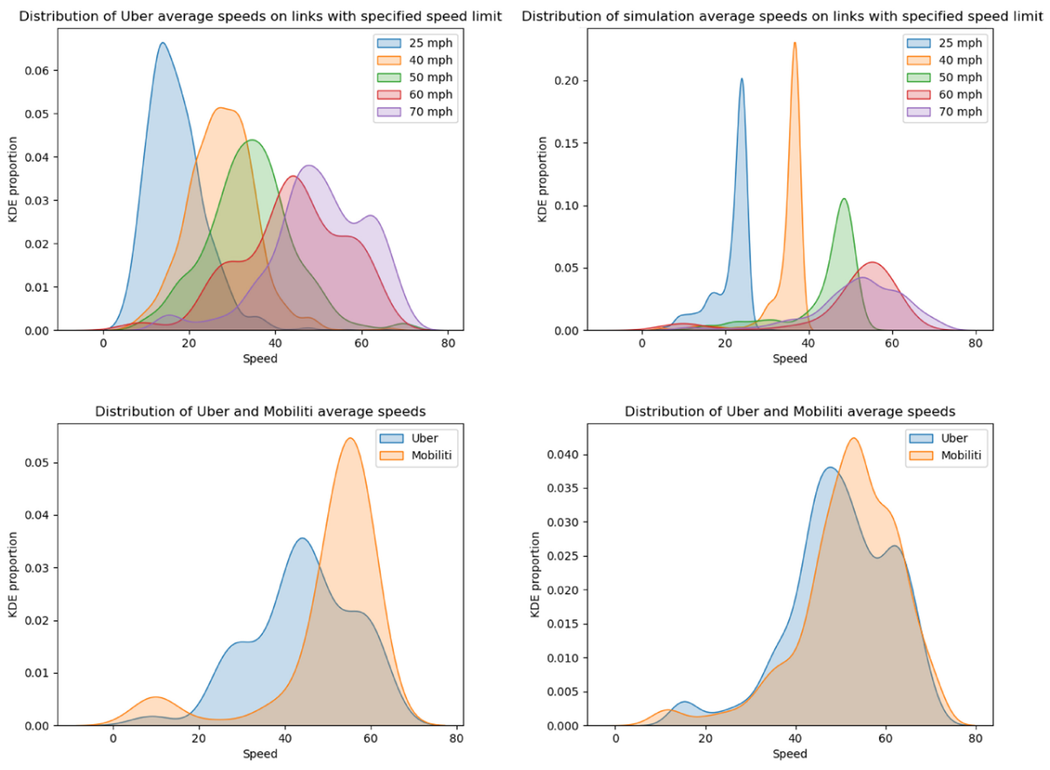


Fig. 22. Average Uber (left) and Mobiliti (right) speeds across all speed limits from 8 am to 9 am in shown in the first row. The second row shows the kernel density plots comparing both speed distributions at 60 mph (left) and 70 mph (right).

Stage 2 in the validation procedure compares simulation speed with (a) Uber Movement data for San Francisco city streets and (b) PeMS speed data for Bay Area highways. For Uber Movement, speed data for the San Francisco region for quarter four, 2019 is obtained from the website [50]. Links from Uber network were matched to Mobiliti links for a total of 139,495 links (20% of total). The speeds were compared from 8 am to 9 am for different speed limits. Figure 22 shows the average speeds from Mobiliti and Uber across all speed limits. The speed distributions from both links with 60 mph and 70 mph speed limit are shown in the second row.

Next, we compared highway links with PeMS speed profiles for the 2,061 matched links. Figure 23 shows the difference in speeds between simulation and PeMS at 9 am and 3 pm for a weekday. Most links are within ±20 mph difference. Further, R^2 values were evaluated for all links to understand the time series trends. Figure 24 shows a time series comparison for six highway links. Of the total, 55% of highway links have R^2 greater than 0.4. We plan to improve the speed models to reflect time series trends closer to real-world data in the future.

Stage 3 is system-level comparisons, network validation, and error checking. Model visualization is used to check for unusual activities in traffic flows and odd roadway network attributes. Error checking and model verification consist of smaller tasks such as checks for link geometry and connectivity, link speeds, and ramp and intersection geometry. Since our travel demand data was obtained from SFCTA, which conducts its validation, we did not conduct additional behavior checks. We conducted system metric checks for VMT and total demand and validated them against the 2017 Environmental Impact Report for the Bay Area [19] in Table 7.

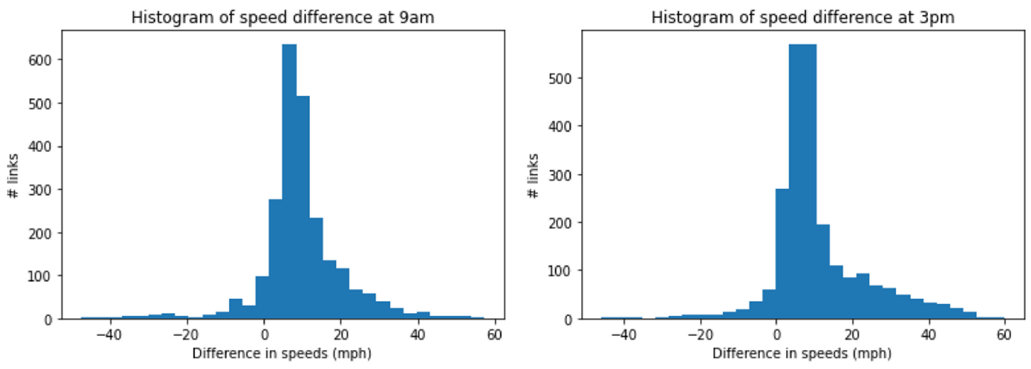


Fig. 23. The figure shows the histogram of the speed difference between simulation and PeMS at 9 am and 3 pm for highways in the Bay Area.

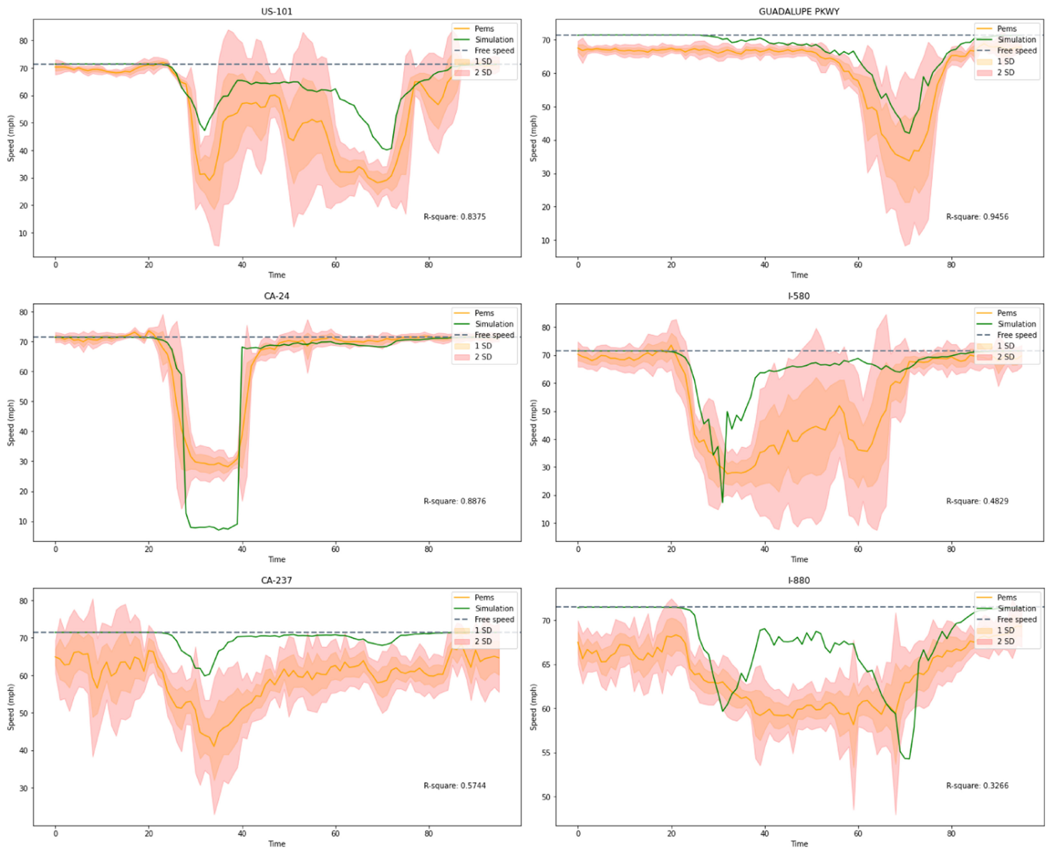


Fig. 24. The figures show the simulation and PeMS speed profile for highways. The yellow line represents the average weekday values from PeMS. The first and second standard deviation bands are also shown. The green line represents the simulation speed for a typical day.

Table 7. System Level Metrics

Metric	Simulation	Field Data	Relative Error (%)
VMT	146,546,360	158,406,800	−7
Daily Trips	19,167,301	21,227,800	−10

5 CONCLUSION

Vehicles are rapidly gaining the ability to utilize up-to-date road congestion information to re-route their paths during their trips through smartphone navigation apps. In this article, we presented a computational methodology to model varying degrees of dynamic rerouting in a large-scale transportation system using the Mobiliti high-performance parallel discrete event simulator. We described updates to our link actor model to capture the effects of link congestion, timing constraints, and storage capacity constraints. We have detailed the implementation of new VehicleController actors and the events required to update their knowledge of the system state and service dynamic rerouting requests. Because we have designed our simulator to scale over distributed memory parallel computing platforms, we can simulate one model day of the San Francisco Bay Area with 19 million vehicle trips and 50 percent dynamic rerouting penetration over a road network with 0.5 million nodes and 1 million links in three minutes of simulation time. We conducted an analysis of system-level impacts when varying the dynamic rerouting penetration rate at 10% increments and examined the varying effects on different functional classes and geographical regions. Finally, we presented a validation of the simulation results compared to real-world data sources.

REFERENCES

[1] M. Abrams. 1989. A common programming structure for Bryant-Chandy-Misra, time-warp, and sequential simulators. In *Proceedings of the 21st Conference on Winter Simulation.* Association for Computing Machinery, New York, NY, 661–670. DOI: https://doi.org/10.1145/76738.76823

[2] Gul Agha and Carl Hewitt. 1987. Actors: A conceptual foundation for concurrent object-oriented programming. In *Proceedings of the Research Directions in Object-oriented Programming*. 49–74.

[3] Joshua Auld, Michael Hope, Hubert Ley, Vadim Sokolov, Bo Xu, and Kuilin Zhang. 2016. POLARIS: Agent-based modeling framework development and implementation for integrated travel demand and network and operations simulations. *Transportation Research Part C: Emerging Technologies* 64, 0968-090X (2016), 101–116.

[4] Joshua Auld, Omer Verbas, and Monique Stinson. 2019. Agent-based dynamic traffic assignment with information mixing. *Procedia Computer Science* 151, 1877-0509 (2019), 864–869.

[5] Henry Baker and Carl Hewitt. 1977. Laws for communicating parallel processes. Technical Report WP-134A. MIT Artificial Intelligence Laboratory.

[6] Pablo Barbecho Bautista, Luis Urquiza-Aguiar, and Mónica Aguilar Igartua. 2020. Evaluation of dynamic route planning impact on vehicular communications with SUMO. In *Proceedings of the 23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems.* 27–35.

[7] Peter D. Barnes, Jr., Christopher D. Carothers, David R. Jefferson, and Justin M. LaPre. 2013. Warp speed: Executing time warp on 1,966,080 Cores. In *Proceedings of the Conference on Principles of Advanced Discrete Simulation* (Montr&#169;al, Qu&#233;bec, Canada). 327–336. DOI: https://doi.org/10.1145/2486092.2486134

[8] David W. Bauer Jr., Christopher D. Carothers, and Akintayo Holder. 2009. Scalable time warp on blue gene supercomputers. In *Proceedings of the Workshop on Principles of Advanced and Distributed Simulation.* 35–44. DOI: https://doi.org/10.1109/PADS.2009.21

[9] Michael Behrisch, Daniel Krajzewicz, and Yun-Pang Wang. 2008. Comparing performance and quality of traffic assignment techniques for microscopic road traffic simulations. *Proceedings of DTA2008* (2008).

[10] Moshe Ben-Akiva, Michel Bierlaire, Haris N. Koutsopoulos, and Rabi Mishalani. 2002. Real time simulation of traffic demand-supply interactions within DynaMIT. In *Proceedings of the Transportation and Network Analysis: Current Trends*. Springer, 19–36.

[11] CAL. 2017. *California Fires: Navigation Apps Like Waze Sent Commuters into Flames, Drivers Say*. Retrieved from https://www.usatoday.com/story/tech/news/2017/12/07/california-fires-navigation-apps-like-waze-sent-commuters-into-flames-drivers/930904001/. Accessed 1 November 2022.

[12] Gordon D. B. Cameron and Gordon I. D. Duncan. 1996. PARAMICS–Parallel microscopic simulation of road traffic. *The Journal of Supercomputing* 10, 1 (1996), 25–53.

[13] Jordi Casas, Jaime L. Ferrer, David Garcia, Josep Perarnau, and Alex Torday. 2010. Traffic simulation with aimsun. *Fundamentals of Traffic Simulation*, J. Barceló (Ed.). International Series in Operations Research & Management Science, Vol. 145, Springer, New York, NY, 173–232.

[14] Cy Chan, Bin Wang, John Bachan, and Jane Macfarlane. 2018. Mobiliti: Scalable transportation simulation using high-performance parallel computing. In *Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems.* IEEE, Maui, HI, 634–641. DOI: https://doi.org/10.1109/ITSC.2018.8569397

[15] Cy P. Chan, Anu Kuncheria, Bingyu Zhao, Theophile Cabannes, Alexander Keimer, Bin Wang, Alexandre M. Bayen, and Jane MacFarlane. 2021. Quasi-dynamic traffic assignment using high performance computing. arXiv:2104.12911. Retrieved from https://arxiv.org/abs/2104.12911.

[16] K. M. Chandy and J. Misra. 1979. Distributed simulation: A case study in design and verification of distributed programs. *IEEE Transactions on Software Engineering* SE-5, 5 (1979), 440–452. DOI: https://doi.org/10.1109/TSE.1979.230182

[17] Julian Dibbelt, Ben Strasser, and Dorothea Wagner. 2014. Customizable contraction hierarchies. In *Proceedings of the Experimental Algorithms.* Joachim Gudmundsson and Jyrki Katajainen (Eds.), Springer International Publishing, Cham, 271–282.

[18] C. A. DOT. 2021. *Traffic Volumes | Caltrans.* Retrieved from https://dot.ca.gov/programs/traffic-operations/census/traffic-volumes. Accessed 1 November 2022.

[19] EIRPBA. 2021. *Environmental Impact Report Plan Bay Area.* Retrieved from https://www.planbayarea.org/2040-plan/environmental-impact-report. Accessed 1 November 2022.

[20] EMarketer. 2021. *Maps and Navigation Apps are Still Essential to Smartphone Experience, and User Penetration Continues to Grow.* Retrieved from https://www.emarketer.com/content/people-continue-to-rely-on-maps-and-navigational-apps-emarketer-forecasts-show. Accessed 1 November 2022.

[21] European Commission, Joint Research Centre (JRC); Columbia University, Center for International Earth Science Information Network - CIESIN. 2015. GHS Population Grid, Derived from GPW4, Multitemporal (2015). Retrieved from http://data.europa.eu/89h/jrc-ghsl-ghs_pop_gpw4_globe_r2015a. European Commission, Joint Research Centre (JRC) [Dataset]. Accessed 1 November 2022.

[22] FHWA. 2021. Traffic Analysis Toolbox Volume XIV: Guidebook on the Utilization of Dynamic Traffic Assignment in Modeling - Section 2. Retrieved from https://ops.fhwa.dot.gov/publications/fhwahop13015/sec2.htm. Accessed 1 November 2022.

[23] Richard M. Fujimoto. 1990. Parallel discrete event simulation. *Communications of the ACM* 33, 10 (1990), 30–53. DOI: https://doi.org/10.1145/84537.84545

[24] Linjie Gao, Zhicai Juan, and Peng Jing. 2008. The design and implement of parallel simulation algorithm of dynamic route solution for traffic network. In *Proceedings of the 2008 Asia Simulation Conference-7th International Conference on System Simulation and Scientific Computing*. IEEE, 230–234.

[25] Robert Geisberger, Peter Sanders, Dominik Schultes, and Christian Vetter. 2012. Exact routing in large road networks using contraction hierarchies. *Transportation Science* 46, 3 (2012), 388–404. DOI: https://doi.org/10.1287/trsc.1110.0401

[26] Frank Harary and Robert Z. Norman. 1960. Some properties of line digraphs. *Rendiconti del circolo matematico di palermo* 9, 2 (1960), 161–168.

[27] HERE Technologies. 2019. Retrieved February 06, 2019 from https://www.here.com/.

[28] David R. Jefferson. 1985. Virtual time. *ACM Transactions on Programming Languages and Systems* 7, 3 (1985), 404–425. DOI: https://doi.org/10.1145/3916.3988

[29] George Karypis and Vipin Kumar. 1998. Multilevelk-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed computing* 48, 1 (1998), 96–129.

[30] Hyunmyung Kim, Jun-Seok Oh, and R. Jayakrishnan. 2009. Effects of user equilibrium assumptions on network traffic pattern. *KSCE Journal of Civil Engineering* 13, 2 (2009), 117–127.

[31] Kyungtae Kim, Seokjoo Koo, and Ji-Woong Choi. 2020. Analysis on path rerouting algorithm based on V2X communication for traffic flow improvement. In *Proceedings of the 2020 International Conference on Information and Communication Technology Convergence.* IEEE, 251–254.

[32] Rafał Kucharski and Guido Gentile. 2019. Simulation of rerouting phenomena in dynamic traffic assignment with the information comply model. *Transportation Research Part B: Methodological* 126, 0191-2615 (2019), 414–441.

[33] Zilu Liang and Yasushi Wakahara. 2014. Real-time urban traffic amount prediction models for dynamic route guidance systems. *EURASIP Journal on Wireless Communications and Networking* 2014, 1 (2014), 85. DOI: https://doi.org/10.1186/1687-1499-2014-85

[34] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. 2018. Microscopic traffic simulation using sumo. In *Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems.* IEEE, 2575–2582.

[35] H. S. Mahmassani 1998. Dynamic traffic simulation and assignment: models, algorithms and application to ATIS/ATMS evaluation and operation. *Operations Research and Decision Aid Methodologies in Traffic and Transportation Management.* M. Labbé, G. Laporte, K. Tanczos, and P. Toint (Eds.). NATO ASI Series, Vol. 166, Springer, Berlin and Heidelberg, 104–135.

[36] Zoltan Majo and Thomas R. Gross. 2011. Memory system performance in a NUMA multicore multiprocessor. In *Proceedings of the 4th Annual International Conference on Systems and Storage.* 1–10.

[37] The Manifest. 2021. *The Popularity of Google Maps: Trends in Navigation Apps in 2018 | The Manifest.* Retrieved from https://themanifest.com/mobile-apps/popularity-google-maps-trends-navigation-apps-2018. Accessed 1 November 2022.

[38] MATSIM. 2020. MATSim.org. Retrieved from https://www.matsim.org/. Accessed 1 November 2022.

[39] NERSC. 2018. Cori Configuration. Retrieved April 27, 2018 from http://www.nersc.gov/users/computational-systems/cori/configuration/.

[40] Kalyan S. Perumalla. 2006. A systems approach to scalable transportation network modeling. In *Proceedings of the 2006 Winter Simulation Conference.* 1500–1507. DOI: https://doi.org/10.1109/WSC.2006.322919 ISSN: 1558-4305.

[41] Moeid Qurashi, Hai Jiang, and Constantinos Antoniou. 2020. Modeling autonomous dynamic vanpooling services in sumo by integrating the dynamic routing scheduler. In *Proceedings of the SUMO User Conference.*

[42] Hesham A. Rakha, Kyoungho Ahn, and Kevin Moran. 2012. INTEGRATION framework for modeling eco-routing strategies: Logic and preliminary results. *International Journal of Transportation Science and Technology* 1, 3 (2012), 259–274. DOI: https://doi.org/10.1260/2046-0430.1.3.259

[43] Marta Rojo. 2020. Evaluation of traffic assignment models through simulation. *Sustainability* 12, 14 (2020), 5536.

[44] RoutingKit. 2019. Retrieved January 28, 2019 from https://github.com/RoutingKit/RoutingKit.

[45] SFCTA. 2019. *SF-CHAMP 6.1: ConnectSF Needs Assessment 2015 Base Year Model Run.* Technical Report. San Francisco County Transportation Authority.

[46] Colin Sheppard, Rashid Waraich, Andrew Campbell, Alexei Pozdnukov, and Anand R. Gopal. 2017. *Modeling Plug-in Electric Vehicle Charging Demand with BEAM: The Framework for Behavior Energy Autonomy Mobility.* Technical Report 1398472. 1398472 pages. DOI: https://doi.org/10.2172/1398472

[47] Statista. 2021. *People who use their Cell Phone for maps/GPS Navigation in the U.S. 2018, by age.* Retrieved from https://www.statista.com/statistics/231615/people-who-use-their-cell-phone-for-maps-gps-navigation-usa/. Accessed 1 November 2022.

[48] Sunil Thulasidasan and Stephan Eidenbenz. 2009. Accelerating traffic microsimulations: A parallel discrete-event queue-based approach for speed and scale. In *Proceedings of the 2009 Winter Simulation Conference.* IEEE, 2457–2466.

[49] Ying-Tsu Tseng and Huei-Wen Ferng. 2021. An improved traffic rerouting strategy using real-time traffic information and decisive weights. *IEEE Transactions on Vehicular Technology* 70, 10 (2021), 9741–9751.

[50] Uber. 2021. *Uber Movement: Let's Find Smarter Ways Forward, Together.* Retrieved from https://movement.uber.com/cities/san_francisco/downloads/speeds. Accessed 1 November 2022.

[51] Juliette Ugirumurera, Gabriel Gomes, Emily Porter, Xiaoye S. Li, and Alexandre M. Bayen. 2018. A unified software framework to enable solution of traffic assignment problems at extreme scale. In *Proceeding of the 21st International Conference on Intelligent Transportation Systems (ITSC'18).* 3917–3922.

[52] Chaojie Wang, Srinivas Peeta, and Jian Wang. 2021. Incentive-based decentralized routing for connected and autonomous vehicles using information propagation. *Transportation Research Part B: Methodological* 149, 0191-2615 (2021), 138–161.

[53] Dali Wei, Feng Chen, and Xinxin Sun. 2010. An improved road network partition algorithm for parallel microscopic traffic simulation. In *Proceedings of the 2010 International Conference on Mechanic Automation and Control Engineering.* IEEE, 2777–2782.

[54] Yadong Xu, Wentong Cai, Heiko Aydt, and Michael Lees. 2014. Efficient graph-based dynamic load-balancing for parallel large-scale agent-based traffic simulation. In *Proceedings of the Winter Simulation Conference 2014.* IEEE, 3483–3494.

[55] Yadong Xu, Wentong Cai, Heiko Aydt, Michael Lees, and Daniel Zehe. 2017. Relaxing synchronization in parallel agent-based road traffic simulation. *ACM Transactions on Modeling and Computer Simulation* 27, 2 (2017), 1–24.

[56] Pavan Yedavalli, Krishna Kumar, and Paul Waddell. 2021. Microsimulation analysis for network traffic assignment (MANTA) at metropolitan-scale for agile transportation planning. *Transportmetrica A: Transport Science* 18, 3 (2022), 1278–1299.

[57] Srikanth B. Yoginath and Kalyan S. Perumalla. 2009. Reversible discrete event formulation and optimistic parallel execution of vehicular traffic models. *International Journal of Simulation and Process Modelling* 5, 2 (2009), 104. DOI: https://doi.org/10.1504/IJSPM.2009.028624

[58] Xiaomei Zhao, Chunhua Wan, Huijun Sun, Dongfan Xie, and Ziyou Gao. 2017. Dynamic rerouting behavior and its impact on dynamic traffic patterns. *IEEE Transactions on Intelligent Transportation Systems* 18, 10 (2017), 2763–2779. DOI: https://doi.org/10.1109/TITS.2017.2655550

Received 4 May 2022; revised 28 November 2022; accepted 2 January 2023