Fast-Forwarding Agent States to Accelerate Microscopic Traffic Simulations

Philipp Andelfinger
TUMCREATE Ltd and
Nanyang Technological University
Singapore
pandelfinger@ntu.edu.sg

Yadong Xu
TUMCREATE Ltd
Singapore
yadong.xu@tum-create.edu.sg

Wentong Cai
Nanyang Technological University
Singapore
aswtcai@ntu.edu.sg

David Eckhoff
TUMCREATE Ltd and
Technische Universität München
david.eckhoff@tum-create.edu.sg

Alois Knoll
Technische Universität München
Germany
knoll@in.tum.de

ABSTRACT

Traditionally, the model time in agent-based simulations is advanced in fixed time steps. However, a purely time-stepped execution is inefficient in situations where the states of individual agents are independent of other agents and thus easily predictable far into the simulated future. In this work, we propose a method to accelerate microscopic traffic simulations based on identifying independence among agent state updates. Instead of iteratively updating an agent's state throughout a sequence of time steps, a computationally inexpensive "fast-forward" function advances the agent's state to the time of its earliest possible interaction with other agents. To demonstrate the approach in practice, we present an algorithm to efficiently determine intervals of independence in microscopic traffic simulations and derive a fast-forward function for the popular Intelligent Driver Model (IDM). In contrast to existing acceleration approaches based on reducing the level of model detail, our approach retains the microscopic nature of the simulation. A performance evaluation is performed in a synthetic scenario and on the road network of the city of Singapore. At low traffic densities, we achieved a speedup of up to 2.8, whereas at the highest considered densities, only few opportunities for fast-forwarding could be identified. The algorithm parameters can be tuned to control the overhead of the approach.

ACM Reference Format:
Philipp Andelfinger, Yadong Xu, Wentong Cai, David Eckhoff, and Alois Knoll. 2018. Fast-Forwarding Agent States to Accelerate Microscopic Traffic Simulations. In *SIGSIM-PADS '18 : SIGSIM-PADS '18: SIGSIM Principles of Advanced Discrete Simulation CD-ROM, May 23–25, 2018, Rome, Italy*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3200921.3200923

ACM ISBN 978-1-4503-5092-1/18/05…$15.00
https://doi.org/10.1145/3200921.3200923

1 INTRODUCTION

Microscopic traffic simulation models represent the traffic in a road network on the level of individual vehicles that update their acceleration, velocity and position according to properties of their environment and nearby vehicles [16]. Typically, the vehicle updates occur at fixed time steps in model time. When considering scenarios spanning the road traffic of an entire city, this detailed simulation approach incurs substantial computational demands and long runtimes. Efforts to accelerate microscopic traffic simulations can be classified into two categories: hybrid modeling approaches, and parallel and distributed simulation. In hybrid modeling [1], areas in the road network are selected for which exact results on the vehicle level are not required. For such areas, vehicle movement is simulated in terms of tasks in a queueing network or as traffic flows. While hybrid modeling can achieve substantial runtime reductions, the microscopic nature of the simulation is partially surrendered. A second approach to reducing the runtime is parallel and distributed simulation, in which the simulation workload is distributed to multiple interconnected processing elements. Since fixed time steps for vehicle updates provide natural points for synchronization across processing elements, microscopic traffic simulators are well-suited for parallelization. However, when synchronizing at every time step, the achieved acceleration tends to scale far from linearly with the number of processing elements [32].

Hence, instead of the commonly applied time advancement using fixed time steps, some previous works have considered *asynchronous* state updates for agent-based simulations [2, 17, 22, 29]. Typically, the neighboring agents considered in an update are limited spatially by an agent's *sensing range*. If an agent is isolated from other agents for multiple time steps into the simulated future, the corresponding state updates can be performed without considering other agents' states. In parallel simulations, it has been shown that asynchronous state updates can reduce processor idle times by prioritizing updates that allow blocked processors to proceed. In the sequential case, the runtime can be reduced by limiting agent updates to those required to reach the simulation's termination criterion, e.g., performing only state updates that affect the state of a particular agent under consideration [29]. However, the applicability of the latter method is limited since satisfying the simulation's termination criterion commonly requires all agents to be at the same time step.

In this paper, we propose an approach for accelerating independent agent state updates using a computationally inexpensive *fast-forward function*, which updates the agent state to the first possible point in model time where an interaction can occur, skipping the intermediate updates. The approach retains the microscopic nature of the simulation. To apply the fast-forward function without violating the correctness of the simulation results, intervals in model time are identified for which agent interactions can be ruled out. A reduction in simulation execution time is achieved if the time spent on identifying such *independence intervals* is smaller than the time saved through the reduction in time steps.

Thus, we propose an algorithm that predicts independence intervals efficiently for microscopic traffic simulations on road networks represented by graphs. Further, we derive a fast-forward function for the well-known Intelligent Driver Model (IDM) [30], which governs the acceleration behavior of the simulated vehicles. The benefits of the proposed approach are evaluated in the city-scale microscopic traffic simulator CityMoS [33], both on a synthetic road network and on a representation of the road network of the city of Singapore. Since the benefit of the approach hinges on the availability and size of independence intervals, the largest performance gains are seen in areas of the road network where traffic is sparse. In cases where independence intervals are small or the overhead for identifying them is large, our approach has limited or even no benefit. The overhead can be balanced with the opportunities for fast-forwarding by adapting the frequency of identifying independence intervals and the maximum interval size.

The contributions of this paper are as follows:

(1) We propose asynchronous state updates using a fast-forward function for microscopic traffic simulation.
(2) We propose an algorithm to determine independence intervals to support asynchronous state updates.
(3) We derive a fast-forward function for the Intelligent Driver Model.
(4) We evaluate the performance benefits of the approach in simulations using two different road networks.

The remainder of this paper is organized as follows: Section 2 sketches the technical background of our work. Section 3 describes the proposed fast-forwarding approach. Section 4 provides validation and performance evaluation results. Section 5 describes remaining limitations and potential enhancements of the approach. Section 6 discusses related work. Section 7 provides a summary of our results and concludes the paper.

2 PRELIMINARIES

2.1 Agent-based modeling and simulation

In agent-based simulation, entities called agents are situated in an environment within the simulation space. An agent's environment is composed of static elements and nearby agents. At each point in model time, each agent has a *state* defined by a set of state variables. During a *state update*, an agent applies *update functions* to update the state variables according to the sensed environment. A *sensing range* limits the distance up to which the environment is considered. We refer to a state update that reads the state variables of nearby agents as an *interaction.*

Execution mechanisms for agent-based models can be classified into two categories: in a *synchronous* execution, the simulation proceeds in cycles. In each cycle, all agents perform a state update to advance their states by one time step, which is a fixed delta in model time. In an *asynchronous* execution, some agents may advance their time further into the simulated future than other agents [29]. The fast-forwarding approach proposed in the present paper is asynchronous. In contrast to existing approaches, instead of using iterative time steps, agent states are advanced into the simulated future through a single invocation of a fast-forward function.

2.2 Microscopic traffic simulation

In microscopic traffic simulations, agents called driver-vehicle-units (DVUs) move through the simulation space according to models of the state and behavior of a human driver as well as of the vehicle operated by the driver. Typically, the simulation space is a road network modeled as a directed graph $G = (V, E)$, where edges represent roads with one or more lanes and vertices represent intersections. At each point in model time, each DVU is situated at a specific position on a lane within an edge.

DVUs perform state updates according to a car-following model (e.g., [10, 30]) and a lane change model (e.g., [11, 15]). Car-following models determine the acceleration of a vehicle according to the characteristics of the driver, the vehicle, and the surrounding traffic conditions. Commonly, the acceleration is chosen according to a desired safety gap to the vehicle ahead. Lane change models decide whether a DVU should change lanes, e.g., based on the current velocity and vehicles on other lanes. The distance up to which nearby DVUs are considered is limited by the sensing range. For simplicity, we refer to DVUs as agents or vehicles throughout the remainder of the paper.

3 PROPOSED FAST-FORWARDING APPROACH

As introduced in the previous section, in an agent-based simulation, agents update their states according to their current environment and the states of neighboring agents within their sensing range. When an agent is spatially isolated from others, the current state update depends only on the environment, which may be static or highly predictable. The proposed approach is based on the observation that if it can be guaranteed that the agent remains isolated up to a certain point in model time, the agent's state can be updated to this point immediately. By computing such updates using a fast-forward function that is less computationally expensive than a sequence of regular state updates, the overall execution time of the simulation can be reduced.

3.1 Problem definition

In this section, we formally describe state updates in agent-based simulations using traditional time-stepped updates and the proposed fast-forward function. We loosely follow the formalization by Scheutz et al. [29], who studied asynchronous state updates either to limit state updates to those required for reaching the simulation's termination criterion or to decrease communication costs in distributed simulations. In contrast to their approach, which still relies on conventional time-stepped agent updates, fast-forwarding accelerates the simulation by avoiding state updates that are guaranteed to be independent of any other agents' states.

Let τ be the time step size of the simulation. An agent state update from t to $t + \tau$ can be represented as applying a *state update function* f_τ:

$$S_a^{t+\tau} = f_\tau(S_a^t, E_a^t, N_a^t)$$

where S_a^t is the state of agent a at simulation time t. We differentiate between the environment E_a^t surrounding agent a at t, and the set of neighboring agents N_a^t that are sensed by a at t. Let f_τ^k denote applying the state update function k times:

$$\begin{aligned} f_\tau^k(S_a^t, E_a^t, N_a^t) &= \\ f_\tau(f_\tau(\dots(f_\tau(S_a^t, E_a^t, N_a^t), E_a^{t+\tau}, N_a^{t+\tau}), \dots),& \\ E_a^{t+(k-1)\tau}, N_a^{t+(k-1)\tau}) \end{aligned}$$

We introduce a *fast-forward function* g, which approximates the result of iteratively applying f_τ given $N_a^{t+i\tau} = \emptyset$, $i \in \{0, \dots, k-1\}$:

$$|g(k\tau, S_a^t, E_a^t) - f_\tau^k(S_a^t, E_a^t, \emptyset)| = \epsilon$$

where ϵ is the approximation error. If agent a does not sense any other agent within the next k time steps, the fast-forward function g successfully approximates the final state for agent a as if iteratively applying the state update function f_τ. However, since the sensing relation may not be symmetric and g does not yield the intermediate states in $(t, t+k\tau)$ required for sensing a, other agents' state updates may deviate when applying g for a. Thus, avoiding deviations across all agents' states requires mutual independence:

$$\forall i \in \{0, \dots, k-1\} : ((N_a^{t+i\tau} = \emptyset) \wedge (\forall a' \in A \setminus \{a\} : a \notin N_{a'}^{t+i\tau}))$$

where A is the set of agents in the simulation, and $\wedge$ denotes logical conjunction. We refer to any interval $[t, t + k\tau]$ for which the above holds as an *independence interval*.

3.2 Identifying independence intervals

To allow for the identification of independence intervals, Scheutz et al. define a *translation function*, which "determines for a given location the maximum distance an agent can travel within one update" [28]. By determining the area that agents may travel to within the next k updates, independence intervals can be identified. In contrast to the translation function, the proposed fast-forward function determines the *full* agent state after k updates in case of independence from other agents' states. Thus, in contrast to the iterative time steps used by Scheutz et al., the fast-forward function allows for agent state updates across multiple time steps through a single function evaluation. We assume that the fast-forward function is accompanied by a *scanning* function, which additionally yields the time at which an agent first arrives at a given target distance if independence from other agents' states is given.

In this section, we propose methods for identifying opportunities for fast-forwarding. For efficiency, the methods are applied on the spatial granularity of edges of a graph representing the simulation space. First, we formulate conditions under which agents can be fast-forwarded across individual graph edges. Subsequently, we propose an algorithm to identify fast-forwarding opportunities across sequences of edges.

We assume that the simulation space, i.e., the road network, is represented by a graph $G = (V, E)$ comprised of a set of directed edges E representing roads, and a set of vertices V representing intersections. During an agent's lifetime, the agent traverses a predefined sequence of connected edges. Each edge traversal may require multiple state updates. Interactions with other agents can increase the number of updates required to traverse an edge. Each edge has an assigned weight l representing its length, l being at least the sensing range. For simplicity, in our description we disregard the spatial extent of the agents themselves, which we do however consider in our implementation of the approach.

3.2.1 *Single-link scanning.*

If an agent a has achieved the highest possible velocity on an edge and is located sufficiently far ahead of all other agents so that it is guaranteed that a will remain outside any other agent's sensing range, a may be eligible for fast-forwarding. To limit our consideration to the agent's current edge, we ensure that a cannot yet sense the next edge on its route and cannot be sensed from the previous edge. More formally, an independence interval for agent a covers the time interval for which the following conditions hold:

$$\begin{aligned} v(a) = v_{\max} \;&\wedge \\ d(a) > r \;&\wedge \\ d(a) < l - r \;&\wedge \\ \forall a' \in \bar{A} \setminus \{a\} : d(a') < d(a) - r& \end{aligned}$$

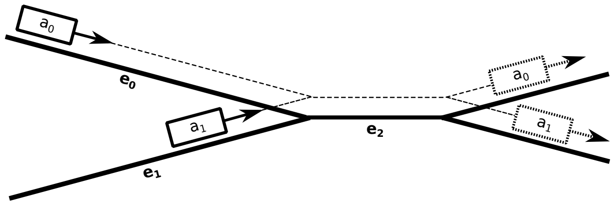
where $v(a)$ and $d(a)$ are agent a's current velocity and position on the current edge, $\bar{A}$ is the set of agents on the same edge as a, r is the sensing range, l is the length of the current edge, and $v_{\max}$ is the speed limit on the edge.

3.2.2 *Multi-link scanning.*

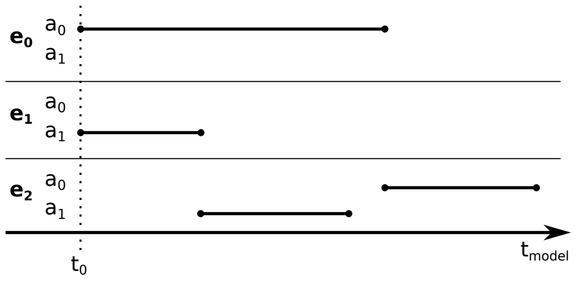
We now extend the identification of independence intervals to sequences of graph edges (cf. Figure 1). Our goal is to determine for each agent an interval during which the agent never shares an edge with another agent. In the following, we describe Algorithm 1, which determines such intervals.

The algorithm proceeds in two stages: in the first stage, each agent registers its occupancy intervals at the edges that may be traversed within a configurable *scanning horizon*. The scanning horizon limits the scanning overheads. Each edge stores the earliest time it is sensed by any agent (*occupiedFrom*), with an initial value of ∞. If a registering agent exits the edge earlier than the current value of *occupiedFrom*, we store the agent as a candidate for fast-forwarding (*earliestAgent*), together with its sensing time and exit time. Otherwise, the registering agent may interact with a previously registered agent; thus, we set *earliestAgent* to *nil*.

In the second stage, each agent once again iterates over the edges reachable within the scanning horizon. Starting at an agent a's current edge, agent a can be fast-forwarded across the longest sequence of edges for which $earliestAgent = a$ and for which the exit time is within the scanning horizon.



(a) Two agents a_0, a_1 passing the same edge e_2.



(b) Occupancy intervals for edges e_0, e_1, e_2. Since a_0 and a_1 never share their starting edges with other agents, they can be fast-forwarded across e_0 and e_1, respectively. Further, since a_0 and a_1 never occupy e_2 at the same time, both can be fast-forwarded across e_2. Thus, at minimum, the independence intervals of both a_0 and a_1 extend to the time at which the successor edge to e_2 is sensed.

Figure 1: Example of identifying independence intervals across multiple graph edges.

By limiting fast-forwarding to the agent who first occupies an edge, some opportunities are not exploited. An example is given in Figure 1: although agent a_0 never occupies edge e_2 at the same time as a_1, a_0 will not be fast-forwarded across edge e_2. To limit the costs of the scanning process, we do not consider such situations.

The size of the independence intervals depends on the scanning horizon as well as the period in model time after which single-link and multi-link scanning are repeated, which must be balanced with the incurred scanning overhead. Our implementation evaluated in Section 4 applies single-link and multi-link scanning with a configurable period length in model time. The effects of varying the scanning parameters are evaluated in Section 4.2.

3.3 Fast-forward function for the Intelligent Driver Model

In this section, we derive a fast-forward function and a scanning function for the Intelligent Driver Model (IDM). Instead of determining a vehicle's future states iteratively throughout multiple time steps, given independence from other vehicles, the functions will directly yield the vehicle state or the time. At a point t_{initial} in model time, we have a vehicle's current state given by its velocity v_{initial} and position d_{initial}. To be able to fast-forward vehicles, our goal is to calculate the vehicle state after t additional units of time. The state is comprised of the velocity v_{final} and the position d_{final}. We refer to the function that yields these two values as the *fast-forward function*. Further, for identifying independence intervals, we require the velocity $v_{\text{projected}}$ and time $t_{\text{projected}}$ at which a vehicle has traveled an additional distance d. We refer to the function that yields these two values as the *scanning function*.

Algorithm 1 Identifying independence intervals across multiple graph edges.

```
1: procedure STAGEONE
2:     for each $a \in A$ do
3:         $e \leftarrow a.currentEdge$
4:         $sensingTime \leftarrow currentTime$
5:         $exitTime \leftarrow currentTime + travelTime(a, e, e.length - a.position)$
6:         $register(e, a, sensingTime, exitTime)$
7:         $sensingTime \leftarrow currentTime +$
8:                 $travelTime(a, e, e.length - a.position - sensingRange)$
9:         $e \leftarrow a.getSuccessorEdgeOnRoute(e)$
10:        while $e \neq nil$ and $exitTime \leq scanningHorizon$ do
11:            $nextSensingTime \leftarrow exitTime +$
12:                    $travelTime(a, e, e.length - sensingRange)$
13:            $exitTime \leftarrow exitTime + travelTime(a, e, e.length)$
14:            $register(e, a, sensingTime, exitTime)$
15:            $sensingTime \leftarrow nextSensingTime$
16:            $e \leftarrow a.getSuccessorEdgeOnRoute(e)$
17: procedure STAGETWO
18:     for each $a \in A$ do
19:         $upperBoundTime \leftarrow -1$
20:         $e \leftarrow a.currentEdge$
21:         while $e \neq nil$ do
22:             if $e.earliestAgent \neq a$ then
23:                 break
24:             $e \leftarrow a.getSuccessorEdgeOnRoute(e)$
25:             $upperBoundTime \leftarrow e.occupiedFrom$
26:         if $upperBoundTime \neq -1$ then
27:             $a.setIndependenceInterval(currentTime, upperBoundTime)$
28: procedure REGISTER($e, a, sensingTime, exitTime$)
29:     if $exitTime >= currentTime + scanningHorizon$ then
30:         $exitTime \leftarrow \infty$
31:     if $exitTime < e.occupiedFrom$ then
32:         $e.earliestAgent \leftarrow a$
33:         $e.earliestAgentExitTime \leftarrow exitTime$
34:     else if $sensingTime \leq e.earliestAgentExitTime$ then
35:         $e.earliestAgent \leftarrow nil$
36:     $e.occupiedFrom \leftarrow min(e.occupiedFrom, sensingTime)$
```

In IDM, vehicles accelerate according to the following differential equation [30]:

$$\frac{dv}{dt} = a_0 \left(1 - \left(\frac{v}{v_0} \right)^{\delta} - \left(\frac{s_0 + vT + (v\Delta v)/(2\sqrt{a_0 b_0})}{s} \right)^2 \right)$$

Here, a_0 is the maximum acceleration, v is the current velocity, v_0 is the target velocity, s_0 is the minimum desired distance to the vehicle ahead, b_0 is the comfortable braking deceleration, and s and Δv are the position and velocity differences to the vehicle ahead. δ is a parameter typically set to 4 [30]. We perform our computations for this value.

When IDM is employed in time-stepped microscopic traffic simulations, the above equation is evaluated at each time step for each vehicle to calculate the vehicle's acceleration and to update its velocity and position accordingly.

Typically, a sensing range r is applied that limits the distance up to which vehicles adapt their acceleration to other nearby vehicles.

For $s > r$, the acceleration is determined solely by the *free road term*:

$$\frac{dv}{dt} = a_0 \left(1 - (\frac{v}{v_0})^\delta\right)$$

For $\delta = 4$, integration after separation of variables yields the time required to accelerate from 0m/s to v:

$$t(v) = \frac{v_0}{4a_0}\left(-\log(v_0 - v) + \log(v_0 + v) + 2\arctan(\frac{v}{v_0})\right)$$

When accelerating from initial velocity $v_{\text{initial}} > 0$m/s, the time elapsed when velocity v is reached is given by:

$$t_{v_{\text{initial}}}(v) = t(v) - t(v_{\text{initial}})$$

The distance that the vehicle has traveled when reaching velocity v can be obtained as follows:

$$d(v) = \int \frac{v}{a(v)}dv = \frac{v_0^2}{2a_0 \operatorname{arctanh}\left((\frac{v}{v_0})^2\right)}$$

Solving for v:

$$v(d) = v_0\sqrt{-\tanh\left(-\frac{2a_0 d}{v_0^2}\right)}$$

Now we have the components of the *scanning function*:

$$v_{\text{projected}} = v\left(d + d(v_{\text{initial}})\right)$$
$$t_{\text{projected}} = t_{\text{initial}} + t\left(v_{\text{projected}}\right) - t(v_{\text{initial}})$$

After skipping state updates, we must allow the simulation to resume regular time-stepped agent state updates. Thus, we round to the nearest smaller timestep t_{final} and calculate the components of the *fast-forward function*:

$$v_{\text{final}} = v\left(t_{\text{final}}\right)$$
$$d_{\text{final}} = d_{\text{initial}} + d\left(v_{\text{final}}\right)$$

We do not have a closed form for $v(t)$. However, since $t(v)$ is twice differentiable, we can postulate $t(v) - t = 0$ and apply Halley's root-finding method [9] to compute v numerically at cubical convergence speed. We terminate once the change in values is below 10^{-10}.

Due to varying speed limits among the roads in a traffic simulation, vehicles may exceed the speed limit if the limit decreases among subsequent roads. An extension to IDM has been proposed to apply decelerations to vehicles using the following differential equation [16]:

$$\frac{dv}{dt} = -a_0\left(1 - (\frac{v_0}{v})^\delta\right)$$

We derive fast-forward and scanning functions for this situation as well. For $\delta = 4$, integration after separation of variables yields:

$$t(v) = -\frac{1}{2a_0}\left(v_0(\arctan(\frac{v_0}{v}) - \operatorname{arctanh}(\frac{v_0}{v})) + 2v\right)$$

We can obtain the distance at velocity v as follows:

$$d(v) = \int \frac{v}{a(v)}dv = -\frac{1}{2a_0}\left((v^2 - v_0^2)\operatorname{arctanh}((\frac{v_0}{v})^2)\right)$$

Here, we apply Halley's method to obtain $v(d)$ and $v(t)$ and proceed as above.

Now we have fast-forward and scanning functions covering both acceleration and deceleration based on the continuous formulation of IDM. Since the time-stepped state updates in a simulation only approximate the acceleration behavior prescribed by the model, a deviation occurs between the fast-forward function and iterative time steps. The magnitude of the deviation depends on the time step size τ and vanishes for $\tau \to 0$. The effects of the deviation are discussed in the following section. We quantify the deviation in Section 4.1.

3.4 Discussion

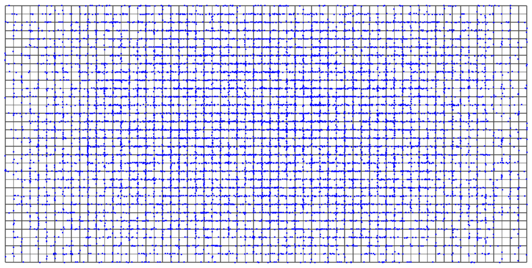
IDM is defined by a time-continuous differential equation specifying a vehicle's acceleration behavior. Time-stepped microscopic traffic simulations approximate the specified behavior by calculating new acceleration values at each time step and updating the vehicles' velocities and positions accordingly. Smaller time step sizes increase the quality of the approximation, but are associated with higher computational cost. In contrast, the fast-forward function produces a "smooth" acceleration behavior without discretization to intermediate time steps. As such, updates performed using the fast-forward function can in fact be considered more in line with the intended acceleration behavior of IDM than iterative time-stepped updates. Still, when applying fast-forwarding, the simulation results will deviate from a purely time-stepped execution. A further potential source of deviations is given by the fact that occupancy intervals are determined using the scanning function, and are therefore affected by deviations as well. When a vehicle does not approach a road segment using the fast-forward function but using iterative time steps, the predicted occupancy interval may slightly deviate from the observed interval during which the vehicle occupies the road segment. Thus, it is possible that a vehicle is fast-forwarded based on the incorrect assumption that it will be isolated on the road segment. However, if the edges of the considered graph are substantially longer than the sensing range, it is unlikely that that such a deviation will lead to a vehicle entering another vehicle's sensing range. We did not observe this latter type of deviation in our experiments.

4 EVALUATION

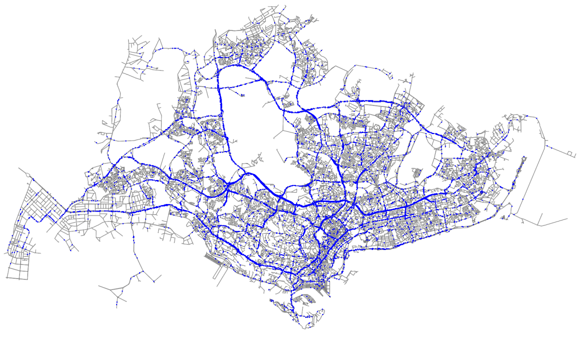
In this section, we aim to answer the following questions:

- *How large is the deviation in the simulation results between a purely time-stepped execution and the proposed fast-forwarding approach?*
- *In which scenarios and to what degree can fast-forwarding accelerate microscopic road traffic simulations?*

Our evaluation is performed using the city-scale microscopic traffic simulator CityMoS [33]. Two road networks are considered: a synthetic grid-shaped road network (cf. Figure 2a), and a representation of the road network of Singapore (cf. Figure 2b). The grid network is comprised of 64×32 rectangles, each edge being 200m in length. There are two edges between two adjacent vertices with opposite traffic directions, resulting in a total length of 1600km. In the Singapore network, the average and total length of the edges is around 36.2m and 8700km, respectively. In both scenarios, origin and destination pairs are chosen uniformly at random on the road network. Route planning is based on Dijkstra's algorithm, using the edges' lengths and speed limits as their weights. Agents start their trips uniformly at random in model time. In the grid scenario, we started the measurements after a warm-up phase of 1800s to achieve



(a) Grid road network with around 10 000 vehicles.



(b) Singapore road network with around 19 500 vehicles.

Figure 2: Considered road networks with traffic, blue dots denoting vehicles.

roughly constant agent populations of 500, 2 000, and 10 000. After the warm-up phase, each measurement continued for 1h of model time. In the Singapore scenario, we used a warm-up phase of 1800s and subsequently measured the performance for 1h of model time while the agent population is ramping up to about 6 000 and 19 500 agents, respectively. Vehicles accelerate according to IDM[30] and perform lane changes according to the rules described in [32]. We configured a sensing range of 40m facing forward.

We varied the following algorithm parameters:

- **Single-link and multi-link scanning period**: the identification of independence intervals and the fast-forwarding are performed periodically. Since the scanning overhead depends on whether individual graph edges or sequences of graph edges are considered, the period length for each variant is varied separately.
- **Scanning horizon**: the overhead of scanning and the size of independence intervals depend on the maximum delta in model time that agents may be fast-forwarded.

For the grid scenario, we performed a parameter sweep to study the effect of different parameter combinations on the simulation performance. The levels in seconds of model time were $\{0.5, 2, 8, 32\}$ and $\{0.5, 2, 8, 32, 128\}$ for the single-link and multi-link scanning period, and $\{16, 64, 256\}$ for the scanning horizon. In the Singapore scenario, we applied a simple auto-tuning approach to select and vary parameter combinations at runtime: a set of preconfigured parameter combinations is set one after the other, measuring the simulation progress per unit wall-clock time for each combination. The simulation then proceeds with the best-performing parameter combination. The auto-tuning process is repeated once either

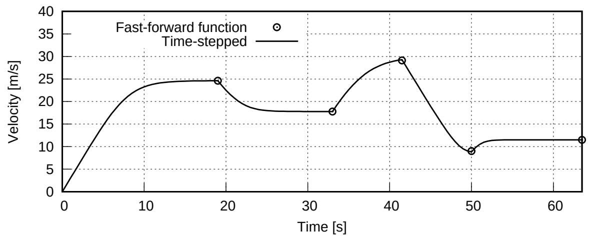


Figure 3: Example of the velocity calculated using time-stepped updates and fast-forwarding on a sequence of road segments with randomized lengths and speed limits, for a time step size of 0.5s.

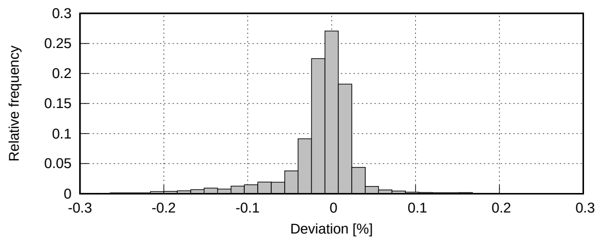
1200s of model time have passed or the simulation performance has changed by more than a factor of 2. We configured the following parameter combinations for the single-link scanning period, multi-link scanning period, and the scanning horizon: (2, 32, 64), (2, 128, 64), (4, 32, 64), (4, 128, 64), and (2, ∞, 64). In the grid scenario, each simulation run was repeated at least 3 times. Each run of the Singapore scenario was repeated 20 times. All performance measurements were performed on a single core of an Intel i5-7400 CPU running at 3.00GHz with 16GiB of RAM.

4.1 Validation

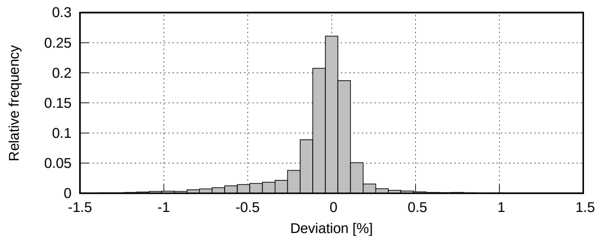
To evaluate the deviation between the simulation results of a purely time-stepped execution and our fast-forwarding approach, we first consider a single vehicle and compare the distance after a certain amount of model time when traversing a sequence of road segments with varying speed limits. We then compare aggregated statistics over all vehicles across entire simulation runs.

4.1.1 Individual fast-forwarding operations. The relationship between time-stepped and fast-forwarding updates is illustrated in Figure 3. We plot the velocities computed using time-stepped updates on a sequence of road segments of randomized lengths and speed limits, and compare the results with those of evaluating the fast-forward function at the end of each road segment. Visually, the deviation between the two methods is marginal.

To quantify the deviation, we recorded the relative deviation between a vehicle's position calculated using fixed time steps and fast-forwarding after 60s of traveling time across a sequence of roads with lengths uniformly distributed between 50m and 500m and speed limits uniformly distributed between 10m/s and 30m/s. We performed 10 000 runs each for step sizes of 0.1s and 0.5s. The overall distance traveled was between about 625m and 1630m. The result is shown in Figure 4. Most deviations are lower than 0.3% for time step size 0.1s, and lower than 1.5% for time step size 0.5s. The largest observed deviations were 0.85% and 3.29%, respectively. The slight bias towards negative deviations is likely due to the use of different equations for acceleration and deceleration behaviors (cf. Section 3.3). However, as discussed in Section 3.4, we note that since the fast-forward function is based directly on the time-continuous formulation of IDM, the results after fast-forwarding may in fact be considered closer to the desired acceleration behavior than the results after a sequence of time-stepped updates.



(a) Time step size 0.1s.



(b) Time step size 0.5s.

Figure 4: Relative deviation between positions calculated using iterative time steps and fast-forwarding after 60s of traveling time across a sequence of road segments with randomized lengths and speed limits.

Table 1: Average trip durations [s] in time-stepped and fast-forwarding runs with 95% confidence intervals.

Scenario	Grid		
Agent count	**500**	**2 000**	**10 000**
Time-stepped	371.4 ± 0.6	371.0 ± 0.6	377.4 ± 0.1
Fast-forwarding	371.4 ± 1.6	371.0 ± 2.6	377.3 ± 0.3

Scenario	Singapore	
Peak agent count	**6 000**	**19 500**
Time-stepped	859.2 ± 4.0	1096.5 ± 5.3
Fast-forwarding	860.0 ± 3.8	1099.7 ± 5.9

4.1.2 Grid and Singapore scenarios. For the grid and Singapore scenarios, we conduct the validation with respect to the average trip duration, which is a commonly studied metric in transportation engineering. The comparison results are shown in Table 1. We performed a parameter sweep across the scanning parameters for the grid scenario. The validation results are given for the parameter combinations resulting in the lowest execution times. For the Singapore scenario, the parameters were configured at runtime using auto-tuning. The time step size was 0.1s. We observe that in both scenarios, there is no significant deviation in the overall simulation results between the time-stepped execution and fast-forwarding.

4.2 Performance measurements

4.2.1 Fast-forward function. To understand the potential for performance gains using the proposed approach, we first compare the computational cost of iterative time-stepped agent updates and updates using the proposed fast-forward function. We simulate a single vehicle on a road segment of 10km length. Initially, the velocity is 0km/h. The vehicle accelerates to the speed limit of 100km/h. In Figure 5, we compare the wall-clock time required to execute a certain number of time steps to the time required to advance a vehicle by the same distance using fast-forwarding. Due to the fine-grained nature of agent updates, each measurement was repeated 10^7 times in the time-stepped case, and 100 000 times for fast-forwarding. The figure shows averages over the repetitions. 95% confidence intervals are plotted but are too small to be visible.

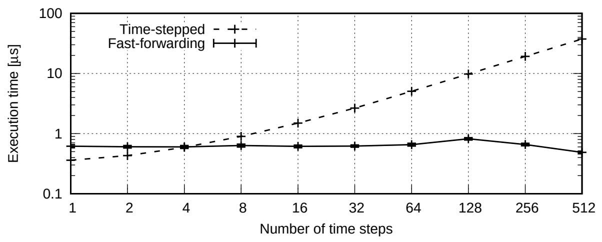


Figure 5: Wall-clock computation time required for a certain number of steps using time-stepped execution and fast-forwarding for a time step size of 0.1s.

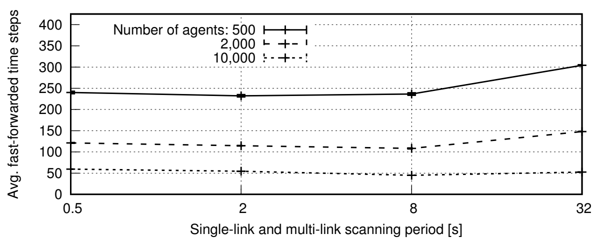
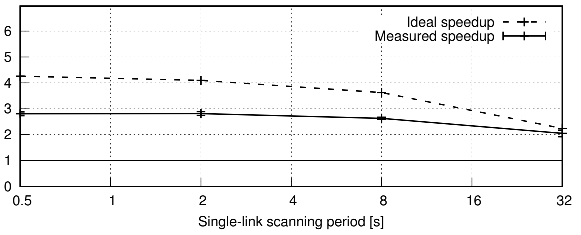


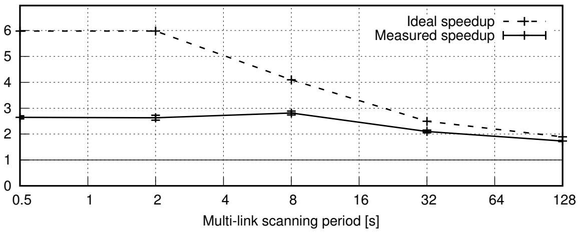
Figure 6: Average number of time steps skipped per fast-forwarding operation in the grid scenario with a fixed scanning horizon of 64s and identical single-link and multi-link scanning periods. Although larger scanning periods provide fewer fast-forwarding opportunities, the number of time steps per individual fast-forwarding operation increases.

Figure 5 shows that the execution time of the fast-forward function is roughly constant, whereas the execution time of time-stepped updates depends approximately linearly on the number of steps. Although the fast-forward function is associated with higher computational cost than an individual time-stepped state update, fast-forwarding outperforms time-stepped updates beyond 4 consecutive steps. Thus, assuming no additional overheads, fast-forwarding is beneficial when the average number of fast-forwarded time steps is larger than 4. Generally, reducing the time step size of the simulation will allow for a larger number of skipped time steps per fast-forwarding operation. However, due to the overall increase in time steps, the proportion of skipped time steps across the simulation run will remain roughly the same.

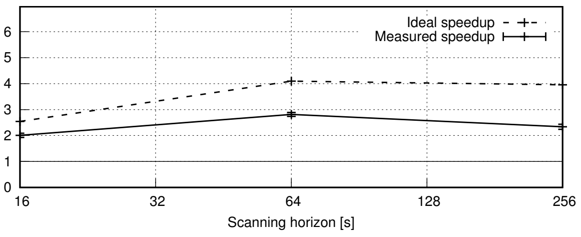
4.2.2 Grid scenarios. Figure 6 shows the average number of time steps skipped per fast-forwarding operation in the grid scenario with a time step size of 0.1s, varying the single-link and multi-link scanning periods. Even in the most congested scenario with 10 000



(a) Multi-link scanning period 8s, scanning horizon 64s.

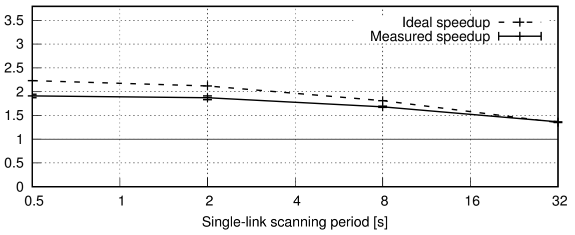


(b) Single-link scanning period 2s, scanning horizon 64s.

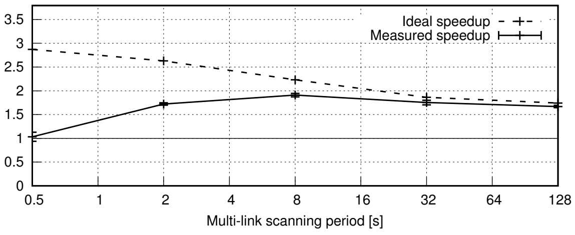


(c) Single-link scanning period 2s, multi-link period 8s.

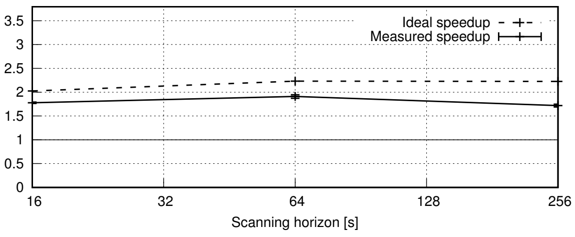
Figure 7: Ideal and measured speedup with 500 agents in the grid scenario.



(a) Multi-link scanning period 8s, scanning horizon 64s.



(b) Single-link scanning period 0.5s, scanning horizon 64s.



(c) Single-link scanning period 0.5s, multi-link period 8s.

Figure 8: Ideal and measured speedup with 2 000 agents in the grid scenario.

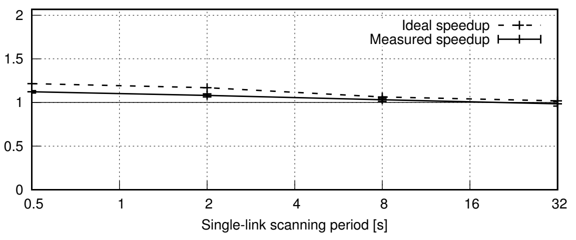
agents, around 50 time steps were skipped per fast-forwarding operation, far beyond the break-even point of 4 time steps shown in Figure 5.

Figures 7 to 9 show the overall speedup achieved using the fast-forwarding approach compared with a purely time-stepped execution for the grid scenario for 500, 2 000, and 10 000 agents, respectively. In addition, we plot the relative reduction in state updates, which indicates the ideal speedup through the fast-forwarding approach when disregarding the overhead for identifying independence intervals and the evaluation of the fast-forward function. For instance, if the number of time steps is reduced by 50%, the ideal speedup is 2.

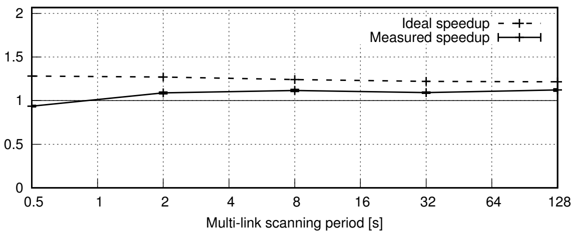
We can observe in Figure 7 that due to substantial opportunities for fast-forwarding with only 500 agents, frequent single-link and multi-link scanning and a large scanning horizon are beneficial. In Figure 7b, when the multi-link scanning period is increased, the ideal speedup decreases substantially from multi-link scanning periods of 2s to 8s. However, the measured speedup is virtually unchanged due to the trade-off between the scanning overhead and the benefit of skipping updates. The largest speedup achieved with 500 agents is 2.82 with single-link and multi-link scanning periods of 2s and 8s, and a scanning horizon of 64s. With 2 000 agents (cf. Fig. 8), the trade-off between identifying opportunities for fast-forwarding and the associated overhead becomes more pronounced: a speedup of 1.91 is achieved with single-link and multi-link scanning periods of 0.5s and 8s, respectively, and a scanning horizon of 64s, whereas no speedup is achieved when decreasing the multi-link scanning period of 0.5s (cf. Fig. 8b). Compared to the above two cases, the most congested scenario with 10 000 agents (cf. Fig. 9) provides fewer opportunities for fast-forwarding. Additionally, since the costs of identifying independence intervals increases with the number of agents, frequent multi-link scanning is not beneficial. A maximum speedup of 1.12 is achieved with single-link and multi-link scanning periods of 0.5s and 128s, and a scanning horizon of 16s.

Overall, the measurements show that controlling the overhead for identifying independence intervals is critical when applying our approach in practice. Further, the opportunities for fast-forwarding decrease with denser traffic.

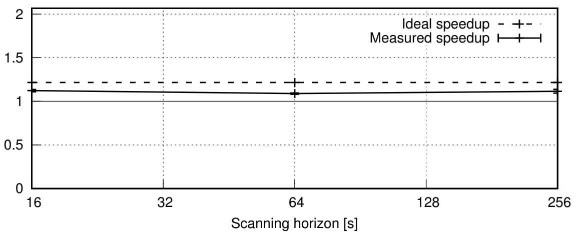
4.2.3 Singapore scenario. The performance results for the Singapore road network are shown in Table 2. We observe that 22.1% and 10.3% of time steps were skipped using auto-tuning for peak agent counts of 6 000 and 19 500, respectively. The average number of time steps per fast-forwarding operation for both cases is much larger than the break-even point of 4 steps (cf. Figure 5). With 6, 000 agents, a speedup of 1.22 is achieved. With 19 500 agents, only few opportunities for fast-forwarding could be identified, allowing for a speedup of 1.05.



(a) Multi-link scanning period 0.5s, scanning horizon 16s.



(b) Single-link scanning period 0.5s, scanning horizon 16s.



(c) Single-link scanning period 0.5s, multi-link period 128s.

Figure 9: Ideal and measured speedup with 10 000 agents in the grid scenario.

Table 2: Performance in the Singapore scenario with scanning parameters configured using auto-tuning.

Peak agent count	**6 000**	**19 500**
Time-stepped execution time [s]	214.3 ± 3.4	789.1 ± 5.9
Steps skipped [%]	22.1 ± 0.1	10.3 ± 0.1
Steps skipped per fast-forwarding	85.45 ± 0.16	81.4 ± 0.15
Scanning overhead [s]	2.5 ± 0.0	7.4 ± 0.1
Fast-forwarding overhead [s]	2.0 ± 0.0	4.9 ± 0.0
Speedup	1.22 ± 0.03	1.05 ± 0.01

5 DISCUSSION

In this section, we discuss limitations and potential enhancements of the proposed fast-forwarding approach.

Applicability to more complex models and other domains: in the present paper, we assume that the fast-forward and scanning functions return a single agent state and time, implying that the routes of the agents (i.e., the sequences of edges) will not change after scanning has been performed. Depending on the considered simulation model, agents may change their routes on interaction with other agents or even spontaneously. For instance, an agent may re-route when it detects traffic congestion. To consider such models, we could either terminate the scanning process at the first point in time when a change is possible, or determine occupancy intervals for all possible branches. Both approaches may substantially reduce the opportunities for fast-forwarding compared to what has been shown in the simulation scenarios of the present paper. If routing decisions are made stochastically, pre-sampling from the pseudo-random number stream may still enable prediction of the agents' routes. In the extreme case of entirely unpredictable routes, fast-forwarding would be limited to disjoint areas reachable by agents according to their maximum velocity.

From the problem analysis in Section 3.1, we can infer that the applicability of our approach to other types of time-stepped agent-based simulations depends on the specific models used. The approach is applicable to models that allow for the prediction of future agent states and to scenarios where independent agent updates occur. For instance, in crowd simulations using predefined routes on a two-dimensional simulation space, the path taken by isolated agents may be fully predictable. The performance benefits of the approach depend on the computational cost of the fast-forward function relative to time-stepped updates as well as the costs for determining independence intervals.

Deviations compared to time-stepped execution: As discussed in Section 3 and evaluated in Section 4.1, state updates performed using the fast-forward function deviate slightly from those performed using iterative time steps. Although the fast-forward function can be argued to be closer to the behavior specified in IDM and the deviations are low, two undesirable properties emerge: first, as with any state alteration in an agent-based simulation, deviations may propagate through the road network. Second, the deviations depend on the parametrization of the fast-forwarding approach, i.e., on the frequency of scanning and on the scanning horizon. Thus, the approach interlinks the *execution* of the simulation and the observed *behavior* of the simulation. Ideally, to allow modelers to clearly identify cause-and-effect relationships when modifying the model or scenario, these two aspects should be decoupled.

Influence on statistics collection and visualization: Typically, in an agent-based simulation, statistics are gathered by periodically aggregating over the states of the agents in the simulation, e.g., over the velocities of the vehicles on a road network. Usually, the period of aggregation is a multiple of the time step size. If updates are performed asynchronously according to the proposed approach, agents may be fast-forwarded beyond the point when statistics are to be collected. A simple solution is to limit the scanning horizon to the next statistics gathering time. Visualization tools could apply interpolation to approximate agent states at time steps that have been skipped through fast-forwarding.

Further opportunities for fast-forwarding: the proposed multi-link scanning approach operates on the granularity of edges in a road network. Thus, if the modeled road network has a high percentage of edges that are substantially longer than the sensing range of vehicles, many opportunities for fast-forwarding are not exploited. By identifying independence intervals on a finer granularity, additional fast-forwarding opportunities could be unlocked. However, more complex scanning may incur an increase in overhead. Further, for efficiency, we limited fast-forwarding to the first agent that senses a graph edge. Fast-forwarding of multiple agents occupying the same edge at disjoint time intervals within the scanning horizon may enable further speedup in some scenarios. Further, in road traffic simulations that consider traffic lights, trivial opportunities for fast-forwarding may be given for vehicles stopped in front of traffic lights. Such vehicles can be fast-forwarded to the next state change of the traffic lights. Finally, we only consider skipping opportunities for individual agents. An increase in fast-forwarding opportunities may be possible if the fast-forward function can be extended to clusters of agents.

Controlling overhead: a number of ways present themselves to control the scanning overhead and thus to enable more complex scanning procedures: first, the proposed algorithm is parametrized with the scanning periods and scanning horizon. Our evaluation showed that the optimal values for the parameters depend strongly on the considered scenario. Thus, we applied a simple auto-tuning scheme to adapt the scanning parameters according to the traffic conditions of the Singapore scenario.

Second, in addition to the variation of congestion across model time, congestion also typically shows variations across the simulated space. For instance, during peak hours a highly congested speedway will provide few opportunities for fast-forwarding, in contrast to sparsely populated roads in residential areas. In such situations, to avoid unnecessary computations, scanning could be restricted to areas outside congested areas. However, further considerations are then required to maintain correctness. Since vehicles may enter or exit congested areas within the considered scanning horizon, excluding agent interactions would require a safety margin around these areas, which could be defined based on static information such as speed limits.

Finally, the scanning operation may be offloaded to a separate processor. Within the accuracy allowed by the time step size, previously identified occupancy intervals may be outpaced by the simulation's progress, but not invalidated. Thus, after scanning, fast-forwarding could be applied to all agents that have not yet progressed beyond the target time. Further, during scanning, the scanning function is evaluated a number of times for each relevant vehicles independently, providing ample opportunities for parallelization, e.g., on graphics processing units.

6 RELATED WORK

In this section, we give an overview of previous work focusing on identifying and exploiting independence between state updates for parallelization of discrete-event simulations and for accelerating sequential and parallel time-stepped agent-based simulations. Further, we discuss hybrid modeling and simulation approaches that execute parts of a simulation microscopically, while applying less detailed and therefore less computationally intensive models for parts of the simulation where full accuracy is not required.

6.1 Exploiting independent state updates

The approach proposed in the present paper bears some similarities with methods from the field of parallel and distributed simulation, which is concerned with the execution of individual simulation runs on a set of inter-connected processing elements [8]. To reduce the cost of synchronization between processing elements, methods have been proposed to exploit *lookahead*, i.e., the difference in model time between an event's creation and execution time [7]. If a lower bound on the lookahead can be determined either prior to the simulation or at runtime [18, 24], intervals in model time can be identified during which processing elements can compute independently. Some previous works consider the minimum model time required for a sequence of events to propagate to a remote processing element [3, 4, 19, 20, 23, 27, 31]. Similarly to our approach, intervals of independence are derived according to the topology of the modeled system. However, instead of exploiting the identified independence for parallel execution, in our work, we accelerate sequential simulations by performing independent agent state updates using a computationally inexpensive fast-forward function. A further similarity exists to optimistically synchronized parallel and distributed simulations [6, 26], where some computations are performed speculatively and rolled back when a violation of the simulation correctness is detected. In our approach, the identification of occupancy intervals can be seen as speculative state updates under the assumption of independence among agents. When independence between the agent updates cannot be guaranteed, the results are discarded.

Some previous works have considered ways of accelerating time-stepped agent-based simulations by identifying independent state updates among agents: Scheutz et al. [13, 28, 29] apply a *translation function* that reflects the furthest possible amount of movement of an agent to determine an *event horizon* in model time. By identifying non-overlapping areas among multiple agents' event horizons, time intervals of mutually independent updates can be identified. Now, in the context of sequential agent-based simulations, agent updates can be prioritized to achieve the simulation's termination criterion with the minimum number of state updates. For instance, if the focus of the simulation study is on a particular agent, only the state updates directly or indirectly affecting this agent must be performed. In distributed agent-based simulations, idle times due to data dependencies can be reduced by prioritizing agent updates according to the data dependencies across processing elements. In contrast to our work, runtime reductions are achieved through changes in the ordering of agent updates, not through accelerating the state updates themselves. Since road traffic simulations are typically executed until all agents have reached a certain point in model time, the approach by Scheutz et al. would not accelerate such simulations.

Buss et al. [2] proposed a discrete-event modeling approach for scenarios involving movement and sensing. Instead of explicitly updating an entity's location over a sequence of time steps, events are scheduled at points in model time where changes in movement occur. However, determining suitable event scheduling times for sets of interacting vehicles may incur substantial overhead. Thus, in contrast to the purely discrete-event approach proposed by Buss et al., our proposed fast-forwarding approach maintains a time-stepped execution for all agents currently involved in an interaction. Further, while the work by Buss et al. and another work with a similar focus by Meyer [22] share with ours the general idea of avoiding explicit intermediate state changes, the main challenge lies in determining the points in model time when interactions between entities may occur and in determining the new agent state. In the present paper, we address these challenges in the context of microscopic road traffic simulations.

Less closely related to our approach is the concept of simulation cloning [14]. In this approach, the total execution time of a set of simulation runs is reduced by computing only the divergent state updates across multiple runs. For instance, if the behavior of only a single agent is modified across multiple runs, state changes of other agents that are unaffected by this agent are not recomputed [25]. Similarly, in updatable and exact-differential simulation [5, 12], intermediate events of an initial full simulation run are stored. Subsequent simulation runs branch off from this initial simulation run, reusing stored events that are unaffected by the branching. As in these approaches, fast-forwarding exploits the independence between state updates to accelerate simulations. However, instead of avoiding recomputation, the fast-forwarding approach proposed in the present paper avoids computation of some updates entirely.

Finally, the term "fast-forwarding" was used previously in other contexts where existing information is exploited to advance a simulated entity in model time. In the updatable simulations proposed by Ferenci et al. [5], some repeated event executions can be avoided, thus "fast-forwarding" the corresponding simulated entity. Mauve et al. [21] use the term "fast forward" to describe the re-execution of events after a rollback in the context of optimistic synchronization for distributed virtual environments.

6.2 Hybrid traffic simulation

In hybrid traffic simulation [1], microscopic models are combined with mesoscopic or macroscopic models to balance simulation fidelity and performance. Spatial or temporal segments of the simulation are selected in which a reduction in modeling detail and accuracy is acceptable. In these segments, vehicles are considered in aggregate, e.g., as sets of tasks in a queuing network or in terms of fluid dynamics. As a consequence, it is not always possible to study an individual vehicle across its entire route. The fast-forwarding approach proposed in the present paper bears a superficial similarity with hybrid traffic simulation in its reliance on an analytical solution for some of the state updates instead of a purely time-stepped execution. However, fast-forwarding is applied only if it is ensured that within the accuracy allowed by the simulation's time step size, the simulation results are unaffected. Since fast-forwarding does not consider agents in aggregate, each vehicle's progress on its route can still be studied individually.

7 CONCLUSIONS AND OUTLOOK

We propose an approach to accelerate microscopic traffic simulation by identifying intervals of independent state updates and performing such independent updates using a computationally inexpensive fast-forward function. The approach maintains the microscopic nature of the simulation. We derived a fast-forward function for the well-known Intelligent Driver Model. We evaluated the approach for a synthetic scenario and the road network of the city of Singapore. Our validation shows that the deviation from a purely time-stepped execution is marginal. The performance benefit of the approach depends strongly on the level of agent density in the scenarios. For scenarios with sparse traffic, a speedup of up to 2.8 was achieved, whereas with dense traffic, the reduced amount of opportunities for fast-forwarding allowed for only limited performance gains. One avenue for future work lies in identifying further opportunities for fast-forwarding, e.g., by increasing the spatial resolution of the approach, or by considering clusters of cars jointly. Further, methods to control the overhead of the approach could improve performance. For instance, avoiding attempts for fast-forwarding in congested areas of the road network could reduce unnecessary computations, while requiring further considerations to maintain correctness. Offloading the computational overhead to a separate processing element could hide some of the overhead. Finally, the fast-forwarding approach could be extended to models with more complex agent movement behaviors such as crowd models.

8 ACKNOWLEDGMENTS

This work was financially supported by the Singapore National Research Foundation under its Campus for Research Excellence And Technological Enterprise (CREATE) programme.

REFERENCES

[1] Wilco Burghout, Haris Koutsopoulos, and Ingmar Andreasson. 2005. Hybrid Mesoscopic-Microscopic Traffic Simulation. *Transportation Research Record: Journal of the Transportation Research Board* 1934 (2005), 218–255.
[2] Arnold H Buss and Paul J Sánchez. 2005. Simple Movement and Detection in Discrete Event Simulation. In *Proceedings of the Winter Simulation Conference*. Winter Simulation Conference, 992–1000.
[3] Moo-Kyoung Chung and Chong-Min Kyung. 2006. Improving Lookahead in Parallel Multiprocessor Simulation Using Dynamic Execution Path Prediction. In *Proceedings of the Workshop on Principles of Advanced and Distributed Simulation*. IEEE, 11–18.
[4] Ewa Deelman, Rajive Bagrodia, Rizos Sakellariou, and Vikram Adve. 2001. Improving Lookahead in Parallel Discrete Event Simulations of Large-scale Applications Using Compiler Analysis. In *Proceedings of the Workshop on Parallel and Distributed Simulation*. IEEE Computer Society, Washington, DC, USA, 5–13. http://dl.acm.org/citation.cfm?id=375658.375659
[5] Steve L Ferenci, Richard M Fujimoto, Mostafa H Ammar, Kalyan Perumalla, and George F Riley. 2002. Updateable Simulation of Communication Networks. In *Proceedings of the Workshop on Parallel and Distributed Simulation*. IEEE Computer Society, 107–114.
[6] Richard Fujimoto. 2015. Parallel and Distributed Simulation. In *Proceedings of the Winter Simulation Conference*. IEEE Press, 45–59.
[7] R. M. Fujimoto. 1988. Lookahead in Parallel Discrete Event Simulation. *Proceedings of the International Conference on Parallel Processing, Vol. 3* (1988), 34–41.
[8] Richard M Fujimoto. 2000. *Parallel and Distributed Simulation Systems*. Wiley New York.
[9] Walter Gander. 1985. On Halley's Iteration Method. *The American Mathematical Monthly* 92, 2 (1985), 131–134.
[10] Peter G Gipps. 1981. A Behavioural Car-following Model for Computer Simulation. *Transportation Research Part B: Methodological* 15, 2 (1981), 105–111.
[11] Peter G Gipps. 1986. A Model for the Structure of Lane-changing Decisions. *Transportation Research Part B: Methodological* 20, 5 (1986), 403–414.
[12] Masatoshi Hanai, Toyotaro Suzumura, Georgios Theodoropoulos, and Kalyan S Perumalla. 2015. Exact-differential Large-scale Traffic Simulation. In *Proceedings of the Conference on Principles of Advanced Discrete Simulation*. ACM, 271–280.
[13] Jack Harris and Matthias Scheutz. 2012. New Advances in Asynchronous Agent-based Scheduling. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*. WorldComp, 1.
[14] Maria Hybinette and Richard M Fujimoto. 2001. Cloning Parallel Simulations. *ACM Transactions on Modeling and Computer Simulation* 11, 4 (2001), 378–407.
[15] Arne Kesting, Martin Treiber, and Dirk Helbing. 2007. General Lane-changing Model MOBIL for Car-following Models. *Transportation Research Record: Journal of the Transportation Research Board* 1999 (2007), 86–94.
[16] Arne Kesting, Martin Treiber, and Dirk Helbing. 2008. Agents for Traffic Simulation. In *Multi-Agent Systems Simulation and Applications*, Adelinde M. Uhrmacher and Danny Weyns (Eds.). CRC Press, Chapter 11, 325–356. http://arxiv.org/abs/0805.0300
[17] Michael Lees, Brian Logan, and Rob Minson. 2005. Modelling environments for distributed simulation. In *Environments for Multi-Agent Systems*. 150–167. http://www.springerlink.com/index/81g6x6elxhx9tbnq.pdf
[18] Y.-B. Lin and E.D. Lazowska. 1990. Exploiting Lookahead in Parallel Simulation. *IEEE Transactions on Parallel and Distributed Systems* 1, 4 (1990), 457–469. https://doi.org/10.1109/71.80174
[19] Jason Liu and David M Nicol. 2002. Lookahead Revisited in Wireless Network Simulations. In *Proceedings of the Workshop on Parallel and Distributed Simulation*. IEEE Computer Society, 79–88.

[20] Boris D. Lubachevsky. 1989. Efficient Distributed Event-Driven Simulations of Multiple-Loop Networks. *Commun. ACM* 32, 1 (1989), 111–123.

[21] Martin Mauve, Jürgen Vogel, Volker Hilt, and Wolfgang Effelsberg. 2004. Local-lag and Timewarp: Providing Consistency for Replicated Continuous Applications. *IEEE Transactions on Multimedia* 6, 1 (2004), 47–57.

[22] Ruth Meyer. 2014. Event-Driven Multi-Agent Simulation. In *International Workshop on Multi-Agent Systems and Agent-Based Simulation.* Springer, 3–16.

[23] Richard A Meyer and Rajive L Bagrodia. 1999. Path Lookahead: a Data Flow View of PDES Models. In *Proceedings of the Workshop on Parallel and Distributed Simulation.* IEEE, 12–19.

[24] D.M. Nicol and J.H. Saltz. 1988. Dynamic Remapping of Parallel Computations with Varying Resource Demands. *IEEE Trans. Comput.* 37, 9 (1988), 1073–1087. https://doi.org/10.1109/12.2258

[25] Philip Pecher, Michael Hunter, and Richard Fujimoto. 2015. Efficient Execution of Replicated Transportation Simulations with Uncertain Vehicle Trajectories. *Procedia Computer Science* 51 (2015), 2638–2647.

[26] Kalyan S Perumalla, Mohammed M Olama, and Srikanth B Yoginath. 2016. Model-Based Dynamic Control of Speculative Forays in Parallel Computation. *Electronic Notes in Theoretical Computer Science* 327 (2016), 93–107.

[27] Patrick Peschlow, Andreas Voss, and Peter Martini. 2009. Good News for Parallel Wireless Network Simulations. In *Proceedings of the International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems.* ACM, 134–142.

[28] Matthias Scheutz and Jack Harris. 2010. Adaptive Scheduling Algorithms for the Dynamic Distribution and Parallel Execution of Spatial Agent-based Models. *Parallel and Distributed Computational Intelligence* 269 (2010), 207–233.

[29] Matthias Scheutz and Paul Schermerhorn. 2006. Adaptive Algorithms for The Dynamic Distribution and Parallel Execution of Agent-based Models. *J. Parallel and Distrib. Comput.* 66, 8 (2006), 1037–1051. https://doi.org/10.1016/j.jpdc.2005.09.004

[30] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. 2000. Congested Traffic States in Empirical Observations and Microscopic Simulations. *Physical Review E* 62, 2 (February 2000), 1805–1824.

[31] Jun Wang, Zhenjiang Dong, Sudhakar Yalamanchili, and George Riley. 2013. Optimizing Parallel Simulation of Multicore Systems Using Domain-Specific Knowledge. In *Proceedings of the Conference on Principles of Advanced Discrete Simulation.* ACM, 127–136.

[32] 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 - Special Issue on PADS 2015* 27, 2 (2017), 14:1–24.

[33] Daniel Zehe, Suraj Nair, Alois Knoll, and David Eckhoff. 2017. Towards City-MoS: A Coupled City-Scale Mobility Simulation Framework. In *5th GI/ITG KuVS Fachgespräch Inter-Vehicle Communication.* FAU Erlangen-Nuremberg.