Proceedings of the 2002 Winter Simulation Conference
E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, eds.

CELL-DEVS QUANTIZATION TECHNIQUES IN A FIRE SPREADING APPLICATION

Alexandre Muzy
Eric Innocenti
Antoine Aiello
Jean-François Santucci

Computer Modeling
University of Corsica
SPE – UMR CNRS 6134
B.P. 52, Campus Grossetti
20250 Corti, FRANCE

Gabriel Wainer

Dept. of Systems and Computer Engineering
Carleton University
4456 Mackenzie Building
1125 Colonel By Drive
Ottawa, ON. K1S 5B6, CANADA

ABSTRACT

We present the use of the CD++ tool to model and simulate forest fire-spread. A semi-physical fire spread model is implemented using the Cell-DEVS formalism. The use of Cell-DEVS enables proving the correctness of the simulation engines and permits to model the problem even by a non-computer science specialist. The high level language of CD++ reduces the algorithmic complexity for the modeler while allowing complex cellular timing behaviors. Different Cell-DEVS quantization techniques are used and developed to decrease execution time. The study is realized regarding time improvement and trades-off between model evolution, simulation time and incurred error. Finally, based on experimentations, interesting perspectives are defined to develop new quantization techniques.

1 INTRODUCTION

At present, concerns for environment engender increasing interest in monitoring and predicting ecosystem changes. The damage induced by devastating fires that occurred over the last few years stressed the necessity for fire fighters to dispose of a tool that would provide rapid and relatively accurate information concerning fire position.

In fire spreading, the use of computer simulation is a good choice to solve the problem due to the complex behavior of the phenomena studied and the volume of data that ecological models have to grasp. However, real-time simulators for fire spread are tricky to elaborate due to both fire complexity and landscape size.

The Cell-DEVS formalism (Wainer and Giambiasi 2001) is well suited to solve this kind of applications. A Cell-DEVS model enables the definition of a cell-space with explicit timing delays. Each cell is defined as an atomic DEVS model (Zeigler et al. 2000), and a procedure to couple cells is depicted. Delay functions allow defining timing behavior explicitly.

Each cell is described as:

$$TDC = < X, Y, \theta, N, delay, d, \delta_{int}, \delta_{ext}, \tau, \lambda, D >$$

X defines the external inputs, Y the external outputs defining the interface of the model. θ is the cell state definition, and N is the set of inputs. Delay defines the kind of delay for the cell (transport or inertial), and d its duration. A transport delay can be associated with each cell, which defers the outputs for the cell. A state change will be discarded if it is not steady during an inertial delay. Each cell takes uses the set of inputs to computes its future state using the τ functions. Finally, there are several functions driving the cell behavior: δ_{int} for internal transitions, δ_{ext} for external transitions, λ for outputs and D for the state's duration.

A Cell-DEVS coupled model is defined by:

$$GCC = < X_{list}, Y_{list}, X, Y, n, \{t_1,\ldots, t_n\}, N, C, B, Z >$$

Here, X_{list} and Y_{list} are input/output coupling lists, used to define the model interface. X and Y represent the input/output events. The n value defines the dimension of the cell space, $\{t_1,\ldots, t_n\}$ is the number of cells in each dimension and N is the neighborhood set. The cell space is defined by C, together with B, the set of border cells, and Z the translation function.

A modeler simply has to focus on three basic aspects: dimension (size and shape of the cell space), influences and behavior. As we can see in Figure 1, the modeler has to define the local computing function, the kind of delay and its length. The influences are specified by defining the neighborhood of the cell.

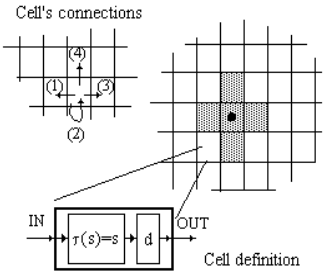

Figure 1: Informal Definition of a Cell-DEVS Model and its Neighborhood

Recently, quantized DEVS theory has been introduced (Zeigler 1998). As depicted in Figure 2, quantization approach consists in quantizing the value space in equal quantum steps. A fixed quantum size q is used. Quantizers are associated with each model. It is a significant event detector which monitors its input and uses a logical condition to decide when a significant change, such as crossing a threshold has occurred. Whenever a crossing occurs the new value is sent to the receiver. The problem consists in a trade-off between reducing number of messages and the error induced by this reduction.

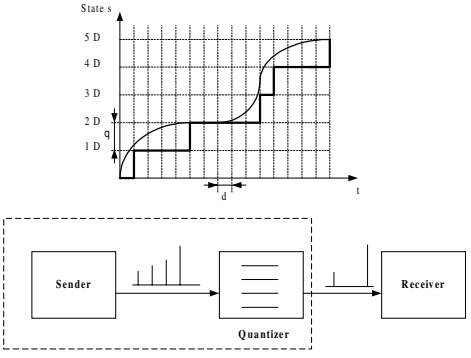

Figure 2: Continuous Models Quantization

Promising empirical results have been obtained when using quantization theory in Cell-DEVS models (Wainer and Zeigler 2000). As seen in Figure 3, quantizers (Q) are associated to each output port of a cell. As long as the output information is not considered as significant, a cell does not send information. This produces a significant reduction in the number of messages involved in the simulation.

We will show how quantized Cell-DEVS models can be applied in reducing and controlling error for complex cellular models of fire spread.

2 FIRE SPREAD MODELING

In a first stage of our research, we studied fire spread across a 1-m² pine needles fuel empirically (Balbi et al. 1998). This study uses elementary cells composed of earth and plant matter, and did not consider wind or slope.

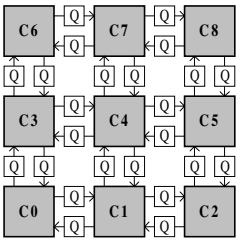

Figure 3: Cell-DEVS Quantization

The phenomenon was described using a set of PDEs. In the domain:

$$\frac{\partial T}{\partial t} = -k(T - T_a) + K\Delta T - Q\frac{\partial \sigma_v}{\partial t} \quad (1a)$$

If $T < T_{ig}$:

$$\sigma_v = \sigma_{v0} \quad (1b)$$

If $T \geq T_{ig}$:

$$\sigma_v = \sigma_{v0}.e^{-\alpha(t-tig)} \quad (1c)$$

At the boundary:

$$T(x,y,t) = Ta \quad (1d)$$

Here, $\boldsymbol{T_a}$ (27 °C) is the ambient temperature, $\boldsymbol{T_{ig}}$ (300°C) is the ignition temperature, $\boldsymbol{t_{ig}}$ (s) is the ignition time, $\boldsymbol{T}$ (°C) is the temperature, $\boldsymbol{K}$ ($m^2.s^{-1}$) is the thermal diffusion, $\boldsymbol{\alpha}$ (s^{-1}) combustion time constant, $\boldsymbol{\sigma_v}$ ($kg.m^{-2}$) is the vegetable surface mass, and $\boldsymbol{\sigma_{v0}}$ ($kg.m^{-2}$) is the initial vegetable surface mass (before the cell combustion).

We discretized the model using finite differences (FDM) and finite elements (FEM) (Santoni 1997). The study domain was meshed uniformly with cells of 1-cm² and we use a time step of 0.01s. Both methods provided similar results, but FEM was more complex to apply, and produce longer execution time. The physical model resolution by FDM led to the following algebraic equation:

$$T_{i,j}^{k+1} = a(T_{i-1,j}^{k} + T_{i+1,j}^{k}) + b(T_{i,j-1}^{k} + T_{i,j+1}^{k}) + cQ\left(\frac{\partial \sigma_v}{\partial t}\right)_{i,j}^{k+1} + dT_{i,j}^{k} \quad (2)$$

Where T_{ij} is the temperature of a grid node. The coefficients a, b, c and d depend on the time step and mesh size. Those coefficients are identified from experimental data of temperature versus time. This equation represents the temperature curve of a cell in the domain, whose sketch is depicted in Figure 4. Above the threshold temperature T_{ig}, the combustion occurs, and under a T_f temperature, the combustion finishes.

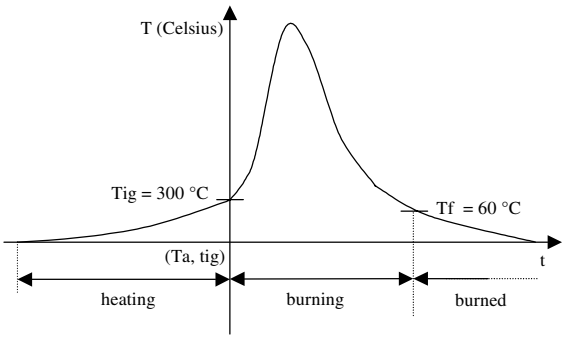

Figure 4: Simplified Temperature Curve of a Cell of the Domain

In order to improve model definition and include more complex behavior, we faced its definition using the Cell-DEVS formalism. The CD++ environment (Rodriguez and Wainer 1999) allows implementing DEVS and Cell-DEVS models. The modeler simply specifies the cell domain dimensions, the cell's neighborhood and the cell's behavior by defining simple logical rules. They have the format {*result*} *delay* {*condition*}. The semantics of the sentences is that, if the *condition* is true, the cell will take the *result* value and will send it through output ports after a *delay* time. If the condition is not valid, the next rule is evaluated (according to the order in that they were defined), repeating this process until a rule is satisfied. The most common operators are included: boolean (*AND*, *OR*, *NOT*, *XOR*, *IMP* and *EQV*), comparison (=, !=, <, >, <= and >=), and arithmetic (+, -, * and /). In addition, different types of functions are available: trigonometric, roots, power, rounding and truncation, module, logarithm, absolute value, minimum, maximum, G.C.D. and L.C.M. Other existing functions allow checking if a number is integer, even, odd or prime. Space *zones*, defined by a cell range, can be associated with a set of rules different from the rest of the cell space.

Each cell uses a single state variable (representing its own state). The language permits to manipulate n-dimensional references. Likewise, a neighborhood can be composed of non-adjacent cells, and the neighborhood's dimension can be similar or inferior to the model's dimension.

With these considerations in mind, we used the high-level specification language included in CD++ to describe the fire model using the Cell-DEVS formalism (Muzy et al., 2002). We used two planes representing different variables in our model. The first plane stores the cell temperature. The second plane stores the ignition time t_{ig} of each cell (see equation 1c). In this plane, every cell detects when the corresponding cell of the propagation plane starts burning. Figure 5 represents the model specification in CD++.

Lines 01 and 02 of this specification, declare the components of the top level coupled model. From line 03 to 18, we include a definition for the Cell-DEVS coupled model representing the fire-spread model. This is done by including the parameters mentioned earlier (neighborhood, dimension, type of delay etc.).

```
00  #include(rules.inc)
01  [top]
02  components : ForestFire
03  [ForestFire]
04  type : cell
05  dim : (100,100,2)
06  delay : transport
07  defaultDelayTime : 1
08  border : nowrapped
09  neighbors : (-1,0,0)(0,-1,0)(1,0,0)(0,1,0)
10  neighbors : (0,0,0)(0,0,-1)(0,0,1)
11  initialValue : 27.0
12  initialCellsValue : init.val
13  zone : cst {(0,0,0)..(0,99,0)}
14  zone : cst {(1,99,0)..(99,99,0)}
15  zone : cst {(99,0,0)..(99,98,0)}
16  zone : cst {(1,0,0)..(98,0,0)}
17  zone : ti {(0,0,1)..(99,99,1)}
18  localTransition : FireBehavior

19  [ti]
20  rule : {time * 0.01} 1 {(0,0,0) = 1.0 AND
(0,0,-1) >= 300}
21  rule : {(0,0,0) } 1 {t}

22  [cst]
%Constant border cells
23  rule : {(0,0,0)} 1 {t}
24  [FireBehavior]
%Heating
25  rule : {#macro(heating)} 1 {(0,0,0) < 300
AND (0,0,0) != 26 AND (#macro(heating) >
(0,0,0) OR time <= 20)}
%Burning
26  rule : {#macro(burning)} 1 {(0,0,0) != 26
AND (((0,0,0)>#macro(burning)AND(0,0,0)>60)OR
(#macro(burning)>(0,0,0)AND (0,0,0) >= 300))}
%Burned
27  rule : {26} 1 {(0,0,0) <= 60 AND (0,0,0)
!= 26 AND (0,0,0) > #macro(burning)}
%Stay Burned or constant
28  rule : {(0,0,0)} 1 {t}
```

Figure 5: Fire Spread Model Specification

In lines 20 and 21, the zone *ti* represents the plane 1 in charge of storing ignition times. These rules show how we store the ignition times: if the corresponding cell in the propagation plane 0 begins to burn, we record the current time in the cell.

In the propagation plane we defined a border zone *cst* corresponding to the model behavior in the boundaries of the cell space (see equation 1d). The cells stay at the ambient temperature (lines 22 and 23). The rules used to compute the cell temperature start in line 24. The first one corresponds to the *unburned* phase. If the computed cell temperature is higher than the present value, the cell will take the computed value. The same occurs during the transient period (simulation time smaller than twenty), when the state of the cell is neither *burning* nor *burned*. Based on the same principles, the rules lines 26 and 27 correspond to the phases *burning* and *burned*.

The last rule in line 28 is used for cells that are away from the fire front or *burned*. These cells keep their value at the next time step.

The model specification was simplified using macros. Figure 6 shows the file where the macros were defined, which contains the rules corresponding to the temperature calculus when the cell is in phase *unburned* or *burning*. The file is included in the model specification using the #include directive in line 00 of Figure 5.

```
#BeginMacro(heating)
(0.98689 * (0,0,0) + 0.0031 * ( (0,-1,0) +
(0,1,0) + (1,0,0) + (-1,0,0) ) + 0.213)
#EndMacro

#BeginMacro(burning)
(0.98689 * (0,0,0) + 0.0031 * ((0,-1,0) +
(0,1,0) + (1,0,0) + (-1,0,0) ) + 2.74 * exp(-
0.19 * ((time + 1) * 0.01 - (0,0,1))) +0.213)
#EndMacro
```

Figure 6: Definition of the Macros: rules.inc

The simulation results obtained when executing this model are displayed in Figure 7 and 8. The white squares represent the experimental front. In Figure 7, a first observation at t=30s of the circular wave front give the same results. Nevertheless, in Figure 8, a difference appears on the simulated fire front width at time 50s. Indeed, the fire front cools more quickly. This can be explained by the end combustion assumption (T_f = 60 °C) effect.

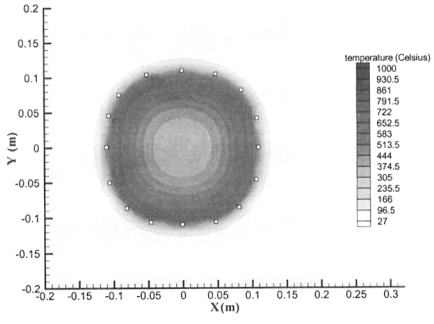

Figure 7: Comparison between Experimental and Simulated Circular Fronts at t=30s

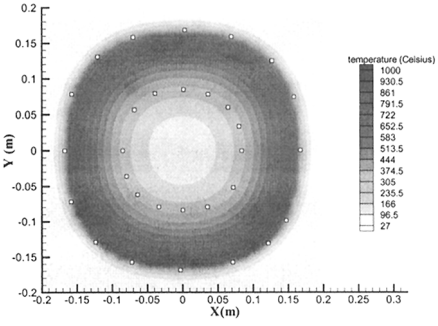

Figure 8: Comparison between Experimental and Simulated Circular Fronts at t=50s

In Figure 9, we can note that after an acceleration stage corresponding to the initialization, the rate of spread of the fire front becomes quite constant.

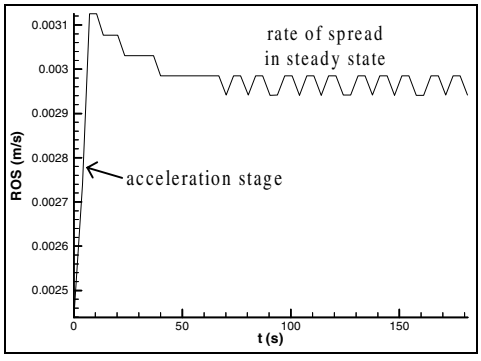

Figure 9: Predicted Rate of Spread Over the Time

The use of Cell-DEVS will enable us to take easily into account the semi-physical model behavioral modifications for wind and slope effects and non-homogenous vegetation. Nevertheless, execution time needs to be improved in order to provide real-time recommendations. Cell-DEVS allows simulating continuous systems by means of events, which produce a high degree of overhead with the message inter-module interactions. In order to minimize these interactions, we have developed and applied quantized Cell-DEVS mechanisms.

3 QUANTIZING CELL-DEVS

Numerical problems are very sensitive to temporal and spatial conditions. Quantizing the space changes the time step and can make the system diverge. Therefore, different methods, with regard to time and space, have to be defined to manage with the error induced by quantization.

We applied Cell-DEVS quantization to a 20x20 propagation domain simulating a real propagation of 20s. The temperature of each cell was quantized using different quantum sizes. In Figure 10, we observe the influence of the message reduction on the execution time and the number of messages involved.

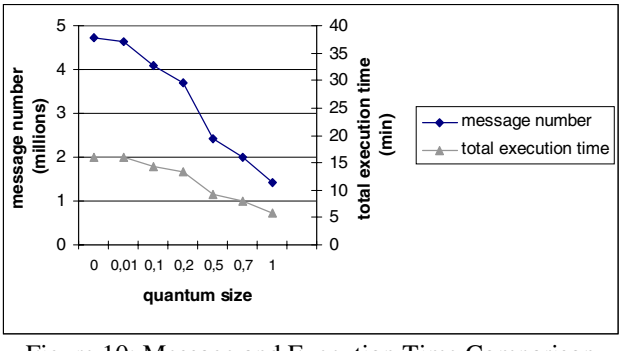

Figure 10: Message and Execution Time Comparison

To make qualitative comparison of the results, we define an average relative error for quantized results:

$$\bar{\varepsilon} = \sum_{i=0}^{n} \frac{1 - \frac{s_i}{q_i}}{n} \quad (3)$$

With n = number of cells
s_i = Value of cell i on the output without quantum
q_i = Value of cell i on the output with quantum

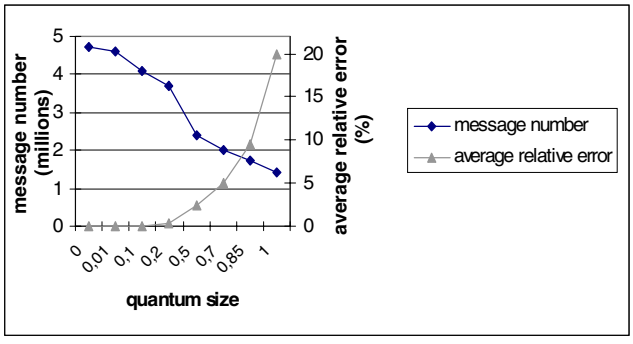

Figure 11: Message and Error Comparison

In Figure 11, we note that the use of quantization introduces an exponential error while the message number diminishes. The average relative error of the quantized domain calculation can be expressed as follows:

$$\bar{\varepsilon} = 0.21\,(e^{4.5q} - 0.21) \quad (4)$$

If we have good results incurring little error, the error does not converge to a fix value for high quanta (see Figure 12) while after the initial ignition the rate of spread of the fire front became quite steady.

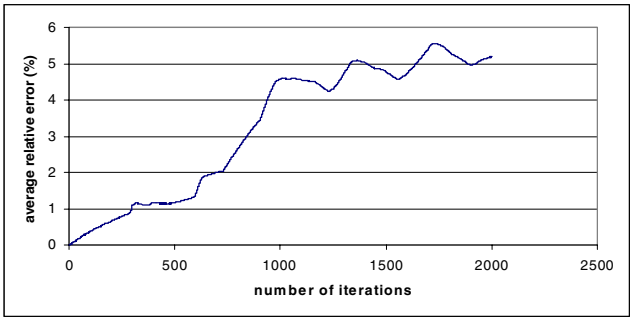

Figure 12: Average Relative Error Convergence for q=0.7

If the quantum is too large and the energy brought by the neighboring cells is not important enough, the temperature of the cell cannot reach the boundary and remains between two states. Hence, the cell is considered inactive. Dynamic Cell-DEVS quantization has been developed to deal with this problem.

4 DYNAMIC QUANTIZATION

In order to improve the error rate while keeping the number of messages small, we dynamically changed the size of the quantum according to a ratio in order to reduce the error introduced by quantization. The quantum size increases or decreases of the ratio according to the level of activity of the cell (see Figure 13). The level of activity is measured by seeing how much the cell changes. If the value of the cell passes the threshold, the quantum is increased of the ratio. If the cell's value does not pass the threshold, the quantum decreases of the ratio.

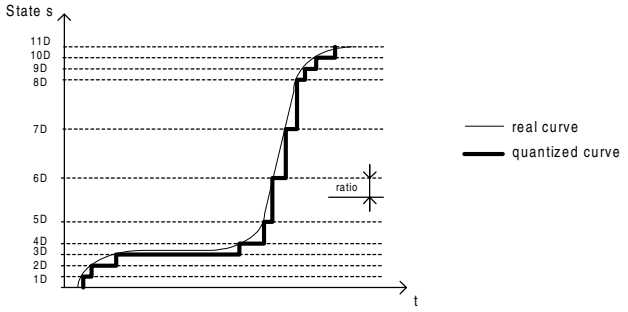

Figure 13: Dynamic Quantization of Real Curve

In Figure 14, we use different ratios to improve the results obtained for q=1. The greater the ratio is the lower the error is the longer the calculation time is. For a ratio of 0.1%, the error decreases from 21 % to 9.2 % inducing small execution time overhead (from 5 min 50 s to 6 min 33 s).

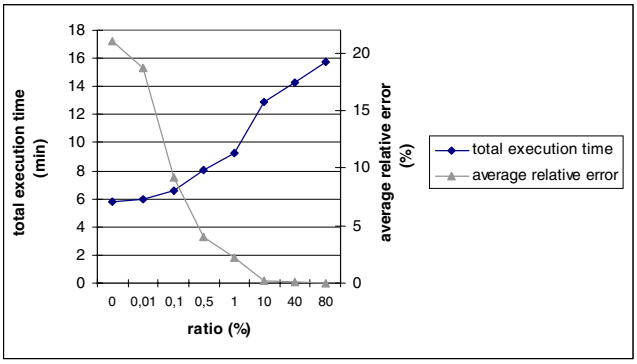

Figure 14: Error and Execution Time Comparison for q=1

We plot the results for Dynamic quantization improvements in Figure 15. This solution consists in reducing the error (inducing small overhead) for high quanta and to reduce the execution time for low quanta, which have very small error.

Comparing standard and Dynamic quantization, the error reduction leads to an execution time reduction. For a quantum of 0.85, we obtained an execution time of 7 min 7 s and an error of 9.6 % with the basic quantization. Using a

dynamic quantum of 1 and a ratio of 0.1%, the error is about 9.2 % for an execution time of 6 min 33s.

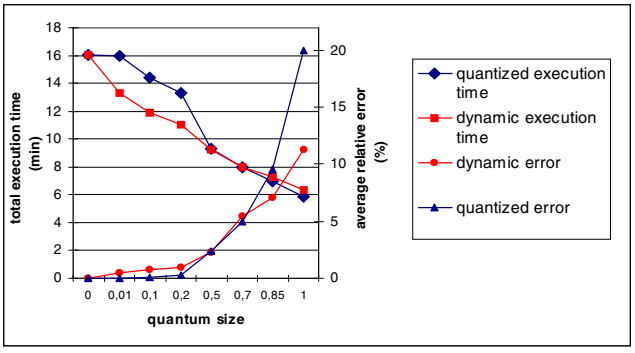

Figure 15: Dynamic Quantization Gain

The Dynamic quantization allowed us to optimize the quantum size of each cell according to cell's phase. Hence, the error and the execution time have been reduced. Nevertheless, error still does not converge for high quanta. Discrete quantization has been investigated to enhance the control of the error.

5 DISCRETE EXPERIMENTATION

In our fire-spread model, time is discrete and hence advances from time t to t+1. Basic quantization does not work this way.

Figure 16 depicts the difference between basic and discrete quantization. We have chosen the case where, with basic quantization, the future value computed by the cell at time t1 does not reach the boundary (i+1)D. Thereby, when the quantizer compares this intermediate value with the old one (the last boundary iD), the cell will be considered to be quiescent. Unless a neighbor reactivates it immediately (which would be pure causality), the cell will not do anything else. This problem appears especially in the *heating* phase, where the temperature growth is not as important as in the *burning* phase (see Figure 4). As basic quantization does not take well into account this heating, the error is accumulated during the whole simulation.

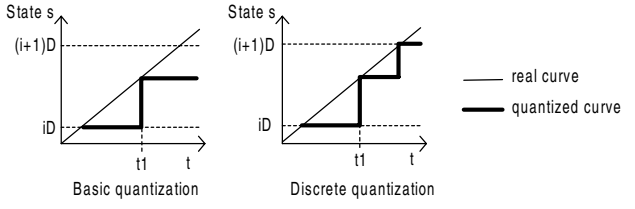

Figure 16: Discrete Quantization of a Real Curve

To experiment discrete quantization we added a neighboring cell to each cell of the propagation domain. This cell changes its state in each time step in order to reactivate the quantized one, which re-computes its state and hence passes the boundary. When the boundary is crossed the cell returns to a quiescent state. This extension takes longer than basic and dynamic quantization but we experiment the error gain of this method.

In Figure 17 and 18, we observe that, due to discrete verification, the quanta used are larger. Nevertheless, above q=30, Discrete quantization has not any more effect on message number and execution time for high quanta.

Physically this message reduction can be explained. In small quanta, the message gain is realised because the active cells amounts to the fire front and the whole cells (even the cells away from the fire front) easily reach the quantum boundary and pass to quiescent state. In higher quanta, all the cells of the domain try to reach the boundary and cells away from the front are constantly calculating their temperatures increasing the number of messages.

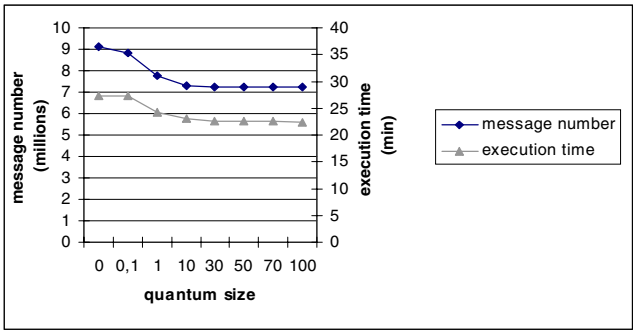

Figure 17: Message and Execution Time Comparison

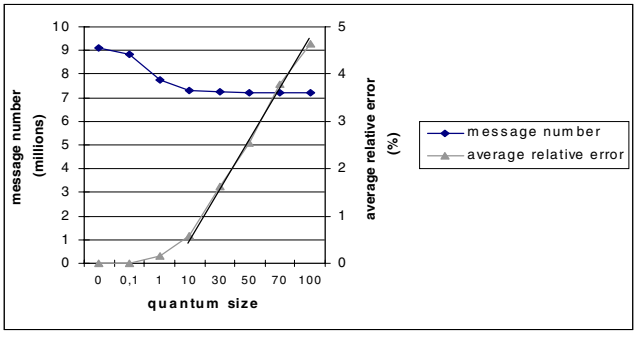

Figure 18: Message and Error Comparison

The average relative error becomes linear and can be approximated by:

$$\bar{\varepsilon} = 0 \qquad for\ q \in [0, 3.4[\qquad (5)$$

$$\bar{\varepsilon} = 1.0272D - 3.5261 \qquad for\ q \in [3.4, 100] \qquad (6)$$

Moreover, in Figure 19, we note that the error converges now for every value of the quantum and is smallest than one of basic quantization.

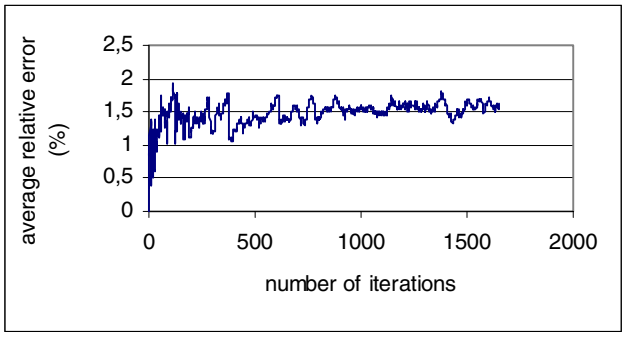

Figure 19: Average Relative Error Convergence for q=30

To reduce the error, we combined Dynamic and Discrete quantization. Figure 20 shows that for a very small execution time overhead, error becomes insignificant for all quanta sizes. Moreover, as depicted in Figure 21, the error for q=30 regresses now to very small value.

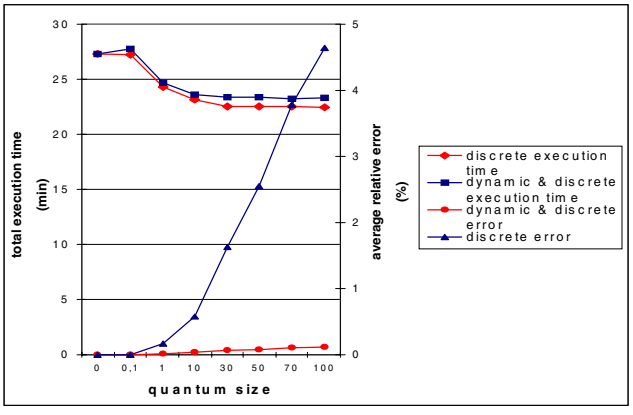

Figure 20: Dynamic and Discrete Quantization Gain

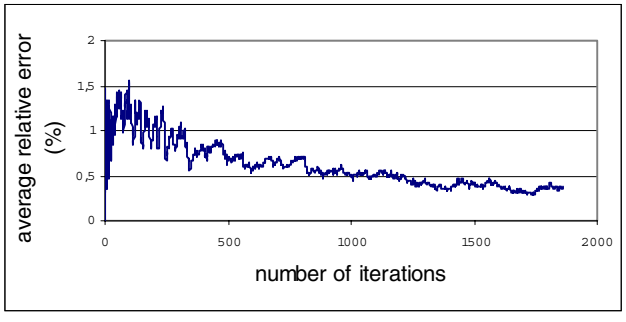

Figure 21: Average Relative Error Convergence for q=30

Figure 22 sums up the error gain obtained by the different quantization methods used. However, observing Figure 11, 18 and 20 we can notice that the error is now totally controlled.

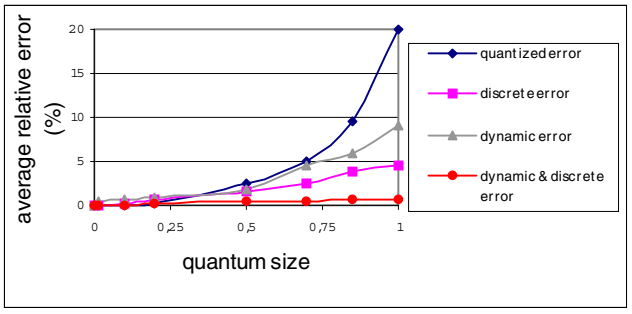

Figure 22: Average Relative Error Comparison

6 CONCLUSION

We have presented the use of the Cell-DEVS formalism to model a semi-physical fire spread model. The advantages of our modeling method can be summarized as follows:

- The evolution of propagation models is eased by the hierarchy description of DEVS and the high-level language of Cell-DEVS,
- Events and quantization allow us to activate only the most heated cells of the fire front,
- Quantization reduces simulation time incurring error,
- Dynamic quantization reduces the error slightly reducing or increasing the simulation time,
- Discrete quantization permits to highly reduce and control the error.

In a first time, we plan to implement the discrete quantization in order to enhance the execution time gain. Once the simulation will be optimized on a single processor, parallel simulation seems to be the only possibility to simulate this type of processes on large scales. In these cases, the use of quantized DEVS will be interesting to reduce the message number of parallel and distributed simulations.

REFERENCES

Balbi J.H., P.A. Santoni, and J.L. Dupuy. 1998. Dynamic modeling of fire spread across a fuel bed, *Int. J. Wildland Fire*, 275-284.

Muzy A., G. Wainer, E. Innocenti, A. Aiello, J. F. Santucci. 2002. Comparing simulation methods for fire spreading across a fuel bed. In *Proceedings of AI, Simulation and Planning in High Autonomy Systems, AIS'2002,* ed. F.J. Barros and N. Giambiasi, 219-224. Lisbon, Portugal.

Rodriguez D. and G. Wainer. 1999. New extensions to the CD++ tool. In *Proceedings of the 32nd SCS Summer Computer Simulation Conference.* Vancouver, Canada.

Santoni P.A. 1997. Propagation de feux de forêt, modélisation dynamique et résolution numérique, validation sur

des feux de litière. *Ph.D. thesis of the University of Corsica.* France.

Wainer G. and B.P. Zeigler. 2000. Experimental results of Timed Cell-DEVS quantization. In *Proceedings of AIS'2000.* Tucson, Arizona. U.S.A.

Wainer G. and N. Giambiasi. 2001. Application of the Cell-DEVS paradigm for cell spaces modeling and simulation. *Simulation.* Vol. 76, No. 1. January 2001.

Zeigler B.P. 1998. DEVS theory of quantized systems. Advanced simulation technology thrust DARPA contract.

Zeigler B.P., H. Praehofer and T. G. Kim. 2000. Theory of modeling and simulation. Second Edition, Academic Press.

AUTHOR BIOGRAPHIES

ALEXANDRE MUZY received his M.Sc. from the University of Corsica, France, in 2001. He is currently pursuing a Ph.D. Degree in the CNRS research laboratory UMR 6134 of the University of Corsica. His current research interests relate to the theory of modeling and simulation of complex systems, the DEVS and Cell-DEVS formalisms and the optimization of discrete-event simulation for environmental problems. His email and web addresses <a.muzy@univ-corse.fr> and <spe.univ-corse.fr/simffweb/debut.htm>.

ERIC INNOCENTI received his M.Sc. from the University of Corsica, France, in 1998. He worked as software developer for the past three years. He is currently pursuing a Ph.D. Degree in the CNRS research laboratory UMR 6134 of the University of Corsica. His current research interests relate to the theory of modeling and simulation of complex systems, the DEVS and Cell-DEVS formalisms and parallel and distributed processing. He can be reached at <ino@univ-corse.fr>.

ANTOINE AIELLO received the Ph.D. (1997) of the University of Corsica, France. He is Assistant Professor at the University of Corsica. His current research interests relate to the modeling and simulation of natural complex systems and the development of multimedia architectures. He is a member of the CNRS research laboratory UMR 6134 of the University of Corsica. He can be contacted at <aiello@univ-corse.fr>.

JEAN-FRANCOIS SANTUCCI obtained his Ph.D. in March 1989 at the Université d'Aix-Marseille – topic: a knowledge based system for testing of digital system. He has been an Associated Professor at the Université de Nîmes, France, from 89 to 95 and from 95 to 96 at the University of Corsica, France. He is Professor in Computer Sciences since 1996 at the University of Corsica. His research interests are modeling and simulation of complex systems and high level digital testing. He published about 100 papers in international conferences and journals. He has been scientific responsible for several European and international industrial contracts. He is since 1998 Adjunct Director of the CNRS Research Laboratory UMR CNRS 6134, University of Corsica. He can be reached at <santucci@univ-corse.fr>.

GABRIEL WAINER received the M.Sc. (1993) and Ph.D. degrees (1998, with highest honors) of the Universidad de Buenos Aires, Argentina, and Université d'Aix-Marseille III, France. He is Assistant Professor at the Systems and Computer Engineering, Carleton University (Ottawa, Canada). He was Assistant Professor at the Computer Sciences Dept. of the Universidad de Buenos Aires, Argentina. He has been the PI of several research projects, and participated in different international research programs. Prof. Wainer held different office positions in the Society for Computer Simulation International (SCS). He is also a Co-associate Director of the Ottawa Center of the McLeod Institute of Simulation Sciences. His email and web addresses <gwainer@sce.carleton.ca> and <www.sce.carleton.ca/faculty/wainer/celldevs/homepage.html>.