Optimization of Cell Spaces Simulation for the Modeling of Fire Spreading

Alexandre Muzy, Eric Innocenti
and Jean-François Santucci
University of Corsica
SPE – UMR CNRS 6134
B.P. 52, Campus Grossetti
20250 Corti FRANCE
e-mail: a.muzy@univ-corse.fr

David R.C. Hill

ISIMA/LIMOS UMR CNRS 6158
Blaise Pascal University
Campus des Cézeaux BP 10125, 63177
Aubière Cedex FRANCE
e-mail: Hill@isima.fr

Abstract

This paper presents a simulation performance improvement of the application of the Multicomponent Discrete Time System Specification (MultiDTSS) formalism to a fire spread. Multicomponent choice is explained through both Multicomponent systems and networks of systems critical study for cell spaces modeling and simulation. A new function has been appended to the MultiDTSS formalism to achieve the Active MultiDTSS formalism extension. This function allows reducing the calculation domain of diffusion problems to active cells thus reducing execution time. The two formalisms have been implemented and are compared with a qualitative and quantitative analysis.

1. INTRODUCTION

Cellular propagation problems are numerous in environmental risks (fire spread, oil spills, floods, gas propagation, etc.). In this type of phenomena, the amount of data to store and the number of operations are so large that computer simulation is the only issue to solve the problem efficiently [1]. However, processing such huge data requires high computer performances. Numerous present day simulators, based on simple Rothermel's semi-empirical model [2], have been developed [3, 4, 5] for fire spread simulation. However, there is no environment for modeling more precise and complex mathematical models, which require high computer capabilities.

The mathematical model we used intends to add a physical dimension to semi-empirical models. The model developed is based on a reaction-diffusion equation with a non-linear combustion energy term [6]. This class of equation does not admit analytical solution. Therefore, a discrete time model is used as an approximation of the continuous solution. The temperature of each cell of the propagation domain is given by a Partial Differential Equation (PDE). After discretizing the PDE model, a Cellular Automaton (CA) [7] is obtained. A CA represents an infinite lattice of discrete time cells. Each cell uses discrete values that change according to a local function. This is computed using the present value of the cell and a finite set of neighbors.

In a first approach we implemented our fire spread model as a simple CA [8]. Despite a resolution algorithm capable of reducing the calculation time the resultant code was very difficult to modify. More methodology was needed to deal efficiently with the model complexity and we retained the Discrete Event System Specification (DEVS) formalism [9]. This formalism allows a modular description of the model behavior encapsulated in the model definition. Nevertheless, despite a better capability for integrating fire spread model evolutions [10], DEVS semantic produced a high simulation time cost and DEVS model description was quite complex to achieve discrete time simulation. Another way to model and simulate fire spread is currently explored with the timed Cell-DEVS formalism [11] application. This formalism is a combination of the DEVS and CA formalisms with timing delays [12]. Each cell is defined as an atomic DEVS model, and a procedure to couple cells is depicted. Simulation enhancements are currently investigated with Cell-DEVS quantization techniques to reduce the message inter-module communication [13].

Always based on DEVS formalism, the Multicomponent Discrete Time System Specification (MultiDTSS), proposed in [9], permits the elimination of messages exchange overhead between cells and hierarchy levels for discrete time systems. A cellular system is described on a single hierarchy level by components (or cells) of a Multicomponent model. The components influence each other directly through their state transition functions. At each time step, the specific transition function of each cell of the infinite grid is executed. Hence, message interaction is eliminated and reduced to system synchronization improving simulation performances.

However, scanning all cells in an infinite space is impossible, but even in a finite space, scanning all cells all the time is clearly inefficient. To enhance performance we reduce the calculation domain (the set of cells which state is computed at a given time step) to the active cells of the domain. To achieve this goal a new function has been appended to Multicomponents models defining the new Active MultiDTSS (AMultiDTSS) formalism. The function behavior is implemented by means of an original algorithm allowing the limitation of the working space to the active cells.

The main contribution of the present paper consists in applying the MultiDTSS and AMultiDTSS formalisms to a fire spread across a fuel bed. The two formalisms have been implemented in C++, and a comparison between MultiDTSS and AMultiDTSS applications is provided.

The following section presents the mathematical model used to model fire spread. Then, modeling and simulation of cellular systems are presented. In the fourth section, the Multicomponent formalism improvement for cellular simulation is exposed. The fifth section is devoted to the description of the reference simulation model of fire spread and to the improved simulation algorithm. In the last section, results of basic and optimized simulation are compared quantitatively and qualitatively before making the final remarks and prospects.

1. Mathematical model of fire spread

Over the last fifty years, several efforts have been made in the field of modeling forest fire spread. The problem consists in calculating fire spread rate, flame front position and temperature distribution in complex fuel. To make real time simulation, complexity of fire spread and data volume require having simple mathematical models capable of predicting the main behavioral features of fire.

Based on Weber's classification [14], three kinds of mathematical models for fire propagation can be identified according to the methods used in their construction. The first type includes *statistical* models [15], which do not consider physical information. The second one incorporates *semi-empirical* models [2] based on the principle of energy conservation but which do not distinguish between the different mechanisms of heat transfer. Finally, *physical* models [16] describe the various mechanisms of heat transfer and take the finest mechanisms involved in fire spread into account. However, solving such models requires very long calculation times and thus they are difficult to integrate into functional fire-fighting tools.

At present, most real-time simulators are based on Rothermel's stationary model. This is a one-dimensional semi-empirical model, in which a second dimension can be obtained using propagation algorithms integrating empirically wind and slope. In the last few years, we planned to add a physical dimension to the semi-empirical models to increase their precision and integrate in a more robust manner the wind and slope effects. This model is a non-stationary two-dimensional *semi-physical* model [6].

The model uses elementary cells of earth and plant matter. The temperature of each cell is represented by a PDE:

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

$$\sigma_v = \sigma_{v0} \text{ if } T < T_{ig} \tag{1b}$$

$$\sigma_v = \sigma_{v0}.e^{-\alpha(t-tig)} \quad \text{if } T \geq T_{ig} \tag{1c}$$

$$T(x,y,t) = T_a \quad \text{at the boundary} \tag{1d}$$

$$T(x,y,t) \geq T_{ig} \quad \text{for the burning cells} \tag{1e}$$

$$T(x,y,0) = T_a \quad \text{for the non burning cells at } t=0 \tag{1f}$$

Where, considering a cell, T_a (*300K*) is the ambient temperature, T_{ig} (*573K*) is the ignition temperature, t_{ig} (s) is the ignition time, T (K) is the temperature, K (m².s^{-1}) is the thermal diffusivity, α (s^{-1}) combustion time constant, σ_v (kg.m^{-2}) is the vegetable surface mass, σ_{v0} (kg.m^{-2}) is the initial vegetable surface mass (before the cell combustion).

The model parameters are identified from experimental data of temperature versus time. The heat transfer of the model is schematized in Figure 1.

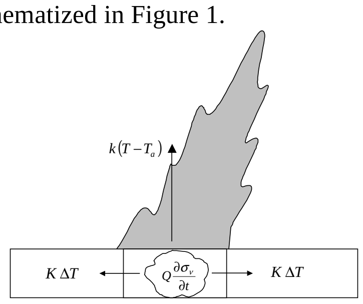

Figure 1. Heat transfer of the semi-physical model

Two numerical methods that can be used to discretize the model: the Finite Element Method (FEM) and the Finite Difference Method (FDM). In a previous study, we applied these two methods [17]. Although they provided the same results, the FEM appeared more complex to implement, and produced longer execution time. Thus, the FDM was chosen because of its simplicity and good performance.

The study domain is meshed uniformly with cells of 1-cm² and a time step of 0.01 s. The physical model solved by the FDM, leads to the following algebraic equation:

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

Where T_{ij} is the grid node temperature. The coefficients *a*, *b*, *c* and *d* depend on the time step and mesh size considered.

2. Modeling and simulation of cellular systems

Our modeling approach [18] is based on reductionism [19]. This concept consists in decreasing the complexity of a problem reducing the behavior of the system to the behavior of its parts. Abstraction and description hierarchies help us to achieve this goal. The abstraction hierarchy [20] consists in understanding the complexity focusing on pertinent information. At the same abstraction level, the description hierarchy [9] allows describing a system by a set of subsystems. Our choice to handle this complexity was to retain the abstract simulator principles [21]. An abstract simulator is an algorithmic description of how to carry out the instructions implicit in DEVS models to generate their behavior. Considering a cellular system as a black box, a distinction has to be made between system structure and behavior. Starting from system structure allows understanding its behavior. The overall system behavior is performed by entities communication. However, the multiplication of the interfaces configuration for each modular component increases algorithmic complexity [22]. It is much easier to coordinate the overall behavior in a single algorithm. This is the purpose of Multicomponent specification. At the top of Figure 2, a first level of description hierarchy is used to split the cellular system into component models. Multicomponent modeling eliminates interface configuration because components models influence each other directly through their state transition functions and the overall behavior is managed by the Multicomponent model. As depicted in Figure 2, using abstract simulation principles, simulation trees independent from the cellular models are automatically generated.

In networks of models specification, simulators are in charge of managing the atomic cells and a coordinator, corresponding to the coupled model, manages the simulators communication. Communication between simulators is performed by the exchange of messages. A root processor implements the overall simulation loop by sending messages to the coordinator.

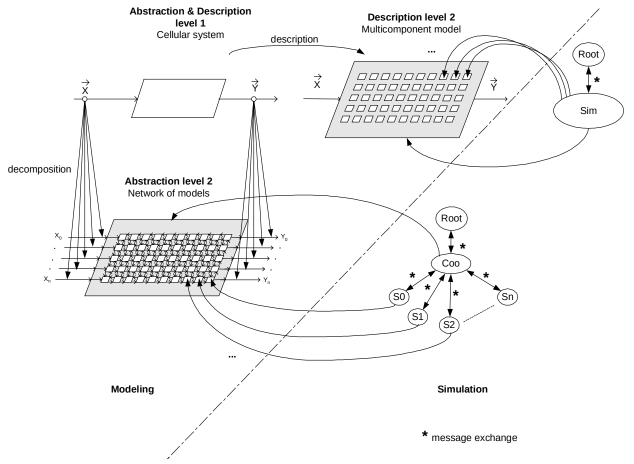

Figure 2. Comparison of modular and non-modular simulation

In Multicomponent models specification, simulators and coordinator are imploded in a single simulator corresponding to both components and Multicomponent model. This global simulator activates component transition function. As a component's state can be directly calculated accessing its influencers, state message interaction between the cells is not needed any more. This results in a message overhead reduction, and thus in a first simulation time reduction. Moreover, canceling interface configuration used for modular communication reduces algorithmic complexity and leads to a second simulation time improvement.

3. Multicomponent improvement

DEVS unicity (or networks of systems) allows modeling and simulating all Multicomponent systems. However, Multicomponent approach reduces modeling and simulation complexity of cellular systems.

In our fire spread model the temperature of each cell is discretized. This means that if a cell is activated at time *t*, it will be reactivated at time *t+h* (where *h* is the time step). One solution could consist in applying discrete event simulation to the fire spread model to concentrate simulation to active cells. However, for numerous active cells, working on a discrete time base, discrete event simulation is inefficient and produces a high simulation cost [10, 22]. Therefore, we choose here to use the MultiDTSS formalism for modeling and simulation of discrete cell spaces. We propose a formalism extension to concentrate simulation to the cells, which can potentially change state at next time step.

3.1. MultiDTSS formalism

A Multicomponent Discrete Time System Specification (MultiDTSS) for cellular models is a structure:

$$MultiDTSS = <X, D, \{M_{ij}\}, h>$$

With

X is an arbitrary set of input values,

h is the interval to define the discrete time base,

$D = \{(i, j) / i \in I, j \in I\}$ is the index set.

For each $d \in D$, the component M_d is specified as

$$M_d = <Q_{i,j}, Y_{i,j}, I_{i,j}, \delta_{i,j}, \lambda_{i,j}>$$

Where $Q_{i,j}$ is an arbitrary set of states of outputs of d, $I_{i,j} \subseteq D$ is the set of influencers of d, $\delta_{i,j} : \underset{i \in I_d}{X} Q_i x X \rightarrow Q_{i,j}$ is the state transition function of d, and $\lambda_{i,j} : \underset{i \in I_d}{X} Q_i x X \rightarrow Y_{i,j}$ is the local output function of d.

In a MultiDTSS, the set of components jointly generate the dynamics of the system. Each of the components owns its local state set, a local output set, a local state transition and an output function. Based on the state of the influencing components $i \in I_d$ and on the current input value, the component determines its next state and its contribution to the overall output of the system.

In propagation problems, an individual component has a set of influencing components, called its neighbors. Its state transition function defines new state values only for its own state variables and does not set other component's states directly. It only influences them indirectly because the state transitions of the other components depend on its own state. Therefore, the next state function of the overall Multicomponent system can be visualized as having all the components look at the state values of their individual influencers. Each component (or cell) has its own state variables, its influencers (its neighborhood) and its behavior.

3.2. Active MultiDTSS extension

In discrete time cellular systems, at every time step only a few components generally undergo a state transition. In this case all the cells do not need to be scanned at every time step as in a CA. Hence, calculating only the future state of the active cells improves the simulation performances.

As schematized in Figure 3, we use here the basic ideas exposed in [9] to predict whether a cell will possibly change state or will definitely be left unchanged in a next global state transition: *A cell will not change state if none of its neighboring cells changed state at the current state transition time.*

We depict here the approach step by step:

- In a state transition mark, the cells that changed state,
- From those, collect the cells that are their neighbors,
- The set collected contains all cells that can possibly change at next step,
- All other cells will definitely be left unchanged.

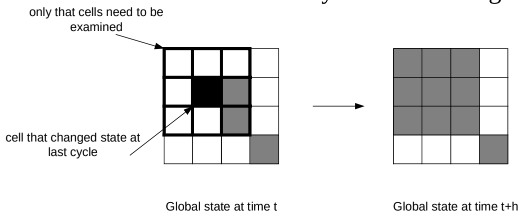
only that cells need to be examined
cell that changed state at last cycle
Global state at time t
Global state at time t+h

Figure 3. Collecting active cells

Based on these principles, we define here the Active Multicomponent Discrete Time System Specification (AMultiDTSS) extension of MultiDTSS:

$$AMultiDTSS = <X, D, \{M_{ij}\}, h, Activate>$$

Where X, h, M_{ij} and D are the same as for the MultiDTSS system. The only difference with the MultiDTSS structure is the *Activate* function, *Activate : D* $\rightarrow$ *D'*, which computes the new index set *D'* from the component states of the old index set *D*. The new index set is obtained scanning the state of each cell of the old set:
$D' = \left\{(i, j) / \ni (k,l) \in I_{i,j} / q_{k,l}^{t} \neq q_{k,l}^{t+h}\right\}$. Hence, component *d* is activated if one of its neighbors or *d* itself changed state.

4. Simulation models of fire spread

The object oriented simulator we developed enables the direct expression of models in MultiDTSS formalism. As depicted in Figure 4, our simulator is composed of four main objects: root coordinator, simulator, Multicomponent and components.

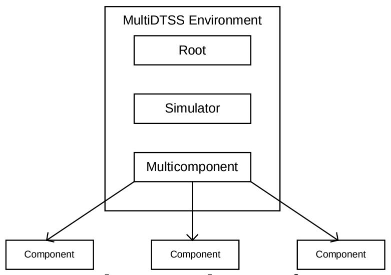

Figure 4. The main objects of our simulator

Based on the numerical model of fire spread (2), basic behavior of the cells has now to be specified. Diffusion processes spread by degrees. As fire proceeds, diffusion slightly heats cells away from the fire front. Neglecting the temperature of these cells allows reducing the calculation domain, thus saving simulation time.

4.1. Basic simulation model

Figure 5 depicts a simplified temperature curve of a cell in the domain. We consider that above a threshold temperature, T_{ig} the combustion occurs and below a T_f temperature the combustion is finished. The end of the real curve is purposely neglected to save simulation time. Three phases corresponding to cell's behavior are defined from these assumptions. A cell has phases *unburned*, *on fire* and *burned*.

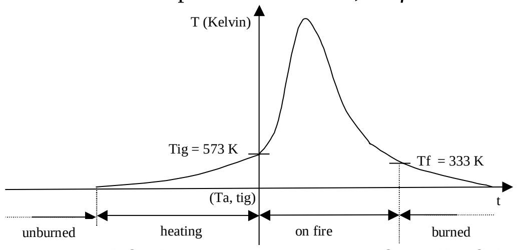

Figure 5. Simplified temperature curve of a cell of the domain

In Figure 6, we present the output and transition state functions used to model fire spreading. The output function is used to return the temperature of a cell to the root for printing. The transition state function allows computing the next state of the component. This new state is computed using current state, input and the influencer set I_d of components.

```
//state variables
temperature=300;
ignitionTime= 0;
temperatureState = ’A’;

//’A’: the cell is at the ambient temp.
//‘U’: the cell is unburned
//‘O’: the cell is on fire
//‘B’: the cell is burned

spreadState=’I’;

//’I’: propagation state = Inactive
//‘Bo’: bordering state
//‘H’: heated state
//‘A’: active state

cellBorder=false;

//output function of the cell
λ() {  return temperature }

//transition state function of the cell
transitionF(I1,I2,I3,I4,x)

  //equation constants
  a, d, c,  alpha, deltat

  If(!cellBorder)
  Then              //limit condition
    Switch(getState())
      Case ‘A’:
        NextTemp ← 300
      EndCase

      Case ‘U’:
        nextTemp←d*temperature+a*(I1+I2+I3+I4)+c*e
        xp(-alpha*((x) *deltat))+0.213

        If(nextTemp >= 573) Then
          setIgnition(time)
            //record ignition time
            //-->new state
          setState(‘B’)
        Endif
      EndCase

      Case ‘O’:
        If(next_temp > 373) Then
          nextTemp←d*temperature+a*(I1+I2+I3+I4)+
          c*exp(-alpha*((x)*deltat-
          ignition*0.01))+0.213
        Else
          setState(‘B’)
          nextTemp←300
        EndIf
      EndCase

      Case ‘B’:
        nextTemp←300
      EndCase

    End Switch
  EndIf
End Function transitionF(I1,I2,I3,I4,x)
```

Figure 6. Cells' behavior algorithm

4.2. Simulation model improvement

As illustrated in Figure 3, if cells have adjacent neighborhood it can be noticed that amongst the cells, which need to be examined, some are already active. In this case, it is easy to understand that if cells remain active during a long simulation time the number of active cells will quickly grow requiring more and more computer capabilities. This is the problem of fires, which spread by degrees.

To obtain good performance the entire active cells cannot be tested. A new algorithm, which consists in testing only the neighborhood of the active bordering cells of a propagation domain, has therefore been defined for this type of phenomena. The algorithm is presented in the "activeCalculus()" method of Figure 6. For comparison, the basic discrete approach calculating all the cells of the domain is also described in the "discreteCalculus()" method of Figure 7.

In discrete approach all the transition functions of the cells are calculated at each time step and stored in a temporary table. Our approach consists in controlling the calculation domain modifications by testing the temperature gradient at the edge of the domain. Hence, the domain is increased or decreased by testing the temperatures at its borders. As long as these temperatures are not very important the domain remains the same.

Propagation phases, defined in the cell algorithm of Figure 6, have been added to the cells:

- *inactive* at the initialization,
- *heated* for the cells in front of the bordering cells,
- *active* for the cells behind the bordering cells,
- *bordering* for the bordering cells.

```
discreteCalculus Function()

    For all the cells of the domain Do
      stateComponent(i,j)←transitionF(i,j,t)
    EndFor

End Function discreteCalculus()
```

```
activeCalculus Function()

For all the cells of the index set Do

    stateComponent(i,j)←transitionF(i,j,t)

    If(stateComponent(i,j)<>'B')
    Then
        // the cell is not burned
       Add the cell to the new index set
    EndIf

    If(getSpreadStateComponent(i,j)=='Bo')Then
      If (all the neighbors<>'H') Then
         setSpreadStateComponent(i,j,'A')
      EndIf
    Else
      If (one of the neighboring cell is heated)
         stateComponent(i,j)←transitionF(i,j,t)

         If(spreadTest(nextTemp)) Then
            setSpreadState(i,j,'Bo')
            add the cell to the new index set
         EndIf
      EndIf
    EndIf
EndFor

Replace the old set by the new one

End Function ActiveCalculus()

bool spreadTest(temp) Function

    If(temp >= (300 + x))
       return true
    Else
       return false
    EndIf

End Function spreadTest(temp)
```

Figure 7. Discrete and active calculus functions embedded in the simulator

As depicted in Figure 8, only the bordering cell tests its neighborhood, this allows reducing the number of testing cells. The test, described in the spreadTest() function of Figure 7, depends on the temperature of the heated cells. If the temperature of a heated cell is greater than *Ta + x* Kelvin, the cell becomes a bordering cell. When a bordering cell has not heated neighbors any more the cell becomes active.

The cells of the calculation domain (inactive and bordering cells) correspond to the index set of the simulator. At the initialization, the index set is composed of all the ignited cells. At each time step, new heated cells are added to the index set and the burned cells are removed from the former to achieve the new index set.

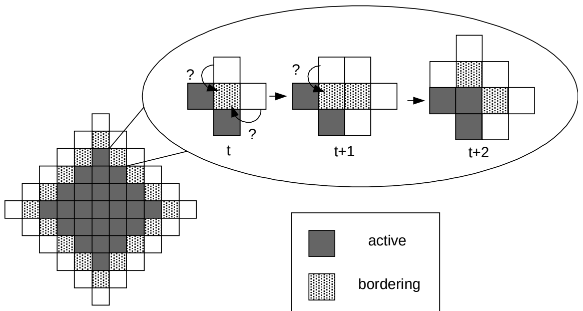

Figure 8. Calculation domain evolution

5. Simulation results

We used experimental fires conducted on Pinus Pinaster litter, in a closed room without any air motion, at the INRA (Institut National de la Recherche Agronomique) laboratory near Avignon, France [23]. These experiments were performed to observe fire spread for point-ignition fires under no slope and no wind conditions. The experimental apparatus was composed of a one square meter aluminum plate protected by sand. A porous fuel bed was used, made up of pure oven dried pine needles spread as evenly as possible on the total area.

As depicted in Figure 9, a point-ignition has been simulated by initializing center cells with a temperature gradient. This gradient avoids not to create thermal shock having more homogenous initial conditions for the semi-physical model. The simulated fire front obtained igniting the center of the plate is represented in Figure 10. Fire spreads circularly and symmetrically up to the plate borders.

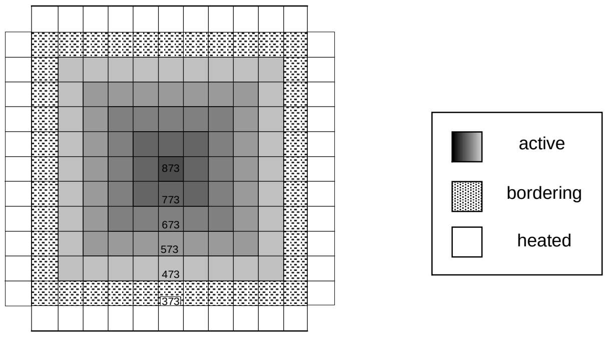

Figure 9. Initial evolving calculation domain

Figure 11 depicts the simulation time gain for different temperature gradient and a real propagation of *200s* on a processor *AMD Duron 500 MHz*. With basic discrete simulation, the propagation is simulated in *160s*. For a temperature gradient of only *1K*, the performance improves by *20s*. From a temperature gradient of *10K* to *30K*, the performance gain remains about *30s*. The simulation time drops to *100s* for temperature gradients greater than *40K*

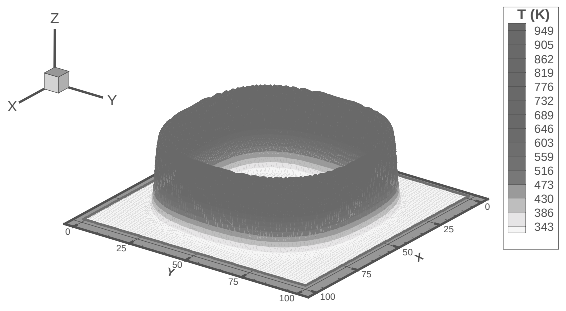

Figure 10. Fire spread simulation of a point ignition

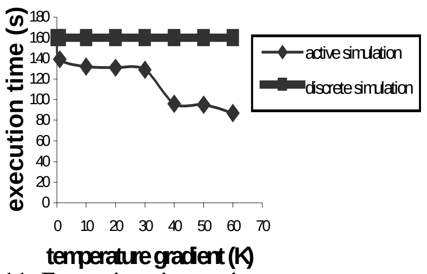

Figure 11. Execution time gain over temperature gradient of the test for a real propagation of *200s*

Figure 12 compares the fire fronts generated for different temperature gradients. For temperature gradients of *1K* and *0K* the fires fronts are similar. Up to a *30K* gradient the fire fronts obtained with active simulation are slightly ahead of the fire front obtained without temperature gradient. Above a temperature gradient of *40K*, we observe that the simulated fire front suddenly becomes incoherent. Small fire fronts propagate according to cardinal directions.

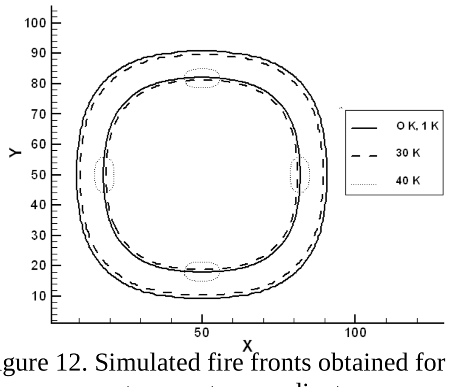

Figure 12. Simulated fire fronts obtained for different temperature gradients

Figure 9 helps to comprehend this result. We understand that the gradient test condition becomes too important above *40K*. The bordering cells of the point ignition continuously test the heated cells. Until the heated cells do not satisfy the temperature gradient test, the fire does not propagate. Hence, because of cardinal neighborhood the corner cells next to the heated ones are not ignited. This results in four linear fire fronts propagation. Below a temperature gradient of *40K*, initial ignition is boosted by the evolving conditions and the fire front is speeded up.

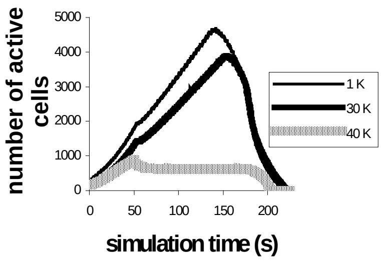

Figure 13. Evolution of the number of active cells during the simulation for different temperature gradients

The evolution of the number of active cells during the simulation for different temperature gradients is presented in Figure 13. We notice that even with a temperature gradient of *1K* the number of active cells never exceeds *5000* cells i.e. less than one half of the total number of cells. Under *50s* fire speed grows quickly, this is the transitory state [6]. Above *50s*, fire front speed reaches a steady state. This explains the change of slope at *50s* for the number of active cells. Above *150s*, the number of active cells decreases because fire reaches the plate borders.

6. Conclusion

A high performance fire spreading simulation based on Multicomponent formalism has been proposed which facilitates the modifications of cells behavior. The Multicomponent formalism has been improved by adding an active function. An original algorithm has been implemented to focus the simulation on active cells of an infinite lattice for diffusion problems. The boundary test of the algorithm allows a generic simulation of diffusion problems whilst incurring slightly controlled error. Fire spread simulation allowed us to study finely active algorithm performance through physical phenomena. Relevance and domain of validity of the algorithm have been studied through the number and percentage of active cells. According to different gradient tests, the error has been quantified.

Interesting time improvements and a very few errors have been obtained for small gradient temperatures. Since there were few active cells during the simulation, performance gain are excepted to be more effective for higher domain sizes. The combination of the Multicomponent formalism with our algorithm enables us to obtain performances under real-time deadlines. We need now to develop a more sound and formal approach enabling to model and simulate cellular systems undergoing structural changes.

Currently, few works exist in structurally dynamic discrete time cellular systems. The most relevant are Structurally Dynamic Cellular Automata (SDCA) [24]. Nevertheless, SDCA are highly restricted by the finite state of cellular automata. Hence, we aim to use the Dynamic Structure Discrete Time System Specification (DSDTSS) formalism [25] for modeling and simulating multicomponent discrete time systems undergoing structural changes (calculation domain restriction, neighborhood changes etc.). Fire spreading applications will allow us to validate this new approach.

10. References

[1] D. Hill "Object-Oriented Analysis and Simulation", Addison-Wesley, 1996.

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

[3] M. Finney and K.C. Ryan "Use of the FARSITE fire growth model for fire prediction in US National Parks", in Proceedings of the *International Emergency Management and Engineering Conference,* San Diego, CA: SCS, 1995, pp. 183-189.

[4] P.L. Andrews "Behave: Fire behavior prediction and fuel modelling system-BURN Subsystem", part1, USDA Forest Service, Intermountain Forest and Range Experiment Station, General technical Report INT-19, 1986.

[5] J.D. Miller and S.R. Youl "Modelling fire in semi-desert grassland/oak woodland: the spatial implications", *Ecological Modelling*, volume 153, issue 3, 2002, pp.229-245.

[6] J.H. Balbi and P.A. Santoni "Dynamic modelling of fire spread across a fuel bed", *Int. J. Wildland Fire*, 1998, pp. 275-284.

[7] S. Wolfram "Theory and applications of cellular automata", Advances series on complex Systems, 1, World Scientific, Singapore, 1986.

[8] P.A. Santoni "Elaboration of an evolving calculation domain for the resolution of a fire spread model", *Numerical heat transfer*, Part A, (33), 1998, pp. 279-298.

[9] B.P. Zeigler, H. Praehofer and T.G. Kim "Theory of modelling and simulation", 2nd Edition, Academic Press, 2000.

[10] A. Muzy, T. Marcelli, A. Aiello, P.A. Santoni, J.F. Santucci and J.H. Balbi "Application of DEVS formalism to a semi-physical model of fire spread across a fuel bed, 13th European Simulation Symposium and Exhibition (South France), 2001, pp. 641-643.

[11] G. Wainer and N. Giambiasi "Application of the Cell-DEVS paradigm for cell spaces modelling and simulation", *Simulation*, 76 (1), 2001, pp. 22 - 39.

[12] N. Giambiasi and A. Miara "SILOG: A practical tool for digital circuit simulation", Proceedings of the *16th D.A.C.*, U.S.A., 1976.

[13] A. Muzy, E. Innocenti, G. Wainer, A. Aiello and J.F. Santucci "Cell-DEVS quantization techniques in a fire spreading application", proceedings of the *Winter Simulation Conference 2002 – Exploring new frontiers,* San Diego, USA, 2002, pp. 542-549.

[14] R.O. Weber "Modelling fire spread through fuel beds", Progress in *Energy and Combustion Science*, 17, 1990, pp. 67-82.

[15] A.G. McArthur "Weather and grassland fire behaviour, Australian Forest and Timber Bureau Leaflet, 100, 1966.

[16] F.A. Albini "A model for fire spread in wildland fuels by radiation", Combustion Science and Technology, 1985, (42) 229 - 258.

[17] P.A. Santoni "Propagation de feux de forêt, modélisation dynamique et résolution numérique, validation sur des feux de litière", PhD thesis of the University of Corsica, 1997.

[18] A. Aiello, J.F. Santucci, P. Bisgambiglia and M. Delhom "An object oriented simulation framework based on hierarchical multi-views modelling concepts", in proceedings of *Object Oriented Simulation*, SCS, Sherton Crescent Hotel, Phoenix, Arizona, 1997, pp. 73-78.

[19] H. Simon "The sciences of the artificial", Cambridge MIT Press, 1969.

[20] C. Oussalah, N. Giambiasi and R. Lbath "A framework for modelling and linking the structure and the behavior of a system", in P. Born, Artificial Intelligence in Scientific Computation : Towards Second Generation Systems. J.C. Baltzer AG, Scientific Publish. Comp. Basel-Switzerland, 1989.

[21] B.P. Zeigler "Multifacetted modelling and discrete event simulation", Academic Press, London and Orlando, FL, 1984.

[22] A. Muzy, G. Wainer, E. Innocenti, A. Aiello and J.F. Santucci "Comparing simulation methods for fire spreading across a fuel bed", in proceedings of the *AIS 2002 – Simulation and planning in high autonomy systems* conference, Portugal, 2002, pp. 219-224.

[23] J.H. Balbi, P.A. Santoni and J.L. Dupuy "Dynamic modelling of fire spread across a fuel bed", *Int. J. Wildland Fire*, 1998, pp. 275-284.

[24] Hilachinski and P. Halpern "Structurally dynamic cellular automata", *Complex Sytems*, 1987 , (1), pp. 503-527.

[25] F. Barros "Dynamic Structure Discrete Event System Specification Formalism", Transactions of the Society for Computer Simulation, 1996, (1), pp. 35-46.