Proceedings of the 2011 Winter Simulation Conference
S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu, eds.

AGENT-BASED DISCRETE-EVENT HYBRID SPACE MODELING APPROACH FOR TRANSPORTATION EVACUATION SIMULATION

Bo Zhang
Wai Kin (Victor) Chan

Rensselaer Polytechnic Institute
Department of Industrial and Systems Engineering
110 Eighth Street
Troy, NY 12180, USA

Satish V. Ukkusuri

Purdue University
School of Civil Engineering
550 Stadium Mall Drive
West Lafayette, IN 47906, USA

ABSTRACT

This paper presents an agent-based discrete-event simulation (AB-DES) modeling approach for transportation evacuation simulation based on a hybrid continuous and cell space. This approach combines advantages of agent-based simulation and discrete-event simulation to allow flexibilities in simulating evacuation scenarios. Its hybrid space of the simulation environment overcomes the limitation of cellular space in cell-based evacuation models. We construct the model by using the Parallel DEVS formalism and develop algorithms for the corresponding DEVS simulators. This modeling approach achieves efficient event-based scheduling by executing only necessary agent interactions. Therefore, this approach has a low computational costs and high degree of scalability compared with traditional approaches.

1 INTRODUCTION

Transportation evacuation is an important issue concerning many researchers in homeland defense preparation and disaster emergency management. Underestimation of this issue and ineffective planning will result in catastrophic outcomes. In 2005, when New Orleans was invaded by Hurricane Katrina and the Houston area was threatened by Hurricane Rita, both evacuations are characterized by tremendous traffic jams and chaos. Due to its low cost and speed, computer simulation has been an effective experimental means for evacuation planning and management. A real-time simulation model can help predict traffic conditions during evacuation and identify bottlenecks of transportation networks. Therefore, a good simulation model can assist the development of a well-coordinated evacuation plan, which could save lives and properties.

In transportation simulation, agent-based modeling methods have been applied to investigate emergent evacuation situations (Lammel and Nagel 2009; Zhang, Ukkusuri, and Chan 2009; Hackney and Marchal 2009). These agent-based simulation models are able to capture the evacuation process and traffic dynamics. However, they generally employ a time-step based updating method in which each agent updates its state at a fixed time step so that the simulation can run in a continuous space. Time-step based method is easy to implement at the cost of high computational burden. Selecting a suitable time step is therefore important (Hu, Muzy, and Ntaimo 2005). Large time step may results in incorrect simulation results (i.e., missing events that should have occurred within two time steps), while a too small time step can result in unnecessary updates, and consequently, incur a high computational cost.

Different from the time-step based method, a discrete-event simulation model executes based on events and only updates its system state when necessary (i.e., occurrences of events). Nothing is changed between events, and therefore any unnecessary state update can be avoided to save computational time.

The discrete-event system specification (DEVS) provides a modular, hierarchical modeling and simulation framework derived from mathematical dynamical system theory (Zeigler, Praehofer, and Kim 2000). This event-based modeling approach enables the development of efficient simulation in the fields of transportation and emergent management (Ntaimo et al. 2004; Wainer 2006; Sun and Hu 2008). Most of the DEVS-based works use a cellular space, in which the space is discretized by a grid of cells. Although cellular space is a popular way of representing and analyzing traffic flow models, it has some limitations. It is not always easy to decide the size of cells. For example, if the cell size is too small, a large number of cells will be kept in memory. Furthermore, information diffusion and agent movements can be restricted if the Von Neuman (i.e., 4 neighbors) or Moore (i.e., 8 neighbors) topology of the cellular space is used (Muller 2009). Finally, the cell-based DEVS models lack flexibilities in modeling people's behaviors and decision-making process during the evacuation.

In this paper, we integrate the agent-based simulation approach and discrete-event simulation approach (AB-DES) to develop a hybrid space modeling approach for transportation evacuation simulation. We integrate flexible-structured cells with a coordinate-based continuous space in the hybrid space. This can better represent real traffic network and reduce the number of cells in the model, thus increasing scalability. We build agent model with decision rules to simulate people's behavior during evacuations. In our model, agents can move towards any direction as desired. This overcomes the limitations of restricted movement in cellular space of the cell-based DEVS. We extend the DEVS to the hybrid space and express the model in the Parallel DEVS formalism. This AB-DES modeling approach achieves efficient event-based scheduling by executing only necessary agent interactions.

The rest of this paper is organized as follows. Section 2 describes the modeling approach and the detailed structure of each atomic model. In Section 3, we construct the atomic models in Parallel DEVS formalism. Section 4 provides algorithms for the DEVS simulators. Section 5 concludes the paper and presents future work.

2 MODELING APPROACH

This section introduces the framework of the AB-DES approach. Section 2.1 discusses the design issues regarding the movement of agents over a hybrid space. Sections 2.2 and 2.3 provide details of the structure of each component.

2.1 System Framework

Figure 1 shows the model framework that integrates agents and hybrid space model for transportation simulation. This system is composed of traveler agents, a simulation space, and a simulation coordinator. The simulation space includes two components: a coordinate-based continuous space and a transportation network composed of roads and intersections. Each road or intersection is modeled as one entity. Roads and intersections are linked by directed ports as illustrated in Figure 1. Each traveler is one agent with its own decision rules and movement rules. A traveler may be riding on a road, passing an intersection, or traveling somewhere off roads. The Simulation Coordinator is the central control module of the simulation. It handles requests and sends commands to its subordinate. Therefore, when a traveler is on a road or intersection, dynamic coupling between the traveler and the road or intersection is maintained by the Simulation Coordinator so that they can exchange information and interact with each other. When the traveler is starting off road, it will be directed to the nearest road or intersection.

2.2 Transportation Network Modeling

The transportation network module is a coupled model composed of roads and intersections. Roads and intersections are atomic models linked by directed ports. Figures 2 (a) and (b) show the schematic view and logical structure of a single-lane road, respectively. Each lane within the road is represented by one pair of input port and output port, which are connected to corresponding intersections. Additional roads with multiple lanes can be easily built by adding more pairs of ports. These ports can be possessed by on-

ly one traveler at any time. The road model is coupled with the intersections during the initialization of the transportation network. When the simulation is running, the road model is dynamically coupled with the travelers by the Simulation Coordinator. In Figure 2(b), the "queryState" and "outState" ports are built for control purposes. When queried, each road reports its state to the Simulation Coordinator.

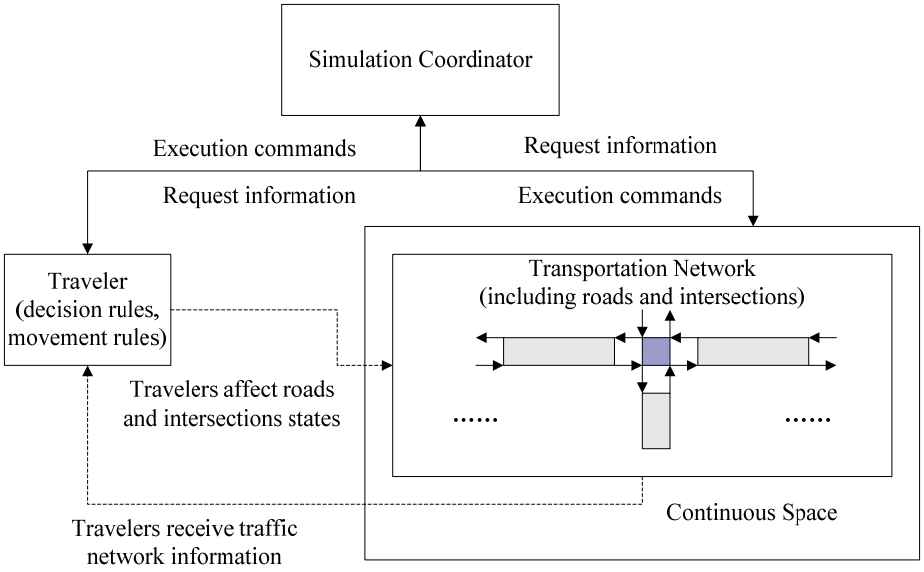

Figure 1: Model framework with agents and hybrid space

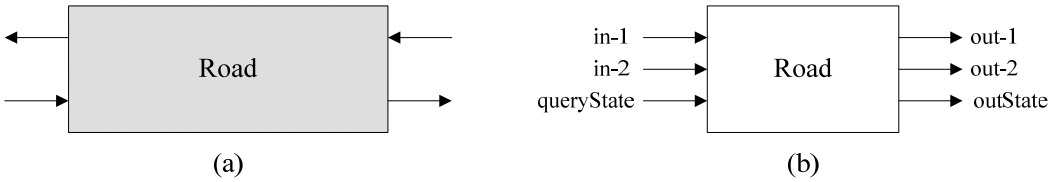

Figure 2: Structure of road atomic model

Figures 3 (a) and (b) show the schematic view and logical structure of an intersection linking four road segments. Although this framework does not restrict the number of roads to which an intersection can connect, most real-world situations require only two to four roads. The intersection model determines travelers' passing priorities based on their arrival times, coming ports and leaving ports. Similar to the road model, an intersection model can also send its state information to the Simulation Coordinator.

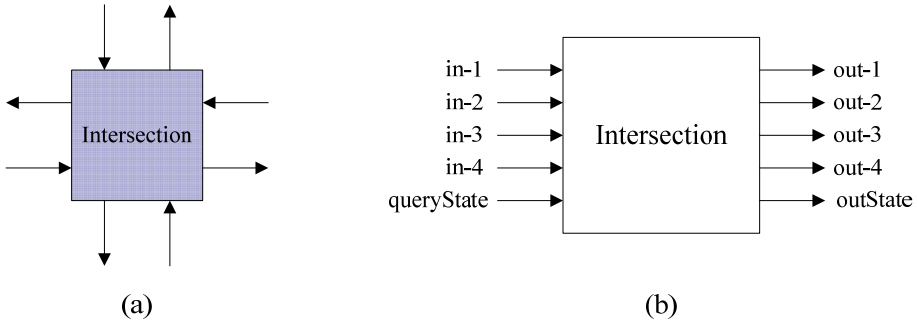

Figure 3: Structure of intersection atomic model

2.3 Traveler Agent Modeling

Different from roads and intersections which are entities, each traveler is modeled as an autonomous agent with the abilities to decide its actions and to perceive the environment. Before an evacuation starts, a traveler needs to decide its destination and evacuation route computed by a shortest path algorithm. After that, the traveler starts from its home and enters the transportation network. Since its home may not locate right at a main road, the traveler first needs to proceed to the nearest main road through a private path (such as a drive way) at a constant speed. Upon reaching a main road, the traveler can gradually speed up if the traffic condition allows. The travel time on each road segment is computed using link performance function based on the congestion level of the road (Sheffi 1985).

Figure 4 shows the structure of the traveler agent model. Besides being queried and reporting its state to the Simulation Coordinator, a traveler can also receive travel times from roads or intersections to determine the occurrence time of its next action. When a traveler enters a new road, it sends update information to the Simulation Coordinator which will remove old couplings and add new couplings between the traveler and the new road.

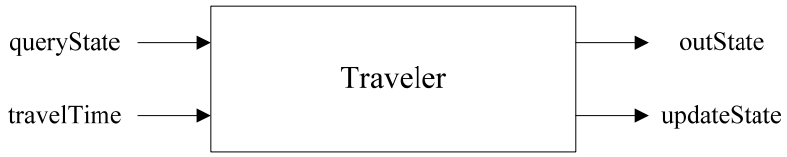

Figure 4: Structure of traveler agent

The traveler agent has predefined movement rules to handle several movement scenarios in the hybrid simulation space. The first scenario is entering a road and moving within the road as shown in Figure 5. We assume that a traveler is moving from left to right. In order to make the model more realistic, we separate the traveler's moving state within a road into three phases: accelerating, constant-speed traveling, and decelerating. Three sequential events will be scheduled to model the traveling process, which includes ending acceleration, starting deceleration, and leaving the road. During the scheduling, the traveler needs to check the states of other travelers on the road. For example, the traveler cannot reach the constant traveling phase earlier than its direct front neighbor, otherwise there will be a collision. Similarly, the scheduled leaving time will be delayed if the leaving port of the road is already possessed at the scheduled time. If the road is too short for the traveler to fully accelerate, the travel process will not be separated into three phases and the traveler will use a relatively low speed to pass the road.

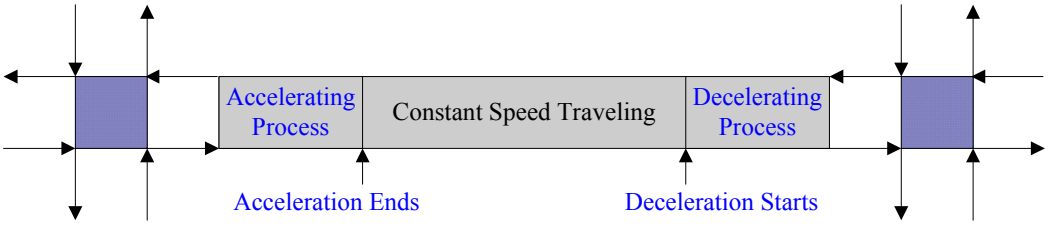

Figure 5: Moving within a road

Between leaving previous road and entering the next road, a traveler must seize the right of way before passing an intersection. When a traveler is coming to an intersection, the time to enter its next road must be determined. If the intersection is currently occupied by another traveler, the scheduled passing time will be recalculated. If multiple travelers from different ports arriving at a very close time and they have the same leaving port, their arriving ports will be used to determine the right of way.

If a traveler is starting from somewhere off road, it will be directed to the nearest road and will also request the Simulation Coordinator to add a dynamic coupling to the road (Figure 6). The traveler needs

to predict the positions of other travelers on the road at current time to find an appropriate gap to enter the road. The prediction method will be described in Section 4. Because we apply three phases to describe the traveling process, the prediction can be more accurate.

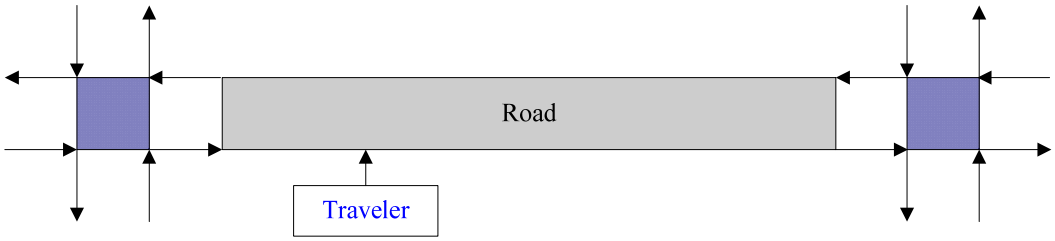

Figure 6: Entering a road

3 DEVS EXPRESSION

In this section, we express the model in terms of the Parallel DEVS formalism (Zeigler, Praehofer, and Kim 2000). A basic Parallel DEVS is defined in the following:

$$DEVS = (X_M, Y_M, S, \delta_{ext}, \delta_{int}, \delta_{con}, \lambda, ta)$$

where

$X_M = \{(p,v)|p \in InPorts, v \in X_p\}$ is the set of input ports and values
$Y_M = \{(p,v)|p \in OutPorts, v \in Y_p\}$ is the set of output ports and values
S is the set of sequential states
$\delta_{ext} : Q \times X_M^b \to S$ is the external state transition function
$\delta_{int} : S \to S$ is the internal transition function
$\delta_{con} : Q \times X_M^b \to S$ is the confluent state transition function
$\lambda : S \to Y^b$ is the output function
$ta : S \to R_{0,\infty}^+$ is the time advance function
$Q := \{(s,e)|s \in S, 0 \le e \le ta(s)\}$ is the set of total states

To define the expression in DEVS, we introduce the following additional variables: $phase$ is a control state used to keep track of the full states in the DEVS models; tl is the time instant that the last event occurred; tn is the time instant scheduled for the next event and $tn = tl + ta(s)$; t is the global current simulation time; e is the elapsed time since the last event and $e = t - tl$; σ is the time left to the next event and $\sigma = tn - t$.

3.1 Expressing Road Model in DEVS

The road atomic model is defined using the Parallel DEVS as follows:

$X_M = \{(p,v)|p \in InPorts, v \in X_p\}$ with $InPorts$ = {"in-1", "in-2", "queryState"}, $X_{\text{in-i}}$ = {TravelerID}, and $X_{\text{queryState}}$ = {queryValue};

$Y_M = \{(p,v)|p \in OutPorts, v \in Y_p\}$ with $OutPorts$ = {"out-1", "out-2", "outState"}, $Y_{\text{out-i}}$ = {TravelerID, leavingTime/NULL}, and Y_{outState} = {outValue};

$\delta_{ext}((phase, \sigma, inport), e, (p,v)) =$
$(phase, \sigma, p, value(v))$ if $p =$ queryState
$(phase, \sigma, p, leavingTime/NULL)$ if $p =$ in-i
$\delta_{int}(phase, \sigma) = (phase, \infty)$

$\lambda(phase, \sigma, inport, leavingTime/NULL) =$
$\quad$ (outState, $value(v)$) if $inport =$ queryState
$\quad$ (out-i, $leavingTime/NULL$) if $inport =$ in-i
$\delta_{con}(s, ta(s), x) = \delta_{int}(\delta_{ext}(s, 0, x))$
$ta(phase) = \infty$

When a traveler requests to enter a road, it receives a leaving time based on the road's current state, or a NULL message if the road is too congested to admit it. In the case of denied admission, the traveler will then make the entering request again after a short time delay.

3.2 Expressing Intersection Model in DEVS

The intersection atomic model is expressed as follows:

$X_M = \{(p, v) | p \in InPorts, v \in X_p\}$ with $InPorts$ = {"in-1", "in-2", "in-3", "in-4", "queryState"}, $X_{\text{in-i}}$ = {TravelerID, nextRoad}, and $X_{\text{queryState}}$ = {queryValue};

$Y_M = \{(p, v) | p \in OutPorts, v \in Y_p\}$ with $OutPorts$ = {"out-1", "out-2", "out-3", "out-4", "out-State"}, $Y_{\text{out-i}}$ = {TravelerID, leavingTime/NULL}, and Y_{outState} = {outValue};

$\delta_{ext}((phase, \sigma, inport), e, (p, v)) =$
$\quad$ $(phase, \sigma, p, value(v))$ if $p =$ queryState
$\quad$ $(phase, \sigma, p, outport, leavingTime/NULL)$ if $p =$ in-i
$\delta_{int}(phase, \sigma) = (phase, \infty)$
$\lambda(phase, \sigma, inport, outport, leavingTime/NULL) =$
$\quad$ (outState, $value(v)$) if $inport =$ queryState
$\quad$ (out-i, $leavingTime/NULL$) if $inport =$ in-i and $outport =$ out-i
$\delta_{con}(s, ta(s), x) = \delta_{int}(\delta_{ext}(s, 0, x))$
$ta(phase) = \infty$

Similar to the road model, when a traveler requests to pass an intersection, it receives a leaving time based on the intersection's current state, or a NULL message if the intersection's right of way has already been possessed by another traveler at that time. The traveler will check the intersection again after a short time delay.

3.3 Expressing Traveler Agent Model in DEVS

Before expressing the traveler agent model in Parallel DEVS, we first introduce the following notations: "AC", "CS", and "DC" denote, respectively, the accelerating phase, constant speed traveling phase, and decelerating phase. "onRoad", "onInt", "offRoad" indicate that a traveler is currently on a road, on an intersection, or off roads, respectively. Δt_{AC} and Δt_{DC} are the times that the traveler spends in the accelerating phase and decelerating phase, respectively. c is the amount of time for a traveler to pass an intersection under normal situation. Other notations and variables are self-explainable by their names. The traveler agent model can be expressed in DEVS as follows:

$X_M = \{(p, v) | p \in InPorts, v \in X_p\}$ with $InPorts$ = {"travelTime", "queryState"}, $X_{\text{travelTime}}$ = {RoadID/IntID, leavingTime/NULL}, and $X_{\text{queryState}}$ = {queryValue};

$Y_M = \{(p, v) | p \in OutPorts, v \in Y_p\}$ with $OutPorts$ = {"updateState", "outState"}, $Y_{\text{updateState}}$ = {add/remove/delay, RoadID/IntID}, and Y_{outState} = {outValue};

$\delta_{ext}((phase, \sigma, inport), e, (p, v)) =$
$\quad$ $(phase, \sigma, p, value(v))$ if $p =$ queryState

$$
\begin{aligned}
&(\text{"onRoad"}, c) \text{ if } p = \text{travelTime and } v = \text{(IntID, NULL)} \\
&(\text{"onRoad"}, leavingTime) \text{ if } p = \text{travelTime and } v = \text{(IntID, leavingTime)} \\
&(\text{"onInt"}, c) \text{ if } p = \text{travelTime and } v = \text{(RoadID, NULL)} \\
&(\text{"onRoad"}, travelTime) \text{ if } p = \text{passTime, } v = \text{(RoadID, leavingTime),} \\
&\qquad \text{and } travelTime < \Delta t_{AC} + \Delta t_{DC} \\
&(\text{"AC"}, \Delta t_{AC}) \text{ if } p = \text{travelTime, } v = \text{(RoadID, leavingTime),} \\
&\qquad \text{and } travelTime \geq \Delta t_{AC} + \Delta t_{DC} \\
&\delta_{int}((phase, \sigma, travelTime), e, (p, v)) = \\
&\quad (\text{"CS"}, (travelTime - \Delta t_{AC} - \Delta t_{DC})) \text{ if } phase = \text{"AC"} \\
&\quad (\text{"DC"}, \Delta t_{DC}) \text{ if } phase = \text{"CS"} \\
&\delta_{con}(s, ta(s), x) = \delta_{ext}(\delta_{int}(s, 0, x)) \\
&\lambda(phase, \sigma, inport, RoadID/IntID) = \\
&\quad (\text{outState}, value(v)) \text{ if } inport = \text{queryState} \\
&\quad (\text{updateState}, IntID) \text{ if } phase = \text{"DC"} \\
&\quad (\text{updateState}, RoadID) \text{ if } phase = \text{"onInt"} \\
&\quad (\text{updateState}, (addPort, RoadID)) \text{ if } phase = \text{"offRoad"} \\
&ta(phase, \sigma) = \sigma
\end{aligned}
$$

4 DEVS SIMULATOR

The DEVS formalism provides a hierarchical simulator and allows the use of four types of messages in the communications between simulators as shown in Figure 7 (Zeigler, Praehofer, and Kim 2000). When the simulation starts, the parent coordinator sends initialization messages (i, t) to its subordinates. During the simulation, the parent coordinator sends internal state transition messages (*, t) to its subordinates to schedule events and input messages (x, t) to trigger external events. The subordinates send output messages (y, t) to their parents to cause output events.

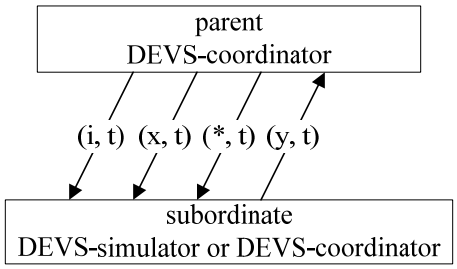

Figure 7: DEVS simulator protocol

The DEVS simulator for road model is shown in Algorithm 1. Upon the receipt of an entering request from a traveler, the road needs to check whether there is enough space to hold the requesting traveler. The road uses the elapsed time and the current states of other travelers who are coupled to this road to estimate their positions at the current time t.

Algorithm 1: Simulator for Road

variables:
    t: the global current simulation time
    h: space threshold for entering a road
when receive i-message (i, t) at time t
    $S = S_0$
when receive x-message (TravelerID, t) at time t

```
    check space
    if no space
        send y-message ((TravelerID, NULL), t) to coordinator
    else
        compute travel time based on current traffic condition
        send y-message ((TravelerID, leavingTime), t) to coordinator
    end if
```

The DEVS simulator for intersection model is shown in Algorithm 2. When receiving a request from a traveler, the intersection obtains the traveler's leaving port based on its next road segment and replies a leaving time to the traveler when there are no conflicts. If multiple travelers make requests simultaneously, the intersection needs to dispatch the right of way based on the arriving ports and leaving ports of these requesting travelers (for example, left-turn travelers yield to the ones going straight). The intersection then admits one traveler and sends NULL messages to the other travelers.

Algorithm 2: Simulator for intersection

```
variables:
    t: the global current simulation time
    c: the time to pass an intersection
when receive i-message (i, t) at time t
    S = S0
when receive x-message ((TravelerID, nextRoad), t) at time t
    check conflicts
    if no conflict
        send y-message ((TravelerID, leavingPort, leavingTime), t) to coordinator
    else
        send y-message ((TravelerID, leavingPort, NULL), t) to coordinator
    end if
```

The DEVS simulator for the traveler agent model is shown in Algorithm 3. When receiving a NULL message from a road or intersection, the requesting traveler is blocked at its current intersection or road due to the space limit. If this also causes delay to subsequent travelers, messages will be sent to the Simulation Coordinator to reschedule the events of those affected travelers. When traveling on a road, each scheduling of tn will be checked with the traveler's direct front neighbor to guarantee no collision.

Algorithm 3: Simulator for Traveler

```
variables:
    t: the global current simulation time
    tl: the time when last event occurred
    tn: the time scheduled for next event
    c: the time to pass an intersection
    v: current travel speed
    VL: a low travel speed
    VH: a high travel speed, usually the maximal legal speed
    VP: the travel speed to pass an intersection
    Δt_AC: the time to accelerate from VP to VH
    Δt_DC: the time to decelerate from VH to VP
when receive i-message (i, t) at time t
    S = S0
when receive x-message ((RoadID, NULL), t) at time t
```

```
    S = onInt
    send y-message ((delay, preRoad), t) to coordinator
    tl = t; tn = tl + c
when receive x-message ((RoadID, leavingTime), t) at time t
    send y-message ((remove, curInt), t) to coordinator
    send y-message ((add, RoadID), t) to coordinator
    if leavingTime – t < Δt_AC + Δt_DC
        S = onRoad; this.v = V_L
        tl = t; tn = leavingTime
    else
        S = AC
        tl = t; tn = tl + Δt_AC
    end if
when receive x-message ((IntID, NULL), t) at time t
    S = onRoad
    send y-message ((delay, currentRoad), t) to coordinator
    tl = t; tn = tl + c
when receive x-message ((IntID, leavingTime), t) at time t
    send y-message ((remove, curRoad), t) to coordinator
    send y-message ((add, IntID), t) to coordinator
    S = onInt
    tl = t; tn = leavingTime
when receive *-message (*, t) at time t
    if t != tn
        ERROR: bad synchronization between coordinator and simulator
    end if
    if S = offRoad
        send y-message ((addPort, nearestRoad), t) to coordinator
        send y-message ((TravelerID, t, dynPort), t) to enter a road
    else if S = AC
        S = CS
        tl = t; tn = leavingTime - Δt_DC
    else if S = CS
        S = DC
        tl = t; tn = leavingTime
    else if S = DC
        send y-message (IntID, t) to coordinator
    else if S = onInt
        send y-message (RoadID, t) to coordinator
    end if
```

Algorithm 4 describes the DEVS simulator for the Simulation Coordinator. The Simulation Coordinator module serves as the root coordinator in the Parallel DEVS formalism. It is the parent of all the travelers, roads and intersections. After initialization, the Simulation Coordinator sends internal state transition messages to its children (e.g., travelers) to execute the simulation. When receiving an output message (y_d, t) from child d, the coordinator first determines a set of receivers, converts the message to input messages using $Z_{d,r}(y_d)$, and then sends the input messages to the children in the receiver set.

Algorithm 4: Simulator for Coordinator

variables:

```
    t: the global current simulation time
    tl: the time when last event occurred
    tn: the time scheduled for next event
    child: direct subordinate DEVS simulator
t = t0
send initialization message (i, t) to child
sort event-list according to tn of its child
t = tn of its child
loop
    send (*, t) message to child
    t = tn of its child
    when receive (yd, t) message with output yd from child d
        receivers = {receive set}
        for each r in receivers
            send (xr, t) message with input value xr = Zd,r(yd) to r
        end for
until end of simulation
```

5 CONCLUSION

In this paper, we present an AB-DES modeling approach on a hybrid-space environment for transportation evacuation simulation. The framework incorporates agents with people's evacuation behavior and includes a flexible-structured hybrid-space transportation network. This approach overcomes the limitation of restricted movements for agents in a cellular space and allows a higher degree of flexibility in simulating transportation evacuation scenarios. We construct the AB-DES model in the Parallel DEVS formalism. We also provide algorithms for the DEVS simulators. This AB-DES employs an efficient event-based scheduling method to avoid unnecessary time-step updates, which are needed in conventional agent-based models. This reduces computational time and increases scalability. Future work of this research includes testing the performance of the AB-DES model under different scenarios, especially for extremely congested conditions. We will also consider modeling and implementing more evacuation behaviors such as flexible routing for travelers with different evacuation strategies.

ACKNOWLEDGMENTS

This work is supported by NSF project Collaborative Proposal: DRU: Incorporating Household Decision Making and Dynamic Transportation Modeling in Hurricane Evacuation: An Integrated Social Science-Engineering Approach, with project No. 0826874.

REFERENCES

Lammel, G., and K. Nagel. 2009. "Multi Agent Based Large-scale Evacuation Simulation." In *Proceedings of the 88th Annual Meeting of the Transportation Research Board*. Washington, D.C.

Hackney, J., and F. Marchal. 2009. "A Model for Coupling Multi-Agent Social Interactions and Traffic Simulation." In *Proceedings of the 88th Annual Meeting of the Transportation Research Board*. Washington, D.C.

Hu, X., A. Muzy, and L. Ntaimo. 2005. "A Hybrid Agent-Cellular Space Modeling Approach for Fire Spread and Suppression Simulation." In *Proceedings of the 2005 Winter Simulation Conference*, edited by M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, 248-255. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Muller, J. P. 2009. "Towards a Formal Semantics of Event-Based Multi-agent Simulations." *Multi-Agent-Based Simulation IX* 5269: 110-126.

Ntaimo, L., B. P. Zeigler, M. J. Vasconcelos, and B. Khargharia. 2004. "Forest Fire Spread and Suppression in DEVS." *Simulation-Transactions of the Society for Modeling and Simulation International* 80(10): 479-500.

Sheffi, Y. 1985. *Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods*. Prentice-Hall, Inc.

Sun, Y., and X. Hu. 2008. "Partial-Modular DEVS for Improving Performance of Cellular Space Wildfire Spread Simulation." In *Proceedings of the 2008 Winter Simulation Conference*, edited by S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, and J. W. Fowler, 1308-1046. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Wainer, G. 2006. "ATLAS: A Language to Specify Traffic Models Using Cell-DEVS." *Simulation Modeling Practice and Theory* 14(3): 313-337.

Zeigler, B. P., H. Praehofer, and T. G. Kim. 2000. *Theory of modeling and simulation*. 2nd ed. New York: Academic Press.

Zhang, B., S. Ukkusuri, and W. K. Chan. 2009. "Agent-Based Modeling for Household Level Hurricane Evacuation." In *Proceedings of the 2009 Winter Simulation Conference*, edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, 2778-2784. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

AUTHOR BIOGRAPHIES

BO ZHANG is a Ph.D. student in the Department of Industrial and Systems Engineering at Rensselaer Polytechnic Institute. He received a bachelor's degree and a master's degree in computer science from Northeastern University, China. His research interests include agent-based simulation, discrete-event simulation and their applications in transportation networks. His e-mail address is zhangb5@rpi.edu.

WAI KIN (VICTOR) CHAN is an Assistant Professor of the Department of Industrial and Systems Engineering at the Rensselaer Polytechnic Institute, Troy, NY. He holds a Ph.D. in industrial engineering and operations research from University of California, Berkeley. He has received the 2006 Pritsker best Ph.D. thesis award, the 2007 NSF CAREER Award, the 2007 and 2008 IEEE CASE Best Paper Award finalists, and the 2010 INFORMS Service Science Best Paper Award. His research interests include discrete-event simulation, agent-based simulation, and their applications in energy markets, social networks, service systems, transportation networks, and manufacturing. His e-mail address is chanw@rpi.edu.

SATISH UKKUSURI is an Associate Professor in the School of Civil Engineering at Purdue University. He holds a Ph.D. in civil engineering from University of Texas at Austin. He has received the CUTC/ARTBA New Faculty Award for outstanding research in 2011. He is the Chair of the Intelligent Transportation Systems SIG at INFORMS and a member of the network modeling committee at Transportation Research Board. His research interests include transportation network modeling, real time operations, dynamic traffic assignment, emergency management issues, logistics and freight transportation and network science. His e-mail address is sukkusur@purdue.edu.