Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

A HYBRID AGENT-CELLULAR SPACE MODELING APPROACH FOR FIRE SPREAD AND SUPPRESSION SIMULATION

Xiaolin Hu

Georgia State University
Department of Computer Science
34 Peachtree Street, Suite 1438
Atlanta, GA 30303, U.S.A.

Alexandre Muzy

Università di Corsica
Laboratory SPE UMR CNRS 6134
Campus Grossetti, BP 52
20250 Corti, FRANCE

Lewis Ntaimo

Texas A&M University
239C Zachry Engineering Center
College Station
TX 77843-3131, U.S.A.

ABSTRACT

This paper presents a first effort to integrate DEVS-based cellular space models with agent models using dynamic structure DEVS for the simulation of forest fire spread and suppression. The main focus is on the interaction between mobile agents (such as fire fighters, air-tankers) and forest cells. DEVS models' dynamic structure modeling capability is applied where couplings between mobile agents and forest cells are dynamically added and removed when the agents move in the cellular space. Two methods, a time-based method and an event-based method, are discussed to update a mobile agent's position in a cellular space. A system architecture is presented and a prototype fire spread and suppression example is implemented. Based on this initial work, we hope to understand more about the nature of this hybrid agent-cellular space approach and to apply it to the modeling and simulation of forest fires and other ecological applications.

1 INTRODUCTION

The ecological problem of wildfires is concerned with propagation processes that are distributed in both time and space. Therefore, the study of fire spread often requires developing simulation models that consider the system evolution in both time and space. Hence, such simulation models are generally of a large-scale nature and are challenging to be efficiently simulated. Furthermore, once a fire starts there is an immediate need to effectively contain it. This calls for incorporating fire fighting or fire suppression modeling into the fire spread simulation models.

In forest fire simulation, Cellular Automata (CA) has been often used to discretize the space into cells. Recently, studies such as (Ameghino et al., 2001; Ntaimo et al., 2004; Muzy et al., 2005) pinpoint the need for developing new classes of CA for fire spreading applications using the Discrete EVent System Specification (DEVS) formalism (Zeigler, et al., 2000). Unlike CA, DEVS models can receive external updated information, and the fire parameters can be updated at any moment due to the continuous time nature of the discrete event specifications. Moreover, founded on the DSDEVS formalism (Barros, 1997), Dynamic Structure Cellular Automata (DSCA) (Barros and Mendes, 1997 ; Muzy et al., 2004) allows to dynamically add active cells, to remove the quiescent ones and to change couplings between cells. Hence, a more faithful modeling and a more efficient simulation can be obtained. Using DEVS to model fire spread includes developing DEVS cellular models and couple them to form a cellular space. This cellular modeling approach has the ability to effectively represent large-scale spatial dynamic phenomena for timely simulations (e.g. Ameghino et al., 2001; Muzy et al., 2002; Zeigler 2003). In the cellular approach, the actual forest is modeled as a cell-space and is divided into forest cells. Fire spread is abstracted and modeled within a forest cell using a fire spread mathematical model of choice. Such a mathematical model uses the forest cell properties such as fuel, topography, wind speed and direction as input to compute the fire spread and other parameters of interest within the cell.

Besides modeling and simulating fire spread, there is also a need to effectively model and simulate fire suppression activities. Towards that end, (Ameghino et al., 2001) has incorporated fire suppression into a cellular Cell-DEVS (Wainer and Giambiasi, 2001) model. (Ntaimo et al., 2004; Ntaimo and Zeigler, 2005) also developed cellular DEVS models of forest fire spread that include fire suppression control measures. These works model both fire spread and fire suppression using the cellular modeling approach. Thus each cell either incorporates fire suppression activities directly or is coupled to a separated

fire suppression model. Different from the above works, this paper proposes a hybrid modeling approach that integrates *cellular* DEVS models and *agent* models using dynamic structure DEVS for fire spread and suppression simulation. We consider this as an alternative approach to model fire spread and fire suppression, which can be treated as separated activities. Fire spreading is a series of ignitions progressing through a fuel bed. Fire suppression refers to the activities connected with restricting the spread of fire following its detection. For forest fire, fire suppression efforts are usually carried out by fire fighters or air-tankers. Based on this observation, our research separates the concerns of fire spread that is modeled by cellular space model from fire fighting that is modeled by agent models. A similar approach has been taken by (Duboz et al., 2004), which proposed a formalization of Multiple Agent System (MAS) using DSDEVS. During the simulation, agents and resources of the environment (contained in the cells) can be destroyed or created and couplings can be added or removed. Our work, while following the same direction, focuses more on the hybrid nature of agents and cellular models and the interactions between them.

The rest of this paper is organized as follows. In the next section design considerations for the hybrid agent and cellular space modeling approach are given. In Section 3 the system architecture of the proposed approach is presented and in Section 4 a prototype fire spread and suppression example is given. Conclusions and discussions of this research, including future research directions, are given in Section 5.

2 DESIGN CONSIDERATIONS FOR HYBRID AGENT-CELLULAR SPACE MODELING

This section discusses some design considerations that are important to support hybrid agent and cellular space modeling and simulation.

2.1 Dynamic Coupling Change between Agents and Cells

Agents are used to model fire fighters and air-tankers. In order to carry out fire fighting actions, an agent needs to know the fire spreading conditions from its environment and then take actions to affect the environment. In our approach, we allow an agent to interact with one (and only one) cell where it locates at any given time (this restriction can be relaxed to allow an agent to interact with multiple local cells.) Couplings are added between the agent and that cell. This allows the cell to send the agent fire spreading condition messages and the agent to send the cell fire fighting action messages. When an agent changes its location in the cellular space, the cell that is coupled to the agent is changed too. Thus the couplings between an agent and the corresponding cells will be dynamically added and removed during simulation. For example, when an agent moves from *cell_A* to *cell_B*, couplings between the agent and *cell_A* are removed and couplings between the agent and *cell_B* are added.

This feature of dynamic coupling change can be modeled by DEVS' dynamic structure modeling capability (Barros, 1997). In (Hu et al., 2005), four dynamic structure change operations: *addModel()*, *removeModel()*, *addCoupling()*, *removeCoupling()*, have been defined. Among them, *addModel()* and *removeModel()* allow models to be added and removed dynamically. *addCoupling()* and *removeCoupling()* allow couplings between models to be added and removed dynamically. To support dynamic couplings, a special atomic model, such as the *couplingManager* in Figure 2, is created and added into the system. This model receives position information from an agent and finds the corresponding cell where the agent locates. It invokes coupling changes if the agent has moved out from an old cell and arrived in a new cell.

Let $LX \times LY$ denote the size of each cell in the two dimensional cellular space. For an agent with position (x,y), the ID (C_ID_x, C_ID_y) of the cell that corresponds to the agent's position can be calculated as follows:

$$C\_ID_x = \text{floor}(x / LX) \quad (1)$$

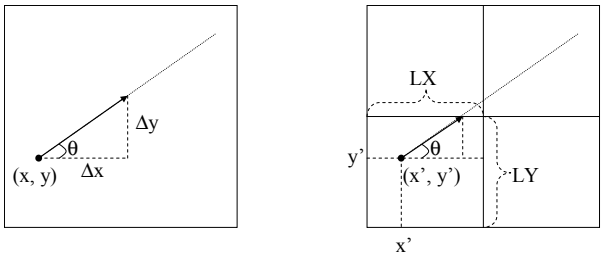
$$C\_ID_y = \text{floor}(y / LY) \quad (2)$$

For a system with multiple agents, each agent has a unique ID and is coupled to its corresponding cell. Multiple agents may occupy (thus are coupled to) the same cell. To allow the *couplingManager* to know which agent is coupled to which cell, the mapping information between *agentID* and *cellID* is maintained and updated as agents move in the cellular space.

2.2 Movement in the Cellular Space

Dynamic coupling changes are triggered by agents' movements in the cellular space. In our approach, each agent keeps track of its own position. To model an agent' movement and to update its position in the cellular space, we define an agent's position as (x,y) in a two dimension coordinate system with $(0,0)$ corresponding to the bottom left corner of the cellular space. An agent's position at time t is defined as $(x(t), y(t))$. Motion of an agent is controlled by its speed $(r(t), \theta(t))$, where $r(t)$ is the value of the speed and $\theta(t)$ is the angular heading at time t. Based on these notations, below we consider two different methods to update an agent's position in the cellular space as illustrated in Figure 1.

The first method, which we refer to as *time-based method*, ignores the existence of cells and treats the cellular space as a continuous space. A time step is defined and the agent updates its position at the end of each time step.



(a) time step (b) event step

Figure 1: Model an Agent's Movement in a Cellular Space

Assuming the time step is Δt and an agent's speed does not change during Δt, the agent's new position can be calculated based on formulas (3) and (4). After calculating its new position, an agent sends it to the *couplingManager*, which finds the corresponding cell as described in Section 2.1.

$$x(t+\Delta t) = x(t) + \Delta x \quad \text{and} \quad y(t+\Delta t) = y(t) + \Delta y \tag{3}$$

$$\Delta x = r(t)\cos\theta(t) \quad \text{and} \quad \Delta y = r(t)\sin\theta(t) \tag{4}$$

The second method, which we refer to as *event-based method*, updates an agent's position only when the agent crosses the boundary of a cell. The justification of this method is that an agent, even though is moving, stays in the same cell unless it passes the boundary of that cell to arrive in a new cell. Only at that point, dynamic coupling change needs to be invoked. In this event-based method, the time interval for position update is not constant. Let (x', y') denote the relative coordinates of an agent within its cell, the time interval *ta* for the next position update can be computed as follows. Note that the *mod* function returns the remainder from division,

$$x' = \text{mod}(x / LX) \quad \text{and} \quad y' = \text{mod}(y/LY) \tag{5}$$

$$ta = \min\left\{\frac{(LX - x')}{r(t)\cos\theta(t)}, \frac{(LY - y')}{r(t)\sin\theta(t)}\right\} \tag{6}$$

Both of these two methods have their advantages and disadvantages. The time-based method is straightforward to be implemented. Furthermore, in this method an agent is largely decoupled from the cellular space model. Thus an agent can be designed and tested "independently" from the cellular space model. But this method is dependent on a suitable time step. As will be discussed in the next section, an inappropriate time step may result in bad simulation performance or even incorrect simulation results. In the event-based method, an agent is closely coupled to the cellular space because the computation of *ta* is dependent on the dimensions of cells and the agent's relative coordinates (x', y') in the cell. However, this method handles time more efficiently because the simulation proceeds based on events that indicate an agent is crossing from one cell to another. Thus this approach is superior to the time-based approach for applications, such as the forest fire application, characterized as temporal and spatial heterogeneity.

While in the above two methods, an agent can move freely in the space, there are other methods that treat the space as a discrete space, *i.e.*, only discrete locations or discrete directions are allowed for an agent to move. For example, (Batty and Jiang, 1999) considers eight discrete directions {0, (1/4)π, (1/2)π, (3/4)π, π, (5/4)π, (3/2)π, and (7/4)π} along which an agent can move. This approach needs some approximation functions to be defined for an agent to find the right direction in each step. (Dijkstra et al., 2000) considers each cell as a discrete dot thus the cellular space is essentially defined as a grid. An agent can move from one point to another point in this grid.

2.3 Different Time Resolution for Simulation

One characteristic of this hybrid agent-cellular space modeling approach is that agents and cellular models may operate in different time resolutions. This is true for the fire spread and suppression application that we consider in this paper. For example, the speed of a typical fire spreading could be *1m/s*. However, a fire fighting agent, such as an air-tanker, may move in the speed of *10m/s*, which is *10* times as fast as the speed of fire spread. Thus during simulation, the time for an agent to move across a forest cell is one tenth of that for fire spreading. This issue of multi-resolution time scales is common for many ecological systems (Ball, et al., 1996).

In discrete time simulation, models' states are updated based on a time step. The time-based method to update an agent' position (described in the previous section) belongs to this category. Thus for this method, selecting a suitable time step is important. If the time step is too small, the change of an agent's position will be insignificant in one time step. This results in unnecessary computation. However, if the time step is too large, an agent may move across multiple cells in one time step. This may result in incorrect simulation, since, for example, the agent may bypass a fire spot that should be suppressed. Different from discrete-time simulation, discrete event simulation treats time in reference to events. In discrete event simulation, the model only performs calculations when it is ready to change states. There is no need to select an appropriate time step as there is an inherent synchronization in this approach, since each model will automatically be staged according to the next event time. The event-based method to update an agent's position (described in the last section) belongs to this category. This approach handles time more efficiently and has the potential to increase simulation performance.

The potential of efficient time handling and simulation can be exploited by the design of advanced simulation

engines. Since agents and cellular models operate in different time resolutions, in most simulation steps only a small subset of models (*i.e.*, the agent models) will change their states. Thus an advanced simulation engine can be developed to focus on only those "active" models in every simulation step. This is the idea behind the work of (Hu and Zeigler, 2004) where a high performance simulation engine was developed. This simulation engine exploits temporal and spatial heterogeneity of the simulation models and develops a data structure to support efficient search of the next event time.

3 SYSTEM ARCHITECTURE

3.1 The System Architecture

Figure 2 shows a system architecture that integrates agents and cellular space models for fire spread and suppression simulation. Only one agent model is shown in Figure 2. However, it can be easily expanded to include multiple agents. The forest is modeled as a two dimensional cellular space model. But this architecture can be upgraded to support three dimensional cellular space models for other ecological problems.

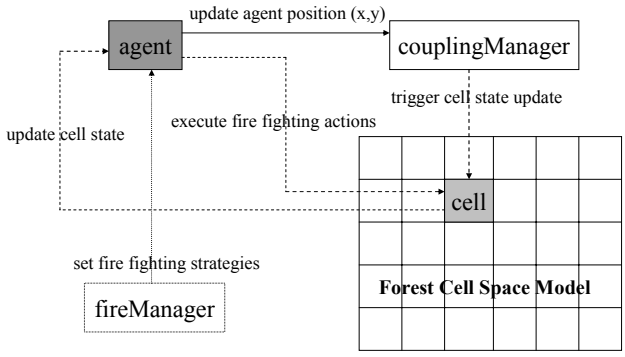


Figure 2: Architecture for Hybrid Agent and Cellular Space Simulation

This system architecture is composed from four loosely coupled components: *Forest Cell Space Model*, *Agent* model, *couplingManager* model, and *fireManager* model.

The *Forest Cell Space Model* is used to model the spatial state of the forest. It is defined according to the conventions of cellular automata modeling. Each cell has 8 adjacent neighboring cells (the Moore neighborhood) surrounding it. A forest cell has states such as *unburned, burning,* and *burned*. It transits from one state to another based on external or internal events. For example, an *unburned* cell, if ignited, may transit to the *burning* state, and then after a period of time may transit to the *burned* state. The time for a cell to stay in the *burning* state can be computed based on a fire spread algorithm, which takes into account factors such as fuel type, wind speed, and fire fighting efforts. Fire in one cell may spread to its neighboring cells. The fire spread speed along a particular direction can be calculated based on some mathematic fire spreading models such as Rothermel's model (Rothermel, 1972). An implementation of a similar cellular space model in the DEVS formalism can be found in (Ntaimo, et al., 2004). In this implementation, couplings are added between a cell and all its surrounding cells. To model fire spread from one cell to another, messages are sent between cells. This model is expanded in our architecture in several ways. For example, a new port, such as the *queryState* port shown in Figure 3, is added for each cell. Whenever a cell receives a message on this port, it sends out a message that contains its current state. Also, a cell will send out its state whenever it transitions to a new state. This allows the agents coupled to this cell to know the current state of the cell.

The *Agent* model is used to model fire fighters or air-tankers. An agent can move in the cellular space with a certain speed. During the movement, it keeps track of its own position and updates it based on moving speed and direction (see Section 2.2). Meanwhile, an agent continuously monitors the condition (state) of its corresponding cell and, if necessary, takes fire suppression actions based on some pre-defined rules. To support the interactions between an agent and the corresponding cell, couplings are added between them. This allows a cell to update its state to the agent and the agent to send fire fighting commands to the cell. When an agent moves in the cellular space, couplings are dynamically added and removed as described in Section 2.1.

The *couplingManager* model takes care of coupling changes when an agent moves in the cellular space. It receives messages that contain the agent's (new) positions (*x,y*) from the agent. This message triggers the *couplingManager* to find the cell where the agent locates. If the cell ID has changed, couplings between the agent and the old cell will be removed and couplings between the agent and the new cell will be added. Furthermore, a coupling will be added from the *couplingManager* to the new cell. This coupling allows the *couplingManager* to inform the new cell to send out its current state. Thus whenever an agent is coupled to a new cell, it will receive an "initial" message from the cell that contains the cell's current state. These couplings that are dynamically added/removed are represented in dashed lines in Figure 2. Using the structure change operations developed in (Hu, et al., 2005), the *couplingManager* can execute the following code to remove a coupling from an agent to an old cell and to add a coupling from the agent to a new cell:

```
removeCoupling( agent , "ffAction", oldCell,
"inFireFight");
addCoupling(agent , "ffAction", newCell,
"inFireFight");
```

The fourth part of this architecture concerns the *fireManager* that is represented in the dotted box in Figure 2. During the process of fire suppression, an agent may receive high-level commands from the *fireManager*, whose role is to allocate fire fighting resources and set fire fighting strategies from the global point of view. In order for the *fireManager* to make timely and optimized decisions, information about fire spread and fire suppression will be continuously updated to the *fireManager* from cellular models and agents. The couplings that support this information update are not shown in Figure 2. Based on this information, the fire manager assesses the current situation and makes a decision. To account for the stochastic nature of fire spread and fire suppression, stochastic programming techniques such as the stochastic server location problem studied by (Ntaimo, 2004) may be applied in order to reach an optimized decision. This fire manager model has not been implemented in our prototype example described in Section 4.

3.2 Towards a Formal Specification

It is desirable to provide a formal specification for the above architecture and all its components. Such a formal specification would provide a clear definition of the system and leave out any possible misinterpretations. Towards this goal, this section presents some previous works from the specification point of view. We discuss the relationship between our work and these previous works and suggest how a complete formal specification may be derived. Our discussion focuses on the specification of dynamic structure changes, the specification of cellular space models, and the specification of agent models. We leave the specification of fire manager for future research.

Barros (Barros, 1997) provides a formal specification for dynamic structure DEVS where the ability to initiate structure changes is supported by a central network executive. Based on Barros' structure, (Duboz et al., 2004) recently proposed a formalization of Multi-Agent Systems (MASs) using DSDEVS. In this work, a MAS network is described by a structure $MAS = < X_{MAS}, Y_{MAS}, \chi, M_\chi >$. X_{MAS} is the set of MAS input events, Y_{MAS} is the set of MAS output events, χ is the name of a special atomic model called the executive model, and M_χ is the executive model. The executive model is a special atomic model described by:

$$< X_\chi, Y_\chi, S_\chi, \gamma_\chi, \Sigma^*_\chi, \delta_{ext,\chi}, \delta_{int,\chi}, \delta_{conf,\chi}, ta_\chi, \lambda_\chi >.$$

X_χ and Y_χ are the input and output sets of the executive model, S_χ is the set of states, $\gamma_\chi : S_\chi \to \Sigma^*_\chi$ is the structure function, Σ^*_χ is the set of network structures, and $\delta_{ext,\chi} : Q_\chi \times X^b_{\chi^b} \to S_\chi$, $\delta_{int,\chi} : S_\chi \to S_\chi$ and $\delta_{conf,\chi} : S_\chi \times X^b_{\chi^b} \to S_\chi$ are the executive model's external, internal, and confluent transition functions. $Q_\chi = \{(s_\chi, e) / s \in S_\chi, 0 \le e \le t_{a\chi}(s_\chi)\}$ is the total state set, with e the time elapsed since the last state change. $t_{a_\chi} : S_\chi \to \Re^+$ and $\lambda_\chi : S_\chi \to Y^b_\chi$ are the executive model's time advance and output functions. In effect, the state of the executive model describes the network structure. Structure changes can occur when the executive model changes its state. This work of MAS bears a close relationship to the work presented in this paper. Especially the executive model is directly corresponding to the *couplingManager* in our architecture. Specifications of the structure change and the *couplingManager* model in our architecture can be derived from the ones developed in DSDEVS and MAS.

Specifications for the cellular space model can be achieved through (Wainer and Giambiasi, 2001) or (Muzy, et al., 2004). These works are based on DEVS. Furthermore, the cellular space model that we employ is derived from an existing model developed in (Ntaimo, et al., 2004). Formal specification of this model has been developed in (Ntaimo and Zeigler, 2004). A specification of our cellular space model can be derived from this formal specification by expanding it to account for the interactions with agents that move in the cellular space.

Several works have been developed that are related to the specification of mobile agents for fire suppression. For example, a specification of fire suppression was developed in (Ameghino et al., 2001) using the Cell-DEVS formalism. Recently, a DEVS specification of an atomic fire fighting model was developed in (Ntaimo and Zeigler, 2005). Unlike a mobile agent, this atomic model is coupled to a single cell and can not move in the cellular space. In (Duboz et al., 2004), mobile agents are specified as dynamic structure networks. These specifications can be described through the level of *Coupled System* specification of the System Specification Hierarchy discussed in (Zeigler, et al., 2000). Considering that we are still at an initial stage to define agents and their interactions with the cellular models, we choose to specify an agent at a more abstract level, the *I/O Behavior* level. The *I/O Behavior* specification defines a system's behavior from the input/output (blackbox) point of view. This leaves spaces for different low-level specifications and detailed implementations (the example presented in the next section shows one implementation). After we understand more about the nature of these agents, a formal specification at the *Coupled System* level will be developed.

To specify an agent at the *I/O Behavior* level, we can define an agent's behavior using a set of rules. For demonstration purpose, below we (informally) show a set of sample rules that an agent may follow. These rules are

partially derived from (Andrews 1986) and (Rothermel and Rinehard 1983). The inputs of the agent model are forest cell's state (*cell_state*), fireline intensity or flame length (*fln*). The outputs of the agent model are fire fighting actions (*ffc = {null, direct attack, indirect attack}*) and movement (*move*). With these inputs and outputs, the rules are presented below.

1. If *cell_state* is "*burning*" or "*burning_wet*" and *fln* < 2.4 meters, agent stops *move* and carries out fire fighting action using *direct attack* method
2. If *cell_state* is "*burning*" or "*burning_wet*" and *fln* ≥ 2.4 meters, agent stops *move* and carries out fire fighting action using *indirect attack* method.
3. If *cell_state* is not "*burning*" and "*burning_wet*" (meaning the cell is not burnable or already burned out), agent's fire fighting action is *null* and it moves with a pre-defined *speed* and *direction*.

4 A PROTOTYPE FIRE SPREAD AND SUPPRESSION EXAMPLE

Based on the architecture presented in Section 3, we have implemented a prototype fire spread and suppression example. This example is implemented in the DEVSJAVA modeling and simulation environment (Zeigler and Sarjoughian, 2003). Figure 3 shows the model of this prototype system with the agent locating at *Cell_0_0*. In this system, there is one agent, one *couplingManager,* and 900 (30×30*)* cell models. Note that for simplicity, Figure 3 displays only two cell models: *Cell_0_0*, and *Cell_1_0*. The fire manager model is not implemented. Furthermore, since this example is only for demonstration purpose, implementation of the *fireFightingAgent* model is very primitive. However, this implementation can be extended to account for more complex simulations.

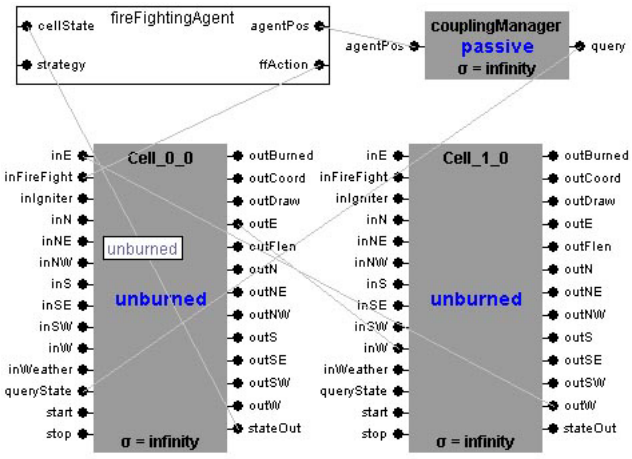


Figure 3: System Model of the Fire Spread and Suppression Example

4.1 The Cellular Space Model

The cellular space model is expanded from the cellular DEVS fire spread model developed in (Ntaimo et al., 2004). In this model, the actual forest is modeled as a cell-space and is divided into forest cells. Each cell is coupled to its eight neighbors. Figure 3 shows the couplings between *Cell_0_0* and *Cell_1_0*. Fire spread is abstracted and modeled within a forest cell as follows. First, a one-dimensional maximum rate of fire spread and direction is computed using Rothermel's mathematical model for fire spread. This model is valid under the assumption of uniform conditions within the forest cell and takes in as input the forest cell fuel properties, topography, wind speed and direction. After the maximum rate of spread and direction are determined, a decomposition algorithm is applied to calculate two-dimensional fire spread in the eight major directions: *N*, *NE*, *E*, *SE*, *S*, *SW*, *W*, *NW*. Fire spread is modeled as spreading from the center of an ignited cell towards the neighbor cells. A cell is ignited when it receives a message from a neighbor cell and its *fireline intensity* is greater than a given threshold for the forest cell fuel model to catch fire. A cell is burned out if the fire spreading on all eight directions reaches its neighboring cell centers.

Following the same approach of (Ntaimo and Zeigler, 2005), we model an agent's fire suppression effort using a random *duration*. This *duration* is determined by the agent and is passed on to the cell model. The cell model transitions to appropriate states based on the fire spreading speed and fire suppression *durations*. More description about this approach can be found at (Ntaimo and Zeigler, 2005).

4.2 The *Agent* Model

The agent model controls the agent to move in the cellular space and to take fire fighting actions if the corresponding cell is burning. In this implementation, we model the agent model, *fireFightingAgent*, as a DEVS coupled model that has two atomic model components: *agent_fireFighting* and *agent_Moving*. The *agent_fireFighting* model is responsible for fire fighting and the *agent_Moving* model is responsible for the agent's movement. Figure 4 shows this agent model.

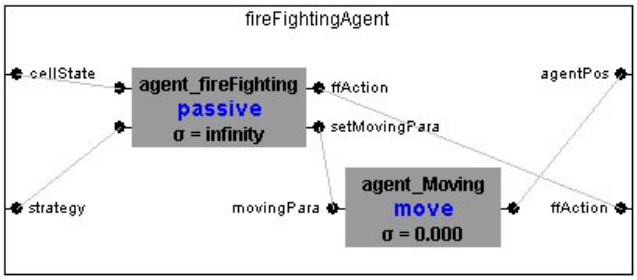


Figure 4: The Agent Model

In our implementation, the *agent_Moving* model employs a time-based method to update its position. A time step of *1s* is chosen and the agent's speed is set to *2m/s*. Note that these values are arbitrarily chosen only for demonstration purpose. The time-based method to update the agent's position will be replaced by an event-based method later. Such a replacement will only affect the *agent_Moving* model and should not affect the *agent_fireFighting* model.

Model *agent_fireFighting* follows after (Ntaimo and Zeigler, 2005) and employs the three rules described in Section 3.2 for fire suppression. Based on these rules, this simple *agent_fireFighting* model consists of three basic states: *passive*, *direct-attack*, and *indirect-attack*. The model is initialized in the *passive* state. It transitions to *direct-attack* state if rule 1 and rule 2 are satisfied. It transitions to *indirect-attack* state if rule 3 is satisfied. For both of these two states, a random *duration* is used to represent the time to perform the suppression effort. The *agent_fireFighting* model can control the agent's movement by sending a speed parameter message to the *agent_Moving* model. In our implementation, it sets the moving speed to 0 when it transits to state *direct-attack* or *indirect-attack*. Otherwise, it sets the moving speed to a predefined value with a random angular direction.

4.3 The *couplingManager* Model

The *couplingManager* model is responsible to change the couplings between the agent and corresponding cells. As described in previous sections, it takes the position information from the agent and computes the corresponding cell. If this cell is different from the old cell, couplings related to the old cell will be removed and new couplings related to the new cell will be added. The external transition function of *coouplingManager* that contains the pseudo code for dynamic coupling change is given in Figure 5.

```
public void deltext(double e,message x){
  if (somethingOnPort(x, "agentPos")) {
    newPos = Get agent's new position( );
    newCellID = Find corresponding cell (newPos);
    if(newCellID != oldCellID){
      removeCoupling( oldCell, "stateOut", agent, "cellState");
      removeCoupling( agent , "ffAction", oldCell, "inFireFight");
      removeCoupling( this,"query", oldCell, "queryState");
      oldCellID = newCellID;
      addCoupling( newCell, "stateOut", agent, "cellState");
      addCoupling(agent , "ffAction", newCell, "inFireFight");
      addCoupling(this,"query", newCell, "queryState");
      holdIn("Query", 0);
    }
  }
}
```

Figure 5: Pseudo Code for Dynamic Coupling Change

5 CONCLUSION

This paper presented the feasibility of a DEVS-based hybrid agent-cellular space modeling approach for fire spread and suppression simulation. Several design considerations of this hybrid approach are discussed. A system architecture is presented and a prototype example is implemented. As compared to the general approach of forest fire simulation where fire suppression activities are incorporated into each forest cell, this alternative approach explicitly separates the concerns of fire spread from fire suppression. It represents a more intuitive and faithful way by modeling the behaviors of fire fighters and air-tankers as mobile agents. The advantages of this hybrid approach become important as fire spread behaviors and fire suppression rules becomes more and more complex. One concern of this approach is that the simulation may become "slower" because couplings are dynamically added and removed between agents and cells. However, since the number of agents is typically much smaller than the number of cells, this influence in simulation performance is tolerable in most cases.

Future work of this research includes extending this initial work and applying it to more complex simulations. To improve simulation performance, we will design advanced simulation engines to exploit the spatial and temporal heterogeneity of this hybrid model. Meanwhile, we will research stochastic optimization techniques and integrate it into the system (such as the fire manager model). This integration of discrete event simulation and stochastic programming promises a powerful tool for fire management prediction and decision making. We note that while this paper focuses on the forest fire application, the hybrid agent-cellular space approach also has the potential to be applied to other complex spatial systems. Integration of this kind of systems will necessitate a sound formal description of agents. Special attentions will be paid to the modeling of agents' behavior and perception. To achieve this purpose (Duboz et al., 2004) can be used as a starting point.

REFERENCES

Ameghino, J., A. Troccoli, G. Wainer. 2001. Models of complex physical systems using Cell-DEVS, in *Proceedings of Annual Simulation Symposium*, Seattle, WA. 266-273.

Andrews, P. L. 1986. BEHAVE: Fire behavior predictions and fuel modeling system-BURN subsystem part 1. General Tech. Rep. INT-194, U.S. Department of Agriculture, Forest Service, Intermountain Forest and Range Experiment Station, Ogden, UT.

Ball, G.L., B.P. Zeigler, R. Schlichting, M. Marefat and D.P. Guertin. 1996. Problems of multi-resolution integration in dynamic simulation, *Third International*

Conference/Workshop on Integrating GIS and Environmental Modeling.

Barros, F. J. 1997. "Modeling Formalisms for Dynamic Structure Systems", *ACM Transactions on Modeling and Computer Simulation*, 7: 501-515.

Barros, F. J., and M. T. Mendes, 1997. "Forest fire modeling and simulation in the DELTA environment", *Simulation Practice and Theory*, 5: 185-197.

Batty, M. and B. Jiang. 1999. Multi-agent simulation: new approaches to exploring space-time dynamics in GIS. CASA Working Papers, no.10. Centre for Advanced Spatial Analysis (UCL), London, UK.

Dijkstra, J., A.J. Jessurun, and H.J.P. Timmermans. 2000. A multi-agent cellular automata system for visualising simulated pedestrian activity. In S. Bandini, and T. Worsch, ed.. Theoretical and Practical Issues on Cellular Automata, *Proceedings of the Fourth International Conference on Cellular Automata for Research and Industry*.

Duboz, R., E. Ramat, and G. Quesnel. 2004. Multi-agent systems and theory of modeling and simulation: an oprational analogy (In French), presented at *Douzièmes Journées Francophones sur les Systèmes Multi-Agents (JFSMA) - Systèmes multi-agents défis scientifiques et nouveaux usages*, Paris, 49-68.

Hu, X., B. P. Zeigler, and S. Mittal, 2005. Variable structure in DEVS component-based modeling and simulation", *Simulation*. 81 (2): 91-102.

Hu, X., B. P. Zeigler: 2004. A high performance simulation engine for large-scale cellular DEVS models", *High Performance Computing Symposium Advanced Simulation Technologies Conference*.

Muzy, A., G. Wainer, E. Innocenti, A. and Aiello, J.F. Santucci. 2002. Comparing simulation methods for fire spreading across a fuel bed, in *Proceedings of AIS'2002*, Lisbon, Portugal, 219-224.

Muzy, A., E. Innocenti, D. R. C. Hill, A. Aïello, J. F. Santucci, and P. A. Santoni, 2004. Dynamic structure cellular automata in a fire spreading application, presented at *First International Conference on Informatics in Control, Automation and Robotics*, Setubal, Portugal, 143-151.

Muzy, A., E. Innocenti, G. Wainer, A. Aïello, and J. F. Santucci, 2005. Specification of discrete event models for fire spreading, *Simulation*. 81: 103-117.

Ntaimo, L., B. Khargharia, B. P. Zeigler and M. J. Vasconcelos. 2004. Forest fire spread and suppression in DEVS, *Simulaiton*, 80 (10): 479-500.

Ntaimo, L. 2004. Decomposition Algorithms for Stochastic Combinatorial Optimization: Computational Experiments and Extensions, Ph.D. Dissertation, University of Arizona.

Ntaimo, L., and B. P. Zeigler. 2004. Expression of a forest cell model in parallel DEVS and Timed Cell-DEVS formalisms, *Proceedings of the 2004 Summer Computer Simulation Conference*, San Jose, CA.

Ntaimo, L., and B.P. Zeigler. 2005. Integrating fire suppression into a DEVS cellular forest fire spread model, *Proceedings of the Spring Computer Simulation Conference*, San Diego, CA.

Rothermel, R. 1972. A mathematical model for predicting fire spread in wildland fuels, Research Paper INT-115. Ogden, UT: U.S. Department of Agriculture, Forest Service, Intermountain Forest and Range Experiment Station.

Rothermel, R. C., and G. C. Rinehard. 1983. Field procedures for verification and adjustment of fire behavior predictions. General Tech. Rep. INT-142, U.S. Department of Agriculture, Forest Service, Intermountain Forest and Range Experiment Station.

Wainer, G., N. Giambiasi, 2001. Application of the Cell-DEVS paradigm for cell spaces modeling and simulation", *Simulation*, 76: 22-39.

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

Zeigler, B.P. and H.S. Sarjoughian. 2003. Introduction to DEVS Modeling & Simulation with JAVA: Developing Component-based Simulation Models, Technical Document, University of Arizona.

Zeigler, B.P. 2003. *Discrete event abstraction: an emerging paradigm for modeling complex adaptive systems, advances in adaptive complex systems*, edited by L. Booker, Santa Fe Institute/Oxford Press, Oxford.

AUTHOR BIOGRAPHIES

XIAOLIN HU is an assistant professor in the Department of Computer Science at Georgia State University. His research interests include discrete event system modeling and simulation, software engineering, and their application to robotic systems, agent systems, and ecological systems. His e-mail address is xhu@cs.gsu.edu.

ALEXANDRE MUZY received a Ph.D. degree (2004) from the University of Corsica. He is currently working at the CNRS research laboratory UMR 6134 of the University of Corsica. His current research interests relate to the modeling and simulation of agents and cellular models using the Dynamic Structure Discrete EVent System Specification (DSDEVS) formalism. His e-mail address is a.muzy@univ-corse.fr.

LEWIS NTAIMO is an assistant professor in the Department of Industrial Engineering at Texas A&M University. His research interests include discrete event modeling and simulation and decomposition algorithms for stochastic mixed-integer programming. His application areas of interest include forest fire spread and containment, telecommunications, stochastic facility location, and stochastic supply chain planning. His e-mail address is ntaimo@tamu.edu.