# Relaxing Synchronization in Parallel Agent-based Road Traffic Simulation

YADONG XU, Nanyang Technological University
WENTONG CAI, Nanyang Technological University
HEIKO AYDT, TUM CREATE Ltd
MICHAEL LEES, University of Amsterdam
DANIEL ZEHE, TUM CREATE Ltd

Large-scale agent-based traffic simulation is computationally intensive. Parallel computing can help to speed-up agent-based traffic simulation. Parallelization of agent-based traffic simulations is generally achieved by decomposing the road network into subregions. The agents in each subregion are executed by a Logical Process (LP). There are data dependencies between LPs which require synchronization of LPs. An asynchronous protocol allows LPs to progress and communicate asynchronously. LPs use *lookahead* to indicate the time to synchronize with other LPs. Larger lookahead means less frequent synchronization operations. High synchronization overhead is still a major performance issue of large-scale parallel agent-based traffic simulations. In this paper, two methods to increase the lookahead of LPs for an asynchronous protocol are developed. They take advantage of uncertainties in traffic simulation to relax synchronization without altering simulation results statistically. Efficiency of the proposed methods is investigated in the parallel agent-based traffic simulator SEMSim Traffic. Experiment results showed that the proposed methods are able to reduce overall running time of the parallel simulation compared to existing methods.

Categories and Subject Descriptors: I.6.8 [**Simulation and Modeling**]: Types of Simulation—*Parallel, Discrete event*; I.6.3 [**Simulation and Modeling**]: Applications

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Agent-based road traffic simulation, asynchronous and conservative synchronization, relaxation

**ACM Reference Format:**
Yadong Xu, Wentong Cai, Heiko Aydt, Michael Lees, and Daniel Zehe, 2015. Relaxing Synchronization in Parallel Agent-based Road Traffic Simulation. *ACM Trans. Model. Comput. Simul.* 0, 0, Article 0000 ( 2016), 24 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

## 1. INTRODUCTION

Today, road traffic in many large cities and mega-cities is facing severe problems such as congestion and high emissions. They diminish the comfort and health of urban inhabitants. To study road traffic and solve urban traffic problems, modeling and simulation of road traffic has been a useful tool to evaluate certain infrastructure

This work was financially supported by the Singapore National Research Foundation under its Campus for Research Excellence And Technological Enterprise (CREATE) programme. Michael Lees acknowledges the support of the Russian Scientific Foundation, Project #14-21-00137.
Authors' addresses: Y. Xu and D. Zehe, TUMCREATE Ltd, 1 CREATE Way, Singapore 138602; emails: {yadong.xu, daniel.zehe}@tum-create.edu.sg; W. Cai, School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798; email: aswtcai@ntu.edu.sg; H. Aydt, Singapore-ETH Centre, Future Cities Laboratory, Singapore 138602; email: aydt@arch.ethz.ch; M.Lees, Instituut voor Informatica, University of Amsterdam, Science Park 904, 1098 XH Amsterdam; email: m.h.lees@uva.nl.

DOI: http://dx.doi.org/10.1145/0000000.0000000

or traffic control strategies. Road traffic can be modeled with different levels of detail: macroscopic [Lighthill and Whitham 1955; Richards 1956], mesoscopic [Paveri-Fontana 1975], microscopic [Gipps 1981], and nanoscopic (a.k.a., sub-microscopic) [Ni 2003]. It is a complex adaptive system whose behavior is influenced by the behaviors of constituent components. The modeling of the behaviors of individual components, i.e., driver-vehicle-units (DVUs), falls within the scope of microscopic and nanoscopic levels of detail. Both microscopic and nanoscopic simulations can be conducted in an agent-based manner. DVUs are agents that interact with the environment and aim to reach certain goals by performing some actions. Agent-based traffic simulation is useful in studying the impact of individual DVU behaviors on the road traffic, for instance, the impact of risky driving on the traffic, and the influence of adopting electric vehicles (EVs) or automatic vehicles on the transportation system.

SEMSim Traffic is a nanoscopic traffic simulator that is able to capture this level of detail [Xu et al. 2012]. It is designed to study how different vehicle designs and different infrastructures may influence the transportation system when EVs are introduced at a large-scale in mega-cities. However, one of the challenges to conduct a simulation at a large-scale (e.g., the whole city) is the issue of high computational resource requirements. To make large-scale agent-based traffic simulations computationally feasible, parallel computing techniques should be employed. There are many critical considerations in developing a parallel simulation, for example, time synchronization [Fujimoto 2000], and partitioning and load balancing [Deelman and Szymanski 1998]. The problem of time synchronization is addressed in this study. The focus is to reduce the overhead of time synchronization in agent-based traffic simulation.

A common way to parallelize agent-based traffic simulation is to decompose the road network into multiple spatial subregions (partitions). The agents in a subregion are executed by a Logical Process (LP) which is usually assigned to a physical processing unit. An agent (i.e., a DVU) interacts with other agents that are spatially situated in its proximity. It subscribes to certain state variables of those agents. If any of the subscribed agents is executed by another LP, the subscription of the state variables incurs data dependencies between the LPs. These agents are usually near the boundaries of the spatial subregions. To maintain the correctness of the simulation, when an LP has data dependencies with other LPs, it can only progress the simulation if the data required are received from relevant LPs. *Synchronization* between LPs is performed to exchange data. In a distributed memory environment, synchronization is usually achieved by messages passing among LPs.

Synchronization in agent-based traffic simulation is generally achieved using global barriers [Nagel and Rickert 2001; Suzumura and Kanezashi 2012]. At a synchronization barrier, all LPs temporarily pauses the execution of agent models. Then all the subscribed states are exchanged among LPs. After the exchange, all LPs resume execution at the same time. Agent models are executed at discrete time intervals which usually have a fixed length that is referred to as time-step or update interval. Global barriers can be deployed at the end of update intervals. This is equivalent to a *synchronous* and *conservative* synchronization protocol in parallel and distributed discrete-event simulation. A conservative synchronization protocol does not allow violation of data dependencies [Nicol 1996]. The limitation is that all LPs have to wait at global barriers despite that some LPs do not have dependencies at certain barriers. This method does not fully exploit the parallelism in the simulation. Another type of conservative synchronization protocol allows LPs to fulfill data dependency requirements *asynchronously* [Nicol 1996; Namekawa et al. 1999]. LPs synchronize with each other in a peer-to-peer manner. The frequencies of synchronization can be different for different LP pairs. A critical element of an asynchronous protocol is the *lookahead*. Lookahead of an LP towards another LP at a certain simulation time is a time interval

in the simulated future within which the LP will not have data dependencies with the other LP. The larger the lookahead values are, the less the synchronization operations are required. However, due to the frequent interactions of agents, agent-based traffic simulation usually has small lookahead. Using an asynchronous protocol alone does not solve the issue of high synchronization overhead in agent-based traffic simulation.

In this work, the aim is to develop heuristics to increase lookahead of LPs to reduce synchronization overhead in agent-based traffic simulation. A simulation is an abstraction of the real system and there is always uncertainty in the processing of modeling and simulation [Helton 1997; Oberkampf et al. 2002]. This fact applies to traffic simulation as well. For example, there is stochastic uncertainty since the traffic at a location in the real world varies from day to day; and there is model uncertainty in the models describing the movement of vehicles. Usually, a traffic simulation is run many times in order to obtain statistical results.

The uncertainties are exploited to reduce synchronization overhead. In our earlier work presented in [Xu et al. 2015], we developed a relaxed synchronization strategy with a heuristic that makes use of uncertainty of the simulation. Relaxation is achieved by skipping some synchronization operations. A dead-reckoning function is utilized to adjust the agent states which should have been updated by synchronization. This paper further extends our earlier work. We propose an alternative approach to relax synchronization with the same philosophy: uncertainty in agent-based traffic simulation allows a certain amount of violation of data dependencies between LPs in the parallel simulation. The new approach reduces synchronization by increasing the update interval of some models of those agents that incur data dependencies (e.g., the agents at boundaries of partitions). Increasing the size of update intervals reduces the temporal resolution of the models. In different traffic conditions, the influence of relaxation can be different. The method should ensure that the simulation results of the parallel simulation are statistically equivalent to those of the sequential simulation.

Our contribution to the literature is the two heuristics that improve the efficiency of synchronization by relaxing it without affecting simulation results statistically (Section 4). The heuristics can be potentially used to make large-scale traffic simulation run at or faster than real-time. This is particularly important when agent-based traffic simulation is used as a forecasting tool in time-critical decision making situations. The challenge is to determine how much relaxation can be introduced without distorting results of the simulation.

The remainder of the paper is organized as follows: Section 2 describes the related work. Synchronization methods in parallel microscopic traffic simulations and work on making use of uncertainties of simulation to improve synchronization in discrete-event simulations are presented. Section 3 introduces some background information: models used in the simulation, partitioning of the simulation, and the synchronization protocol used in the parallel simulation. Section 4 describes the two heuristics of relaxing synchronization, starting with a discussion on uncertainties in traffic simulation. Section 5 presents experiment results and analysis of the results. Section 6 provides a summary of this work and recommendations for future work.

## 2. RELATED WORK

The synchronization issue of parallel discrete-event simulations has been extensively studied in the literature [Fujimoto 2000; Nicol 1996]. Nevertheless, developing synchronization protocols usually involves tuning the protocols to the models used in particular simulations. For parallel agent-based traffic simulations, there have been many approaches to conduct synchronization. Employing global barriers is the most frequently used approach due to its simplicity. LPs are blocked from executing agent models at the end of update intervals, then they exchange messages with relevant

LPs and proceed to the next update interval simultaneously [Nagel and Rickert 2001; Suzumura and Kanezashi 2012]. For the simulations parallelized by multi-threading, all threads access the shared memory directly at barriers instead of communicating with each other by message passing [Barceló et al. 1998]. This simplicity frequently comes at the cost of LPs waiting at global barriers. It does not fully exploit the parallelism of the simulation. The work in [Namekawa et al. 1999] used a conservative time window synchronization. It is a similar approach to our mutual appointment protocol introduced in [Xu et al. 2015] which allows LPs to communicate asynchronously (which is also briefly described in Section 3.3). LPs analyze their event lists and determine lower bounds of simulation time for neighboring LPs in which they do not affect the neighboring LPs. LPs collect the time bounds from *all* neighboring LPs and take the minimum values as their time windows. LPs execute local events within the time windows, then they synchronize again. This is inefficient since LPs still need to communicate with all neighbors during synchronization. Communicating to all neighbors is unnecessary.

Another attempt is to use an optimistic approach [Yoginath and Perumalla 2008; Hanai et al. 2015]. LPs are allowed to progress over the synchronization point and violations of data dependencies are examined. If there is a violation, the simulation is rolled-back to the point before the violation happens. An optimistic approach may be able to explore more parallelism compared to a conservative approach. However, the disadvantage of the optimistic approach is the overhead of state saving and performing roll-back operations. Some optimistic simulation frameworks use reverse computation to avoid state saving overhead [Carothers et al. 2000]. Our work aims to improve the efficiency of conservative synchronization methods in agent-based traffic simulation. Particularly, we focus on improving lookahead by exploring uncertainty in traffic simulation.

Utilizing uncertainty to improve the performance of parallel and distributed simulation has also been studied. An approximate time causal order was proposed in [Fujimoto 1999] to increase the concurrency of events. It relaxes the conventional strict time stamp order of events. Events can be executed in an approximate order instead of exact time order. More events can be executed concurrently. It takes advantage of the fact that temporal uncertainty of events always exists in simulation. The approach was later studied with a queuing model in [Loper and Fujimoto 2004]. Another similar work is reported in [Nicol 2012]. The simulation application analyzes the delays that packets experience when they pass through switches in a switch network. Certain approximation of individual latencies of packets is accepted provided that the average latency is preserved. Allowing such approximation improves the performance of the simulation significantly. Our relaxed synchronization strategy follows the same direction. However, the way to achieve relaxation depends on the characteristics of models in the simulation. The behaviors of traffic models must be studied for an effective relaxation synchronization method for agent-based traffic simulation.

There is also work on trading the accuracy of simulation for performance. In [Park and Fishwick 2011], events are clustered into intervals and synchronization is only performed at the end of the intervals. It is intended to organize synchronization operations into a synchronous manner which is more suitable for the GPU architecture. Another study is presented in [Rihawi et al. 2013] which studies the effect of relaxing the synchronization of distributed agent-based simulations. However, it was not investigated how to maintain the correctness of a distributed simulation using relaxed synchronization.

## 3. PARALLEL AGENT-BASED TRAFFIC SIMULATION

### 3.1. Models and Model Execution

The simulation space of an agent-based traffic simulation is a road network which is a *spatial network*. The spatial network consists of links and nodes. Links represent roads, and they are the containers of agents. A link has one or more lanes. Nodes do not represent any physical entities. They contain the connectivity information of links and lanes. There are two nodes at the two ends of a link. The traffic flow on a link is unidirectional from the *start node* to the *end node*. A small road network is illustrated in Figure 1. For simplicity, lanes are not shown. In the figure, $node_2$ and $node_3$ are the start node and end node of $link_2$ respectively.

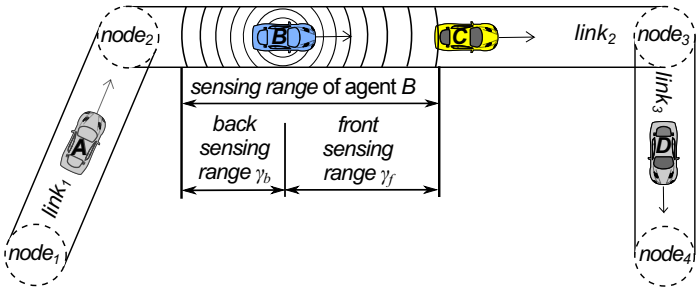


Fig. 1. Agents with front sensing ranges $\gamma_f$ and back sensing ranges $\gamma_b$ in a road network. Agent $C$ is in the sensing range of agent $B$; thus, agent $B$ subscribes to the agent-based state variables of agent $C$.

The *agent* in the simulation contains driver behavior models and vehicle component models. Examples of driver behavior models are the acceleration model and the lane-changing model, and examples of vehicle component models are the motor model and the battery model for EVs. An agent has a *state* at a certain instant of the simulation time. The state contains multiple state variables. *Agent-based state variables* are visible to other agents, e.g., velocity and position. *Component-based state variables* are not visible to other agents, e.g., a state-of-charge state variable in the battery model. An agent has a *sensing range* which is the area around the agent within which the states of other agents affect the agent's behavior. An agent subscribes to certain state variables of other agents inside its sensing range, in order to perform certain actions. An illustration is shown in Figure 1.

The state of an agent changes as the simulation progresses with the execution of timestamped *events* which contain certain update functions. Events are scheduled by driver behavior models and vehicle component models [Xu et al. 2012]. We assume that agent-based state variables that affect other agents are updated periodically. The period is referred to as an *update interval*. Denote it as $\delta$. In agent-based traffic simulation, the update interval $\delta$ can be interpreted as drivers' finite attention to the traffic: drivers only look at the traffic and instantaneously adapt the acceleration to the situation with a resolution of $\delta$ [Kesting and Treiber 2008]. A detailed description of the driver behavior models used in this work is presented in the appendix.

### 3.2. Partitioning and Dependency of Logical Processes

Denote the whole simulation space (i.e., road network) as $G$, and agent population at simulation time $t$ as $A_t$. To parallelize the simulation, the simulation space is partitioned into $I$ disjoint spatial subregions, $G_i$ $(0 \leq i < I)$, where $G = \cup_{i=0}^{I-1} G_i$. An LP is responsible for executing events of the agents in one partition. We assume that LPs only have access to the agents in their local spaces. The subset of $A_t$ that resides in partition $G_i$ at the simulation time $t$ is denoted as $A_{i,t}$. By definition, $A_t = \cup_{i=0}^{I-1} A_{i,t}$.

The LP that executes events in partition $G_i$ is $LP_i$. The partitioning of the network is performed on links. The links that are cut are named *boundary links*. A boundary link is evenly divided between two partitions. For instance, in Figure 2a, $link_2$ is a boundary link. The left half belongs to $G_1$, and the right half belongs to $G_2$. Since the direction of traffic on $link_2$ is from $G_1$ to $G_2$, we say $link_2$ is an *outgoing boundary link* of $G_1$, and an *incoming boundary link* of $G_2$. $LP_1$ and $LP_2$ are *neighboring processes*.

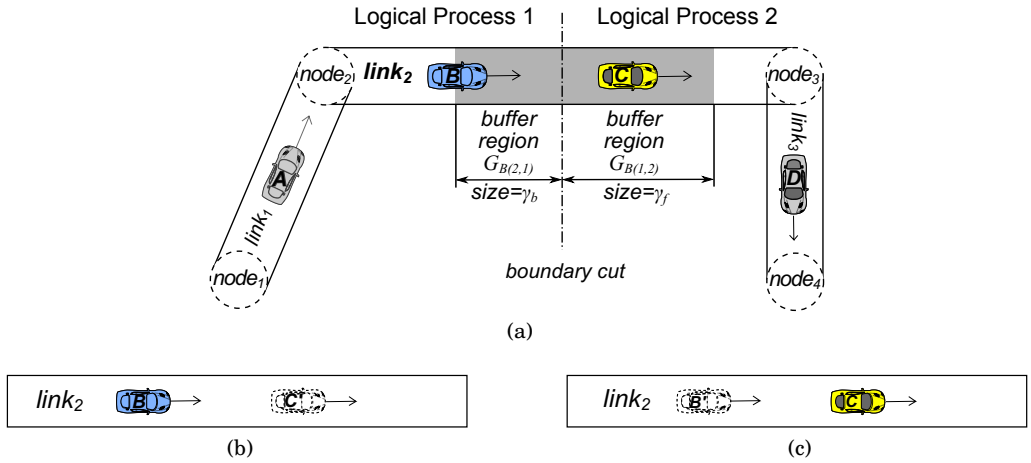


Fig. 2. Road network partitioning: (a) illustration of boundary cut and buffer regions; (b) view of $link_2$ from $LP_1$; and (c) view of $link_2$ from $LP_2$.

As mentioned earlier, agents subscribe to the state variables of other agents in their sensing ranges. The subscription incurs data read dependencies between LPs which is defined as follows:

*Definition* 3.1 (*data read dependency*). Data read dependency between $LP_i$ and $LP_j$ $(i \neq j)$ at simulation time $t$ is a condition in which there exists an agent in $LP_i$ that has subscription of state variables of an agent in $LP_j$ at simulation time $t$.

An example is shown in Figure 2a: the position of agent $C$ in $LP_2$ is inside the sensing range of agent $B$ in $LP_1$. Agent $B$ subscribes to the agent-based state variables of agent $C$. The state variables should be sent over to $LP_1$ by $LP_2$ and kept updated. In this case, there is a data read dependency between $LP_1$ and $LP_2$ at this simulation time. The subscribed state variables are referred to as *shared states*. The LPs that receive shared states use the shared states to create *proxy agents* which act as representatives of the *real agents* in the original LPs. There is a proxy agent $C'$ in $LP_1$ of real agent $C$ (Figure 2b), and a proxy agent $B'$ in $LP_2$ of real agent $B$ (Figure 2c). The set of proxy agents in $LP_i$ at the simulation time $t$ is denoted as $P_{i,t}$.

To update agents in an LP, states of agents in the neighboring LP near the partition boundary cut are required. To identify those agents whose states need to be shared, *buffer regions* are defined. They are the regions along the boundary cut of partitions with the size equal to the sensing range of agents. So, if an agent falls inside a buffer region, it is possible that the agent is in the sensing range of some agents in the neighboring LP. Denote the buffer region for partition $G_i$ inside partition $G_j$ as $G_{B(i,j)}$ $(0 \leq i < I, 0 \leq j < I, i \neq j)$. For example, in Figure 2a, $G_{B(2,1)}$ is the buffer region for $G_2$ in $G_1$, and $G_{B(1,2)}$ is the buffer region for $G_1$ in $G_2$. Given the traffic flow direction of the link, the size of $G_{B(2,1)}$ equals to the back sensing range of agents, and the size of $G_{B(1,2)}$ equals to the front sensing range of agents.

Agents move in the road network. When an agent moves beyond the boundary of one partition and enters another partition, *migration* of the agent from one LP to another LP happens. In Figure 2a, suppose agent $B$ in $LP_1$ continues moving on $link_2$ and its new position falls inside $G_2$ which is executed by $LP_2$, $LP_2$ would be responsible for executing future events of agent $B$. Agent $B$ is removed from $LP_1$ and created anew in $LP_2$. The information about the entire agent $B$ should be sent from $LP_1$ to $LP_2$ through a synchronization message. In this case, there is a *data write dependency* between $LP_1$ and $LP_2$ at time $t$. Data write dependency is defined as follows:

*Definition* 3.2 (*data write dependency*). Data write dependency between $LP_i$ and $LP_j$ $(i \neq j)$ at simulation time $t$ is a condition in which there exists an agent that was inside partition $G_i$ and changes its position to be inside partition $G_j$ at simulation time $t$.

Data dependencies between LPs are only affected by agent-based state variables. For instance, an event that updates the position, velocity, and acceleration of a vehicle affects surrounding vehicles; on the contrary, an event scheduled by a battery model of a vehicle that updates the state-of-charge of the battery in a vehicle does not affect surrounding vehicles. Hence, events that update component-based state variables are not considered in the following discussions.

### 3.3. Mutual Appointment Synchronization Protocol

The synchronization protocol used in this work is named mutual appointment (MA) protocol [Xu et al. 2015]. The idea of MA synchronization is that an LP communicates with other LPs by making *appointments* individually with them at certain *mutually agreed* simulation time. An appointment contains two tasks: exchange data required by data read and write dependencies and make the next appointment. Appointments are scheduled as synchronization events. The processing of an MA synchronization event in $LP_i$ with time stamp $t$ is shown in Algorithm 1.

**ALGORITHM 1:** MA synchronization event in $LP_i$ at simulation time $t$

**Data:**
$A_{i,t}$ set of local agents in $LP_i$ at $t$
$P_{i,t}$ set of proxy agents in $LP_i$ at $t$
$C_{i,t}$ set of LPs having appointments with $LP_i$ at $t$
$M_{i,j,t}$ set of agents migrating from $LP_i$ to $LP_j$ at $t$
$S_{i,j,t}$ set of shared states that should be sent by $LP_i$ to $LP_j$ at $t$
$l_{i,j,t}$ lookahead of $LP_i$ towards $LP_j$ at $t$
**Result:** time interval before the next appointment between $LP_i$ and $LP_j$, $\Delta t_{i,j,t}$

1 **foreach** $LP_j \in C_{i,t}$ **do**
  // 1. prepare the content of the message
2  determine $M_{i,j,t}$, $S_{i,j,t}$, and $l_{i,j,t}$;
  // 2. exchange messages
3  send $M_{i,j,t}$, $S_{i,j,t}$, and $l_{i,j,t}$ to $LP_j$;
4  receive $M_{j,i,t}$, $S_{j,i,t}$, and $l_{j,i,t}$ from $LP_j$;
  // 3. update the set of local agents
5  $A_{i,t} \leftarrow A_{i,t} \cup M_{j,i,t} \setminus M_{i,j,t}$;
  // 4. update the set of proxy agents
6  update the set of proxy agents $P_{i,t}$ with $S_{j,i,t}$;
  // 5. make a new appointment
7  $\Delta t_{i,j,t} \leftarrow \min(l_{i,j,t},\ l_{j,i,t})$;
8  make an appointment with $LP_j$ at time $t + \Delta t_{i,j,t}$;
9 **end**

Associated with each synchronization event, there is a set of LPs that currently have appointments with $LP_i$, denoted as $C_{i,t}$. An LP only synchronizes with its direct neighbors; therefore, $C_{i,t}$ only contains neighboring LPs. Note that $C_{i,t}$ includes all or only a subset of the neighboring LPs of $LP_i$. For each $LP_j$ in the set $C_{i,t}$, $LP_i$ performs the following five steps.

(1) The first step is to prepare the data for fulfilling data dependencies (migrating agents and shared states). The lookahead for the next appointment is calculated.
(2) In the second step, $LP_i$ sends all that information to $LP_j$. A single synchronization message is used. At the same time, it receives a message from $LP_j$ which contains the migrating agents $M_{j,i,t}$, shared states $S_{j,i,t}$, and lookahead $l_{j,i,t}$.
(3) The third step is to update the local agent set $A_{i,t}$. It is done by removing the agents in $M_{i,j,t}$, and adding the agents in $M_{j,i,t}$.
(4) The fourth step is to update the proxy agent set $P_{i,t}$ with the latest shared state $S_{j,i,t}$. Proxy agents outside of buffer regions are removed. Remaining proxy agents are updated, and new proxy agents are created if necessary.
(5) The last step is to make the next appointment. The next synchronization event with $LP_j$ will be scheduled at time $t + \Delta t_{i,j,t}$, where $\Delta t_{i,j,t}{=}min(l_{i,j,t}, l_{j,i,t})$.

The initial synchronization event for an LP is scheduled at the beginning of the simulation. Lookahead is calculated using the initial positions of agents. The targets of the initial synchronization event of an LP are *all* its neighboring LPs. Subsequent synchronization events are scheduled based on lookahead of LPs.

$l_{i,j,t}$ is the lookahead of $LP_i$ towards $LP_j$ at time $t$. It is the minimum time that $LP_i$ needs to update its shared states (i.e., agents' states in buffer region $G_{B(j,i)}$) to $LP_j$. If there are agents that are already inside $G_{B(j,i)}$ or ready to be migrated to $LP_j$, since agents' states are updated periodically with an interval $\delta$, $l_{i,j,t}$ is set to $\delta$. Otherwise, $l_{i,j,t}$ is set to the minimum time that any agent in $A_{i,t}$ or any agent to be created in $LP_i$ requires to travel to buffer region $G_{B(j,i)}$. So, the worst case of lookahead is that there are always agents inside $G_{B(j,i)}$ or ready for migration, in which case the lookahead always equals to one update interval.

## 4. RELAXATION ON THE SYNCHRONIZATION

### 4.1. Uncertainty in Traffic Modelling and Simulation

When modeling and simulation of a physical system is conducted, it always involves uncertainty and error [Helton 1997; Oberkampf et al. 2002]. This is also true for traffic simulation. Traffic simulation mimics the behavior of the real world traffic but can never duplicate the real world. There is always uncertainty in a traffic simulation [de Jong et al. 2006]. Uncertainty is categorized into two types in the literature according to their sources. The first category is *aleatory uncertainty*, a.k.a., stochastic uncertainty, irreducible uncertainty, inherent uncertainty, or variability. It is the inherent variation associated with the physical system, in this context, real road traffic. It is highly unlikely that the traffic conditions of two days at the same location are identical. The traffic always varies from day to day. Aleatory uncertainty in a computer simulation is represented as a number of streams of random variables drawn from specified probability distributions, for example, the distribution of the trip starting time of agents in traffic simulation. Aleatory uncertainty can be quantified by repeated simulation runs with different random variable streams. In scientific studies, simulations are usually run multiple times to get statistically meaningful results. The second category of uncertainty is *epistemic uncertainty*, a.k.a., reducible uncertainty, subjective uncertainty, or cognitive uncertainty. It results from a lack of knowledge about the simulated system and is further classified into input uncertainty and model

uncertainty. The input for traffic simulation, such as traffic demand, is usually an estimation from observed or forecast real world data. *Input uncertainty* arises from the process of estimation and forecast. Models used in the traffic simulation are the abstraction of the real world. *Model uncertainty* exists in both the model equation (e.g., certain assumption on the function form and omitted variables) and the values of model coefficients (usually estimated by calibration) [de Jong et al. 2006]. The models shown in the appendix are all based on certain assumptions. They are the major models in microscopic traffic simulation. Various car-following models exist based on different assumptions. A review of car-following models can be found in [Brackstone and McDonald 1999].

Traffic simulations have inherent stochastic uncertainties. The result of the simulations has certain variability and there is no single correct result of the simulation. This indicates that the parallel simulation needs not to produce the *exact* result as the sequential simulation; instead, a statistically equivalent result is sufficient. Thus, the synchronization of LPs can be relaxed to improve the performance of parallel traffic simulation as long as the relaxation preserves the result of the simulation statistically. This is similar to the ideas proposed by Fujimoto in [Fujimoto 1999] and Nicol in [Nicol 2012] respectively, where uncertainties of models are exploited to improve the synchronization of parallel and distributed simulations. Our relaxation strategies are presented in the remainder of this section.

### 4.2. Synchronization Skipping and Dead-reckoning

Consider a scenario where the traffic is jammed on a boundary link, and all agents are not moving, the states of the agents such as positions and velocities do not change. Since buffer regions on the link are always non-empty, synchronization is performed every update interval. This frequent synchronization is not necessary since skipping some synchronization operations here does not have much influence on the result of the simulation.

A convenient way of relaxing the synchronization is to simply increase lookahead values and skip some synchronization operations. It is imperative to analyze the influence of skipping synchronization operations. Some agents may not be migrated in time and some proxy agents may not be updated in time either. So, the consequence is that some agents use obsolete states of proxy agents as input in their models. The states of proxy agents can be estimated with a dead-reckoning function. However, discrepancy may exist between the states obtained by dead-reckoning and synchronization. An illustration of possible discrepancy is shown in Figure 3.

Agent $B$ is a local agent and agent $C'$ is a proxy agent in $LP_i$ at time $t$ (Figure 3a). The real agent $C$ of proxy agent $C'$ is in $LP_j$ $(i \neq j)$. As the simulation progresses to time $t+\delta$, states of agent $B$ and $C$ are calculated by $LP_i$ and $LP_j$ respectively. The proxy agent $C'$ should be updated by a synchronization operation with the shared states of agent $C$. When synchronization is not performed, $C'$ is updated by dead-reckoning. The state of $C'$ may be different due to the difference between the update function and the dead-reckoning function. Figures 3b and 3c show a case where there is discrepancy of positions between the dead-reckoned $C'$ and the updated $C'$ by synchronization. Figure 3d shows another case where the real agent $C$ has performed lane-change. The lanes of the dead-reckoned $C'$ and the updated $C'$ by synchronization are different. The discrepancy of states of proxy agent $C'$ may or may not lead to a discrepancy on agent $B$ in the simulated future depending on their relative position, relative speed, and the sensitivity of the driver's behavior models. If there is a discrepancy on agent $B$, the discrepancy may affect the results of the simulation.

It is not difficult to see that for a large-scale simulation, the amount of relaxation on synchronization is influenced by the dead-reckoning function as well as traffic con-

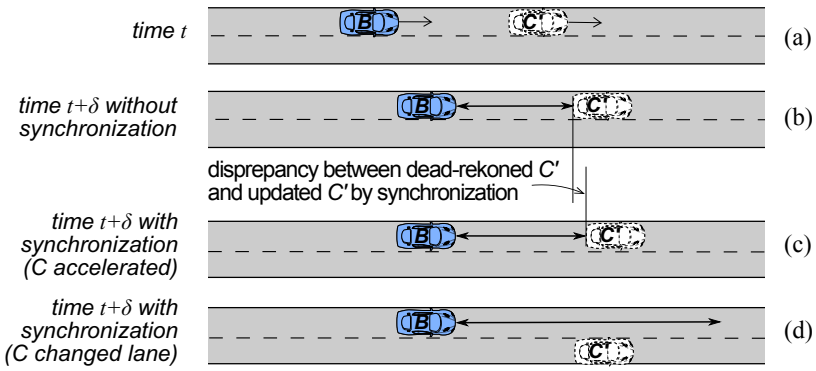


Fig. 3. Positions of agent $B$ and proxy agent $C'$ on a boundary link seen by $LP_i$. The real agent $C$ of proxy agent $C'$ is in $LP_j$ $(i \neq j)$. (a) Agents are at time $t$. (b) Synchronization is skipped at time $t+\delta$. The position of proxy agent $C'$ is calculated by dead-reckoning. (c) Synchronization is performed normally at time $t+\delta$. Proxy agent $C'$ is updated by synchronization. Agent $C$ accelerated in this case. (d) Synchronization is performed normally at time $t+\delta$. Agent $C$ changed its lane in this case.

dition on the boundary links. A simple dead-reckoning function can just assume that agents move at a constant speed until the next synchronization. It keeps the velocity of the proxy agents constant and updates the positions accordingly. In this case, the discrepancy is correlated with how much the velocities of the real agents change. Lookahead can be increased in the traffic conditions where the discrepancy is potentially insignificant.

Traffic condition is typically characterized by traffic density, speed, and flow. *Density* is the number of vehicles per unit length of a roadway. *Speed* is the average distance that vehicles travel per unit time. *Flow* is the number of vehicles passing a reference point per unit time. Flow is the product of speed and density. *Stability* of traffic describes how traffic is affected by perturbations (e.g., a sudden drop of speed of a vehicle). A traffic is unstable if it has a tendency to oscillations of speed, traffic waves, and stop-and-go traffic [Treiber and Kesting 2013c]. A *stability diagram* which plots the different stabilities of traffic under different traffic density ranges is shown in Figure 4 [Helbing et al. 2009]. Traffic density is denoted as $\rho$. In the entire density range $[0,\ \rho_{max}]$, there are four thresholds $\rho_{c1}$, $\rho_{c2}$, $\rho_{c_3}$, and $\rho_{c_4}$ $(0 < \rho_{c1} \leq \rho_{c2} \leq \rho_{c_3} \leq \rho_{c_4} \leq \rho_{max})$. They are the densities where traffic transitions between different stability states. In the stable range ($\rho < \rho_{c1}$ or $\rho \geq \rho_{c4}$), perturbations in traffic remain small and eventually dissolve. In the metastable range ($\rho_{c1} < \rho < \rho_{c2}$ or $\rho_{c3} < \rho < \rho_{c4}$), sufficiently small initial perturbations dissolve while larger perturbations develop into persistent traffic waves. In the linear unstable range ($\rho_{c2} < \rho < \rho_{c3}$), arbitrarily small perturbations will grow through time. A stability state may not exist for certain traffic models or parameter specifications. For example, under some configurations, a traffic may not transition from an unstable state to a stable state as the traffic density increases ($\rho_{c3}$=$\rho_{c4}$=$\rho_{max}$).

We consider skipping synchronization as a source of perturbation to traffic since it exerts certain disturbance on the normal movement of vehicles. Synchronization operations can be skipped more under stable traffic and less in unstable traffic. We notice from Figure 4 that when traffic flow is very low, density is either less than $\rho_{c1}$ or greater than $\rho_{c4}$. Therefore, the conceptual basis for our first heuristic is that lookahead can be larger when flow is lower (more stable traffic) and lookahead should be smaller when flow is higher (more unstable traffic).

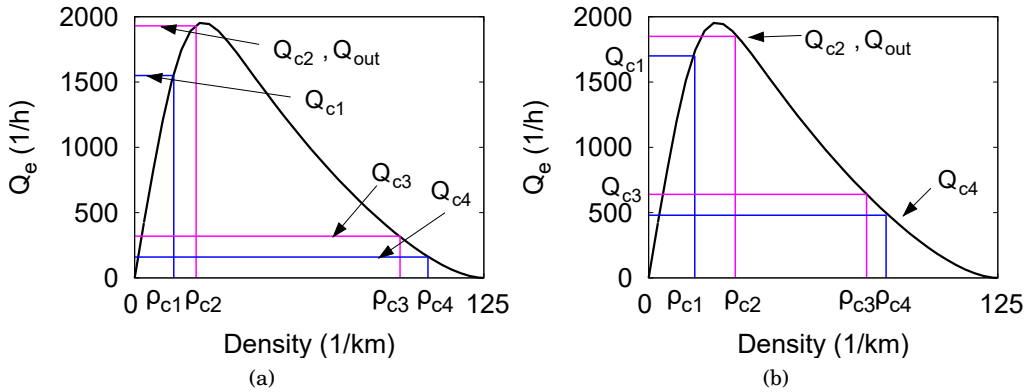


Fig. 4. Illustration of stable, linearly unstable, and metastable density regimes within flow-density diagrams $Q_e(\rho)$ [Helbing et al. 2009]. Traffic is stable for density $\rho < \rho_{c1}$ and $\rho \geq \rho_{c4}$; metastable in the intervals $\rho_{c1} < \rho < \rho_{c2}$ and $\rho_{c3} < \rho < \rho_{c4}$; and linearly unstable for $\rho_{c2} < \rho < \rho_{c3}$. The relative position of the thresholds with respect to the density at maximum flow $Q_{max}$ (denoted as $\rho_{Q_{max}}$) influences the possible types of congested traffic patterns. (a) shows the situation for $\rho_{c2} < \rho_{Q_{max}}$, (b) shows the situation for $\rho_{c2} > \rho_{Q_{max}}$. The traffic flows at densities $\rho_{c1}$, $\rho_{c2}$, $\rho_{c3}$, and $\rho_{c4}$ are $Q_{c1}$, $Q_{c2}$, $Q_{c3}$, and $Q_{c4}$ respectively. $Q_{out}$ is the out flow of traffic.

A simplistic but effective heuristic for relaxation is proposed. It contains only one input parameter. We first define a time window $window_{l,t}$ of a link $l$ as the longest time period at time $t$ in which synchronization can be skipped without affecting the traffic flow of link $l$. Here, $l$ is an outgoing boundary link from partition $G_i$ to $G_j$. The window is calculated using

$$window_{l,t} = \alpha \cdot \frac{1}{Q_{l,t}} \tag{1}$$

where $\alpha$ is a sensitivity factor, and $Q_{l,t}$ is the traffic flow on link $l$ at time $t$. When $Q_{l,t} = 0$, $window_{l,t} = maxw_l$, where $maxw_l$ is a maximum window. In fact, the physical meaning of $\frac{1}{Q_{l,t}}$ is the *time headway* which is the time difference between two consecutive vehicles passing a reference point on the road. Thus, this equation can also be interpreted as the time window on a link is proportional to the time headway on the link. The lookahead between two LPs should be the minimum of the time windows of all boundary links between them. Since the appointments are negotiated by both of the synchronizing LPs, it is sufficient to consider only outgoing boundary links of the partition for an LP. The lookahead from $LP_i$ to $LP_j$ at time $t$ is

$$l_{i,j,t} = \min_{l \in O_{i,j}} (window_{l,t}) \tag{2}$$

where $O_{i,j}$ is the set of outgoing boundary links from $G_i$ to $G_j$. The sensitivity factor $\alpha$ controls the amount of relaxation introduced. The optimal value of $\alpha$ is the one that maximizes lookahead and does not distort the statistical result of the simulation (e.g., traffic flow of roads, or trip durations of agents). The value of $\alpha$ is influenced by the models used in the traffic simulation, the road network topology, and partitioning. When a large-scale road network is simulated, discrepancy that occurs on boundary links may propagate to the upstream and downstream directions of the traffic. Therefore, it is not straightforward to obtain the optimal $\alpha$ value. An appropriate $\alpha$ value can be obtained by conducting trial experiments if this heuristic is used in practical applications.

The lookahead determination algorithm for this relaxation method is shown in Algorithm 2. The algorithm begins with checking the local agent population: if the local agent set is empty, the lookahead is set to the minimum value of maximum windows directly; otherwise, the outgoing boundary links are checked one by one and the minimum value of time windows is taken as the lookahead. If the lookahead is less than one update interval, it will be set to one update interval. It is evident that the $l_{i,j,t}$ is limited by the link with the highest flow. The maximum window of $l$ is calculated using the minimum traveling time of $l$ (length divided by speed limit).

**ALGORITHM 2: Lookahead of synchronisation relaxed with skipping and dead-reckoning**

**Data:**
$A_{i,t}$ set of agents in $LP_i$ at time $t$
$O_{i,j}$ set of outgoing boundary links from $G_i$ to $G_j$ at time $t$
$Q_{l,t}$ traffic flow on link $l$ at time $t$, $l \in O_{i,j}$
$maxw_l$ maximum window of link $l$, $l \in O_{i,j}$
$\delta$ an update interval
**Result:** lookahead from $LP_i$ towards $LP_j$ at time $t$, $l_{i,j,t}$

Initialize $l_{i,j,t} \leftarrow maximum\ double$;
**if** $A_{i,t} = \emptyset$ **then**
  $l_{i,j,t} \leftarrow \min_{l \in O_{i,j}} \left(maxw_l\right)$;
**else**
  **foreach** $l \in O_{i,j}$ **do**
    **if** $Q_{l,t} = 0$ **then**
      $l_{i,j,t} \leftarrow \min\left(l_{i,j,t},\ maxw_l\right)$;
    **else**
      $l_{i,j,t} \leftarrow \min\left(l_{i,j,t},\ \alpha \cdot \frac{1}{Q_{l,t}}\right)$;
    **end**
  **end**
**end**
$l_{i,j,t} \leftarrow \max(l_{i,j,t},\ \delta)$ ;

### 4.3. Reducing Temporal Resolution of Agent-based Models

The necessity of synchronization is originated from the update of agent-based states near boundaries of partitions. If the update interval of those agent-based states is increased, synchronization will be required less frequently. This inspired another way of relaxing synchronization. A heuristic that relaxes synchronization by increasing the update interval of agent-based models is introduced in this section. Increasing update interval reduces temporal resolutions of agent-based models, which may alter simulation results. The challenge is not to alter simulation results statistically.

Acceleration, lane-changing, and gap-acceptance models are the common behaviors that agents possess. Acceleration models describe the longitudinal movement of agents. Agents update their accelerations every update interval and the acceleration is kept constant within an update interval. The positions and velocities of agents are calculated using numerical integration of the acceleration through time [Treiber and Kesting 2013a]. Lane-changing and gap-acceptance models describe the discrete decisions of an agent whether to change its lane and enter another lane [Treiber and Kesting 2013b]. At time $t$, agents scan the surrounding environment and calculate their states for time $t + \delta$. The update interval represents a driver's finite attention to

traffic, that is, how fast an agent is able to adapt to the surrounding traffic condition [Kesting and Treiber 2008]. Update interval varies in different studies of agent-based traffic simulation. It ranges from a fraction of a second to a few seconds [Gipps 1981; Park and Qi 2005; Kesting and Treiber 2008; Basak et al. 2013]. It is also reported that lengthening the update interval within a certain range does not significantly influence the stability of traffic [Kesting and Treiber 2008].

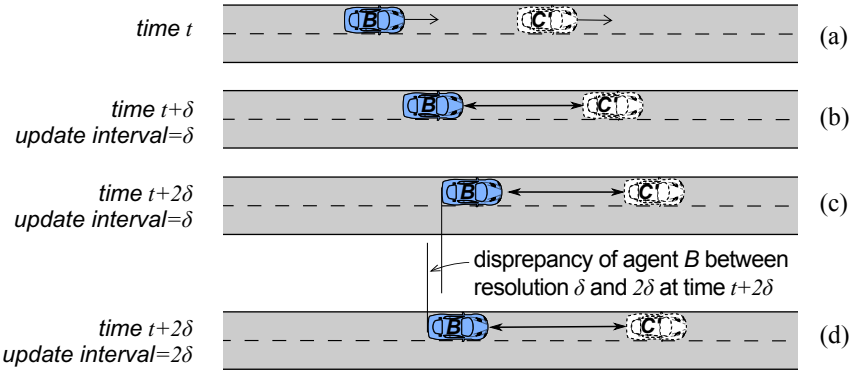


Fig. 5. Positions of agent $B$ and proxy agent $C'$ on a boundary link seen by $LP_i$. (a) Agents are at time $t$. (b) Agents are at time $t+\delta$, updated from $t$ with update interval $\delta$. (c) Agents are at time $t+2\delta$, updated from $t+\delta$ with update interval $\delta$. (d) Agent are at time $t+2\delta$, updated from $t$ with update interval $2\delta$.

Increasing update interval reduces the frequency of agents making decisions on their movement. An illustration is given in Figure 5. The positions of agent $B$ in two update intervals from time $t$ to $t+2\delta$ are shown in Figures 5a, 5b, and 5c. Agent $B$ and proxy agent $C'$ are in $LP_i$. Real agent $C$ of the proxy agent $C'$ is in $LP_j$ $(i \neq j)$. When the update interval equals to $\delta$, agent $B$ makes driving decisions twice. The accelerations it exerts from $t$ to $t+\delta$ and from $t+\delta$ to $t+2\delta$ are $\dot{v}_B(t)$ and $\dot{v}_B(t+\delta)$ respectively. In comparison, Figure 5d is evolved from Figure 5a with the update interval equal to $2\delta$. Agent $B$ only makes driving decisions once. It has an acceleration $\dot{v}_B(t)$ from $t$ to $t+2\delta$ in Figure 5d. If $\dot{v}_B(t) \neq \dot{v}_B(t+\delta)$, there will be a discrepancy between the position of agent $B$ in the two cases. Note that state of agent $B$ will be used to update its proxy in $LP_j$. So, $LP_j$ will see different positions of agent $B$ under the two cases.

To minimize discrepancy, the number of agents simulated with reduced temporal resolution should be as small as possible. We know that an agent affects other LPs when its position is inside a buffer region. Therefore, only the agents that are inside buffer regions are required to be simulated with a relaxed update interval (i.e., reduced temporal resolution). The buffer regions between $LP_i$ and $LP_j$ are $G_{B(i,j)}$ and $G_{B(j,i)}$. Denote the relaxed update interval as $\delta_{i,j,t}$ for $G_{B(i,j)}$ and $G_{B(j,i)}$ at time $t$, where $\delta_{i,j,t}{=}k \cdot \delta$ $(k \in \mathbb{Z}, k > 1)$. Note that $\delta_{i,j,t} = \delta_{j,i,t}$. The agents inside $G_{B(i,j)}$ and $G_{B(j,i)}$ are simulated with update interval $\delta_{i,j,t}$ at time $t$. Other agents outside of buffer regions are simulated with the original update interval $\delta$. Since agents move in the road network, when they move into $G_{B(i,j)}$ and $G_{B(j,i)}$, their update intervals are increased to $\delta_{i,j,t}$. When they exit the buffer regions, their update intervals are reverted to the original value, $\delta$. In other words, update intervals of agents will change when they move through a boundary link between partitions.

Recall that traffic density is segmented into five ranges in the stability diagram in Figure 4. We propose another heuristic that applies suitable relaxed update intervals according to which ranges the traffic densities of boundary links fall in. The amount of relaxation should be customized to stable, metastable, and unstable traffic states.

The heuristic utilizes four density thresholds $\rho_1$, $\rho_2$, $\rho_3$ and $\rho_4$. They correspond to the thresholds $\rho_{c1}$, $\rho_{c2}$, $\rho_{c3}$, and $\rho_{c4}$ in the stability diagram respectively. Denote the traffic density on boundary link $l$ at time $t$ as $\rho_{l,t}$, and the relaxed update interval allowed on boundary link $l$ as $\delta_{l,t}$, then $\delta_{l,t}$ is determined by the following rules:

$$\delta_{l,t} = \begin{cases} \delta_{gen}, & \text{if } \rho_{l,t} < \rho_1 \text{ or } \rho_{l,t} \geq \rho_4 \\ \delta_{mod}, & \text{if } \rho_1 \leq \rho_{l,t} < \rho_2 \text{ or } \rho_3 \leq \rho_{l,t} < \rho_4 \\ \delta, & \text{otherwise} \end{cases}$$

where $\delta_{gen} = k_{gen} \cdot \delta$ and $\delta_{mod} = k_{mod} \cdot \delta$ which satisfy $k_{gen} \in \mathbb{Z},\ k_{mod} \in \mathbb{Z},$ and $1 < k_{mod} < k_{gen}$. The rationale for the relaxation heuristic is as follows:

(1) When the traffic density of a link is below $\rho_1$, vehicles are traveling freely on the road. When the density is higher than $\rho_4$, the traffic is jammed. In both cases, relevant agent-based models can have a *generously* relaxed update interval $\delta_{gen}$.
(2) When the density of a link is between $\rho_1$ and $\rho_2$, or between $\rho_3$ and $\rho_4$, the traffic is in a metastable state. A moderate amount of relaxation can be applied. Relevant agent-based models have a *moderately* relaxed update interval $\delta_{mod}$.
(3) Otherwise, the traffic density is between $\rho_2$ and $\rho_3$. The traffic on the link is highly sensitive to perturbations. Drivers may tend to be more attentive to traffic, so relaxation should not be applied.

The values of $\rho_1$, $\rho_2$, $\rho_3$, and $\rho_4$ can be tuned to adjust the density ranges that allow different degrees of relaxation. The values of $\delta_{mod}$ and $\delta_{gen}$ can be tuned to adjust the update intervals for different degrees of relaxation. Note that the stable state or metastable state may not exist when density is high (i.e., $\rho_3{=}\rho_4{=}\rho_{max}$), which depends on the traffic models and parameter values used in the simulation. The relaxed update interval $\delta_{i,j,t}$ should take the minimum value of the allowed relaxed update intervals on all boundary links between $G_i$ and $G_j$. Thus

$$\delta_{i,j,t} = \min_{l \in O_{i,j} \cup O_{j,i}} (\delta_{l,t}) \tag{3}$$

The calculation of the relaxed update interval $\delta_{i,j,t}$ is shown in Algorithm 3. $\delta_{i,j,t}$ is initialized to $\delta_{gen}$. Then traffic densities of the outgoing boundary links from $G_i$ to $G_j$ are checked one after another against the four density thresholds. The minimum update interval is taken. After this, $LP_i$ sends $\delta_{i,j,t}$ to $LP_j$ and receives $\delta_{j,i,t}$ from $LP_j$. We need to ensure that relaxation should not be applied too much for any boundary link. Thus, the smaller one of $\delta_{i,j,t}$ and $\delta_{j,i,t}$ is taken as the resultant relaxed update interval. All agents inside the buffer regions $G_{B(i,j)}$ and $G_{B(j,i)}$ now have update interval $\delta_{i,j,t}$ (now $\delta_{i,j,t}{=}\delta_{j,i,t}$). It is evident that relaxation is limited by the boundary link that allows the least relaxation. The relaxed update intervals may be different for the buffer regions between different LP pairs. Algorithm 3 is run by every LP for every neighboring LP.

The relaxed update intervals need to be re-evaluated when the traffic densities of links have changed as the simulation progresses. One way to do so is to periodically check the traffic densities of the links in the road network, and trigger Algorithm 3 if a significant change is detected. In this way, the suitable amount of relaxation between LP pairs can be automatically adjusted.

The lookahead determination for this relaxation heuristic is similar to the case without relaxation described in Section 3.3. It is shown in Algorithm 4. If there is any agent inside $G_{j,i}$ or ready for migration from $LP_i$ to $LP_j$, the lookahead is set to $\delta_{i,j,t}$. With relaxed update intervals, $\delta_{i,j,t}$ will always be greater than or equal to $\delta$, as shown in

**ALGORITHM 3: Relaxed update interval $\delta_{i,j,t}$ by $LP_i$**

**Data:**

| | |
|---|---|
| $A_{i,t}$ | set of agents in $LP_i$ at time $t$ |
| $O_{i,j}$ | set of outgoing boundary links from $G_i$ to $G_j$ at time $t$ |
| $\rho_{l,t}$ | traffic density on link $l$ at time $t$, $l \in O_{i,j}$ |
| $\delta$ | original update interval |
| $\delta_{gen}$ | update interval when generous relaxation is applied |
| $\delta_{mod}$ | update interval when moderate relaxation is applied |
| $\rho_1$, $\rho_2$, $\rho_3$, $\rho_4$ | density thresholds separating different degree of relaxation, $\rho_1 \leq \rho_2 \leq \rho_3 \leq \rho_4$ |

**Result:** relaxed update interval in buffer regions $G_{B(i,j)}$ and $G_{B(j,i)}$, $\delta_{i,j,t}$

```
Initialize δ_{i,j,t} ← δ_{gen};
if A_{i,t} ≠ ∅ then
    foreach l ∈ O_{i,j} do
        if ρ_{l,t} < ρ_1 ∨ ρ_{l,t} ≥ ρ_4 then
            δ_{i,j,t} ← min(δ_{i,j,t}, δ_{gen});
        else if ρ_1 ≤ ρ_{l,t} < ρ_2 ∨ ρ_3 ≤ ρ_{l,t} < ρ_4 then
            δ_{i,j,t} ← min(δ_{i,j,t}, δ_{mod});
        else
            δ_{i,j,t} ← δ;
        end
    end
end
send δ_{i,j,t} to LP_j, and receive δ_{j,i,t} from LP_j ;
δ_{i,j,t} ← min(δ_{i,j,t}, δ_{j,i,t})
```

Algorithm 3. The effectiveness of this relaxation method has been investigated with experiments which are presented in the next section.

**ALGORITHM 4: Lookahead of synchronisation relaxed with reduced temporal resolution**

**Data:**

| | |
|---|---|
| $A_{i,t}$ | set of agents in $LP_i$ at time $t$ |
| $M_{i,j,t}$ | set of agents migrating from $LP_i$ to $LP_j$ |
| $S_{i,j,t}$ | set of shared states that should be sent by $LP_i$ to $LP_j$ |
| $\Delta t_{new_{i,j}}$ | minimum time for any new agent to travel to the buffer region $G_{B(j,i)}$ |
| $\Delta t_{exist_{i,j}}$ | minimum time for any agent in $A_{i,t}$ to travel from its current position to buffer region $G_{B(j,i)}$ |

**Result:** lookahead from $LP_i$ towards $LP_j$ at time $t$, $l_{i,j,t}$

```
Initialize l_{i,j,t} ← maximum double;
if A_{i,t} = ∅ then
    l_{i,j,t} ← Δt_{new_{i,j}};
else if M_{i,j,t} = ∅ ∧ S_{i,j,t} = ∅ then
    l_{i,j,t} ← min(Δt_{exist_{i,j}}, Δt_{new_{i,j}});
else
    l_{i,j,t} ← δ_{i,j,t};
end
l_{i,j,t} ← max(l_{i,j,t}, δ_{i,j,t}) ;
```

## 5. EXPERIMENTS AND DISCUSSION

### 5.1. Experiment Set-up

Experiments were conducted using a real-world road network. It is the road network of Singapore city consisting of approximately 80,000 links and 40,000 nodes in our representation. The trips of agents are derived from the data of the Household Interview and Travel Survey (HITS). No traffic lights are included in the simulation due to lack of traffic lights data. The lengths of vehicles are 3 meters. However, no vehicle component models are included. Agents have a front sensing range of 40 meters and a back sensing range of 20 meters. The traffic of 19 hours from 5 am to the midnight of the day is simulated. The update interval is 0.5 seconds. The parallel simulation is partitioned using a dynamic load-balancing algorithm described in [Xu et al. 2014] which uses METIS as the graph partitioning algorithm [Karypis and Kumar 1999]. A partitioning of Singapore road network into four partitions is shown in Figure 6. Different intensities of gray represent different partitions. Work load of LPs is checked every 10 minutes of simulation time. A repartitioning operation will be triggered if the imbalance exceeds the threshold of 500 agents. During repartitioning, the partitioning algorithm attempts to cut the links with lower traffic flow. This is to maximise the relaxation. The simulation is implemented using C++. Communication between LPs is realized using OpenMPI. The experiments were run on a cluster that is composed of three compute nodes each of which has the following hardware configurations: two $Octacore\,Intel(R)\,Xeon(R)$ CPUs with 2.60 GHz clock frequency (i.e., 16 physical processors), and 192 GB RAM. The compute nodes are connected via 56Gbps InfiniBand.

Fig. 6. Singapore road network partitioned into four partitions.

### 5.2. Results

*5.2.1. Amount of relaxation.* Denote MA synchronization relaxed using dead-reckoning as *MA-DR*, and MA synchronization relaxed using lower temporal resolutions as *MA-LR*. We first need to find suitable parameter values (i.e, $\alpha$ for MA-DR, $\rho_1$, $\rho_2$, $\rho_3$, $\rho_4$, $\delta_{mod}$, and $\delta_{gen}$ for MA-LR) for the relaxation heuristics while maintaining equivalence of results between simulations with and without relaxation. We experimented on a series of potential parameter values. The sequential simulation is run twelve times with different random seeds, as the referential group. Then the parallel simulation is run with different parameter values and different number of LPs. For each parameter value set and each number of LPs, the parallel simulation is run twelve times as one group using the same twelve random seeds as in the sequential group. Statistical tests are conducted to make sure that the parallel simulation with relaxed synchronization has equivalent results as the sequential simulation. The null hypothesis of the tests, $H_0$, is that the simulation results of the parallel simulation and the sequential sim-

ulation in terms of the average trip duration of agents are equivalent. Fail to reject the hypothesis means that there is insufficient evidence to prove that the results from both simulations are different.

The statistical test used is the *Bootstrap method*. Suppose the sequential group is $A=\{a_1, a_2, \ldots, a_{12}\}$, and a parallel group is $B=\{b_1, b_2, \ldots, b_{12}\}$. The first step of the Bootstrap method is resampling. In each resampling, two new sample groups $A'=\{a'_1, a'_2, \ldots, a'_{12}\}$ and $B'=\{b'_1, b'_2, \ldots, b'_{12}\}$, where $a'_i \in A$ and $b'_i \in B$, are obtained. Then the difference of the means of the two resampled groups is calculated by $d=\overline{A'}-\overline{B'}$. Resampling is performed 1000 times. Then we obtain 1000 values of the difference of means which follow a certain distribution. Using a 95% confidence interval, $H_0$ will be rejected if zero falls outside of the percentile range between 2.5% and 97.5%. Otherwise, $H_0$ could not be rejected. Tables I and II show the statistical test results. The values of $\delta_{mod}$ and $\delta_{gen}$ are fixed to 1.0s and 1.5s. They are reasonable values for the agent-based models and their parameter set-ups used in this experiment.

Table I. Statistical test of equivalence using different sensitivity factor values in MA-DR

| 8 LPs | 16 LPs | 32 LPs |
|---|---|---|
| 1.0 (rejected) | 1.0 (rejected) | 1.0 (rejected) |
| 0.4 (rejected) | 0.4 (rejected) | 0.4 (rejected) |
| **0.3 (not rejected)** | 0.3 (rejected) | 0.3 (rejected) |
| | **0.2 (not rejected)** | 0.2 (rejected) |
| | | 0.1 (not rejected) |
| | | **0.15 (not rejected)** |

Table II. Statistical test of equivalence using different traffic density threshold values in MA-LR

| Tested parameter groups (in the order of $\rho_1$, $\rho_2$, $\rho_3$, $\rho_4$)*, $\delta_{mod}$=1.0s, $\delta_{gen}$=1.5s | | |
|---|---|---|
| 8 LPs | 16 LPs | 32 LPs |
| 50, 100, 250, 333 (rejected) | 50, 100, 250, 333 (rejected) | 50, 100, 250, 333 (rejected) |
| 50, 100, 333, 333 (rejected) | 50, 100, 333, 333 (rejected) | 50, 100, 333, 333 (rejected) |
| **25, 100, 333, 333 (not rejected)** | 25, 100, 333, 333 (rejected) | 25, 100, 333, 333 (rejected) |
| | 25, 50, 333, 333 (rejected) | 25, 50, 333, 333 (rejected) |
| | **25, 30, 333, 333 (not rejected)** | 25, 30, 333, 333 (rejected) |
| | | **25, 25, 333, 333 (not rejected)** |

*the unit is $vehicle/km/lane$

In Tables I and II, the order of the parameter values that are experimented is from top down. Some estimated values are tried first, and if the statistical test is rejected, the values are decreased and tried again. According to Table I, the suitable sensitivity factors in MA-DR are 0.3, 0.2, and 0.15 for 8, 16, and 32 LPs, respectively. For MA-LR, the suitable parameter values are boldfaced in Table II. There, $\rho_3=\rho_4=\rho_{max}=333$ for all various numbers of LPs. We notice that when there are more partitions, the parameter values for both MA-DR and MA-LR are smaller. This is because the partitioning of the simulation also influences the effect of relaxation. More partitions lead to more total boundary links; thus, there will be less opportunities for synchronization relaxation. It is worth to notice that it is impractical to traverse all possible parameter values to look for the optimum for relaxation. Thus, the parameter values used are conservative estimates.

*5.2.2. Running time and speed-up.* Denote barrier synchronization as *barrier*, and MA synchronization without relaxation *MA*. We compare the running time and speed-up of the parallel simulation using different synchronization methods: barrier, MA, MA-DR, and MA-LR. The parallel simulation is run with various number of LPs: 8, 16, and

32. LPs are evenly distributed in the three compute nodes and each LP is bonded to one physical CPU core. The running time and speed-up of the parallel simulation are shown in Figure 7.

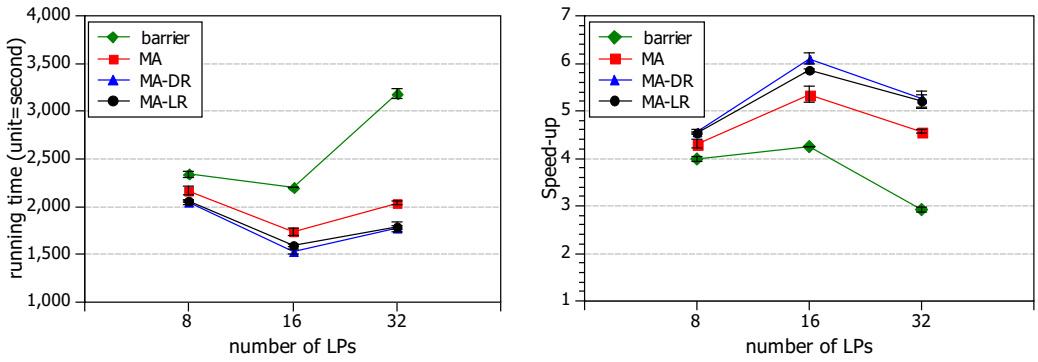


Fig. 7. Running time of the parallel simulation using different synchronization methods.

From Figure 7, we can observe that both MA-LR and MA-DR relaxation methods have reduced the running time. The least running time and greatest speed-up are achieved with 16 LPs for all the methods. There is no significant difference between the running time of MA-LR and MA-DR for 8 LPs and 32 LPs. The running time of MA-DR is slightly less than MA-LR for 16 LPs. The performance gain of the two heuristic is the result of the reduced communication between LPs. The number of synchronization messages exchanged during the simulation is shown in Figure 8.

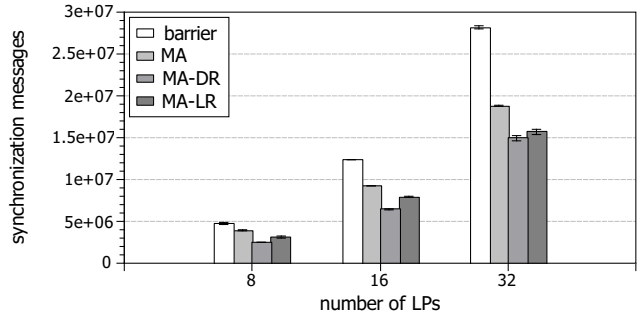


Fig. 8. Number of synchronization messages exchanged in the parallel simulation using different synchronization methods.

Both relaxation heuristics have reduced synchronization messages by a large percentage. Since the relaxation parameter values are conservative estimates, the speed-ups gained are also conservative measures. The optimum speed-up should be larger. A concrete conclusion cannot be drawn on which heuristic of the two has a better speed-up either. In theory, the second heuristic possesses more input parameters, and thus is more flexible in fine-tuning the relaxation.

A question may rise here is whether it is possible to simply skip synchronization operations or reduce temporal resolutions without considering traffic flow or density. We have conducted experiments using the same set-up of road network and agent trips as the previous experiments. Relaxation is achieved by synchronizing LPs every two update intervals with (a) skipping synchronization operations, and (b) doubling the up-

date interval of agents in buffer regions. In both scenarios, the simulation result (i.e., average trip duration of agents) is distorted. Based on this observation, considering traffic flow or density is necessary when synchronization is relaxed.

*5.2.3. Algorithm Overhead.* MA-DR does not incur notable overhead, since the traffic flow can be easily tracked during the simulation. Algorithm 3 which determines the suitable relaxed update interval is triggered periodically during the simulation. A re-evaluation is performed every 30 seconds of simulation time in the experiments. 30 seconds is a conservative estimation of the interval within which traffic densities of links may change. Thus with a 19 hours simulation, the re-evaluation is performed 2280 times. The overhead of running the algorithm is shown in Table III.

Table III. Total time spent on Algorithm 3 and the average time per invocation

| | 8 LPs | 16 LPs | 32 LPs |
|---|---|---|---|
| Total (second) | 9.8 | 10.1 | 15.6 |
| Average (second) | 0.0043 | 0.0044 | 0.0068 |

The overhead of the algorithm is not significant compared to the total running time of the simulation. Because the re-evaluation is performed every 30 seconds, if there were any emergency, such as vehicle breaking down, the resolution could be adapted to the situation in less than 30 seconds. The frequency of re-evaluations can be adjusted according to the dynamics of traffic flow and density. If it is known prior to the simulation that traffic flow and density will not vary much during the simulation, the re-evaluation can be less frequent.

## 6. CONCLUSION AND FUTURE WORK

Large-scale agent-based road traffic simulation is a useful tool for evaluating new infrastructure or control strategies of road traffic. One of the challenges in developing large-scale agent-based traffic simulation is the computational requirement. Parallel computing helps to conquer the challenge by utilizing more computational resources. However, existing synchronization methods of parallel agent-based traffic simulation are inefficient.

We proposed two heuristics to relax the synchronization of parallel agent-based road traffic simulation without altering simulation results statistically. The first relaxation heuristic ignores some data dependencies and skips synchronization operations. The lookahead at simulation run-time is determined using traffic flow of boundary links. The second relaxation heuristic reduces the temporal resolution of agent-based models for the agents at boundaries of partitions so as to reduce data dependencies. The amount of relaxation of the two heuristics is limited by the equivalence of simulation results. Experiments have shown that the two approaches are both able to increase lookahead of LPs and improve speed-up of the parallel simulation.

We suggest three directions of future work. Firstly, the second heuristic can be improved. When the temporal resolution of some models is reduced, input parameters of the models can be adjusted to minimize the difference of model outputs between different resolutions. Using this approach, lookahead may be further increased. Alternatively, a mesoscopic or macroscopic traffic model can replace the model with reduced temporal resolution. If the models with coarser resolution are calibrated, statistical tests may not be required to check simulation results. This makes the relaxation easier for practical use. Secondly, the heuristics can be tested for simulating emergency situations, for example, when there is a traffic accident or break-down. It is interesting to investigate whether the heuristics are responsive to sudden changes of traffic conditions in simulation. Thirdly, the relaxation methods can be made aware of conditions of the underlying system resources (e.g., CPU and memory usage, latency and available

bandwidth of interconnection network). The amount of relaxation can then be tuned to fit certain simulation execution environment.

## APPENDIX

This appendix describes the three driver behavior models used in this study.

### Acceleration Model

Intelligent Driver Model (IDM) model is a car-following model that describes the dynamics of the positions and velocities of single vehicles [Treiber et al. 2000]. It is a time-continuous car-following model. For a vehicle, the acceleration is assumed to be a continuous function of its velocity $v$, the gap $s$ between the vehicle and the vehicle ahead, and the velocity difference $\Delta v$ to the vehicle ahead:

$$\dot{v} = a\left[1 - \left(\frac{v}{v_0}\right)^4 - \left(\frac{s^*(v,\Delta v)}{s}\right)^2\right] \tag{4}$$

where $v_0$ is the desired velocity of the vehicle, and $a$ is the maximum acceleration of the vehicle. The expression consists of two terms: a free road acceleration term $\dot{v}_{free}(v) = a[1-(v/v_0)^4]$, and a braking deceleration term $\dot{v}_{brake}(s,v,\Delta v) = -a[s^*(v,\Delta v)/s]^2$. The deceleration term dominates when the vehicle is too close to the vehicle ahead. It depends on the ratio between the effective desired minimum gap and the actual gap $s$. The desired minimum gap is determined by:

$$s^*(v,\Delta v) = s_0 + vT + \frac{v\Delta v}{2\sqrt{ab}} \tag{5}$$

where $s_0$ is a safe distance to the vehicle ahead, $T$ is the minimum safe time gap, and $b$ is the comfortable deceleration of the vehicle ($b$ has a positive value). When there is no vehicle ahead in the same lane, the deceleration term will be excluded from the acceleration model. IDM is a collision-free model due to the deceleration term. When the city traffic is simulated, there are always roads with different speed limits connecting with each other. If it happens that the velocity of a vehicle is greater than the desired velocity which is usually smaller than or equal to the speed limit, the free acceleration is replaced by:

$$\dot{v}_{free}(v) = -b\left[1 - \left(\frac{v_0}{v}\right)^4\right] \tag{6}$$

The velocities of vehicles are obtained by numerical integration of the acceleration over time. The positions of vehicles are calculated with the equations of motion. There is approximation error in the integration while the continuous model IDM is integrated by discrete time-steps (update intervals). The acceleration is commonly assumed to be constant within an update interval. The velocity and position of a vehicle at the end of an update interval are $v(t+\delta) = v(t) + \dot{v}(t)\cdot\delta$ and $x(t+\delta) = x(t) + v(t)\delta + \frac{1}{2}\dot{v}(t)\delta^2$, where $v(t)$ and $x(t)$ are the velocity and position of the vehicle at time $t$, and $\delta$ is the update interval of the vehicle.

The size of update intervals should be as small as possible. However, a smaller update interval is computationally inefficient. The update interval is typically set between 0.1 seconds and 0.2 seconds [Kesting et al. 2008]. When the size of update interval is large (for example, 0.5 seconds or 1 second), the velocity at the end of an update interval can be greater than the speed-limit. The road network used in our experiment consists of a large number of links with various speed limits. Due to the large update interval we use (0.5 seconds), the acceleration output from IDM sometimes results in

the velocity of a vehicle greater than the speed limit or less than 0. Therefore, the output of IDM is monitored and $v(t+\delta)$ is bounded to be greater than or equal to 0 and less than or equal to the speed limits of roads. Collisions of vehicles may also occur under some traffic conditions. There is also a collision detection mechanism. If two vehicles are detected to be overlapping, the vehicle behind will be forced to come to a halt. Another modification to the original IDM model is the introduction of sensing range. Agents only take actions when neighboring vehicles are in their sensing ranges, which is not considered in the IDM model. This is necessary for controlling the data required for fulfilling data dependencies of LPs.

#### Lane-changing Model

The lane-changing model outputs a decision whether to stay in the current lane or change to an alternative lane. We develop a new lane-changing model which is shown in Figure 9. The lane-changing decision starts with checking whether an alternative lane exists. All the lanes that belong to the route of an agent are assigned weights. If there is a turn or a junction, the lanes that allow turning are assigned greater weights than those that do not allow turn. Therefore, agents should tend to change to lanes with higher weights. When multiple alternative lanes exist, the lane with the higher weight is chosen as the target lane. Similar to other lane-changing models, there are discretionary lane-changing and compulsory lane-changing. If the position of an agent is within a certain distance to a turn or a junction, it makes a compulsory lane-changing decision. The agent changes its lane if the target lane has a higher weight than the current lane and it is safe to perform lane-changing according to a gap-acceptance model (presented in the next subsection of this appendix). If the agent is outside the distance to a turn or a junction, it makes a discretionary lane-change decision. The decision is based on whether the agent is able to gain an acceleration benefit after a lane change. The weights of the lanes do not play a role here. The agent first checks whether there is a slower preceding vehicle blocking its way in the current lane. If there is none, the agents stay in the current lane. If there is one, the agent checks the target lane to see if it can get an acceleration benefit. If it can obtain a higher acceleration, it will check whether there is enough space for it to change without causing a collision. If so, the agent changes to the target lane. Otherwise, it remains in the current lane.

#### Gap-acceptance Model

The gap-acceptance model here describes the judgment of a driver whether the gap between a lead vehicle and lag vehicle in the target lane allows a safe lane change. It is a new model.

There are a lead gap and a lag gap between the subject vehicle and the lead and lag vehicles, respectively, which are shown in Figure 10. The lead gap and the lag gap should ensure that no collision happens if the subject vehicle changes its lane.

Denote the lead gap as $s_{lead}$, the lag gap as $s_{lag}$, the velocity of the lead vehicle as $v_{lead}$, the velocity of the lag vehicle as $v_{lag}$, and the velocity of the subject vehicle as $v$. To accept the gaps in the target lane, the following two conditions must be satisfied:

$$s_{lead} \geq s_0 + \frac{\delta}{2} \cdot (v + v_{max} - v_{lead}) \tag{7}$$

$$s_{lag} \geq s_0 + \frac{\delta}{2} \cdot (v_{lag} + v_{lag_{max}} - v) \tag{8}$$

where $v_{lag_{max}}$ and $v_{max}$ are the maximum velocities that the lag vehicle and the subject vehicle can reach at the end of the update interval, respectively. The first condition

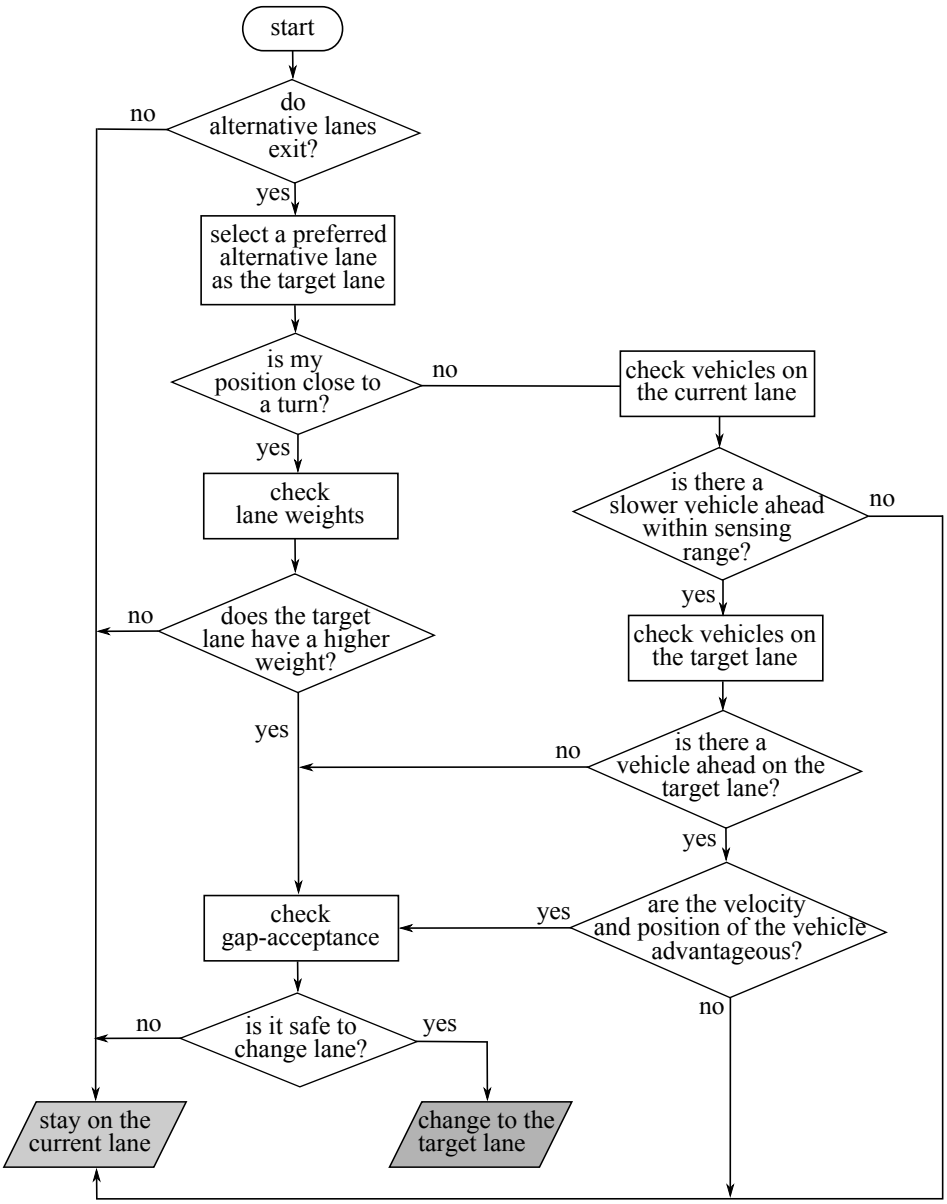


Fig. 9. Flow diagram of the lane-changing decision of agents.

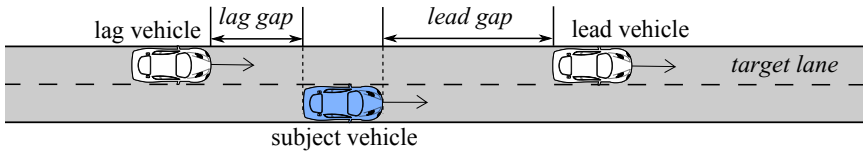


Fig. 10. Lead gap and lag gap in the target lane

depicts that the lead gap should be at least as big as the safe gap even when the lead vehicle comes to an emergency stop and the subject vehicle accelerates with the maximum acceleration. The second condition depicts that the lag gap should be at least as big as the safe gap even when the subject vehicle comes to an emergency stop and the lag vehicle accelerates with the maximum acceleration.

## REFERENCES


Jaume Barceló, Jaime L. Ferrer, and David Garcia. 1998. Microscopic traffic simulation for ATT systems analysis. a parallel computing version. *Contribution to the 25th Aniversary of CRT* (1998), 1–16.

Kakali Basak, Seth N. Hetu, Carlos Lima Azevedo, Harish Loganathan, Tomer Toledo, and Moshe Ben-Akiva. 2013. Modeling reaction time within a traffic simulation model. In *Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE, The Hague, Netherlands, 302–309. DOI:http://dx.doi.org/10.1109/ITSC.2013.6728249

Mark Brackstone and Mike McDonald. 1999. Car-following: a historical review. *Transportation Research Part F: Traffic Psychology and Behaviour* 2, 4 (dec 1999), 181–196. DOI:http://dx.doi.org/10.1016/S1369-8478(00)00005-X

Christopher D Carothers, David Bauer, and Shawn Pearce. 2000. ROSS: a High Performance Modular Time Warp System. In *Proceedings of the 14th Workshop on Parallel and Distributed Simulation*. IEEE, Bologne, Italy, 53–60. DOI:http://dx.doi.org/10.1109/PADS.2000.847144

Gerard de Jong, Andrew Daly, Marits Pieters, Stephen Miller, Ronald Plasmeijer, and Frank Hofman. 2006. Uncertainty in traffic forecasts: literature review and new results for The Netherlands. *Transportation* 34, 4 (Dec. 2006), 375–395. DOI:http://dx.doi.org/10.1007/s11116-006-9110-8

Eva Deelman and Boleslaw K. Szymanski. 1998. Dynamic load balancing in parallel discrete event simulation for spatially explicit problems. In *Proceedings of the 12th Workshop on Parallel and Distributed Simulation*. IEEE, Banff, AB, Canada, 46–53. DOI:http://dx.doi.org/10.1145/278009.278015

R. M. Fujimoto. 1999. Exploiting temporal uncertainty in parallel and distributed simulations. In *Proceedings of the 13th Workshop on Parallel and distributed simulation*. 46–53. DOI:http://dx.doi.org/10.1109/PADS.1999.766160

R. M. Fujimoto. 2000. *Parallel and distribution simulation systems*. Wiley Interscience, New York, NY, USA.

P. G. Gipps. 1981. A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological* 15, 2 (1981), 105–111. DOI:http://dx.doi.org/10.1016/0191-2615(81)90037-0

Masatoshi Hanai, Toyotaro Suzumura, Georgios Theodoropoulos, and Kalyan S. Perumalla. 2015. Exact-differential large-scale traffic simulation. In *Proceedings of the 3rd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. 271–280. DOI:http://dx.doi.org/10.1145/2769458.2769472

Dirk Helbing, Martin Treiber, Arne Kesting, and Martin Schönhof. 2009. Theoretical vs. empirical classification and prediction of congested traffic states. *The European Physical Journal B* 69, 4 (2009), 583–598. DOI:http://dx.doi.org/10.1140/epjb/e2009-00140-5

J. C. Helton. 1997. Uncertainty and sensitivity analysis in the presence of stochastic and subjective uncertainty. *Journal of Statistical Computation and Simulation* 57, 1-4 (April 1997), 3–76. DOI:http://dx.doi.org/10.1080/00949659708811803

George Karypis and Vipin Kumar. 1999. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing* 20, 1 (1999), 359–392. DOI:http://dx.doi.org/10.1137/S1064827595287997

Arne Kesting and Martin Treiber. 2008. How reaction time, update time, and adaptation time influence the stability of traffic flow. *Computer-Aided Civil and Infrastructure Engineering* 23, 2 (2008), 125–137. DOI:http://dx.doi.org/10.1111/j.1467-8667.2007.00529.x

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

M. J. Lighthill and G. B. Whitham. 1955. On kinematic waves. II. a theory of traffic flow on long crowded roads. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 229, 1178 (May 1955), 317–345. DOI:http://dx.doi.org/10.1098/rspa.1955.0089

M. L. Loper and R. M. Fujimoto. 2004. A case study in exploiting temporal uncertainty in parallel simulations. In *Proceedings of the 33rd International Conference on Parallel Processing*. IEEE, Montreal Quebec, Canada, 161–168. DOI:http://dx.doi.org/10.1109/ICPP.2004.1327916

Kai Nagel and Marcus Rickert. 2001. Parallel implementation of the TRANSIMS. *Parallel Comput.* 27 (2001), 1611–1639. DOI:http://dx.doi.org/10.3929/ethz-a-009900754

Mitsuhiro Namekawa, Akira Satoh, Hideki Mori, Kunio Yikai, and Toshio Nakanishi. 1999. Clock synchronization algorithm for parallel road-traffic simulation system in a wide area. *Mathematics and Computers in Simulation* 48, 4-6 (1999), 351–359. DOI:http://dx.doi.org/10.1016/S0378-4754(99)00015-4

Daiheng Ni. 2003. 2DSIM: a prototype of nanoscopic traffic simulation. In *Intelligent Vehicles Symposium, 2003. Proceedings. IEEE*. IEEE, 47–52. DOI:http://dx.doi.org/10.1109/IVS.2003.1212881

David M. Nicol. 1996. Principles of conservative parallel simulation. In *Proceedings of the 1996 Conference on Winter simulation*. DOI:http://dx.doi.org/10.1109/WSC.1996.873270

David M. Nicol. 2012. Exploiting uncertainty and error to accelerate simulations.. In *Keynote Addresses of International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2012)*. http://dblp.uni-trier.de/db/conf/simultech/simultech2012.html#Nicol12

William L. Oberkampf, Sharon M. DeLand, Brian M. Rutherford, Kathleen V. Diegert, and Kenneth F. Alvin. 2002. Error and uncertainty in modeling and simulation. *Reliability Engineering & System Safety* 75, 3 (2002), 333–357. DOI:http://dx.doi.org/10.1016/S0951-8320(01)00120-X

Byungkyu Park and Hongtu Qi. 2005. Development and evaluation of a procedure for the calibration of simulation models. *Transportation Research Record: Journal of the Transportation Research Board* 1934 (2005), 208–217. DOI:http://dx.doi.org/10.3141/1934-22

Hyungwook Park and Paul a. Fishwick. 2011. An analysis of queuing network simulation using GPU-based hardware acceleration. *ACM Transactions on Modeling and Computer Simulation* 21, 3 (2011), 1–22. DOI:http://dx.doi.org/10.1145/1921598.1921602

S. L. Paveri-Fontana. 1975. On Boltzmann-like treatments for traffic flow: a critical review of the basic model and an alternative proposal for dilute traffic analysis. *Transportation Research* 9, 4 (1975), 225–235. DOI:http://dx.doi.org/10.1016/0041-1647(75)90063-5

P. I. Richards. 1956. Shock waves on the highway. *Operations Research* 4, 1 (1956), 42–51. DOI:http://dx.doi.org/10.1287/opre.4.1.42

Omar Rihawi, Yann Secq, and Philippe Mathieu. 2013. Relaxing synchronization constraints in distributed agent-based simulations. *Jurnal Teknologi* 3 (2013), 65–76.

Toyotaro Suzumura and Hiroki Kanezashi. 2012. Highly scalable X10-based agent simulation platform and its application to large-scale traffic simulation. In *Proceedings of the 2012 IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time Applications*. Ieee, 243–250. DOI:http://dx.doi.org/10.1109/DS-RT.2012.44

Martin Treiber, Ansgar Hennecke, and Dirk Helbing. 2000. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E* 62, 2 (2000), 1805. DOI:http://dx.doi.org/10.1103/PhysRevE.62.1805

Martin Treiber and Arne Kesting. 2013a. Elementary car-following models. In *Traffic Flow Dynamics*. Springer Berlin Heidelberg, 257–301. DOI:http://dx.doi.org/10.1007/978-3-642-32460-4

Martin Treiber and Arne Kesting. 2013b. Lane-changing and other discrete-choice situations. In *Traffic Flow Dynamics*. Springer Berlin Heidelberg, 239–255. DOI:http://dx.doi.org/10.1007/978-3-642-32460-4

Martin Treiber and Arne Kesting. 2013c. Stability Analysis. In *Traffic Flow Dynamics*. Springer Berlin Heidelberg, 157–180. DOI:http://dx.doi.org/10.1007/978-3-642-32460-4

Yadong Xu, Heiko Aydt, and Michael Lees. 2012. SEMSim: a distributed architecture for multi-scale traffic simulation. In *ACM/IEEE/SCS 26th Workshop on Principles of Advanced and Distributed Simulation (PADS)*. 178–180. DOI:http://dx.doi.org/10.1109/PADS.2012.40

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 2014 Winter Simulation Conference*. 3483–3494. DOI:http://dx.doi.org/10.1109/WSC.2014.7020180

Yadong Xu, Wentong Cai, Heiko Aydt, Michael Lees, and Daniel Zehe. 2015. An asynchronous synchronization strategy for parallel large-scale agent-based traffic simulations. In *Proceedings of the 3rd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. ACM, 259–269. DOI:http://dx.doi.org/10.1145/2769458.2769461

Srikanth B. Yoginath and Kalyan S. Perumalla. 2008. Parallel vehicular traffic simulation using reverse computation-based optimistic execution. In *Proceedings of the 22nd Workshop on Principles of Advanced and Distributed Simulation (PADS)*. Roma, Italy, 33–42. DOI:http://dx.doi.org/10.1109/PADS.2008.14

Received December 2015; revised May 2016; accepted August 2016