Discrete Event Front-tracking Simulation of a Physical Fire-spread Model

Jean-Baptiste Filippi, Frédéric Morandini, Jacques Henri Balbi and David RC Hill

SIMULATION 2010 86: 629 originally published online 4 August 2009
DOI: 10.1177/0037549709343117

Discrete Event Front-tracking Simulation of a Physical Fire-spread Model

Jean-Baptiste Filippi
Frédéric Morandini
Jacques Henri Balbi
UMR CNRS 6134,
Systèmes physiques pour l'environnement,
University of Corsica
Corte, France
filippi@univ-corse.fr

David RC Hill
ISIMA/LIMOS UMR CNRS 6158,
Computer Science and Modeling Laboratory,
Blaise Pascal University,
France

Simulation of moving interfaces such as a fire front usually requires resolution of a large-scale and detailed domain. Such computing involves the use of supercomputers to process the large amount of data and calculations. This limitation is mainly due to the fact that a large scale of space and time is usually split into nodes, cells, or matrices and the solving methods often require small time steps. In this paper we present a novel method that enables the simulation of large-scale/high-resolution systems by focusing on the interface and its application to fire-spread simulation. Unlike the conventional explicit and implicit integration schemes, it is based on the discrete-event approach, which describes time advance in terms of increments of physical quantities rather than discrete time stepping. In addition, space is not split into discrete nodes or cells, but we use polygons with real coordinates. The system is described by the behavior of its interface and evolves by computing collision events of this interface in the simulation. As this simulation technique is suitable for a class of models that can explicitly provide the rate of spread, we developed a radiation-based propagation model of wild land fire. Simulations of a real large-scale fire performed by implementation of our method provide very interesting results in less than 30 s with a 3-m resolution with current personal computers.

Keywords: fire spread, DEVS, simulation, asynchronous, discrete events, flame, forest, envelope, interface, front tracking

1. Introduction

Simulation of a spatial phenomenon requires the description of an environment in a way that can be efficiently processed by a computer and adapted to numerical manipulations. Typically, in a forest fire, the phenomenon evolves on an area composed of non-burnable roads, rivers, and fuel breaks and burnable large patches of uniform land. One of the main problems of this spatial distribution is that non-burnable areas have significant effects on the fire dynamics despite being several orders of magnitude smaller than the vegetative fuel areas.

In order to take all of the details into account, most simulation techniques discretize space and time in more

SIMULATION, Vol. 86, Issue 10, October 2010 629–644

DOI: 10.1177/0037549709343117
Figures 12–14 appear in color online: http://sim.sagepub.com

or less regular meshes or grids and use those meshes to solve the model. Using a mesh implies that before performing a simulation, a trade-off must be made between the resolution and the size of the domain as a computer will only be able to process a limited number of nodes or cells. For a large forest fire that will spread over an area greater than 100 km^2, at best a 2-m resolution is needed to take into account roads and fuel breaks; thus, 25 million cells are necessary, which has a impact on the simulation time and memory usage.

There are many ways to work around these limitations, such as increasing computer speed and memory, distributing calculus when it possible, or optimizing the algorithms to enhance performance.

Nevertheless, such large fires will develop in a few hours and require a forecast that can be delivered in a few minutes in order to try different fire-fighting scenarios.

The goal of the work proposed in this paper is to propose a method that is able to simulate in a few minutes the propagation of a large wildfire with high resolution.

For that, we propose to work around the limitations of current methods by using a different representation of the environment and the phenomenon.

The proposed method combines discrete event simulation (DES) [1] and front tracking [2–5].

Front-tracking methods are used to study the physical interface or boundary dynamics. A 'physical interface' is defined here as the frontier line between two states of a system.

For example, a model of a sea oil slick may be defined by two states: oil is present and oil is absent. In a forest fire model, the two states for vegetation are burned and unburned, and the fire front is the interface between these two states.

While common methods simulate the spatial evolution of the state of the system, front-tracking methods are designed to directly simulate the spatial evolution of the frontier between those two states. With this view on systems, the environment can be described in a different fashion because the interfaces can be processed as a set of points or vertices with continuous coordinates. The entire map on which the system is evolving does not need to be discrete in order to study interface dynamics. The map state can be a function of the location (a hill can be described by a sine function), as long as it is possible to evaluate a state for a given set of coordinates.

Front tracking has been seldom applied to forest fire simulation. In the proposed approach, it is coupled with DES, as opposed to discrete time simulation. If discrete time stepping is used, the resolution of the system has similar limitations to regular meshes; trade-offs have to be made between the temporal resolution and the temporal scale of the simulation. Describing front models as discrete event systems permits us to define time as a continuous value; then like the other dimensions of the system and domain, simulation then becomes asynchronous in an approach similar to that of Karimabadi et al. [6].

In the following section we present a review of related work on front tracking, asynchronous simulation, and fire-spread simulation. The overall simulation method, concepts, and detailed simulation algorithms are presented in Section 3. The proposed physical fire rate of spread (RoS) model is described in Section 4. Section 5 presents simulations and an implementation of the fire-spread model that shows that the method is appropriate to simulate in a few minutes the evolution of large wildfires at high resolution.

2. Background

Topics of interest related to this study are transversal to DES, numerical simulation, front-tracking methods, and fire propagation. The general problem that motivates the development of the method involves finding a suitable method to quickly perform a physical simulation of fire spread on a wide domain.

Fire spread models are generally classified into three families [7], namely empirical, semi-empirical, and physical. Empirical models try to capture the behavior of a front using purely statistical and stochastic methods based on observation. Semi-empirical models, such as that proposed by Rothermel [8], are the most widely used and are derived from a physical formulation of the problem, but include some parameters that are deduced (fitted) from observations. Physical models attempt to represent the behavior of a wildfire with no fitted parameters; among them, complex multiphase formulations attempt to represent both combustion and propagation using high-computational-cost computational fluid dynamics (CFD), while reduced physical models focus only on propagation.

The advantages of physical models are that more diagnoses can be derived from the simulation, such as fire intensity or emission of atmospheric pollutants. Physical models are also more generic, since empirical and quasi-empirical models can only reproduce a fire behavior that has already been observed [7]. The interested reader will find in [9] a summary of currently available forest fire models.

Simulation results of different models may not be directly comparable, for example, the RoS will be diagnostic (usually deduced from a fully resolved temperature field) in a complex physical model, while it will be prognostic (calculated directly) in reduced physical, empirical, and quasi-empirical models.

Complex models are also usually more calculation intensive, and therefore are restrained to either smaller domains or will take more time to calculate than the actual wildfire will take to propagate.

Consequently, numerical methods adapted to simulate those models can be very different, such as finite-elements methods [10], ellipse reconstruction [11], front tracking [12], discrete event system specification (DEVS)-based cellular simulation [13–15], or even cellular automata [16].

Cellular automata methods in the field of forest fires and DES have also been fields of numerous studies that have shown interesting results, but may be limited to a certain resolution in order to have sufficiently fast simulation time for larger domains. Optimization methods have been applied to overcome these limitations, most notably by keeping track of the active areas [15, 17] but the approach developed in this work is very different from cellular models as it does not require a regular mesh or cells. A complete description of the numerous cell-based models of fire spread will be out of the scope of this paper.

Most work done in the field of numerical methods of fire spread has been done with models that calculate the RoS. RoS for every portion of the front is then used to reconstruct the shape of the fire over time from an initial solution.

Among the reconstruction methods, the most commonly used is the method of ellipses where ellipses of the size of a fire advance for a given time step are drawn around an initial front to reconstruct the front one step forward. We are more interested here in the family of front-tracking methods from which the proposed method is derived.

Front tracking is the general denomination for any numerical scheme that calculates the movement of an interface between two phases of a system (burned/unburned, water/oil, etc.). Of these methods, three main approaches are relevant for the simulation of fire fronts.

In the volume of fluid method [18], the front is represented by a plane in a grid cell, and the volume of each phase is estimated by the amount of each grid cell in each part of the plane. Such methods usually represent well the topological changes of the front and are adequate to perform a budget for any physical variable of each phase. Nevertheless, as the front is approximated locally as a straight line parallel to the mesh, it is difficult to evaluate the norm vector of the front advance, which is critical in fire-spread models.

For problems where performing budgets are less critical, the most common approach is the level set method [12]. Like the volume of fluid method, the level set method requires the use of a grid to calculate the position of the interface. The main advantage of the level set method is that it directly deals with topological problems (such as front merging and convergence). An application of the level set method to forest-fire-spread model can be found in [12]. Some computationally efficient algorithms exist for the level set method, but it implies automatic refinement of the grid near the interface using a memory-intensive data structure, and therefore may limit the size of the simulated domain.

Finally, marker methods are also used for the resolution of front tracking [19]; in this method, the front is discretized by a set of points, and at each step, markers are moved according to a speed function. With this low-computational-cost Lagrangian technique, it is possible to simulate the evolution of an interface without an underlying grid to represent the state of the system.

The method proposed in this paper derives from the marker method. As stated earlier, one of the main problems in forest fire modeling is that the phenomenon is evolving on large patches of uniform condition with a few small-scale details that have strong effects on the propagation. Using this method helps overcome this problem as large uniform patches, small roads, or even large rocks can be represented as arbitrarily shaped polygons with no underlying matrix to represent the state of the system. The main problem of this method is that collision or intersection must be checked each time a marker is moved so as to take care of the topological changes.

If all markers are moved synchronously using an explicit time step, there will be as many topological checks as there are markers at every step. The originality of the proposed front-tracking markers method is to move markers asynchronously without any global time step by using DES.

In a time-driven simulation, all state changes must be performed synchronously at each time step. As the time step is a spatially uniform time interval, it is conditioned by the smallest detail that needs to be taken into account in the simulation. This limitation is known as the Courant–Friedrichs–Levy (CFL) condition $\Delta t < \Delta x / V$, where V is the fastest speed in the whole domain, and Δx is the size of the smallest detail or the size of a unitary cell in a mesh.

Specifically in the markers method, the time step will be constrained by the fastest marker at each step. In a forest fire, the speed for the same front can range from a few centimeters per second (back-fire) to a few meters per second (head-fire). In such configurations, markers are recalculated, regardless of the actual distance covered, thus resulting in many unnecessary computations (at each time step, the slower markers will be moved by a very small distance, while the faster markers will be moved by the maximum acceptable distance given by the CFL condition).

In DES, the local state change of a system is triggered by an event. Each event has an occurrence time, which does not have to be spatially uniform. If the time advance of each marker is event based, markers do not have to share the same time step. With different time advances for each marker, the CFL condition only applies locally to the given speed of individual markers. Computations become dependent on the actual distance covered, whatever the speed of the marker.

While DES is often used to simulate agents [20] and cellular models [21] that share some of the goals developed in the proposed application, it has seldom been used in the field of CFD and physical simulation.

Some recent studies are proposing DES as a numerical integration method for fluid (or field) dynamics models; in particular, Karimabadi et al. [6] propose a way to calculate an electrical field in one dimension using a DES

of particles moving through cells without a global CFL condition. Nevertheless, as DES is a totally different way of thinking, the simulation of fields or particle advances and every method such as the method of markers in two dimensions have to be reformalized.

Specific formalisms exist for the specification of DES, among these DEVS [1] has already been applied to the study of discrete, continuous, and hybrid systems. Many specializations of DEVS exist for different applications; Vector-DEVS [22] has been developed for the specification of models with moving dynamic interfaces. In the next section, Vector-DEVS is used to formalize the markers method.

3. Front-tracking Markers Method

To use the DEVS formalism, the model has to be decomposed into basic models that represent the atomic behavior of a subpart of a system and coupled models that manage the links between these basic models. One limitation of standard DEVS is that the structure of the model is static during simulation; it is not possible to add a basic model or change the coupling.

3.1 Modeling Formalism

Moving interfaces show a naturally dynamic behavior; front lines expand, collide, merge, and are dislocated. If the basic model is a portion of an interface, it will merge, collide, or disappear during the simulation. Owing to this, it is necessary to use the DS-DEVS (dynamic structure) [23] variant of DEVS, which is a formalism that is adapted from DEVS for the specification of systems with dynamic behavior.

In DS-DEVS, couplings are handled by another component, namely the executive that can trigger the creation, destruction, and links of basic models.

Vector-DEVS derives from DS-DEVS for the specification of spatially explicit systems with moving interfaces. In Vector-DEVS, basic models are always a portion of an interface and are called *geographic agents.* The connected agents that represent the interface are coupled into a shape by a *shape manager.*

It is not the topic of this paper to provide a complete description of Vector-DEVS; this description is available in [22, 24]. Basically, the three components cited are to be specified in order to implement a specific simulation method such as front tracking. Definitions of these components are as follows.

- The *geographic agent*, a basic DEVS model representing a point on the interface. A *geographic agent* has a reference position and a displacement vector. The agent can trigger the generation of new agent as well as a position change.

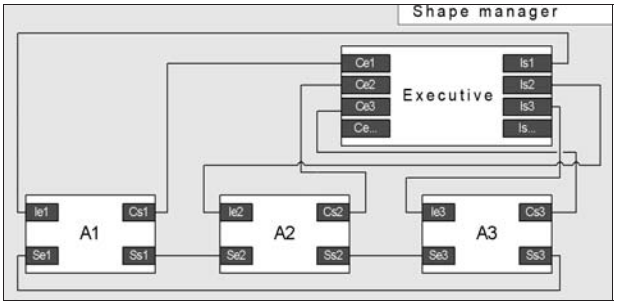

Figure 1. Structural view of the internal coupling in a shape manager (light gray), the *executive* is the shape executive of the manager, models *A1, A2, A3* represent the geographic agents; dark gray squares represent the ports, and the lines represent the internal coupling of the shape.

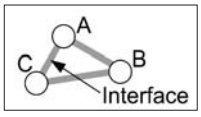

Figure 2. System view of a front polygon. Thick gray lines represent the simulated interface.

- The *shape manager*, a classical DS-DEVS network. It is a container for all agents in charge of activation and structure change. The structure is a set of connection between the agents that represents a polygonal shape. This structure is defined in a shape manager *executive*. There are as many executives as fronts and each front may have inner fronts.

- The *executive* links all agents in a front; the set of links is defined in an internal set. A structure change function defines the dynamics of the shape structure by triggering the creation, linking, and destructions of agents.

Figure 1 presents a structural view of the Vector-DEVS component. A shape is contained in a *shape manager* containing *geographic agents A1 A2, A3*, which are coupled by ports *SE* and *SS* and linked to the *shape executive* by ports *Ie* and *Cs*. The *shape executive* is in charge of the behavior of the shape as a whole, while agents are only in charge of their behavior with a limited view to their direct neighbors.

In the markers method, the phenomenon of behavior is usually represented by a polygon, which is the discrete view of the continuous system front [2]. In Vector-DEVS, this polygon is decomposed into *geographic agents* that represent angular points. The system view of the structure in Figure 1 is the polygon presented in Figure 2.

Agents evolve in an environment; for the front-tracking method, each agent has a set of two-dimensional coordi-

nates, and based on this set of coordinates, it is able to retrieve the state of the system at its specific location from data maps.

3.2 Data Driving the Simulation

Handling of geographical data is usually performed in a geographical information system (GIS).

There are two kinds of data formats usually available in GIS.

- Raster maps represent the environments in a matrix; this kind of data is very common because much of the data is based on satellite or aerial imagery. Moreover, most data obtained from simulations will be available in this form because the computation of the fields has been performed using matrices. The main problems of raster are the resolution of the data (at best, 50 m × 50 m for forest fire applications) and the actual data size of the raster that can be too large to fit in memory. Some vegetation cover, wind, and even elevation models are provided in raster.

- Vector maps that represent the environment in polygons or lines such as a highly non-uniform mesh; this kind of data is used increasingly, thanks to the use of the Global Positioning System (GPS). There are no resolution problems with vector data as all points have real coordinate values, and the area covered by a vector map is virtually infinite as memory problems will arise only if there are too many polygons. Nevertheless, polygons can only provide a delimitation of homogeneous areas and not the whole state of a system as raster does. Some vegetation covers, road networks, most land use maps, and tracking of vehicles are available in vector format.

Simulation in Vector-DEVS is performed using natively a vector format; the front interface is a polygon with agents having real coordinates. The main problem is the availability of the data driving the simulation in this format.

Basically, it is viewed as simulation on an arbitrary meshed map that can provide the state of the system at the point location of the agent.

Simulation on arbitrary meshed maps cannot be applied to study all two-dimensional evolutionary systems. It implies that interfaces have a predictable behavior in homogeneous areas. For this class of systems, there are four main reasons that make such a simulation more efficient.

- The space does not have to be discrete. There are no questions about mesh resolution and size. Some details of major importance that must be neglected with a common matrix are taken into account without increasing exponentially the overall simulation complexity (a forest fire simulation on a 10 km × 10 km grid must have a 2-m resolution to take into account the roads, thus resulting in a huge matrix on a regular mesh).

- The redundant calculus that occurs for a front with a stable speed on a homogeneous front can be simplified. Instead of propagating a front from cell to cell in a homogeneous area, if the front has a constant speed, the next calculation will occur when the front enters a different area.

- There is no need to transform maps from vector to raster format. This transformation usually introduces some information loss dependent of the resolution of the target raster.

- Different kinds of regular meshes (hexagonal cells, square cells, etc.) often introduce an effect known as ghosting, resulting in a propagation that mimics the underlying grid. Vector maps are not affected by this problem, because points do not have to be aligned.

All data are virtually parts of the state parameters of all models in Vector-DEVS; in fact, each model should have knowledge of every state value of every point in the entire environment.

Formally, a copy of (the infinity of) points should be made available in memory for all models, but from an implementation point of view, the availability of the data for the method is made through a data broker that provides the state values for a pinpoint location. The data broker has two-point data retrieval algorithm for raster and vector maps.

- For raster maps, the local data is obtained by bicubic interpolation with the four neighboring matrix values of the requested point; this method has the advantage of ensuring continuity for the state value over space.

- For vector maps, the local data is obtained by a neighborhood function that finds which polygon of the map contains the requested point. As the actual simulated interface is also in vector format, the broker can also check whether the point is contained in an area already covered by the same mechanism. The returned value is the state value of the polygon found.

The only information available to the agent is the actual map state value at its location and the location of its two direct neighbors. From this information, each agent has to calculate how it will move over space and time during simulation.

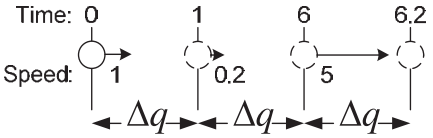

Figure 3. One-dimensional computation of agent advance.

4. Simulation Mechanisms

As for every DES, agents react to events that are triggering their advance in space and time. In addition to the structure, there are two key concepts that form the base of the simulation in Vector-DEVS as they generate the driving events.

- *Collision*. A collision happens when an agent is moving into a different area (from vegetative fuel to a fuel break or from an unburned area to an already burned area). Collisions also occur if an agent and its neighbor are separated by more than the minimum distance allowed during simulation, i.e. the quantum distance. Each collision will trigger a dynamic modification of the shape by adding or deleting an agent.
- *Quantum distance*. Denoted by Δq, it defines the maximum distance (in meters) that is allowed to be covered by an agent advance. The actual resolution of the simulation is limited by this quantum distance, and details that are smaller than this quantum distance may not be taken into account.

All of the motion and all events are generated by either a collision or by the agent planning an advance in space and time. Collisions trigger the activation of the *executive* that will resolve the overall shape modification. Agents actually move by self-activation until they stop.

4.1 Computation of Agent Advance in Space and Time

Unlike the conventional Lagrange method for the computation of a particle advance in a flow, DES resolution of an agent advance has to resolve the inverse problem of fixing a quantum distance instead of fixing a time step. The activation time for agents is given by time advance (t_a), which is computed by calculating how long it will take for the agent to travel Δq. Figure 3 depicts a simplified view of how an agent advances in one dimension.

A set of state parameters S_0 is valid during the time given by t_a. A set corresponding to the parameters in the next state is denoted by S_n.

As defined in Vector-DEVS, two state variables are attached to each state:

- the agent coordinates, denoted by P;
- the agent propagation vector, denoted as that is computed from the next and previous agent locations (for position of the agent on the left and for position of the agent on the right).
- the next and previous agent locations (P_l for position of the agent on the left and P_r for position of the agent on the right).

The current set of map parameters, denoted by M, contains the local properties of the map at point P (elevation, type of fuel, etc.).

Here $(P_0, V_o, M_o) \in S_0$ is valid at time t_o for the duration of t_a and $(P_n, V_n, M_n) \in S_n$ is valid at time $t_n = t_o + t_a$.

An agent is a basic DEVS model; as such, it has an external transition function, an internal transition function, a time advance function, and an output function. In addition to these functions, the agent has a velocity or RoS function, denoted by R.

The external transition function reacts to external events that are used here by the shape manager for one of the following purposes:

- immediately stop an agent if it has to be removed from a shape;
- inform the agent of the next and previous agent locations in the shape if they have moved;
- inform the agent of the new map parameters M_n at location P_n.

An output function is used to send an event to inform the shape manager of the updated location.

Calculation of t_a function is performed explicitly with

$$t_a = \frac{\Delta q}{\overline{v_o}}.$$

The internal transition function is triggered just after the output function, at time $t_n = t_o + t_a$. This function updates S_0 to the values of S_n and calculates the next state parameters with the equations

$$P_n = P_0 + t_a V_o \quad \text{and} \quad V_n = R(M_o, P_0, P_r, P_l)$$

so that the agent advance is a simple Euler first-order integration and the agent direction and speed are dependant on the velocity function. This function $R(M_o, P_0, P_r, P_l)$ will define the specificity of the models (fire spread, oil leak, shockwave, etc.). In the fire-spread model, the direction of the propagation vector is given along the bisector of the angle formed by P_l, P_0, P_r as presented in Figure 4. This definition enables us to handle naturally the direction of propagation of the unburned area as always directed to the outside of the clockwise-linked segments.

As the front is approximated as a polygon, no normal can be defined at the singular point of the agent location.

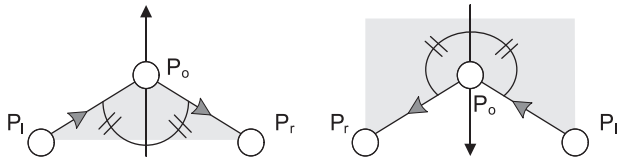

Figure 4. Direction of the propagation vector for an agent located at P_o in the propagation model is along the bisector of the local angle. The unburned area is in light gray; P_l and P_r lie to the left and to the right of P_o, respectively, depending on the linking direction of segments shown by the gray arrows. Propagation vector for P_o is shown in black and is always directed to the left of the clockwise-linking direction.

The bisector is used as an approximation of the normal angle because it may be computed efficiently. This approximation is made because transformation along the bisector presents the property of preserving the orthocenter of regular polygons and triangles.

At the local scale, quantum distance drives the numerical integration of an agent path; the behavior of the overall interface is handled by the shape manager that reacts to collisions.

4.2 Computation of the Shape Behavior

Each structural state change occurs when a collision happens between the interface and the environment or between the interface and another interface. Three kinds of collisions can occur.

- An internal collision occurs when there is an intersection of the shape with itself or another shape. It triggers a front recomposition.
- A self-collision occurs if part of the interface has moved by a certain distance or if the polygon has reached a critical size. It triggers a self-decomposition that refines the shape.
- An external collision occurs when the interface collides with an element of the environment. It triggers a decomposition that adapts the shape to the new environment.

In Vector-DEVS, these three functions (recomposition, self-decomposition, and decomposition) must be defined for each of these collision events.

The decomposition function is called by the external transition function of the executive (as it is inputted by an agent). This function must implement the modifications in the shape resulting from a collision with an agent of a different area.

The self-decomposition function, also inputted by an agent, is triggered by the internal transition function of the executive. The resulting shape must be a refined view of the shape.

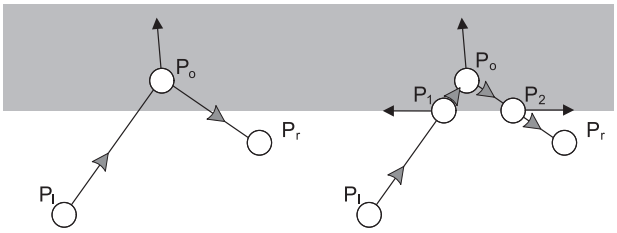

Figure 5. Decomposition of a front occurs when an agent (located at P_o) just enters a different area. Two new points (P_1, P_2) are created along the separation line between the two areas, with an initial propagation vector collinear with the separation line.

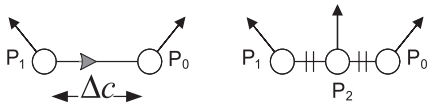

Figure 6. Self-decomposition of a front occurs when two agents located at two points (P_0, P_1) are separated by a critical distance Δc. A new agent is created at the median point (P_2).

The recomposition function is triggered by the internal transition function of the executive when the executive detects a collision between non-consecutive agents. This function must resolve the resulting front of the different intersections.

In the forest-fire-spread model, these three functions are defined as follows.

The decomposition function is activated when an agent has just entered a different area. The change is detected by different state parameters that are received from the shape manager. After detection, the agent sends a decomposition message to the executive, triggering the immediate creation of two new agents located at the boundary of the new area as described in Figure 5.

Self-decomposition, which is presented in Figure 6, refines a shape near an agent if two agents are separated by more than a critical distance or perimeter resolution denoted by Δc, with $\Delta c > 2\Delta q$, to ensure that the two agents cannot crossover. Each agent is aware of the position of the previous and next agents in the front in the current state, and self-decomposition is detected by the agent by calculating the distance between its projected location and these two neighboring agents. If self-decomposition is necessary, a decomposition message is sent to the executive, triggering the immediate creation of an agent located at the median location.

In Figure 6, self-decomposition occurs when the distance (P_0, P_1) is greater than Δc. The new agent is created at the median location P_2, which is given by $P_2 = P_1 + \frac{1}{2}(P_1 - P_0)$.

The recomposition function is the most complex function as it must handle the deletion of agents and is not

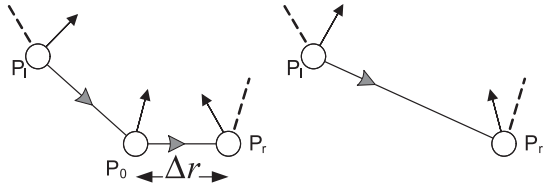

Figure 7. Deletion of an agent located at P_0 after a recomposition.

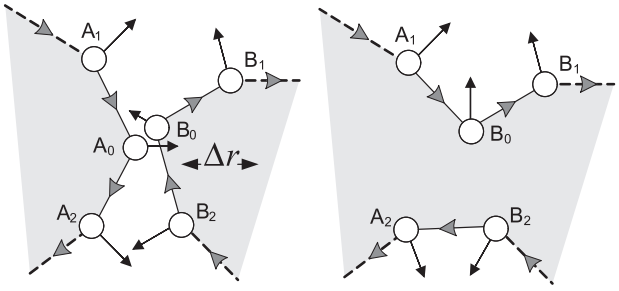

Figure 8. Collision of agent (A_0) with a front. Previous agent (A_1) is linked with the colliding agent (B_0). Previous agent of the colliding agent (B_2) is linked with the next (A_2).

detected directly by the agent itself, because it requires whole knowledge of all evolving fronts. The shape manager handles recomposition, and a proximity check for collision is performed each time an agent has moved. For this test, the distance is calculated between the moving agent and all other active agents. Any distance lower than Δr will require deletion of the moving agent. The stability condition is given by $\Delta r = \frac{\Delta c}{2} \geq \Delta q$. This approximation is made because two agents at a distance less than the perimeter resolution of the simulation (Δc) can be considered as located at the same point.

If deletion is required, the new structure is not updated immediately but after an elapsed time, which is given by the t_a function of the shape manager and is equal to t_n of the deleted agent, so that the structure change occurs when the actual agent would have reached Δr proximity. Three scenarios of recomposition can occur, which are presented here.

The first case corresponds to an agent being at a distance less than Δr from the next or previous agents in the front. In this case, the agent is deleted. The front the shape is as shown in Figure 7. For this case of decomposition, the shape manager t_a is set equal to t_a of the deleted agent.

The second case concerns a moving agent A being at a distance less than Δr from a colliding agent B of another front. The resulting shape is composed of the two fronts merged at the location of the agents, as shown in Figure 8.

In this case, agent A is deleted and the previous and the next agent are relinked with B and the previous agent of B, respectively. All agents that composed the front containing A are reassigned to the network executive of the front containing B.

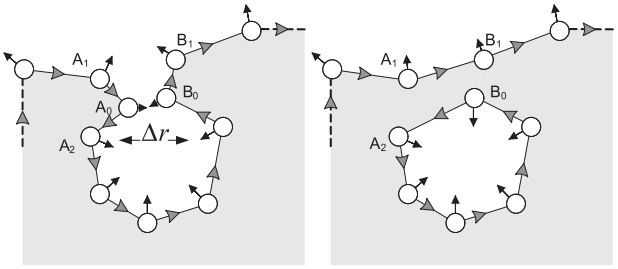

Figure 9. Collision of agent (A_0) with its own front. Previous agent (A_1) is linked with the next of colliding agent (B_1). Colliding agent (B_0) is linked with the next (A_2), and a new inner front is created.

The third and last case, shown in Figure 9, involves a moving agent A being at a distance less than Δr from a non-successive agent of the same front B. Agent A is deleted. The previous agent of A is connected to the next of B, and B is connected to the next of A. All agents that used to be between A and B are reassigned to a new front.

Geometrical operations are computer intensive but only have to be performed locally. If the simulation is synchronous, these three geometrical checks have to be made between every agent at every step so that the complexity of the check is N^2, where N is the number of agents. Asynchronous simulation reduces these checks to the very agent that has moved so that there are N checks at each agent advance. Overall, only two geometrical functions are required to determine the collision events.

- The neighborhood function is used to test the inclusion of every agent in environment shapes to determine the new state parameters and decomposition. Neighborhood check is performed using a 'point in polygon' method [25].
- The proximity test is given for two agents located at P_1 and P_2 by $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} < \Delta d$ where Δd is the proximity distance to be checked. Here (x_1, y_1) are the coordinates of P_1, and (x_2, y_2) are the coordinates of P_2.

The next section presents some implementation issues and limitations due to memory and processing usage.

4.3 Precision, Limitations, Processing and Memory Usage

The precision of the method is highly dependent on Δq and Δc. The method is subject to two kinds of errors: integration and truncation.

The integration error is introduced by the first-order Euler method used to perform the agent advance, which

is highly related to the QSS1 method [26], except that the agent path integration is in two dimensions. A demonstration of the precision of the method as a quantized front-tracking method would be out of the scope of this paper and has already been discussed in detail [26, 27].

As a first-order quantized system, the integration error, denoted as *ei*, is linearly dependent on Δq over the distance (denoted as D) to be traveled by an agent, with a maximum (worst case) of $ei = \Delta q D$. This worst case could be reached in highly turbulent winds with eddies of the same size as Δq. In real case scenarios, Δq is of a much higher resolution than wind or elevation data, and, thus, error is minimal.

The truncation error, denoted as *et*, occurs during decomposition and is induced by the deletion of the agent's front. This error is also linearly dependent on Δc, with a worst case of $et = \frac{\Delta c}{2}\left(\frac{D}{\Delta q}\right)$ or $et = \Delta r\left(\frac{D}{\Delta q}\right)$, meaning that each agent is deleted at every move. The worst case would typically be reached by multiple parallel fronts colliding. In any case, *et* only occurs for agents that are not located at a reflex angular point, and hence, there is no *et* observed at convex fronts.

The overall maximum error $e = ei + et$ is given by $e = D\left(\Delta q + \frac{\Delta r}{\Delta q}\right)$, with a minimum stability of $\Delta r = \Delta q$. This gives $e = D(\Delta q + 1)$.

A small Δq is required to minimize the error. However, it will also increase the computational cost of the simulation. To understand these limitations, it is important to have an overview of the data structures used in the implementation of the method.

An agent of the shape must memorize its current position, next position, displacement vector, neighboring (linked) point, link to the *shape manager*, valid time, next time, and a few state variables (dependant on the model), thus summing up to 128 bytes in a 64-bit computer. For each of these points, an activation event must be stored in a general scheduled list. An event is an activation time, a link to the model to activate and a state, 24 bytes. With this count, it is possible to fit a 50-km-long front with points at every centimeter (50 million points) in a gigabyte of memory.

In terms of processing, self-decomposition is typically the kind of collision that occurs most often (as the refinement critical distance is as small as the smallest detail in the point neighborhood).

A point triggers an event at least every time it moves by this quantum distance; the faster the point, the greater the number of events it will generate for a given period of time. If all points move at the same speed, a discrete event is less appropriate because using a common 'time step' for all points will keep the same level of detail while saving the handling of the scheduled event list.

A large variation of RoS over the simulation area is a good indicator to select DES.

Optimization of the event list and events is a key issue in this application in order to keep a quick processing time and small memory footprint. The event list is implemented as a linked list with the first element being the most imminent event. As most events are generated by fast-moving agents, most event insertions occur at the beginning of the list and a sorted insertion starting from the head provides good performance.

In order to limit the memory footprints of the numerous events that must be generated, all events that are processed are not removed from memory, but are kept in a recycling list. New events are only allocated in memory if the recycling list is empty, thus the number of events ever allocated never exceeds the number of moving agents.

With this simple optimization, handling the event list does not account for much of the simulation time; instead, part of the calculation time is spent searching for new collisions and obtaining properties of the new location. With a large number of points, much of the processing time needed for topological checks is avoided by the proximity test to estimate the neighborhood, collisions, and intersections.

In addition to this minimum set of calculations, agent update also requires a new calculation from the RoS model that accounts for most of the calculation time.

With the proposed implementation of the method, computational time and memory footprint grow linearly with the number of agents. As the number of agents is directly related to the fire perimeter, the calculation time is proportional to the fire perimeter.

Most raster- or cells-based methods must update at least the 'active' area [15, 17]. Thus, with raster-based methods, the simulation time is dependant on the burning surface of the simulated shape.

It should be noted that a low perimeter-to-surface ratio is a good indicator to select the front-tracking method.

We can eventually draw the following conclusions on the limitations of the discrete event/front-tracking method.

- If the RoS is similar for every direction and location (isotropic growth), it is more interesting to use a discrete time step, thus saving the sorting of the scheduled list.
- If the expected results are highly inhomogeneous (a lot of small shapes), the perimeter/surface ratio will be high and it is likely that a small discrete time step and a fine raster will make a better use of the processor.
- If the level of detail required is within one or two orders of magnitude of the overall simulation area (typically no need to take into account roads in a 1 km^2 area simulation), using raster/cells will be just as fast (small raster) and at the same time will keep track of the whole state of the system.

As forest fires have a low perimeter-to-surface ratio and a high variability of RoS over a given (large) area, the discrete events/front-tracking method is appropriate.

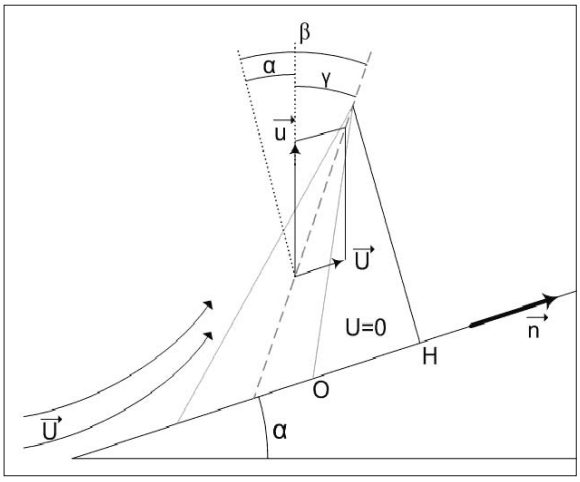

Figure 10. Calculation of the flame angle. 6 Calculation of the flame angle

The Vector-DEVS method has thus far only been applied in the field of forest fire simulation. To perform this application, we have developed a physical model that can explicitly provide the RoS in any given direction, the R function. The behavior described in this section is coupled with the physical spread model to serve as a basis for all numerical experiments.

5. Fire-spread Model

The proposed front-tracking asynchronous method can be considered as a general-purpose front-tracking method. As such, it falls in the same category as the level set, the volume of fluid, or the marker methods. In order to perform a numerical simulation, it is necessary to use an envelope RoS model. In the case of a forest fire, this model should provide the rate of fire spread in a given direction and the local parameters corresponding to a formulation of the R function for fire spread.

5.1 Physical Formulation of the Fire RoS Model

Our model is based on the description of physicochemical laws (mass, energy, and momentum balances) that drives the flame and propagation. This approach is explained in full detail in [28, 29].

This physical-based model has been developed to provide an analytical formulation of the RoS, given the slope angle, wind speed, and fuel parameters. Figure 10 represents the geometry of the flame, which is considered as a uniform radiating panel.

It is based on the following hypothesis:

- the heat transfer is due to the radiation of the front fire assimilated to its tangent plane;
- the radiation factor decreases with the surface-to-volume ratio of the flame;
- the gas velocity in the flame is the vectorial sum of the wind velocity and the upward gas velocity (in still air) due to buoyancy**.**

These hypotheses lead to a simple expression of the physical laws: mass, species, moment, and energy balances in the flame and in the fuel, providing the following model:

$$r = 1 + A\frac{r(1 + \sin\gamma - \cos\gamma)}{1 + \frac{r}{r_0}\cos\gamma}$$

$$\tan\gamma = \tan\alpha + \frac{U}{u_0}.$$

Here $r = R/R_0$ is the reduced RoS, R is the RoS (in m s^{-1}), R_0 is the RoS without wind and spread (in m s^{-1}), γ is the tilt angle, U is the normal wind velocity (in m s^{-1}), α is the local slope angle, u_0 is the vertical velocity in the flame without wind and slope (in m s^{-1}), and A is the ratio of radiated energy to ignition energy.

Using this formulation, we have to identify three parameters (R_0, u_0, and A) in order to characterize a kind of vegetation (or fuel type).

With a set of fitted parameters, the model is able to provide the RoS in any direction and for any given combination of slope and wind.

As the output of the model is directly a RoS given any direction of the front normal, a natural representation of the model output is a RoS polar diagram, as shown in Figure 11. This aspect is important because most other fire models provide only a maximum RoS and derive the RoS in all directions using an ellipse analogy [11].

Before using the RoS submodel, it is necessary to check whether it is able to reproduce results that are consistent with the experimental results.

5.2 Idealized Simulation Cases

We define as idealized tests the simplified tests used to analyze the behavior and error of both the simulation algorithm and the fire-spread model. All of these simple simulation tests can be compared with the analytical results derived from the model formulation. Four types of tests are employed to assess the performance of the fire-spread model. The first test is a spot fire in zero-wind and no-slope conditions to evaluate the model in an isotropic growth. The second is propagation under a series of wind conditions to verify the non-ghosting ability of the model. The third is propagation in areas of different RoSs. The last test is a rotating wind test, where the initial conditions are two symmetrical ignition lines with a fire break located on the rear of each line. Wind conditions for the last case are taken as rotation around the center of symmetry.

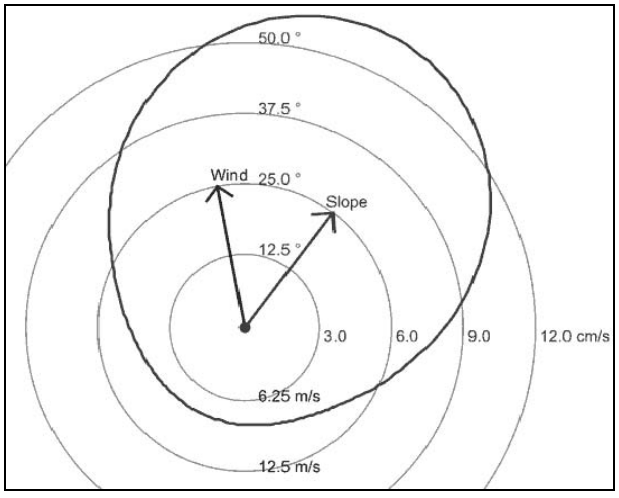

Figure 11. The RoS polar diagram of the model output for a given wind and slope vector and a given set of fuel parameters. The black ovoid line represents the polar view of the RoS in all directions and the two arrows represent the wind and slope directions and intensities.

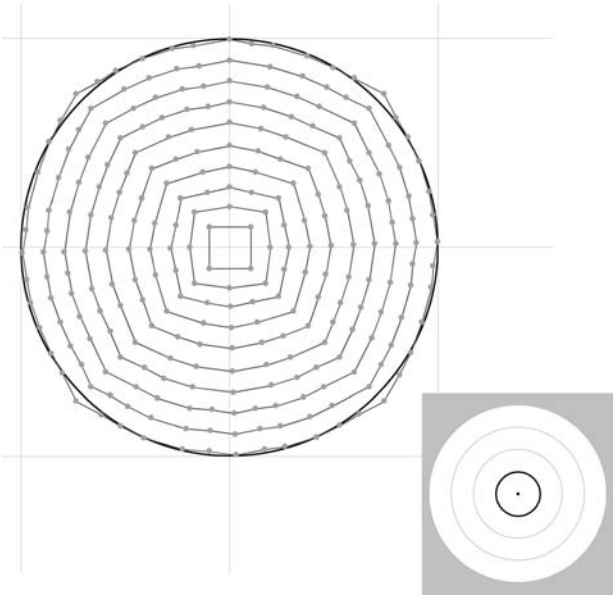

Figure 12. Spot fire case under zero–wind conditions. Circles correspond to 1-s isochronous fronts. Gray points correspond to agent locations. Grid size is 10 m. The speed polar diagram is shown at the bottom right corner.

Figure 12 shows the evolution of a spot fire (represented by a square) under no-wind and no-slope conditions. Simulation is performed with 2 $\Delta q = \Delta r = 1$ m. The speed polar diagram in this case is a circle, with a 1 m s^{-1} radius to simulate an isotropic growth. Given the initial size of the spot fire (a 10 m × 10 m square), the simulation converges well into a diffusive circle while transporting the initial angular points of the ignition square.

Figure 13 shows similar evolution for two spot fires in a 7 m s^{-1} wind. While the wind for the first spot fire is directed to the exact north, the wind for the second spot fire is directed in the north-northeast direction, totally unaligned with the grid. Simulation is also performed with 2 $\Delta q = \Delta r = 1$ m. The speed polar diagram in this case is an oval, with a minimum speed of 1 m s^{-1} and a maximum speed of 3 m s^{-1}. We can see from this figure that the head-fire in both cases reaches exactly the same distance of 30 m and that isochronous fronts in the first case are exactly the same as those in the second case with rotation. This test shows that because no underlying grid is necessary, there is no ghosting effect.

Figure 14 shows a spot fire ignition under zero-wind conditions, but in a non-homogeneous environment composed of three different fuel areas. Simulation is also performed with 2 $\Delta q = \Delta r = 1$ m. Owing to the different fuel parameters, the RoS is different in all areas. We can observe that the RoS for the front in each area is preserved and that angular points are formed at the boundary.

Figure 15 shows two line fire ignitions located symmetrically to the left and the right sides of the center of a circular wind flow. Wind at the location of ignition is 4 m s^{-1}, resulting in a maximum RoS of 2.2 m s^{-1}. Simulation is also performed with 2 $\Delta q = \Delta r = 1$ m. We can observe that the head-fire is totally following the wind streams, terminating exactly at the symmetrical point of ignition. Time to travel the half perimeter is about 71 s, corresponding to a distance of 156.2 m at 2.2 m s^{-1}, which is in good accordance with the theoretical half-perimeter of 157 m (50 m radius). Fronts merge at $t = 80$ s with the creation of an inner front.

These tests represent a verification of the front-tracking method in idealized cases. Validation of the fire RoS model and the large-scale fire propagation model is presented in the next section.

6. Validation

As the proposed simulator is a combination of the fire RoS model and the simulation method, both have to be validated. Validation of the fire RoS model is performed using data collected in a fire tunnel. A second experiment validates the model with the simulation method in the reanalysis of a large experimental fire.

6.1 Fire RoS Model Validation

Weise and Biging [30] performed a large number of studies in a fire tunnel with variable wind speed and slope configurations (collinear in each test). For a combination

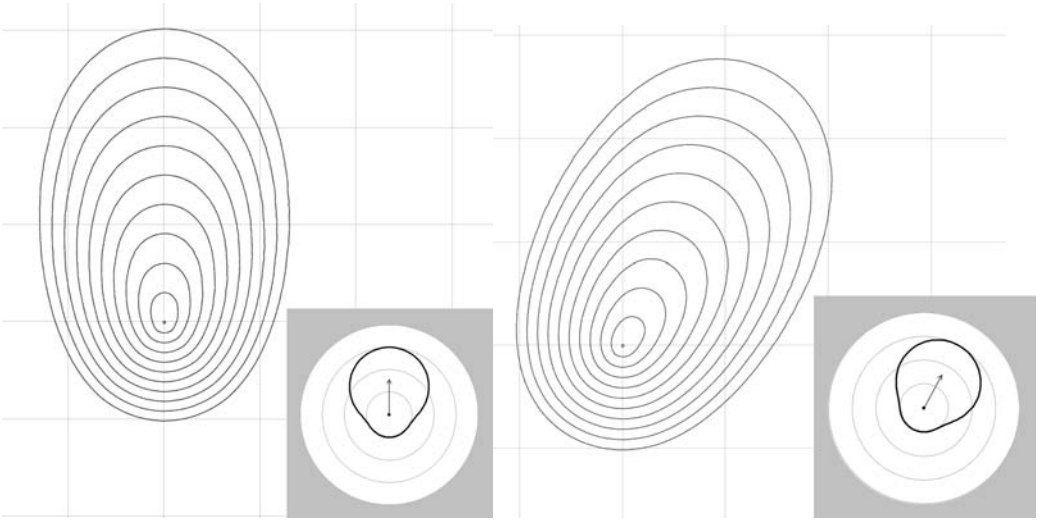

Figure 13. Ghosting test in north oriented (left) and north-northeast oriented light wind (7 m s^{-1}). Lines correspond to 10-s isochronous fronts. Grid size is 100 m. The speed polar diagram is shown in the bottom right corner, with a head-fire speed of 3 m s^{-1} and a back-fire speed of 1 m s^{-1}.

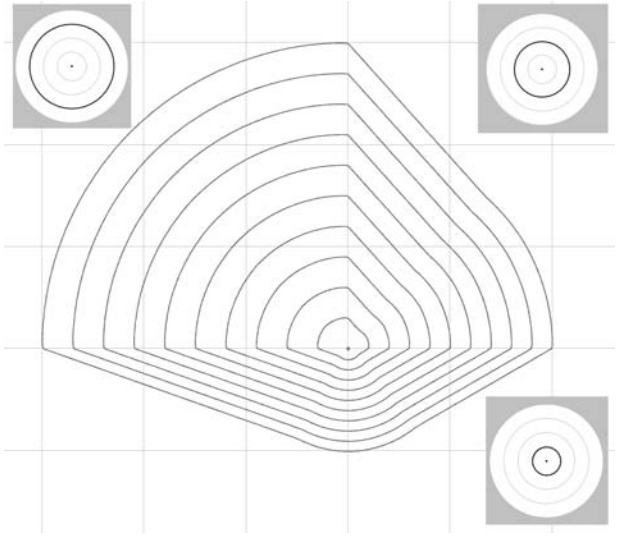

Figure 14. Non-homogeneity test. Lines correspond to 10-s isochronous fronts. Grid size is 100 m. Three regions have different isotropic RoSs: bottom region, 1 m s^{-1}; top right, 2 m s^{-1}; and top left, 3 m s^{-1}.

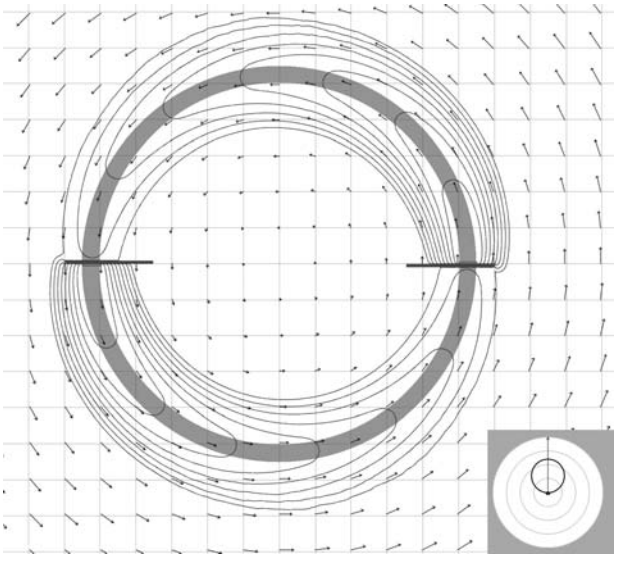

Figure 15. Fan case. Lines correspond to 10-s isochronous fronts. Arrows correspond to wind forcing, with the gray circle as the flow line passing by the ignition points. Black lines correspond to fuel breaks preventing the fire from propagating backwards to the flow. Grid size is 10 m. The speed polar diagram is shown in the bottom right corner, with a head-fire speed of 2.2 m s^{-1} and a back-fire speed of 0.1 m s^{-1}.

of parameters, the author observed the RoS and compared it with existing models. The authors eventually concluded that the model from Rothermel and Albini (see [31]) provided the most accurate results.

We used the experimental data of this study to verify the proposed fire-spread model.

Two measurement samples are needed and have been used to perform the regression:

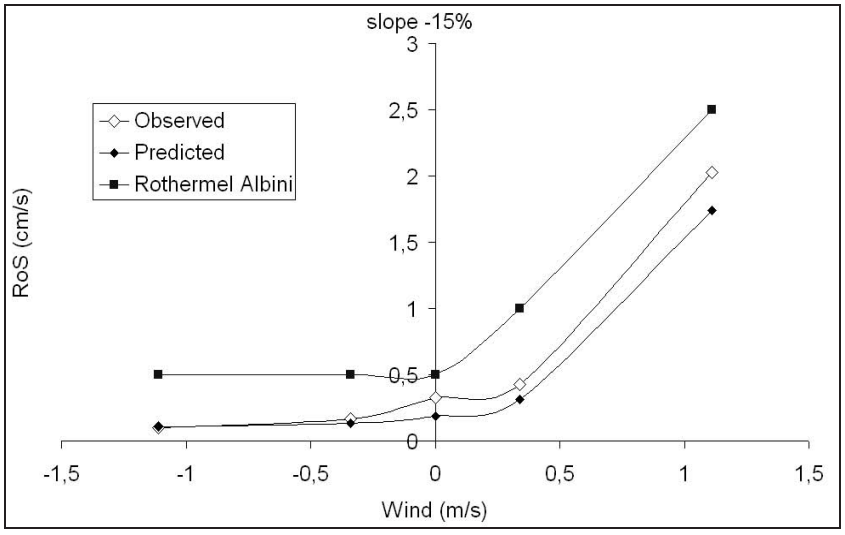

Figure 16. RoS measured and calculated with a –15% slope.

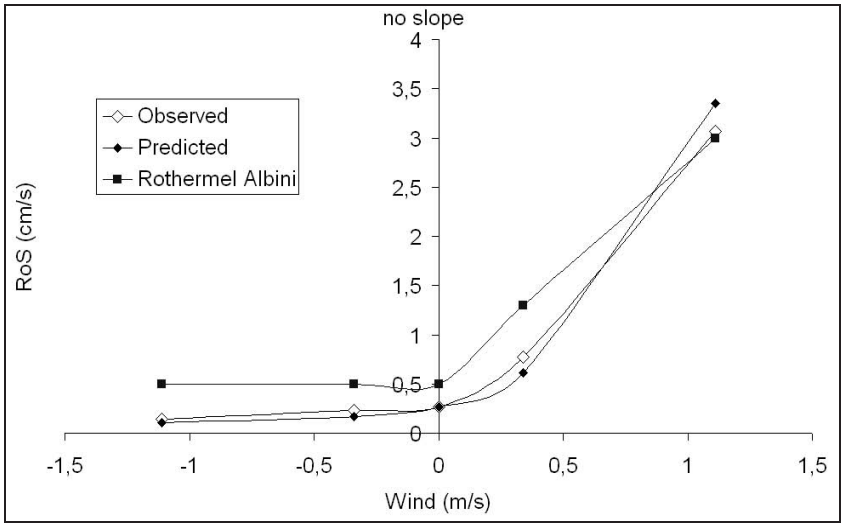

Figure 17. RoS measured and calculated with no slope.

1. the RoS with no slope and no wind;

2. the RoS with no slope and a wind speed of 1 m s^{-1}.

The first input of the model is the rate of fire spread expected under no-slope and no-wind conditions, namely R_0 whose value is 3×10^{-3} m s^{-1}. For the experiments, we considered the upward gas velocity (u_0) and fuel parameter (A) to be 1.72 m s^{-1} and 22, respectively. These values were deduced using a least-squares regression based on the experimental data without wind and with a wind speed of 1 m s^{-1}.

Figures 16, 17, and 18 show the results given by the proposed model compared with the experimental results and those obtained with the Rothermel and Albini model for different combinations of collinear winds and slopes.

The model predictions are in good agreement with the experimental data. The model also has another important advantage: only three parameters have to be set, and they all have a physical meaning that is easier to guess qualitatively. The Rothermel model requires the adjustment of more parameters (five or more depending on the revision), which makes the original parameter setup less robust. Moreover, the proposed model can also provide other

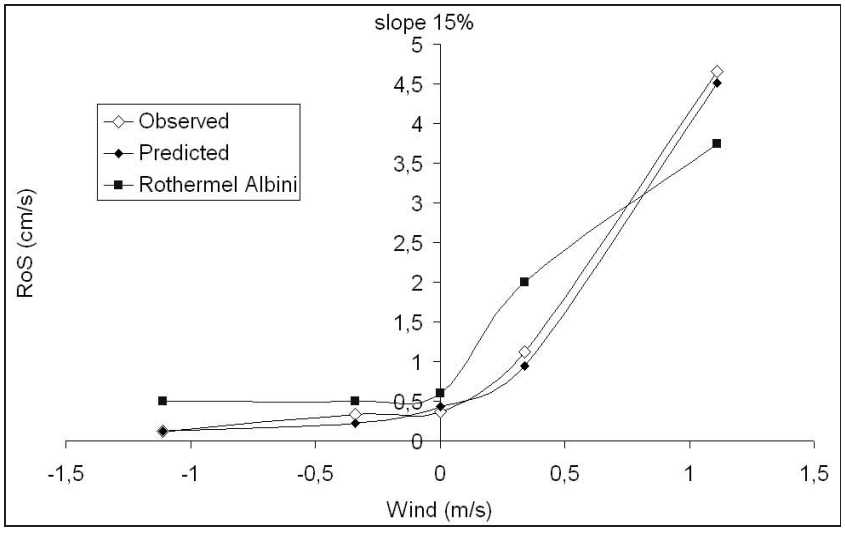

Figure 18. RoS measured and calculated with a 15% slope.

important fire front variables such as radiant heat flux, flame height, and tilt angle.

This RoS model is also directly applicable to cases where the slope and wind are not collinear, but this cannot be verified directly without simulating the whole front shape evolution with a fire propagation model. The following section presents the comparison with real fire data on a higher scale.

6.2 Comparison with Real Fire Data

To verify the robustness and speed of the method in propagating two-dimensional fire fronts, we ran the model to simulate a fire based on a real scenario. We selected the 2005 Lançon fire that took place in the south of France and burned about 800 Ha of shrubs and forest. On the day a north-westerly wind of 50 km h^{-1} was blowing, providing extreme conditions of propagation. Fire started at 09:40 and lasted until 16:30.

The wind map has been calculated as a stationary solution using a mainstream CFD software (Fluent). The resolution of the wind map is the same as that of the digital elevation model, 50 m × 50 m.

The simulation has been run for $\Delta q = 3$ m and $\Delta c = 9$ m. As the vegetation is quite homogeneous, the simulation is mostly driven by self-decomposition. The interested reader can perform the simulation directly using a java applet available online at http://spe.univ-corse.fr/filippiweb/simulation/.

The fire-spread model needs three parameters (A, R_0, u_0) to be determined in order to take into account the vegetative fuel properties. As vegetation is relatively unknown, these three parameters are fitted so that the simulation matches the first contour (at 12:00).

The simulation results and observations are presented in Figure 19. The simulation started at 09:40 at the exact same points as the estimated ignition points and stopped at 16:30 (last-known contour). Although no adjustments have been made on the parameters during the simulation run, the simulation is in good accordance with the observed fire fronts. Nevertheless, the simulation results show a strong development on the right front that is not observed in the actual fire. This difference is due to the attack of the fire fighters that prevented the fire from developing on the right flank. These results indicate that the proposed fire simulator succeeds in rendering the wind and slope conditions in a complex topography.

The simulation time for these parameters is 28 s on a computer capable of showing a 3-Gflops peak performance (Intel Pentium 4) and having 2 GB main memory. It is largely possible to achieve better performance with larger Δq and Δc, and evaluation of the performance of the method with different quantum sizes is the subject of our ongoing research. Ultimately, the goal of simulating a large fire with a 3-m resolution in less than a minute was reached.

7. Conclusion

In this paper we have presented a new method, formalism, and software for the simulation of interface dynamics. Vector-DEVS, a simulation methodology using front-tracking method has been developed to simulate fire growth.

We showed that where front tracking can be applied, a different set of solutions can emerge from the system study on a larger temporal and spatial scale.

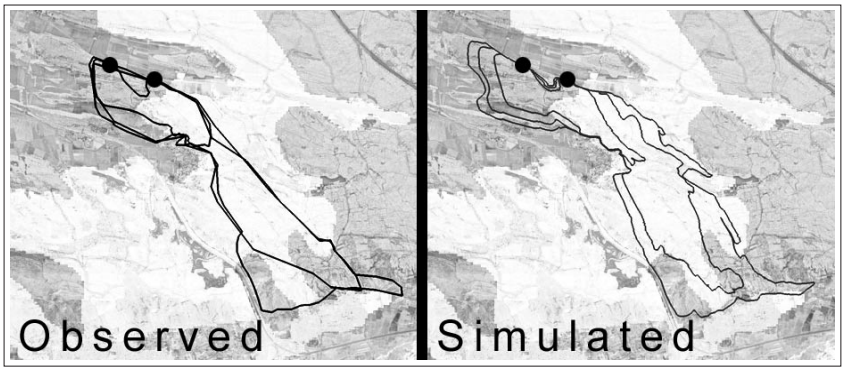

Figure 19. Observations (left) and simulation results (right) obtained for the Lançons fire. Black dots represent ignition points (at 09:40). The contours correspond to the fire front at 12:00, 14:30, and 16:30.

To take advantage of the method in forest fire simulation, we have developed a specific physics-based propagation model of fire spread that generalizes the behavior of the fire front at any given point.

Given a level of detail of 3 m, which is necessary to take into account roads in the simulation, computing speed is, by far, faster than real-time and other methods; this is because only the very perimeter of the phenomenon is simulated. The main goal of this method was to be able to provide the evolution of a large wildfire in less than a minute of calculation time, which we show is possible with a validation on a large wildfire accident that took less than 30 s to simulate.

The method is not tied to the simulation of forest fire spreads as it may be possible to forecast the shape evolution of any system that can be derived as a RoS model.

8. References

[1] Zeigler, B., 2000. Theory of Modeling and Simulation. Academic Press, 2nd Edition, ISBN: 0127784551.

[2] Glimm, J., Simanca, S., Smith, T., Tangerman, F., 1996. Computational physics meet computational geometry. Tech. Rep. SUNYSB-96-19, State University of New York at Stony Brook.

[3] Du, J., Fix, B., Glimm, J., Jia, X., Li, X. and Li, Y., Wu, L., 2006. A simple package for front tracking. Journal of Computational Physics 213 (2), 613–628.

[4] Risebro, N., Holden, H., 2002. Front Tracking for Hyperbolic Conservation Laws. Academic Press.

[5] Risebro, N., Tveito, A., 1992. A front tracking method for conservation laws in one dimension. J. Comp. Phys. 101, 130–139.

[6] Karimabadi, H., Driscoll, J., Omelchenko, Y.A., Omidi, N. 2005, A new asynchronous methodology for modeling of physical systems: breaking the curse of courant condition. Journal of Computational Physics Volume 205, Issue 2, Pages 755–775.

[7] Pastor, E., Zarate, L., Planas, E., and Arnaldos, J. (2003). Mathematical models and calculation systems for the study of wildland fire behaviour. Progress in Energy and Combustion Science, 29(2):139–153.

[8] Rothermel, R. 1972. A mathematical model for predicting fire spread in wildland fuels. Research Paper INT-115, USDA Forest Service.

[9] Sullivan, A.L. 2009. A review of wildland fire spread modelling, 1990-present 3: Mathematical analogues and simulation models, International Journal of Wildland Fire 18(4) 387–403 DOI: 10.1071/WF06144.

[10] Linn, R., Reisner, J., Colman, J. J., and Winterkamp, J. 2002. Studying wildfire behavior using FIRETEC. International Journal of Wildland Fire, 11(3–4):233–246.

[11] Finney, M., Andrews, P., 1994. The farsite fire area simulator: Fire management applications and lessons of summer 1994. In: in Proceedings of Interior West Fire Council Meeting and Program. pp. 209–216, coeur Alene, USA.

[12] Mallet, V., Keyes, D. E, Fendell, F. E. 2009. Modeling Wildland Fire Propagation with Level Set Methods Authors, Computers & Mathematics with Applications, 57(7) doi:10.1016/j.camwa.2008.10.089.

[13] Ntaimo, L., Zeigler, B. P., Vasconcelos, M. J., and Khargharia, B. 2004. Forest fire spread and suppression in devs. Simulation, 80(10):479–500.

[14] Vasconcelos, M. Guertin, D. 1992. FIREMAP – simulation of fire growth with a geographic information system. International Journal of Wildland Fire, 2(2):87–96.

[15] Muzy, A. Innocenti, E.Wainer, G. Aïello, A. Santucci, J. F. 2005. Specification of discrete event models for fire spreading, Simulation: transactions of the society of modeling and simulation international, SCS, 81, 103–117.

[16] Dunn, A. Milne, G. 2004, Modelling Wildfire Dynamics via Interacting Automata. Proccedings of Cellular Automata, 6th International Conference on CellularAutomata for Research and Industry, ACRI 2004, Amsterdam, The Netherlands, October 25–28, pp

[17] Hirt, C. W. and Nichols,B. D. 1981, Volume of fluid (VOF) method for the dynamics of free boundaries. Journal of Computational Physics Volume 39, Issue 1, Pages 201–225.

[18] Lallemand, P., Luob, L. Pengb, Y. 2007, A lattice Boltzmann front-tracking method for interface dynamics with surface tension in two dimensions, Journal of Computational Physics, Volume 226, Issue 2, 1 October, Pages 1367–1384.

[19] Uhrmacher, A.M. 1997 Concepts of object- and agent-oriented simulation, In Simulation, volume 14, Issue 2, 59–67.

[20] Wainer, G. 2006, Applying Cell-DEVS Methodology for Modeling the Environment, In Simulation, Transactions of the SCS. Vol. 82, No. 10, 635–660.

[21] Filippi, J.-B., Bisgambiglia, P., 2002. Enabling large scale and high definition simulation of natural systems with vector models and jdevs. In: Proceedings of 2002 IEEE, SCS, ACM Winter Simulation Conference. pp. 1964–1970, san Diego, CA, USA.

[22] Barros, F., 1997. Modeling formalism for dynamic structure discrete event systems. ACM Transactionson modeling and Computer Simulation 7, 501–515.

[23] Filippi, J., 2003. Une architecture logicielle pour multi-modélisation et la simulation a évènements discrets de systèmes naturels complexes. Ph.D. thesis, University of Corsica.
[24] Haines, E. 1994, Point in Polygon Strategies, *Graphics Gems IV*, ed. Paul Heckbert, Academic Press, p. 24–46.
[25] Kaufman, E. 2003. Quantization-Based Simulation of Differential Algebraic Equation Systems. Simulation. 79(7). pp 363–376.
[26] Kaufman, E. 2002. A Second Order Approximation for DEVS Simulation of Continuous Systems. Simulation. 78(2). pp 76–89.
[27] Rossi, L., Balbi, J., Rossi, J., Marcelli, T., Pieri, A., 2006. Front fire propagation model: use of mathematical model and vision technology. Advanced Technologies, Research–Development–Applications, 745–760Ed Bojan Lalic.
[28] Balbi, J., Rossi, J., Marcelli, T., Santoni, P., 2005. A simple 3d physical model for forest fire. In: Proceedings of the 2005 Journées internationales de thermique. Tanger, Maroc.
[29] Weise, D., Biging, G., 1997. A qualitative comparison of fire spread models incorporating wind and slope effects. Forest Science 43 (2), 170–180.
[30] Albini, F., Baughman, R., 1979. Estimating wind speed for predicting wildland fire behavior. Tech. Rep. INT-221, USDA Forest Research Service.

Jean Baptiste Filippi *is a computer science researcher at the CNRS SPE laboratory of the University of Corsica, France. His main research interests are in forest fire simulation.*

Frédéric Morandini *a full-time physics research engineer at the CNRS SPE laboratory of the University of Corsica, France.*

Jacques Henri Balbi *is an emeritus professor in physics in the University Of Corsica, France and was also the president of the University of Corsica, France. His main research interests are in forest fire modeling.*

David RC Hill *is a computer science professor in and currently the Vice President of Blaise Pascal University, France in charge of ICT. His main research interest concerns simulation and software engineering for biological and environmental systems. He is also director of the Inter-Univeristy Computing Center (CIRI), Auvergne, France.*

Journal of Simulation – Table of Contents Volume 3 (2009)

[1]In co-operation with other journals in the field of M&S, Simulation, Transactions of the Society for Modeling and Simulation International, is publishing bibliographies to facilitate rapid diffusion and impact of research results, while facilitating interpenetration of ideas across subdisciplines of M&S. In this issue, we are listing articles from Journal of Simulation (JOS), Volume 3, (2009)

For full listings, please visit www.palgrave-journals.com/jos.

Journal of Simulation aims to publish both articles and technical notes from researchers and practitioners active in the field of simulation. In ***JOS***, the field of simulation includes the techniques, tools, methods and technologies of the application and the use of discrete-event simulation, and covers a wide range of domains including manufacturing, service, defence, health care and general commerce. ***JOS*** particularly seeks topics that are not "mainstream" in nature but which are interesting and evocative to the simulation community; particular interest is paid to significant success in the use of simulation. ***JOS*** is an official publication of The OR Society, and is published by Palgrave Macmillan.

Volume 3, Issue 1 (March 2009)

Editorial

Simulation software: evolution or revolution?
S J E Taylor and S Robinson

Articles

A survey on distributed simulation in industry
C A Boer, A de Bruin and A Verbraeck

Simulation as a tool for gaming and training in operations management—a case study
ED-J van der Zee and J Slomp

A fast computational algorithm for simulating round-robin service
D Finley, J R Ramos, V Rego and J Sang

Towards a unified conceptual model representation: a case study in healthcare
B S S Onggo

Effective design of an assembly line using modelling and simulation
F Longo and G Mirabelli

A conceptual model for carpooling systems simulation
G Correia and J M Viegas

Volume 3, Issue 2 (June 2009)

Articles

Simulation-based cycle-time quantile estimation in manufacturing settings employing non-FIFO dispatching policies
J M Bekki, J W Fowler, G T Mackulak and M Kulahci

An integrated supply chain model with dynamic flow and replenishment requirements
G M Magableh and S J Mason

Method for assessing and selecting discrete event simulation software applied to the analysis of logistic systems
A K da Silva and R C Botter

Modelling, simulation and analyses of systems with breakdown imposed scrapping
T Ilar, J Powell and A Kaplan

Generic interface specifications for integrating distributed discrete-event simulation models
A K Garg, J Venkateswaran and Y-J Son

Volume 3, Issue 3 (September 2009)

Special Issue: Simulation in Healthcare: Part 1

Guest Editor: Hazel Pilgrim

Articles

An analysis of the academic literature on simulation and modelling in health care
S C Brailsford, P R Harper, B Patel and M Pitt

Developing model requirements for patient flow simulation studies using the Unified Modelling Language (UML)
C Vasilakis, D Lecznarowicz and C Lee

Service optimization with patient satisfaction in healthcare systems
T Gonsalves and K Itoh

Application of synthetic patient data in the assessment of rapid rule-out protocols using Point-of-Care testing during chest pain diagnosis in a UK emergency department
S Robinson, F FitzGibbon, J Eatock, T Hunniford, D Dixon and B J Meenan

Success and failure in the simulation of an Accident and Emergency department
J Bowers, M Ghattas and G Mould

Incorporating remote visits into an outpatient clinic
J Eatock and T Eldabi

Book Review

Operations research and health care: a handbook of methods and applications
Martin Pitt

Volume 3, Issue 4 (December 2009)

Articles

Improved energy-efficient production using discrete event simulation
P Solding, P Thollander and P R Moore

Subset selection procedures
E J Chen

Using simulation and goal programming to reschedule emergency department doctors' shifts: case of a Tunisian hospital
B Jerbi and H Kamoun

Simulating hospital evacuation—the influence of traffic and evacuation time windows
E Tayfur and K Taaffe

Optimization of ship evacuation procedures as part of tsunami preparation
T Pitana and E Kobayashi

Book Review

Crowd Simulation
Reviewed by: Vassilis Zachariadis and James Amos