An Asynchronous Synchronization Strategy for Parallel Large-scale Agent-based Traffic Simulations

Yadong Xu
School of Computer Engineering
Nanyang Technological University
Singapore 639798
xuya0006@ntu.edu.sg

Wentong Cai
School of Computer Engineering
Nanyang Technological University
Singapore 639798
aswtcai@ntu.edu.sg

Heiko Aydt
TUM CREATE Ltd.
CREATE Tower
1 Create Way
Singapore 138602
heiko.aydt@tum-create.edu.sg

Michael Lees
Informatics Institute
University of Amsterdam
Amsterdam 1098 XH, The Netherlands
m.h.lees@uva.nl

Daniel Zehe
TUM CREATE Ltd.
CREATE Tower 1 Create Way
Singapore 138602
daniel.zehe@tum-create.edu.sg

ABSTRACT

Large-scale agent-based traffic simulation is a promising tool to study the road traffic and help solving traffic problems, such as congestion and high emission in megacities. Such simulation requires high computational resource which triggers the need for parallel computing. The parallelization of agent-based traffic simulations is generally performed by decomposing the simulation space into spatial subregions. The agent models contained by each subregion are executed by Logical Processes (LPs). As the simulated system evolves over the simulation time in individual LPs, synchronization among LPs is required due to data dependencies. Existing work has used global barriers for synchronization which is a type of synchronous synchronization method. However, global barriers have very low efficiency due to the waiting of processes at barriers. High synchronization overhead is still one of the major performance issues in parallel large-scale agent-based traffic simulations. In this paper, we proposed a novel asynchronous conservative synchronization strategy named Mutual Appointment (MA) to address this issue. MA removes global barriers and allows LPs to communicate individually. Since the efficiency of conservative synchronization relies on the lookahead of the simulated system, a heuristic was developed to increase the lookahead in agent-based traffic simulations. It takes advantage of the intrinsic uncertainties in traffic simulations. MA together with the lookahead heuristic forms the Relaxed Mutual Appointment (RMA) strategy. Its efficiency was investigated in the parallel agent-based traffic simulator SEMSim Traffic using real world traffic data. Experiment results showed that the MA strategy improved the speed-up of the parallel simulation compared to the barrier method, and the RMA strategy further improved the MA strategy by reducing the number of synchronization messages significantly.

SIGSIM-PADS'15, June 10–12, 2015, London, United Kingdom.

DOI: http://dx.doi.org/10.1145/2769458.2769461.

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

Keywords

Agent-based traffic simulation, Asynchronous conservative synchronization, Relaxation

1. INTRODUCTION

With the fast urbanization of our modern society, road traffic in large cities and mega-cities are facing problems such as congestion and high emission which negatively impact the comfort and health of urban inhabitants. To study road traffic and solve urban traffic problems, the modeling and simulation of road traffic has been a useful tool. The modeling and simulation of road traffic can be approached in various ways, depending on the level of abstraction necessary. The levels of abstraction are commonly known as macroscopic [16, 23], mesoscopic [22], microscopic [9], and nanoscopic (a.k.a., sub-microscopic) [18]. Road traffic is a complex system whose behavior is difficult to predict. As a complex system, the behaviors of constituent components have an impact on the whole system. From this perspective, the modeling of the behaviors of individual components, i.e., driver-vehicle-units falls within the scope of the microscopic and nanoscopic levels of detail. Both microscopic and nanoscopic simulations can be conducted in an agent-based approach where driver-vehicle-units are agents

that have certain behaviors and try to reach certain goals in the simulation. Large-scale agent-based traffic simulation is a promising method for studying the road traffic, for instance, the impact of various driving behaviors on the traffic and the influence of adopting electric vehicles (EVs) or automatic vehicles in the transportation system. SEMSim Traffic is a nanoscopic traffic simulator that is able to capture this level of detail [28]. It is designed to study how different vehicle designs and different infrastructures will 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 requirement of high computational resource. To make large-scale agent-based traffic simulations computationally feasible, certain computing techniques should be deployed. To harness more computational resource, parallel computing can be used. There are many critical aspects of considerations on developing a parallel simulation, for example, time synchronization [7], partitioning and load balancing [28], and interest management [14]. The problem of time synchronization is addressed in this study. The focus is to reduce the overhead of the time synchronization in agent-based traffic simulation.

Existing microscopic and nanoscopic traffic simulations are conventionally executed in a time-stepped fashion [1–3, 17, 26]. In parallel microscopic and nanoscopic traffic simulations, the decomposition of the simulation is usually achieved by decomposing the simulation space into multiple spatial subregions. The agent models contained by each subregion are executed by Logical Processes (LPs). The LPs are usually assigned to different physical processing units. Due to the interaction of agents, there are usually data read and write dependencies between LPs. When read or write dependency happens between two LPs at a certain simulation time, the LPs should not progress over the simulation time until the dependency is fulfilled by the LPs exchanging necessary data. This operation to fulfill the dependencies is referred as a *synchronization* operation. The simulation analyzed in this work is a discrete-event simulation, where agents schedule time-stamped events which are ordered by their time-stamps and the simulation progresses by executing the events. Synchronization strategies have been well studied in the parallel discrete-event simulation community. Synchronization strategies can be broadly categorized into conservative and optimistic [7, 20]. Conservative strategies prohibits any causality error from occurring, whereas optimistic strategies uses a detection and recover approach: causality errors are detected, and a rollback mechanism is invoked to recover [7]. In existing work, synchronization in the traffic simulations is generally achieved using global barriers either in a shared memory environment [1, 2] or distributed memory environment [3, 17, 26]. This method is straightforward to use. Global barriers can be deployed at the end of time steps. This is equivalent to a *synchronous* conservative synchronization strategy where all LPs participate in a synchronization at the same time and usually global reduction is used. The limitation of this synchronization method is that all processes have to wait in front of a global barrier during the synchronization. This may decrease the parallel efficiency. Another type of conservative synchronization strategy is *asynchronous*. LPs do not wait at a global barrier when synchronizing in an asynchronous strategy. To the best of our knowledge, asynchronous conservative synchronization has not been used in parallel agent-based traffic simulations.

In this study, we have developed a novel asynchronous conservative synchronization strategy for agent-based traffic simulations named Mutual Appointment (MA). It considered the characteristics of the behavioral models of agents. In MA, LPs synchronize with each other using appointments. An appointment is an event scheduled at a specific simulation time where two processes exchange data by sending messages and whose time stamp is mutually made by the two communicating partners according to their *lookaheads*. Lookahead of an LP_1 towards another LP_2 at simulation time t is a time interval Δt in the simulated future within which LP_1 will not have data read and write dependency with LP_2. The lookahead values represent the ability of an LP predicting its future behavior that may affect other LPs. The larger the lookahead values, the less the synchronization operations. Due to the characteristic of behavioral models of agents, agent-based traffic simulations have inherently low lookahead. Low lookahead limits the efficiency of conservative synchronization strategies, therefore, a way to increase the lookahead is required. Therefore, we developed a heuristic to increase the lookahead. The lookahead heuristic takes advantage of the intrinsic *uncertainties* of traffic simulations. We claim that as long as the parallelization does not increase the uncertainty of the simulation, certain dependencies between the processes of the parallel simulation can be violated. The heuristic allows a certain amount of violation of dependency but keeps the output of the simulation statistically unaltered. It uses the traffic flow information at run-time to exploit the lookahead. This heuristic together with MA forms Relaxed Mutual Appointment (RMA) strategy. Statistical tests were performed to ensure that the output of the parallel and sequential simulations are statistically indistinguishable. The synchronization strategies are experimented in the traffic simulator SEMSim Traffic. Our contribution to the literature is

- an asynchronous conservative synchronization strategy for parallel agent-based traffic simulations (Section 3); and
- a heuristic to solve the problem of low lookahead of agent-based traffic simulations taking advantage of the uncertainties in traffic simulations (Section 4.2)

Examples of potential usage of the proposed methods are: enable large-scale traffic simulations to operate at or faster than real-time; let agent-based traffic simulations generate results faster when used as a forecasting tool in time critical decision making situations; and reduce the running time of scientific experiments which requires repeated runs of traffic simulations.

The remainder of the paper is organized as follows: The agent models and the partitioning of the traffic simulation that our algorithm operates on is described in Section 2. Terminology and more detail on the background are presented. Then we introduce the MA synchronization strategy in Section 3, and the RMA strategy in Section 4. These two sections are the main contribution of this work. Following that, experiments and results are presented in Section 5. Subsequently, related work is presented in Section 6. In the end, the conclusions and future work are presented in Section 7.

2. PARALLEL AGENTED-BASED TRAFFIC SIMULATION

The synchronization strategies are fundamentally determined by the behavior of the simulated models and partitioning of the simulation. Agent models, the partitioning of the simulation, and more review on conservative synchronization are introduced in this section.

2.1 Models and Model Execution

The simulation space of our agent-based traffic simulation is a road network which is a *spatial network*. The spatial network consists links and nodes. Links are containers/placeholders of agents. A link may have one or more lanes. Nodes contain the connectivity information of links and lanes. A small road network is illustrated in Figure 1. For simplicity, lanes are not shown.

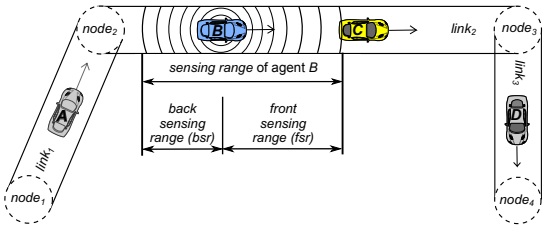

Figure 1: Agents with sensing ranges in a road network.

The *agent* in the simulation is a driver-vehicle-unit that 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 which are classified into two groups according to there visibility: agent-based state variables and component-based variables. Agent-based state variables belong to the agent and are visible to other agents, e.g., velocity and position. The component-based variables belong to a specific model and are usually not visible to other agents, e.g., a state-of-charge state variable in the battery model. The state of an agent changes as the simulation evolves with the execution of timestamped *events* which contain certain update functions. Events are ordered in ascending order of their time stamp in an *event list*. There is one event list per LP. An LP repeatedly executes a three-step cycle: advance the simulation time to the time stamp of first event in the event list, execute the event (and schedule future events triggered by this event if any), and remove the event from the event list. Events are scheduled by driver behavior models and vehicle models [28]. This event-based execution enables the models to be executed with suitable temporal resolution individually. An agent has a *sensing range* which is the area around the agent within which the states of other agents may have an effect on the agent's behavior. An illustration is shown in Figure 1. Even though the traveling direction of the agent is unidirectional in the road network, the sensing range is omni-directional, since the agent needs to examine the area in front to decide appropriate accelerations, and the area both in front and behind to decide safe lane-changes.

The models that form the main computational components are currently the driving behavior models, i.e., the acceleration and lane-changing models. Examples are Gipps' acceleration model [9] and Intelligent Driver Model (IDM) [27], and their corresponding lane-changing models [10, 13]. Acceleration models generates the actual velocity or acceleration in the next time-step based on the current surrounding information and characteristics of the driver and the vehicle. Similarly, lane-changing models decides if a lane change should be performed. The models schedule *move events*. A move event contains both acceleration and lane-changing and are scheduled periodically with a fixed interval. We refer to this fixed interval as a *move interval* in the rest of this paper. Move events change the dependency between agents since they change the positions of agents in the road network.

2.2 Partitioning and Dependency of Logical Processes

The road network is partitioned into multiple spatial subregions. A partitioning of Singapore into four partitions is shown in Figure 2. Different intensities of gray represent different partitions. An LP is responsible for executing the events of the agents in one partition, and only has access to the agents in the local space. The cutting of the network is performed on links. The links that are cut and spanning two partitions are named *boundary links*. A boundary link is evenly divided between two partitions. For instance, in Figure 3a, $link_2$ is a boundary link. The left half belongs to LP_1, and the right half belongs to LP_2. Since the driving direction of $link_2$ is from LP_1 to LP_2, we say $link_2$ is an *outgoing boundary link* of LP_1, and an *incoming boundary link* of LP_2. LP_1 and LP_2 are *neighboring processes*.

Figure 2: Singapore road network with four partitions.

As the simulation progresses, *migration* of an agent happens when the agent moves beyond the boundary of one partition and enters the space of another. For example, in Figure 3a, suppose agent B in LP_1 continues moving on $link_2$ and its new position falls within LP_2 at time t, it should be migrated to LP_2, and LP_2 would be responsible for executing future events of agent B. In this case, there is a *write dependency* between LP_1 and LP_2 at time t. If the position of agent C in LP_2 is inside the sensing range of agent B in LP_1, the state variables of agent C that may affect the behavior of agent B should be sent over to LP_1 and kept updated to maintain the correctness of the simulation. In this case, there is a *read dependency* between LP_1 and LP_2. These state variables of agent C are referred as *shared states*. To help determining where shared states are, *buffered regions* are defined. They are the areas along the boundary cut of partitions with the size equal to the sensing range of agents. The states of the agents that are inside buffered regions and directly next to the boundary cut are

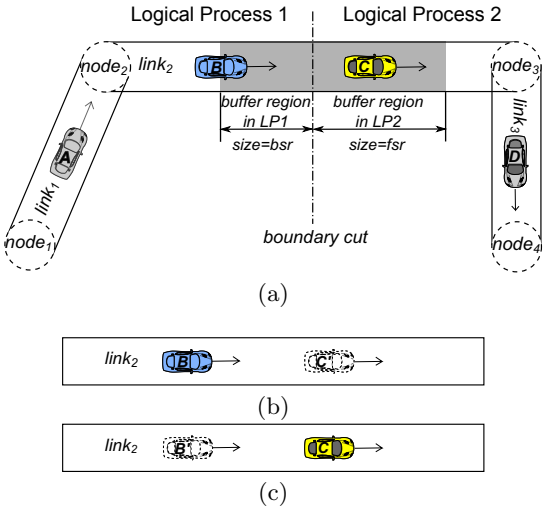

Figure 3: Road network partitioning: (a) illustration of boundary cut and buffer regions; (b) view of $link_2$ from LP_1; and (c) view of $link_2$ from LP_2.

considered as shared states. For example, in Figure 3a, the shaded area on $link_2$ are two buffered regions, with the left side inside $LP1$ and right side inside LP_2. The size of the buffer region inside LP_1 equals to the back sensing range of agents, and the size inside LP_2 equals to the front sensing range of agents. Since agent B and C are both inside the buffer region, shared states of agent B are sent to LP_2 and those of agent C to LP_1.

In addition, the LPs that receive shared states use the shared states to create non-local *proxy agents*. Proxy agents act as representatives of the *real agents* in other LPs, so that the agents can see the shared states as agents. For example, there is a proxy agent C' in LP_1 of real agent C (Figure 3b), and a proxy agent B' in LP_2 of real agent B (Figure 3c). This is for managing the shared states more easily and simplifying the implementation of the simulator.

2.3 Synchronization of Logical Processes

The need for synchronization originates from the read and write dependencies between LPs. An LP should not progress the simulation over the point when read or write dependency happens until the dependency is fulfilled by exchanging information with the relevant LPs. Synchronization here means the management of the sending and receiving of migrating agents and shared states. Synchronization strategies can be broadly categorized into conservative and optimistic [7,20]. Conservative strategies prohibits any causality error from occurring, whereas optimistic strategies allows causality errors to occur, and a rollback mechanism is invoked to recover the errors [7]. This study only deals with the conservative approach. A synchronization strategy is *synchronous* if global synchronization points are used, i.e., a barrier or a global reduction, where all LPs participate in the communication. *All* LPs are blocked until the communication is finished. In contrast, in an *asynchronous* strategy, blocking does not happen globally. When LP_2 is specially told by LP_1 that it might be affected by LP_1 at time t, only LP_2 is blocked at time t [20]. All other irrelevant LPs are free to proceed their execution without participating in this synchronization operation. Asynchronous strategy has the potential advantage over synchronous strategy that it may reduce the waiting of LPs and decrease the number of synchronization messages.

A key ingredient of an conservative synchronization strategy is the *lookahead*. Lookahead defines the ability of an LP predicting its behavior that might affect other LPs. The lookahead in our simulation can be defined as follows: if an LP_1 is at simulation time t, and it predicts that it does not have any read and write dependency with LP_2 until simulation time $t + \Delta t$, then LP_1 has a lookahead of Δt towards LP_2. To determine the lookahead values is to predict when the read and write dependencies change in the simulated future. Determining the lookahead is a critical component of a conservative synchronization strategy.

3. MUTUAL APPOINTMENT STRATEGY

This section presents the MA strategy and the lookahead determination algorithm.

3.1 Mutual Appointment

The basic idea of MA is that an LP communicates with other LPs by making *appointments* individually with them at certain mutually agreed time points of the simulation. The execution of the LP will be blocked at an appointment until the appointment is fulfilled. This blocking only happens between this LP pair. An MA contains two tasks: exchange information of current dependency and make the next appointment.The MAs are scheduled as synchronization events. The logic of an MA synchronization event is shown in Algorithm 1.

Algorithm 1: MA Synchronization Event

Definitions:

t	time stamp of the synchronization event in LP_i
$A_{i,t}$	set of agents in LP_i at t
$C_{i,t}$	set of LPs having appointments with LP_i at t
$M_{ij,t}$	set of agents migrating from LP_i to LP_j at t
$S_{ij,t}$	set of shared states sent by LP_i to LP_j at t
$l_{ij,t}$	lookahead of LP_i towards LP_j at t
$\Delta t_{ij,t}$	time interval until next appointment between LP_i and LP_j at t

foreach $LP_j \in C_{i,t}$ **do**

```
// 1. prepare the content of the message
```
determine $M_{ij,t}$, $S_{ij,t}$, and $l_{ij,t}$;

```
// 2. exchange messages
```
send $M_{ij,t}$, $S_{ij,t}$, and $l_{ij,t}$ to LP_j;
receive $M_{ji,t}$, $S_{ji,t}$, and $l_{ji,t}$ from LP_j;

```
// 3. update the set of local agents
```
$A_{i,t} \leftarrow A_{i,t} \cup M_{ji,t} \setminus M_{ij,t}$;

```
// 4. update proxy agents with shared states
```
update proxy agents with $S_{ji,t}$;

```
// 5. make a new appointment
```
$\Delta t_{ij,t} \leftarrow \min(l_{ij,t},\ l_{ji,t})$;
make an appointment with LP_j at time $t + \Delta t_{ij,t}$;

end

The event is in LP_i with the time stamp t. In the 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 $C_{i,t}$ may *not* include all

the neighboring LPs of LP_i and can be empty. For each LP_j in the set $C_{i,t}$, LP_i performs the following five steps. The first step is to prepare the data for fulfilling the read and write dependency, and calculate lookahead for the next appointment. Migrating agents $M_{ij,t}$ and shared states $S_{ij,t}$ are determined by scanning the boundary links between LP_i and LP_j. Then in the second step, LP_i sends all that information to LP_j. A point-to-point communication is used. To reduce the number of messages, a single compound synchronization message is used which contains three parts of information: migrating agents (write dependency), shared states (read dependency), and lookahead (predicted time interval until the next MA event). At the same time, it receives a message from LP_j which contains the migrating agents $M_{ji,t}$, shared states $S_{ji,t}$, and lookahead $l_{ji,t}$. The partner LP_j is inside an MA event with the time stamp t as well. After receiving the message, the third step is to update the local agents. The local agent set $A_{i,t}$ is updated by removing the agents in $M_{ij,t}$, and adding the agents in $M_{ji,t}$. Then the next step is to update the proxy agents in LP_i with the latest shared state $S_{ji,t}$. Unnecessary proxy agents are removed, and new proxy agents are created if necessary. The last step is to schedule the next synchronization event. LP_i determines the time interval until the next appointment with LP_j as $\Delta t_{ij,t} = min(l_{ij,t},\ l_{ji,t})$, and the next synchronization event with LP_j will be scheduled at time $t + \Delta t_{ij,t}$. If there is no synchronization event scheduled at the simulation time $t + \Delta t_{ij,t}$ yet, a new synchronization event is scheduled. Otherwise, the id of LP_j will be added to the set $C_{i,(t+\Delta t_{ij,t})}$ of the synchronization event that already scheduled at time $t + \Delta t_{ij,t}$ instead of scheduling a new event.

The initial MAs are scheduled at the beginning of the simulation for all LPs. The targets of the initial MAs of an LP are all of its neighboring LPs. All subsequent MAs are made based on lookaheads.

3.2 Lookahead Determination

The lookahead in the MA synchronization strategy is *directed*, which means that an LP may have different lookahead values towards each neighboring LPs at a certain simulation time. The lookahead values are used to make individual appointments. There are three cases considered according to different traffic conditions.

The first case is that no agents exist in the local space. This usually happens when the simulation just starts up. The lookahead should be the minimum time for any newly created agent to travel into buffer regions on outgoing boundary links. Suppose the set of agents to be created locally in LP_i is NA=$\{na_1, na_2, \ldots, na_{|NA|}\}$; agent na_k ($1 \leq k \leq |NA|$) is scheduled to be created nat_k later in the future, and requires minimum traveling time tt_k to travel from the point of creation to the buffer region connecting to LP_j on its route (if the agent does not pass by LP_j, tt_k=∞), then the earliest time for newly created agents to possibly affect LP_j is $min(nat_k + tt_k)$ later from the current time. Thus the current lookahead towards LP_j should be $min(nat_k + tt_k)$. If the schedule to create new agents is not available, the effect of new agents should be considered in another way. Suppose the set of outgoing boundary links towards LP_j is OL_{ij}, the minimum traveling time from the start of a link $l\,(l \in OL_{ij})$ to the buffer region of that link is obt_l, then the minimum time for any new agent to travel to a buffer region is $\min(obt_l)$. An example of obt_l is shown in Figure 4a. The lookahead should be $\min(obt_l)$. Note that all the estimation of traveling time here is a lower bound. The actual traveling time is usually longer, since the actual velocities of agents are usually smaller than the speed limit, in other words, since agents usually do not travel with speed-limit, tt_k and $\min(obt_l)$ are rather conservative estimations.

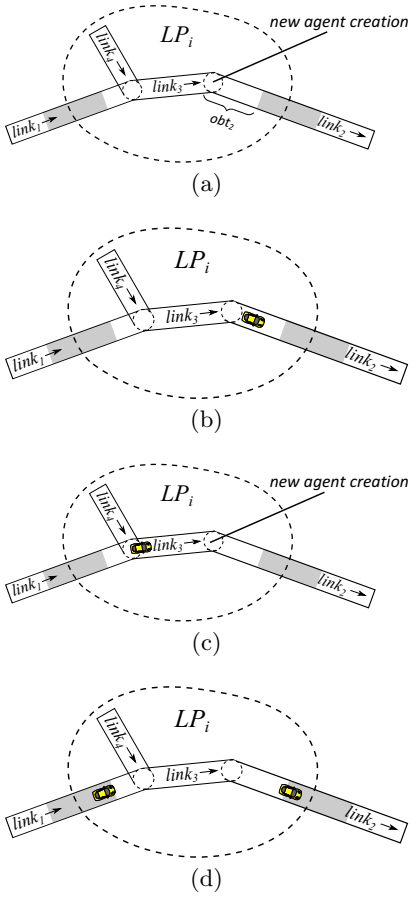

Figure 4: Scenarios of MA lookahead calculation: (a) no local agents exist, a new agent is being created in a network node; (b) buffer regions are empty and no migrating agents; (c) same with (b) with agents being created; and (d) buffer regions are not empty.

The second case is that there are local agents, but there are no agents inside buffer regions or migrating before the next move event, as depicted in Figures 4b and 4c. The lookahead is the minimum time for a local agent to travel to a buffer region. Migration of local agents only happens on outgoing links, therefore, the agents whose routes pass through outgoing boundary links should be examined. Note that new agents may be created on an outgoing link (Figure 4c). This case should also be considered. Suppose the minimum traveling time of any existing agent to a buffer region connecting to LP_j on its route is at_j, considering the agents to be created NA, the lookahead in this case is $min(at_j, min(nat_k + tt_k))$. If the schedule to create new agents is not available, similar to the first case, $min(obt_l)$ should be used instead of $\min(nat_k + tt_k)$. Then, the lookahead is $\min(at_j, min(obt_l))$.

The third case is that there are agents inside local buffer regions or to be migrated before the next move event. Lookahead can only be as big as one move interval, since the shared

Algorithm 2: MA lookahead determination

Definitions:

A_i	set of agents in LP_i
L_{ij}	set of boundary links between LP_i and LP_j
M_l	set of agents migrating on link l, $l \in L_{ij}$
S_l	set of shared states on link l in LP_i, $l \in L_{ij}$
nat_k	time interval before new agent k is created
tt_k	minimum time for new agent k to travel to any outgoing boundary link towards LP_j on its route
obt_l	traveling time from the start of link l to the buffer region of l, l is an outgoing link towards LP_j
at_j	minimum time for any agent in A_i to travel from its current position to the buffer region of any outgoing boundary link towards LP_j on its route
sa	boolean - if the schedule to create agents available

Result: lookahead towards LP_j

Initialize $lookahead \leftarrow maximum\ double$;
// the first case
if $A_i = \emptyset$ **then**
  if sa **then**
    $lookahead \leftarrow min(nat_k + tt_k)$;
  else
    $lookahead \leftarrow min(obt_l)$;
  end
// the second case
else if $\forall l \in L_{ij}: M_l = \emptyset \wedge S_l = \emptyset$ **then**
  if sa **then**
    $lookahead \leftarrow min(at_j, min(nat_k + tt_k))$;
  else
    $lookahead \leftarrow min(at_j, min(obt_l))$;
  end
// the third case
else
  $lookahead \leftarrow one\ move\ interval$;
end
$lookahead \leftarrow max(lookahead,\ one\ move\ interval)$;

states must be updated before the next move event takes place. An example is shown in Figure 4d. There is an agent inside the buffer region of an incoming boundary link $link_1$ of LP_i and another inside the buffer region of an outgoing boundary link $link_2$.

The minimum value of a lookahead is one move interval, because one move interval is the assumed period of time in which no states of the agents that may affect other agents are updated. After estimating the lookahead, if the lookahead value is smaller than one move interval, its value is set to one move interval. An illustration of process of determining the MA lookahead is shown in Algorithm 2. The worse case of this lookahead determination algorithm happens when there are always migrating agents or shared states, in which case the lookahead always equals to one move interval.

4. RELAXED MUTUAL APPOINTMENT STRATEGY

The lookahead described in Section 3.2 approaches to one move interval when the traffic is always dense along the boundary of LPs and there are often migrating agents and shared states. In this case, MA may hardly have advantage over a global barrier synchronization. This section presents a method to increase the lookahead by exploiting the uncertainty in the simulation and potentially trading-off certain accuracy of the simulation.

4.1 Uncertainties in the Simulation

Error and uncertainty exist throughout the process of modeling and simulation [11, 21]. 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 are always uncertainties in a traffic simulation [5]. Uncertainty is categorized into two types in literature. The first category is *aleatory uncertainty*, a.k.a., stochastic uncertainty, irreducible uncertainty, inherent uncertainty, and variability. It is the inherent variation associated with the physical system, in this context, real road traffic. Aleatory uncertainty is represented as a number of streams of random variables drawn from specified probability distributions in a computer simulation, for example, the distribution of the trips 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, and cognitive uncertainty. It results from a lack of knowledge about the simulated system. For example, the input for traffic simulation, such as traffic demand, is usually estimated from real world observed data or forecast, and *input uncertainty* arises from the estimation and forecast process. Models used in the traffic simulation are the abstraction of the real world, and *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) [5]. Uncertainty in traffic simulation may be utilized to improve the synchronization of parallel traffic simulation, which is presented in the following subsection.

4.2 Lookahead in RMA

Consider a situation where the traffic is jammed on a boundary link, and all agents are hardly moving, the states of the agents do not change much, but the synchronization is performed once per move interval according to the MA lookahead algorithm. This frequent synchronization may not be necessary because skipping some synchronization events here may not have much influence on the output of the simulation. As discussed, traffic simulations have inherent stochastic uncertainties. The output of the simulations has certain variability and there is no single correct output of the simulation. This indicates that the parallel simulation does not necessarily produce the exact output as the sequential simulation; instead, a statistically equivalent output is sufficient. Thus, a relaxed lookahead algorithm can be developed, as long as the algorithm does not distort the output of the simulation statistically. This is a similar idea to Fujimoto's work in [8], where the temporal uncertainty of models are exploited to improve the synchronization of parallel and distributed simulations.

When the lookahead is relaxed, the migration of agents may not be achieved in time and the shared states may not be updated in time. The direct consequence is that agents use the obsolete states of proxy agents until the next synchronization. To reduce the discrepancy between the real agents and proxy agents, dead-reckoning is used to update

the proxy agents. An illustration of possible discrepancy between a strict synchronization and a relaxed synchronization is shown in Figure 5. Suppose agent B is a local agent

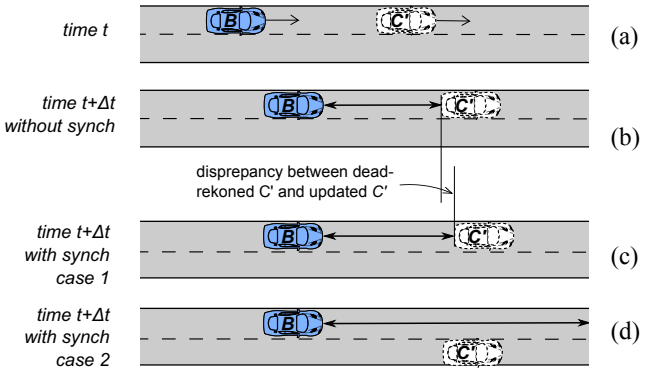

Figure 5: The effect of relaxed synchronization: (a) agent B and proxy agent C' at time t; (b) proxy agent C' at time $t+\Delta t$ calculated by dead-reckoning; (c) proxy agent C' at time $t+\Delta t$ updated by synchronization; (d) same with (c), but agent C performed lane-changing.

and agent C' is a proxy agent in the LP at time t (Figure 5a). The real agent of C', C, resides in another LP. As the simulation progresses to time $t + \Delta t$ (Δt is a move interval), agent B and C calculate their new states. The proxy agent C' should be updated with the shared states of agent C at time $t + \Delta t$ by a synchronization operation. When a synchronization is not performed, C' is updated by dead-reckoning. Discrepancy of the state of C' may occur here due to the difference between the real update function and the dead-reckoning function. Figure 5b and 5c show a case where there is discrepancy of positions between the updated C' and the dead-reckoned C'. Another case where the real agent C has performed lane-change is shown in Figure 5d. The discrepancy of proxy agent C' may or may not lead to discrepancy on agent B in the simulated future depending on their relative position, relative speed, and the sensitivity of the driver's behavior models. Further more, even if there is discrepancy on agent B, the discrepancy may or may not affect the final output of the simulation.

Two factors may influence the allowed relaxation of lookahead: dead-reckoning function and traffic condition. A simple dead-reckoning function can just assume that agents move in 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. Therefore the potential discrepancy in different traffic conditions can be estimated by investigating how the velocities of agents change. Lookahead can be lengthened 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. An experiment studying the relationships between the average change of velocities and these three metrics is shown in Figure 6. A road network with two connecting links l_1 and l_2 were used. The traffic direction is from l_1 to l_2. Agents are created on link l_1, and they travel through both links.

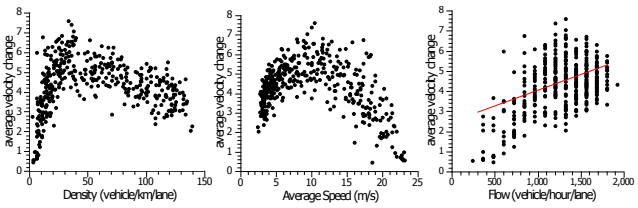

Figure 6: The change of velocities with respect to: (a) density; (b) speed; and (c) flow (the line here is a linear regression line).

Traffic flow, density, average speed, and the average velocity change on a segment of l_1 were recorded every move interval. To create different traffic conditions on l_1, we varied the inter-arrival time of agents, the speed limit of l_1, and the number of lanes in l_2. The average velocity change is calculated with $\sqrt{\frac{\sum_{a=1}^{N}[v_{(a,t+\Delta t)}-v_{(a,t)}]^2}{N}}$, where N is the number of agents on the road segment, and $v_{(a,\,t)}$ is the velocity of agent a at time t, and Δt is the move interval. The result in Figure 6 shows that there is a linear correlation between the change of velocities and traffic flow. The correlation between the change of velocities and density or velocity is not linear. This indicates that if the synchronization is relaxed, the discrepancy is more likely to be greater when traffic flow is higher. A lookahead model can be developed using traffic flow.

We firstly define a time window $window_{l,t}$ of a boundary link l at time t as the longest time period within which synchronizations can be skipped without altering the simulation output. The window is calculated using

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

where α is a sensitive factor, and $flow_l$ is the traffic flow on link l at time t. When $flow_{l,t} = 0$, $window_{l,t} = mw_l$, where mw_l is the maximum window. In fact, the physical meaning of $\frac{1}{flow_{l,t}}$ 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 partners, it is sufficient to consider only outgoing boundary links. Let OL_{ij} be the set of outgoing boundary links from LP_i and LP_j, then the lookahead from LP_i to LP_j at time t is

$$lookahead_{ij,t} = min(window_{l,t}),\ l \in OL_{ij} \tag{2}$$

The value of α controls the amount of relaxation introduced. The optimal value of α is the one that can maximize lookahead and do not distort the statistical output of the simulation. It may be influenced by the models used in the traffic simulation, the road network, and partitioning. The discrepancy on boundary links may propagate to the upper stream and the lower stream of the link. Therefore, it may be not possible to obtain the optimal α value. A proper α value can be gained with testing experiments.

The lookahead determination algorithm for RMA is shown in Algorithm 3. The algorithm begins with checking the

Algorithm 3: RMA lookahead determination

Definitions:
A_i set of agents in LP_i
OL_{ij} set of outgoing boundary links from LP_i to LP_j
$flow_l$ current traffic flow on link l, $l \in OL_{ij}$
mw_l maximum window of link l, $l \in OL_{ij}$
Result: lookahead towards LP_j

Initialize $lookahead \leftarrow maximum\ double$;
if $A_i = \emptyset$ **then**
  $lookahead \leftarrow \min(mw_l)$, $l \in OL_{ij}$;
else
  foreach $l \in OL_{ij}$ **do**
    if $flow_l = 0$ **then**
      $lookahead \leftarrow \min(lookahead,\ mw_l)$;
    else
      $lookahead \leftarrow \min(lookahead,\ \alpha \cdot \frac{1}{flow_l})$;
    end
  end
end
$lookahead \leftarrow max(lookahead,\ one\ move\ interval)$;

local agent population: if the local agent set is empty, the lookahead is set to the minimum value of maximum windows directly; if not, the outgoing boundary links are checked one by one and the minimum value of time windows is taken as the lookahead. If the value is less than one move interval, set the lookahead to one move interval. The maximum window of l is calculated using the minimum traveling time of l (length divided by speed limit).

4.3 Output Measurement

It is necessary to ensure that the output of the parallel simulation with the RMA synchronization strategy has no more uncertainty than the sequential simulation. If we treat the sequential simulation as the reference, we should make sure that the output of the parallel simulation is statistically identical to the output of the sequential simulation. For statistical test of equivalence, the Kolmogorov-Smirnov two-sample test or the Bootstrap method [6] can be used. They do not make assumptions on the probability distribution of the tested variable, thus fit for our study. The output variables measured may be different for different studies, for example, some studies are interested in trip durations of agents, and some studies are interested in traffic flows on roads [5].

5. EXPERIMENTS

We conducted experiments to investigate the performance of the MA method and the RMA method in terms of the average lookahead, total synchronization messages sent, and overall speed-up of the simulation. Comparison with the conventional barrier synchronization method was made. In the MA lookahead algorithm, the schedule of creating new agents was not used. The simulation is implemented using C++, and the communications are realized using OpenMPI.

5.1 Set-up

Real world data were used in the traffic simulation of our experiments, including the road network and trips of agents. The experiments were set up as follows: The road network is the network of whole Singapore that consists of 43,392 nodes and 84,343 links (124,589 lanes). The trip distribution was derived from the data of the Household Interview and Travel Survey (HITS). The acceleration model used was the IDM model. Due to the lack of real data on the traffic flow, the agent models are not calibrated, thus, only approximate values of the input parameters of models were used. Therefore, the simulated traffic may not accurately reflect the real world. The maximum number of agents during the peak traffic hours of the day was approximately 75,000, which is presumably smaller than the real traffic. The models were collision-free, therefore, no accidents or emergencies occurred in the simulation. The traffic of 24 hours from the midnight of one day to the midnight of the next day was simulated. The simulation warm-up period was from the midnight to 5 am of which the statistics was excluded. The size of a move interval was 0.5 second. The parallel simulation was partitioned using METIS [12]. Partitioning was only performed once at the beginning of the simulation, and no dynamic load-balancing was performed. Partitioning was performed in a way that the boundary links between LPs tended to be the long road links.

The experiments were run on a cluster that is composed of 4 compute nodes each of which has the following hardware configurations: 2 Octa-core $Intel(R)\,Xeon(R)\,CPU\,E5-2670\,0\,@2.60GHz$, 192GB of RAM. The compute nodes are connected via 56Gbps InfiniBand.

5.2 Output Equivalence to Sequential Simulation

We expect the suitable values of α vary with different partitioning on the network. The number of links cut usually increases as the number of partitions increases, thus, the same α may have higher impact on the accuracy of the simulation with more LPs. We experimented on a range of α values to find out the effect. The sequential simulation was firstly run twelve times with different random seeds, as the referential output group. Then the parallel simulation was run with different α values and different number of LPs. For each α value and each number of LPs, the parallel simulation was run twelve times as one group using the same twelve random seeds as the sequential group. Statistical tests of equivalence were performed between each parallel group and the sequential referential group. The output variable investigated was the *average trip duration* of agents throughout the simulation. The statistical test of equivalence used was the *Bootstrap method*. The null hypothesis H_0 of the test was that there is no difference between the average trip duration of agents in the sequential simulation and that in the parallel simulation. The alternative hypothesis H_1 was that the average trip durations are different. Suppose the current sequential group is $A=\{a_1, a_2, \ldots, a_{12}\}$, and the 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 $B'=\{b'_1, b'_2, \ldots, b'_{12}\}$ were obtained, where $a'_i \in A$ and $b'_i \in B$. Then the difference of the means of the two resampled groups was calculated by $d=\overline{A'}-\overline{B'}$. We performed resampling 1000 times. Thus, we obtained 1000 values of the difference of means. The 1000 values formed the distribution of the difference of means. Using a 95% confidence interval, H_0 would be rejected if zero falls outside of the percentile range between 2.5% and 97.5%. Otherwise, H_0 could not be rejected. The testing result is shown in Table 1. For some α values, data were

Table 1: Acceptance of the null hypothesis using the Bootstrap method

	4 LPs	8 LPs	16 LPs	32 LPs
$\alpha = 0.2$	accepted	accepted	accepted	accepted
$\alpha = 0.4$	*	*	*	accepted
$\alpha = 0.5$	accepted	accepted	accepted	rejected
$\alpha = 1.0$	accepted	rejected	rejected	rejected
$\alpha = 1.5$	rejected	*	*	*

* data not collected

not collected since they were not necessary. For example, if H_0 could not be rejected when α=0.5, H_0 must not be rejected when α<0.5 either. This is because a smaller sensitivity factor results in a smaller average lookahead which means a more frequent synchronization, and it should lead to less discrepancy between the parallel simulation and the sequential simulation.

From Table 1, we can take 1.0 as a suitable α value for 4 LPs, 0.5 for 8 LPs and 16 LPs, and 0.4 for 32 LPs. The values may not be the optimum; however, the optimum value cannot be gained without traversing through all possible α values, which is not practical. Judging by the values we obtained, we noticed that the suitable α value decreased as the number of LPs increased. Experiments were not conducted for more than 32 LPs due to hardware constrains. If the RMA synchronization strategy would be used in practice, certain prediction model for α value should be developed to reduce the effort of obtaining α instead of running the simulation itself. The value of α using more than 32 LPs may also be estimated. Developing a model for α can be a piece of future work.

5.3 Lookahead and Synchronization Messages

The direct impact of different sensitivity factor α values is the lookahead and the number of synchronization messages. The average lookahead values of the simulation using the MA method and the RMA method with suitable α values (1.0 for 4 LPs; 0.5 for 8 LPs and 16 LPs; and 0.4 for 32 LPs) are shown in Figure 7, and the total numbers of synchronization messages are shown in Figure 8. The lookahead

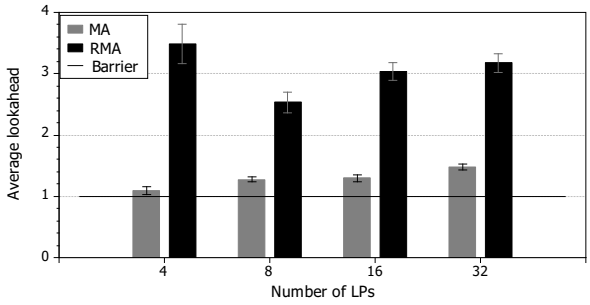

Figure 7: Average lookahead with the MA and RMS methods.

was measured in terms of move intervals. Figure 7 shows that the lookahead values in the MA method were slightly larger than one move interval. This indicates that the worse case, in which the lookahead is one move interval, did not always happen. The lookahead values were larger in the RMA method. When the average lookahead was larger, the total synchronization messages sent in the simulation were less (with the same number of LPs), as shown in Figure 8. The number of total messages was reduced by a large proportion in the RMA method compared to the MA method. The message count in the barrier method is not shown in Figure 8 because the barrier method uses an *all-to-all* MPI communication every move interval to determine the neighboring processes to communicate. The all-to-all communications acts as barriers. The message count is not comparable to the MA and RMA methods.

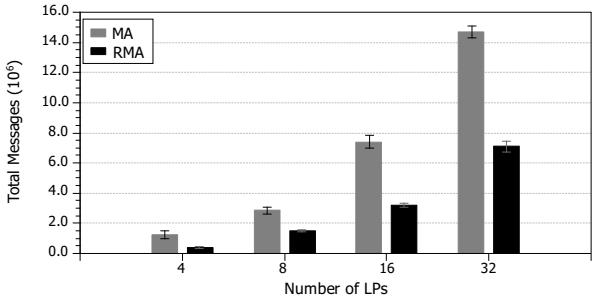

Figure 8: Total synchronization messages with the MA and RMS methods.

5.4 Speed-up

We investigated whether there was improvement on the overall speed-up of the parallel simulation using the MA and RMA methods over the conventional barrier method. We again used α=1.0 for 4 LPs, 0.5 for 8 LPs and 16 LPs, and 0.4 32 LPs in the RMA method. The speed-ups are shown in Figure 9. The speed-up was calculated using the

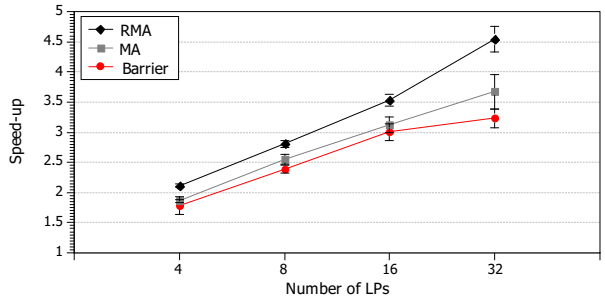

Figure 9: Speed-up of the three synchronization methods.

running time of the sequential simulation divided by the running time of the parallel simulation. Each configuration was run multiple times and the average was taken. The sequential simulation took approximately 4 hours to simulate the whole day's traffic. The parallel simulations using 32 LPs took around 50 to 70 minutes. The simulation time might be much longer if a full city-scale traffic was simulated. We observed that the MA method always performed better than the barrier method, and the RMA method always performed better than the MA method. The speed-up improvement of the MA method over the barrier method was in the range of 2.1% (4 LPs) to 13.6% (32 LPs), and the RMA method over the barrier method was in the range of 17.2% (8 LPs) to 40.7% (32 LPs). The advantage of the MA method over the barrier method has two sources: less waiting time of LPs

since global barriers do not exist; and less communication overhead due to the omission of global all-to-all communications. The RMA method has even less synchronization overhead than the MA method. The reduction of the synchronization overhead comes mainly from two sources: the total size of data sent over is reduced since less shared states are sent over (despite that the number of migrated agents are the same); data are packed into larger messages and less messages are sent which reduces the overall message passing time (less start-up time for message passing).

6. RELATED WORK

The closest related work is the synchronization strategies in other parallel agent-based traffic simulations [17,26]. The simulation time in agent-based traffic simulations are conventionally progressed in a time-stepped fashion. Synchronization is performed synchronously among all LPs at the end of time-steps. To date, we have not encountered any literature studying asynchronous conservative synchronization in agent-based traffic simulations.

There is work on improving the synchronization of general agent-based simulations. One attempt is to use optimistic synchronization [15]. With an optimistic approach strategy, dependencies between LPs can be temporarily violated, and rollback is performed if any violation is detected. However, the method needs to save simulation states and perform rollbacks which are extra overheads and require more effort to implement. The optimistic approach might be not beneficial when the dependencies among agents are heavy along the boundary. Another attempt to reduce the synchronization is through asynchronous agent scheduling [25]. The dependencies between agents are analyzed every update cycle and the agents can be in different update cycles if they are not dependent on each other. In [25], synchronization was done by a central server that keeps the states of all agents. Processes only communicate with the server when proxy agents need to be updated.

Asynchronous conservative synchronization strategies have been well studied in the parallel discrete-event simulation community [4, 8, 20]. The performance is good in applications where lookahead can be well exploited such as queuing networks [19]. Our MA strategy is very similar to the appointment protocol in [19] which was used in a queuing network simulation. The difference is that in our simulation the appointments between LPs should be mutually agreed due to the characteristic of agent-based simulations.

Another related area of work is about relaxing the synchronization among LPs. An approximate time causal order were proposed in [8] which relaxes the conventional strict time stamp order of events. It took advantage of the fact that temporal uncertainty of events always exists in simulations. Events can be executed in an approximate order and more events could be executed concurrently. Our relaxed synchronization strategy focuses on improving the lookahead of LPs without affecting the accuracy of the simulation. We considered the characteristics of the agent-based models in traffic simulation and analyzed the discrepancy between the sequential simulation and the relaxed parallel simulation. A study on the impact of relaxing the barrier synchronization in distributed agent-based simulations was presented in [24], however, asynchronous synchronization was not mentioned and it was not studied how to maintain the accuracy of a distributed simulation.

7. CONCLUSIONS AND FUTURE WORK

We have proposed a new asynchronous conservative synchronization strategy for parallel agent-based traffic simulations. Its asynchronous nature reduces the waiting of processes at barriers compared to the conventional global barrier synchronization method. To address the low lookahead issue in agent-based traffic simulations, we have also developed a relaxed lookahead heuristic which improves the lookahead by taking advantage of the uncertainty in traffic simulations. Experiments have shown that it has improved the parallel speed-up to a certain degree when keeping the outputs of the parallel simulation statistically indistinguishable to the sequential simulation.

Future work may include: firstly, developing a prediction model for the suitable value of α to reduce the effort of obtaining α, which makes the lookahead model in the RMA method easier for practical use; the characteristics of the road traffic should be considered; and secondly, investigating more intelligent dead-reckoning functions under different traffic conditions which may improve the relaxation of lookahead.

8. ACKNOWLEDGMENT

This work was financially supported by the Singapore National Research Foundation under its Campus for Research Excellence And Technological Enterprise (CREATE) programme. M.L. acknowledges the support of the Russian Scientific Foundation, Project #14-21-00137.

9. REFERENCES

[1] H. Aydt, Y. Xu, M. Lees, and A. Knoll. A multi-threaded execution model for the agent-based semsim traffic simulation. In *Proceedings of the 13th International Conference on Systems Simulation*, pages 1–12, Singapore, November 06–08, 2013. Springer Berlin Heidelberg.

[2] J. Barceló, J. Ferrer, D. García, M. Florian, and E. Saux. Parallelization of microscopic traffic simulation for att systems analysis. In P. Marcotte and S. Nguyen, editors, *Equilibrium and Advanced Transportation Modelling*, pages 1–26. Springer US, 1998.

[3] G. D. Cameron. PARAMICS–Parallel Microscopic Simulation of Road Traffic. *The Journal of Supercomputing*, 53(1):25–53, 1996.

[4] K. Chandy and J. Misra. Distributed simulation: A case study in design and verification of distributed programs. *IEEE Transactions on Software Engineering*, SE-5(5):440–452, 1979.

[5] G. de Jong, A. Daly, M. Pieters, S. Miller, R. Plasmeijer, and F. Hofman. Uncertainty in traffic forecasts: literature review and new results for The Netherlands. *Transportation*, 34(4):375–395, 2006.

[6] B. Efron. Bootstrap methods: another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.

[7] R. Fujimoto. Parallel Discrete Event Simulation. *Communications of the ACM*, 33(10):30–52, 1990.

[8] R. Fujimoto. Exploiting temporal uncertainty in parallel and distributed simulations. In *Proceedings of the 13th Workshop on Parallel and Distributed Simulation*, pages 46–53, Atlanta, GA, USA, May 01-04, 1999. IEEE.

[9] P. Gipps. A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, 15(2):105–111, 1981.

[10] P. Gipps. A model for the structure of lane-changing decisions. *Transportation Research Part B: Methodological*, 20(5):403–414, 1986.

[11] J. Helton. Uncertainty and sensitivity analysis in the presence of stochastic and subjective uncertainty. *Journal of Statistical Computation and Simulation*, 57(1-4):3–76, 1997.

[12] G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1999.

[13] A. Kesting, M. Treiber, and D. Helbing. General Lane-Changing Model MOBIL for Car-Following Models. *Transportation Research Record: Journal of Transportation Research Record*, 1999(1):86–94, 2007.

[14] M. Lees, B. Logan, R. Minson, T. Oguara, and G. Theodoropoulos. Modelling environments for distributed simulation. In D. Weyns, H. Van Dyke Parunak, and F. Michel, editors, *Environments for Multi-Agent Systems*, volume 3374 of *Lecture Notes in Computer Science*, pages 150–167. Springer Berlin Heidelberg, 2005.

[15] M. Lees, B. Logan, and G. Theodoropoulos. Adaptive optimistic synchronisation for multi-agent distributed simulation. In *Proceedings of the 17th European Simulation Multiconference*, pages 77–82, Nottingham, UK, June 09-11, 2003. Society for Modelling and Simulation International.

[16] M. Lighthill and G. Whitham. 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):317–345, 1955.

[17] K. Nagel and M. Rickert. Parallel implementation of the TRANSIMS. *Parallel Computing*, 27(12): 1611–1639, 2001.

[18] D. Ni. 2DSIM: a prototype of nanoscopic traffic simulation. In *Proceedings of the 2003 Intelligent Vehicles Symposium*, pages 47–52, Columbus, OH, USA, June 09-11, 2003. IEEE.

[19] D. M. Nicol. Parallel discrete-event simulation of fcfs stochastic queueing networks. In *Proceedings of the ACM/SIGPLAN Conference on Parallel Programming: Experience with Applications, Languages and Systems*, pages 124–137, New Haven, CT, USA, July 19-21, 1988. ACM.

[20] D. M. Nicol. Principles of conservative parallel simulation. In *Proceedings of the 1996 Winter Simulation Conference*, pages 128–135, Coronado, CA, USA, December 08-11, 1996. IEEE.

[21] W. L. Oberkampf, S. M. DeLand, B. M. Rutherford, K. V. Diegert, and K. F. Alvin. Error and uncertainty in modeling and simulation. *Reliability Engineering & System Safety*, 75(3):333–357, 2002.

[22] S. Paveri-Fontana. 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):225–235, 1975.

[23] P. Richards. Shock Waves on the Highway. *Operations Research*, 4(1):42–51, 1956.

[24] O. Rihawi, Y. Secq, and P. Mathieu. Relaxing Synchronization Constraints in Distributed Agent-based Simulations. *Jurnal Teknologi*, 63(3):65–76, 2013.

[25] M. Scheutz and J. Harris. Adaptive scheduling algorithms for the dynamic distribution and parallel execution of spatial agent-based models. In F. de Vega and E. Cantú-Paz, editors, *Parallel and Distributed Computational Intelligence*, volume 269 of *Studies in Computational Intelligence*, pages 207–233. Springer Berlin Heidelberg, 2010.

[26] T. Suzumura and H. Kanezashi. 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*, pages 243–250, Dublin, Ireland, October 25-27, 2012. IEEE.

[27] M. Treiber, A. Hennecke, and D. Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E (Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics)*, 62(2):1805–1824, 2000.

[28] Y. Xu, H. Aydt, and M. Lees. SEMSim: A Distributed Architecture for Multi-scale Traffic Simulation. In *Proceedings of the 2012 ACM/IEEE/SCS 26th Workshop on Principles of Advanced and Distributed Simulation*, pages 178–180, Zhangjiajie, China, July 15-19, 2012. IEEE.