From classical infinite space–time CA to a hybrid CA model for natural sciences modeling

C.R. Calidonna a,⇑, A. Naddeo b, G.A. Trunfio c, S. Di Gregorio d

a CNR-ISAC Istituto di Scienze dell’Atmosfera e del Clima, Area Industriale, Comparto 15, 88046 Lamezia Terme (CZ), Italy
b CNISM, Unità di Ricerca di Salerno and Dipartimento di Fisica “E.R. Caianiello”, Università degli Studi di Salerno, Via Ponte Don Melillo, 84084 Fisciano (SA), Italy
c Facoltà di Architettura, Università degli Studi di Sassari, 07041 Alghero (SS), Italy
d Dipartimento di Matematica, Università della Calabria, Via P. Bucci, 87036 Rende (CS), Italy

ABSTRACT

Complex phenomena occurring in natural sciences are usually characterized by a non trivial interplay between microscopic and macroscopic dynamics, which can be successfully captured by the cellular automata (CA) computational paradigm [1]. In this paper we show that some approximation of the classical CA paradigm is needed in order to properly deal with complex dynamical systems. Real phenomena can be efficiently modeled and simulated by introducing a modified CA approach, the CANv2 [2]. In this way one takes into account multiscale dynamics, through approximate infinite and/or infinitesimal dynamical stages, by means of a hybrid network of standard CA components and global operators. The power of the CANv2 approach is fully exploited by discussing three examples borrowed from the realm of natural science: debris flows after a landslide [3–5], superconductive devices [2] and forest fires spread [6,7]. Advantages and limitations of our computational model explicitly arise when examples are discussed.

© 2011 Elsevier Inc. All rights reserved.

1. Introduction

Cellular automata (CA) are a powerful tool for modeling and simulation of complex dynamical systems, whose evolution can be described by considering only local interactions between their elementary parts. The strategy is the decomposition of a complex phenomenon into a finite number of elementary processes, the overall dynamics being the combination of them. Thus CA provide useful models for a lot of applications in natural sciences, ranging from the simulation of fluid dynamics to physical, chemical and geological processes.

Often, in modeling and simulating complex systems some properties of locality get lost and the application of the standard CA model becomes very difficult. As a consequence an extension of the original CA model turns out to be useful in order to take advantage of the parallelism source of such a model. Dealing with real phenomena leads one to introduce some modifications and generalizations of the classical CA paradigm. Introducing real numbers in the CA simulation of macroscopic real complex systems forces sometimes to work with a huge number of entities almost infinite; on the other hand, dealing with microscopic complex systems obliges us to work with very small involved particles: for instance water and debris in a landslide, water, combustion particles and molecules in a forest fire or single spins in a quantum device. Another consideration about the possible states, infinite or finite, arises in the case of reversibility of a system whose dynamic representation requires a succession of an infinite and infinitesimal number of states such that reversibility is guaranteed. Recently, in order to work in a homogeneous way, a hybrid computational model, the Cellular Automata Network version 2 (CANv2) model, has

⇑ Corresponding author.
E-mail address: cr.calidonna@isac.cnr.it (C.R. Calidonna).
been proposed [2], which has a unique capability to simulate microscopic as well as macroscopic phenomena [1]. The CANv2 model [2] extends the standard CA model introducing the possibility to have a hybrid network of standard cellular automata components and global operators. Each automaton of the network represents, for instance, a component of the physical system to be simulated and the connections among the network automata represent a disjoinable evolving law which characterizes the evolution of the physical system under study. That makes possible to exploit two different types of parallelism: a data parallelism, which comes from the use of CA classical model, and a task parallelism, which could arise when considering the network of CA. Furthermore, the presence of global operators is mandatory when an external influence, to the macroscopic complex phenomena, assumes a control role and allows for expressing any mechanism that could not be expressed in terms of local interactions. In this way the CANv2 model [2] provides the possibility to simulate a multistage evolutionary process. It allows the following facilities: to simplify the modeling of a complex system, and to codify and to classify the kind of the adopted model of simulation. We stress that we deal with a hybrid model between a functional model, in which there are directional flows of signals among transfer functions, and a spatial model. Intuitively our model allows us to determine all the components of a complex system, which are ruled by a defined transfer function. Furthermore signals are also determined, which follow a well defined directional flux: they represent properties of the components which act as input and output of each transfer function. In this way the CANv2 model is a hybrid vision and finally the whole evolution of a complex system can be divided into more than a stage. In conclusion, such a model offers global information-processing capabilities which are not explicitly represented in the elementary components of the network or in their interconnections.

The aim of this contribution is to show, through concrete examples, how the CANv2 approach works as a tool to model and simulate the dynamics of a wide range of phenomena in the realm of natural sciences. We set up a general methodological approach which proceeds starting from a standard CA model via a mapping to a network of CA and global operators. The key point in such a refinement process is the determination and simplification, in terms of interconnection of partial components and effects, of the complex phenomenon under study [8].

The paper is organized as follows. In Section 2 some considerations on infinite space and time limitation of classical CA paradigm for the simulation of complex systems are presented. The formal definition of a CANv2 network is recalled together with its main properties in Section 3, then the subsequent steps of our simulation methodology are presented in detail. In Section 4 the power of the CANv2 model in simulating micro- and macroscopic dynamics is fully exploited by focusing on three examples: debris flows after a landslide [3,8], superconductive devices [2] and forest fires spread [6]. Finally some conclusions and perspectives of this work are outlined.

2. Classical CA infinite space and time properties: Toward CA for simulation of micro and macro complex systems

The concept of cellular automaton was introduced by John von Neumann in late 1940 on looking for a formal model of self-reproducing organisms behavior. In Ref. [9] von Neumann introduced the notion of cellular space and discussed the construction of automata embedded in such a space. The cellular space is a particular space endowed with the following characteristics:

- an infinite plane is divided into squares;
- each square contains the copy of the same finite automaton (a cell is the square that embeds an automaton);
- neighboring cells are associated with the cell, so that they consist in itself and the immediate surrounding cells but the diagonal cells;
- at the time \( t + 1 \) the cell is uniquely determined by the state of the neighboring cells at time \( t \), together with the transition function \( f \) of the automaton which is associated with every cell;
- the finite automaton associated with each cell has a distinguished state \( r_0 \) called quiescent state, such that \( f(r_0, r_0, \ldots, r_0) = r_0 \);
- at each time step all but a finite number of cells are in the state \( r_0 \);
- the number of distinct states for the finite automaton associated with each cell is 29;
- a particular transition function \( f \) is specified.

Starting from this view, the simulation of CA on computers implies programming features that reproduce the CA execution model. The classical CA definition requires infinite lattices in each dimension, but the consideration of storage and computational time requires bounded lattices that lead to assign boundary conditions.

In fact simulation of CA has many implications in:

- mapping cells with available memory;
- equipping each cell with a local memory for the storage of the updated state value;
- assuring the reading only access on neighboring cells;
- creating and maintaining a lookup table for the transition rule;
- furnishing a mechanism for the update of the lattice boundary;
- creating a common clock in order to assure a synchronous cell updating;
- parallel execution.
In the past the activity of modeling complex systems led to the definition of several approaches. In general, hierarchical modeling has been showed as a powerful means to handle model complexity [10], where hierarchies define a relation in which the components are grouped into levels; the relation occurs only between adjacent levels (parent and child) and, according to a general point of view, children are not related hierarchically between them: i.e., there is not a direct relation. Although different hierarchical relations can exist for the same entities, two of them could have a main role in our modeling activity: composition, in which the higher level has its own being and employs the lower level in its behavior (in our case lower level properties could be used in the transition phase of a higher level phase); and substitution, in which there is the substitution for the lower level by the higher level. It encompasses the abstraction by reduction: it involves reducing the complexity of the lower level with a simpler formalism defined by abstraction. In our case the individuation of the lower level coincides essentially by a phase and its properties.

According to this consideration the stage of subprocesses determination is crucial in the modeling activity. Such a vision guides the use of a specialized method despite to others. First of all our models are macroscopic ones and in this context we find the same patterns and behaviors when moving from macro to micro vision. Much more complex problems then arise. Indeed when we decompose processes in order to find subprocesses we are really describing the process at the same level of the others. In a certain sense processes and subprocesses are at the same level of description, so that our model is not anymore a hierarchical model when subprocesses are made explicit. In conclusion, some limitations arise on considering the CA classical model within the context of hierarchical modeling. In particular, with respect to this vision the main characteristics of our approach are:

**Spatio-temporal evolution** in order to perform simulations, neither space nor time could be infinite or infinitesimal as it depends on computer clock unit which is finite but not infinitesimal. The simulation of a complex phenomenon is finite in space as well as in time.

**Computation** CPU and memory have a finite computational and storage unit; in the realm of classical CA this limitation should be adopted [11,12].

**Morphology** if we consider for instance environmental complex phenomena we cannot describe morphology in a simple way by decomposition in subprocesses; main and averaged altitudes should be taken especially if the simulation region is quite extended.

All these considerations on limitations of classical CA paradigm lead us to introduce the concept of a network of CA and global operators, i.e., the CANv2 model [1,2], whose nodes are the CA components and the edges are the transfer functions between the components. Summarizing, the network should not be viewed as a topological one: indeed it establishes connections between the components of the complex system to be modeled and each component is a CA. In this way the network gives rise to an information transfer and processing. A brief introduction to CANv2 model is given in the following section.

### 3. CANv2: A hybrid paradigm for the simulation of complex systems

In this section we introduce the CANv2 approach. The CANv2 model [2] provides the possibility to simulate a multistage evolutionary process. It allows to simplify the modeling of a complex system, and to codify and classify the kind of the adopted simulation model.

In fact we decided to adopt a hybrid model between a functional model, in which there are directional flows of signals among transfer functions, and a spatial model. Intuitively our model allows us to determine all the components of a complex system, which are ruled by a defined transfer function. Furthermore signals are also determined, which follow a well defined directional flux: they represent properties of the components which act as input and output of each transfer function. In this way the CANv2 model is a hybrid vision and finally the whole evolution of a complex system can be divided into more than a stage. In conclusion, such a model offers global information-processing capabilities which are not explicitly represented in the elementary components of the network or in their interconnections.

#### 3.1. CANv2: An un-informal definition

In the CANv2 model the behavior of an automaton is described by a (possibly empty) set of properties (the automaton grids), by a neighborhood type, by a boundary condition and finally by a transition function. A property corresponds either to a physical property of the system to be simulated, such as the temperature, the volume and so on, or to some other feature of the system, such as the probability for a particle to move. Furthermore each property corresponds to a computational grid of a standard CA. In this scheme a cell of an automaton consists of an array of values, each one given by the corresponding value of the property’s cell.

According to the component decomposition method, see Ref. [2] for details, it is possible to model a complex system in terms of its components (let’s denote them with c); these could be cellular automata together with global operators. The last ones could account for phase transitions (stages) or global behaviors respectively. The application of the transition function
and/or the global operators must follow a well defined order. Once stages are determined, correlations between stages must be fixed in order to build the network. Subsequent stages, whose order is derived from complex system phenomenology, can employ the output signal flows (properties) from their previous stages by means of the composition relation. A precedence relation between a global operator and a cellular automaton occurs if a global operator accesses a property of the cellular automaton; or, in an opposite direction, a cellular automaton accesses a global variable defined through a global operator.

Therefore we need to employ a CANv2 network in all the cases where different values of the properties/global variable must be used at the same time step evolution. Summarizing, the introduction of the CANv2 model implies that:

1. from a functional point of view, the transition functions and/or the global operator functions could be built up as parts or phases of the whole function describing the network function evolutionary law;
2. from an execution point of view, the network execution consists, at each macro-step, of executing all cellular automata/global operator nodes belonging to the network; the executions must obey to the precedence relations between nodes.

3.2. CANv2: A formal definition

Let us now give a formal definition of a CANv2 network in a one dimensional cellular space, which can be trivially generalized to higher dimensions.

A CANv2 network is the following tuple:

\[
(L, X, S, G, P, P_{var}, F, g, F).
\]

- \(L = \{x: x \in \mathbb{N}, 0 < x < l_x\}\) is the set of points with integer coordinates in the finite region where the system evolves, \(N\) is the set of natural numbers, and \(l_x\) is the upper bound of the set of points, i.e., it determines the bounds of the region of the system evolution. Here the one-dimensional case was introduced, but a three-dimensional region can be trivially generalized.
- \(X^{\#N_{var}} = \bigcup_{i=1}^{M} X^{\#N_i}\) where \(M\) is the number of components, \(\#N_{var}\) is the cardinality of the total neighborhood set, and \(X^{\#N_i} = [i - r, i + r] \subseteq L\) is the set which identifies the geometrical pattern of the cells which influence the cell state change, i.e., the neighborhood set for each cell \(i\), where \(r\) is the radius. It will be opportunely arranged in the case of more than one dimension.
- \(S\) is a finite set of states, where \(S = \times_{i=1}^{m_i} S_i\) is the Cartesian product of all the sets of sub-states and \(m_i\) is the total number of such sets.
- \(G\) is a finite set of global variables, where \(G \subseteq \mathbb{R}\) (\(\mathbb{R}\) is the set of real numbers).
- \(P\) is a finite set of parameters, occurring in transition functions, \(P \subseteq \mathbb{R}\).
- \(P_{var}\) is the set of global parameters, occurring in global operator functions, where \(P_{var} \subseteq \mathbb{R}\).
- \(f\) is the set of the cellular automata transition functions.
- \(g\) is the set of global operator functions.
- \(F : S^{\#N_{var}} \times G \times P \times P_{var} \rightarrow S \times G \times P_{var}\) is the general transition for all the cells in \(L\) (\(\times\) is the Cartesian product).

Complex systems, which are made of different components, give rise to different transition functions, i.e., different cellular automata nodes for the network. In such a case all the components can be enumerated according to two main driving procedures: dependence relation constraints and consequent sequential transition function application respectively. Sometimes, when a direct dependence is not present, a different sequence of application of the transition function between components can be adopted.

In order to simplify the definition of the total transition function for the whole system, it could be useful to separately define the CA component and the global operator components as follows. Let \(f_1 \ldots f_M\) be the transition functions for a system made of \(M\) components; the total transition function of the network, denoted with \(f\), can be expressed in terms of a composition law such as:

\[
f = f_1 \circ f_2 \circ \cdots \circ f_M,
\]

where \(\circ\) is the composition operator. From an execution point of view the following precedence relations hold:

\[
f_1 \prec (f_2 \ldots f_M) \quad f_2 \prec (f_3 \ldots f_M) \quad \cdots \quad f_{M-1} \prec f_M.
\]

This implies that any transition function must be executed before the \(M\)th, \(\prec\) being the precedence relation.

Let \(g_1 \ldots g_K\) be global operators for the system made of \(K\) components, then the global function \(g\) of the network can be expressed in terms of a composition law as follows:

\[
g = g_1 \circ g_2 \circ \cdots \circ g_K
\]

where \(\circ\) is the composition operator. From an execution point of view, if relation (3) holds the following precedence relations must be satisfied:

\[
g_1 \prec (g_2 \ldots g_K) \quad g_2 \prec (g_3 \ldots g_K) \quad \cdots \quad g_{K-1} \prec g_K.
\]

which implies that any global operator, but the \(K\)th, must be executed before the last one.
Let $A$ be a cellular automaton, whose property is denoted by $p$ and transition function by $f$, and $GO$ a global operator, whose global variable is denoted by $g_a$, and whose function is denoted by $g$. If $A$ needs to know the value/s of a global variable $g_a$ at the same macro-step, defined as the whole network time step evolution, in order to evolve at each micro-step, defined as the time step evolution for each component cellular automaton or global operator node, then the execution of the function $g$ must precede the execution of the transition function $f$. So, merging the previous definitions we obtain the following

$$ F : g \circ f, $$
$$ F : \sigma_{ij} \tau_i \circ f_i, $$

where the $[g_i]$ symbol implies that its presence could be optional and the corresponding precedence relations must be expressed according to it.

### 3.3. CANv2: Further specifications

The requirement of introducing components and their composition mechanism is needed in order to have the possibility to represent a complex system as a composition of more than one cellular automaton or global operators. In the CANv2 model this is made possible through the abstraction of a network of cellular automata. The introduction of a precedence relation between the network components is entirely due to the owner-rule which obeys to the following requirements:

- each cellular automaton/global operator is the owner of its properties/global variables;
- only owners can update their properties/global variables;
- non-owners can access, in the same macro-step, properties/global variables only after they were updated.

A network of cellular automata can be represented as a graph, the CAN precedence graph, where nodes represent cellular automata/global operators components and edges represent the precedence relations between nodes.

Here we develop two points of view:

**top–down** from macroscopic scales to subprocesses, as for debris flows and forest fire examples, which are first modeled by the empirical methodological approach by Di Gregorio and Serra [1] and, then, refined by means of the application of CANv2 owner rule and the introduction of global operators;

**bottom–up** from a core model to an enlarged one by adding more specifications, as for the superconductive device, where we consider reversible issues by adding global operators.

To make clearer the proposed approach it is useful to examine the steps which result in the building of a model for the top–down view. In particular, in the latter case the main modeling phases can be outlined as follows:

1. identify the $n$ atomic processes which are relevant to the dynamics of the phenomenon under study. According to the empirical methodological approach described in [1], the first step can lead to two different types of local processes, typically modeled in terms of CA:
   - internal transformations, which accounts for the cell’s state changes that do not depend on the states of the neighboring cells;
   - local interactions, which accounts for the cell’s state transformation related to the state of its neighborhood.

Further processes involved in the phenomenon dynamics may take place on a non-local basis and represent global influences, according to the CANv2 methodology, they are modeled by global operators;

2. define a suitable ordered sequence of CANv2 components:

$$ \Sigma = [c_1 \rightarrow c_2 \rightarrow \cdots \rightarrow c_n], $$

in which: (i) each $c_i$ corresponds to a different atomic process identified at the previous step; (ii) the input of $c_i$ does not depend on the output of $c_j$ if $i < j$. Hence, a sequential macro-step of the CANv2 model consists of the ordered execution of all components in $\Sigma$ from $c_1$ to $c_n$. In particular, let a CANv2 component $c_i$ be defined as:

$$ c_i = (\tau_i, I_i, O_i), $$

where $\tau_i$ can be either a CA transition function or a global function, more specifically $\tau_i \in f$ OR $\tau_i \in g$ where $f$ and $g$ are defined in [1], $I_i$ is the set of input substates/variables (note that it may also be $I_i = \emptyset$) and $O_i$ is the set of of output substates/variables, more specifically $\{I_i, O_i\} \in S$ where $S$ is defined in (1). Thus, by construction:

$$ I_i \cap O_j = \emptyset \quad \text{if} \quad i < j. $$

Depending on the relationships between the CANv2 components, there may be more than one sequence $\Sigma$ that satisfies the condition of Eq. (8). Each of these can be considered a correct execution sequence;
3. find an optimized precedence graph for the parallel execution of the CANv2 components in \( R \). This is accomplished taking advantage of any further independence between the involved variables and substates. In particular, if for two components \( c_i \) and \( c_j \):
\[
O_i \cap I_j = \emptyset \quad \text{with} \quad i < j.
\]
then the parallel execution of the components \( c_i \) and \( c_j \) is allowed. On the basis of this consideration the algorithm described later is able of building the optimal CANv2 precedence graph.

The latter can be specified in terms of a sequence of \( q \leq n \) parallelization levels \( PL_i \):
\[
Seq = [PL_1 \rightarrow PL_2 \rightarrow \cdots \rightarrow PL_q],
\]
where each level \( PL_i \) collects CANv2 components of \( \Sigma \) that can be executed in parallel. For example, in the precedence graph represented in Fig. 1 six CANv2 components were grouped into three levels:
\[
\begin{align*}
PL_1 &= \{c_1, c_2\}, \\
PL_2 &= \{c_3, c_4, c_5\}, \\
PL_3 &= \{c_6\}.
\end{align*}
\]
This means that, in addition to Eq. (8), the following conditions also hold:
\[
O_1 \cap I_2 = O_3 \cap I_4 = O_3 \cap I_5 = O_4 \cap I_5 = \emptyset.
\]

The idea underlying the algorithm for building the CANv2 graph consists of generating all the sequences of parallelization levels which are compatible with the dependency relationships. Among such sequences, the one with the minimum length is taken as an optimal dependency graph. In particular, the algorithm operates on a set \( K \) which contains at first only the sequence \( Seq_1 = \{(c_1)\} \). The set \( K \) is then updated as follows:

```
for each (component \( c_i \in \Sigma \setminus \{c_1\} \}) {
    for each (sequence \( Seq_j \in K \)) {
        PL \leftarrow \text{last level of } Seq_j;
        if \((O_k \cap I_l = \emptyset, \forall c_k \in PL) \) {
            PL' \leftarrow PL \cup \{c_i\};
            Seq' \leftarrow Seq_j;
            In Seq' replace the last level with \( PL' \);
            Insert Seq' in \( K \);
            Append to Seq_j the new level \( \{c_i\} \);
        }
    }
}
```
Note that the same procedure should be carried out for each of the different sequences \( \Sigma \) that satisfies the condition of Eq. (8). At the end of the steps outlined above, any of the sequences in the set \( \mathcal{K} \) with the minimum length represents an optimal CANv2 dependency graph. Consider for example, the following sequence of ordered CANv2 components:

\[
\Sigma = [(c_1 \rightarrow c_2 \rightarrow c_3 \rightarrow c_4 \rightarrow c_5 \rightarrow c_6)]
\]

and suppose that, besides the dependency relationships defined by Eq. (8) also the conditions specified by Eq. (14) hold and that \( O_i \cap I_j \neq \emptyset \) in any other case. Thus, the algorithm outlined above gives rise to the following evolution of the set \( \mathcal{K} \):

1. \( \mathcal{K} = \{S_1\} \), with \( Seq_1 = \{(c_1)\} \);
2. since \( O_1 \cap I_2 = \emptyset \), the new level \( \{c_1\} \cup \{c_2\} \) is defined. Then, the new sequence \( Seq_2 = [(c_1, c_2)] \) is inserted in \( \mathcal{K} \). Also, the new level \( \{c_2\} \) is defined and appended to \( Seq_1 \). At the end of the step: \( \mathcal{K} = \{Seq_1, Seq_2\} \), with \( Seq_1 = [(c_1) \rightarrow (c_2)] \) and \( Seq_2 = [(c_1, c_2)] \);
3. since \( O_2 \cap I_3 = \emptyset \), the new level \( \{c_3\} \) is defined and appended to both \( Seq_1 \) and \( Seq_2 \). At the end of the step: \( \mathcal{K} = \{Seq_1, Seq_2, Seq_3\} \), with \( Seq_1 = [(c_1) \rightarrow (c_2) \rightarrow (c_3)] \) and \( Seq_2 = [(c_1, c_2) \rightarrow (c_3)] \);
4. since \( O_3 \cap I_4 = \emptyset \), the new level \( \{c_4\} \) is defined. Then, the new sequences \( Seq_3 = [(c_1) \rightarrow (c_2) \rightarrow (c_3)] \) and \( Seq_4 = [(c_1, c_2) \rightarrow (c_3)] \) are inserted in \( \mathcal{K} \). Also, the preexisting sequences are redefined as: \( Seq_1 = [(c_1) \rightarrow (c_2) \rightarrow (c_3) \rightarrow (c_4)] \) and \( Seq_2 = [(c_1, c_2) \rightarrow (c_3) \rightarrow (c_4)] \), and \( Seq_3 = [(c_3) \rightarrow (c_4)] \). At the end of the step: \( \mathcal{K} = \{Seq_1, Seq_2, Seq_3, Seq_4\} \);
5. since \( O_4 \cap I_5 = O_4 \cap I_5 = \emptyset \), the following new sequences are created and inserted in \( \mathcal{K} \): \( Seq_5 = [(c_1) \rightarrow (c_2) \rightarrow (c_3, c_4, c_5)] \), \( Seq_6 = [(c_1, c_2) \rightarrow (c_3, c_4, c_5)] \), \( Seq_7 = [(c_1) \rightarrow (c_2) \rightarrow (c_3) \rightarrow (c_4, c_5)] \) and \( Seq_8 = [(c_1, c_2) \rightarrow (c_3) \rightarrow (c_4, c_5)] \). At the end of the step: \( \mathcal{K} = \{Seq_1, Seq_2, Seq_3, Seq_4, Seq_5, Seq_6, Seq_7, Seq_8\} \);
6. since \( O_i \cap I_j = \emptyset \; \forall \; k \), the new level \( \{c_k\} \) is defined and appended to all the current elements of \( \mathcal{K} \).

As a final result, the sequence \( Seq_8 = [(c_1, c_2) \rightarrow (c_3, c_4, c_5) \rightarrow (c_6)] \), which is represented in Fig. 1, is characterized by the lowest length and thus can be considered the optimal one.

To some extents, the procedure above can be considered a way to start from the methodology described in [1] and then to refine the complex system model, when possible, according to CANv2.

In the following Section we show how the CANv2 paradigm works by discussing in detail three applications in the realm of natural sciences.

4. Simulation of microscopic and macroscopic dynamics via CANv2 approach

The aim of this Section is to show how the CANv2 approach can be successfully exploited as a simulation tool for complex systems, where the dynamical evolution is characterized by more than one mechanism and different scales. That appears to be very useful in a variety of contexts, ranging from physics to biology and natural and environmental sciences. Here we concentrate our attention on three examples, borrowed from both the realm of natural macroscopic phenomena and that of condensed matter physics and microelectronics respectively, whose complex dynamic evolution depends on local interactions between the components: the debris flows after a landslide [8], the operation of a topologically protected qubit based on a superconductive device [2] and, finally, the spread of a forest fire. The examples within the debris flows and forest fires contexts are obtained by generalizing the original SCIDDICA [3] and ABBAMPAAU [6] models, in order to build up a network of cellular automata. Regarding the two sources of parallelism exploitation, data parallelism and task parallelism. In particular, in SCIDDICA modified vs CANv2 approach the evidence of the double level of parallelism was explored. As for the subsequent example of simulation of a superconductive device we have to say that the device under study is very simple; the advantages of parallelism will be fully exploited when we will deal with a more complex device, for instance by considering the aggregation of more blocks. Likewise, the simulation of forest fire spread via CANv2 is at the very beginning but, also in this case, we expect to exploit all the power of our proposal in a future work.

4.1. Debris flows

Landslides are usually characterized by a dominant flow–type, i.e., earth flows, debris flows, debris avalanches; that makes possible to think about them as dynamical systems whose elementary parts evolve exclusively as a consequence of local interactions. In this context a CA based approach is well appropriate because the governing flow equations (Navier–Stokes equations for the debris flow) cannot be easily solved without making a lot of simplifications and approximations. In this Subsection we show how a mud/debris flow after a landslide can be reliably simulated by means of a network of CA, i.e., the CANv2 model. This task is accomplished by properly generalizing the original debris flow model [8].

As we will show in the following, the CANv2 model for the simulation of mud/debris flows is obtained as the final result of a sort of heuristic refinement method, which allows one to map the CA SCIDDICA [3] application into a CA network [8].
4.1.1. SCIDDICA original model

The CA model for the simulation of landslide phenomena is represented by the quadruple:

\[ \text{SCIDDICA} = (R, X, Q, P, \sigma) \]

where:

- \( R = \{(x,y) : x,y \in \mathbb{N}: 0 < x < l_x, 0 < y < l_y\} \) is the set of points with integer coordinates in the finite region where the phenomenon evolves, \( \mathbb{N} \) is the set of natural numbers, and \( l \) is the vector, i.e., the upper bound of the set of points, which determines the bounds of the region under consideration.
- \( X = \{(i,j) : -1 < i < 1, -1 < j < 1\} \) is the set that identifies the geometrical pattern of cells which influences the cell state change, i.e., the neighborhood set for each cell.
- \( Q \) is a finite set of states.
- \( \sigma : Q^R \rightarrow Q \) is the deterministic state transition for the cells in \( R \), where the apex of the \( Q \), i.e., \( 9 \) is the cardinality of the neighbor set.
- \( P \) is a finite set of parameters and their detailed description is in [3].

The main mechanisms of the landslide are described by one internal transformation and two local interactions, as follows:

- **Mobilization effect** (\( \sigma_1 \)): it takes into account the effects after the mobilization of the cell region, its influence on the thickness of the cell, its altitude, the detrital cover, in the cell, and the energy that can be added to the amount of the considered debris/mud.
- **Debris/mud, run-up and outflows** (\( \sigma_2 \)): it takes into account the ability of the amount of the considered debris/mud which can overcome an obstacle.
- **Mobilization propagation** (\( \sigma_3 \)): it takes into account the possibility of movement if one of the neighbors is already moving.

SCIDDICA substates are the following:

- \( Q_a \), that is correlated to the altitude of the cell;
- \( Q_{th} \), that is correlated to the thickness of debris/mud in the cell;
- \( Q_e \), that is correlated to a measure of the energy of the cell debris/mud given by the product of debris/mud thickness with run-up highness;
- \( Q_{dc} \), that is correlated to the type of detrital cover of the cell and it individuates the maximum depth of detrital cover that can be transformed by the erosion in debris/mud;
- \( Q_{th} \), that is correlated to the mobilization activation of the detrital cover which becomes debris/mud; that depends on the altitude difference between the central cell and its neighbors.

In this phenomenon local interactions are the determination of outflows (e.g., debris/mud outflows) to the neighboring cells, depending on the value of some substates value in its own neighborhood. In order to determine the total variation of the properties cells, each cell must apply the procedures not only to compute internal transformations of the substates and their own outflows of, but also the neighboring cells’ outflows (corresponding to their own inflows if). As a consequence the overall cell neighborhood must include not only the cells necessary to calculate their own outflows but also the cells necessary to calculate the inflows. According to the classical CA model, that involves a more extended neighborhood and a heavy repetition of the same computations.

4.1.2. Mapping SCIDDICA into a CANv2 model

According to the CAN model, a more extended neighborhood and a heavy repetition of the same computations can be overcome by specializing the role of each automaton, and by composing the network, according to internal transformations and local interactions, where each automaton has its own neighborhood different from the other automata. The assignment is not so immediate. First of all, any internal transformation and local interaction in CA SCIDDICA is described as a transition function and roughly we could map any function of these in a component of CAN, i.e., a cellular automaton, but as we will discuss later the local interaction \( I \) cannot be mapped directly. Then the switching from CA SCIDDICA to the network prece- dences, according to the procedure outlined in Section 3.3, is mainly driven by the outflows of an automaton that are the inflows of another automaton. Outflows and inflows represent the means by which the transfer functions between components act. Let us now discuss about outflows and inflows. Outflows, within CAN, represent automata properties. They are written as outflows by a cell of an automaton, that owns it, and then can be read as inflows from the adjacent cells of other automata. All the other properties are associated to automata according to the owning rule: this results in a network of 5 automata according to the CAN model, as depicted in Fig. 2. The first advantage in using a network of automata is the following: it does not require the repetition of the same computation as in the classical model, and that allows outflows to be...
specialized according to different neighborhoods. In this way all the outflows become properties of the CAN model and a new property, $Q_e$, is introduced as follows:

1. $Q_e$ that is correlated to the energy of the cell debris/mud that can overcome an obstacle;
2. $of_{th}$ individuates a debris/mud outflow from the central cell;
3. $of_r$ individuates the run-up outflows;
4. $of_m$ individuates a mobilization outflow, that accounts for the propagation of mobilization from the central cell;
5. $Q_m$ that is correlated to the mobilization activation of the detrital cover which becomes debris/mud; it depends on the altitude difference between the central cell and its neighbors.

We observe that the processes described by the local interaction $I_1$ are decomposed into two subprocesses:

1. the first computes the thickness of the outflow overcoming the cell due to its run-up energy;
2. the second computes the run-up energy of the outflow leaving the cell.

Network automata are identified as follows:

- $A_1$ includes one property, its transition function computes part of the $\sigma_{t1}$ debris/mud outflows; it reads $Q_{th}$, $Q_r$ and $Q_a$ and it writes $of_{th}^t$.
- $A_2$ includes one property, its transition function computes the remainder part of $\sigma_{t1}$; it reads $Q_b$ and $Q_c$, and the inflow $if_{th}^t$, and writes $of_r^t$.
- $A_3$ includes one property and its transition function computes $\sigma_{t2}$; it reads $Q_{th}$, $Q_r$, $Q_a$ and $Q_m$ and it writes $of_m^t$.
- $A_4$ includes five properties, its transition function reads $Q_a$, $Q_r$ and $Q_{dc}$ and the inflows $if_{th}^t$ and it writes: $of_m^t$ and $of_r^t$. If the mobilization occurs, the detrital cover partially disappears because it is transformed into debris/mud, so that a flow of debris/mud plus run-up is generated inside the cell and the altitude is reduced.
- $A_5$ is composed by two properties and its transition function computes new values of the properties $Q_e$ and $Q_{th}$ adding inflows and subtracting outflows to their old values; it reads the properties $Q_a$ and $Q_r$ and the outflows $of_{th}^t$ and $of_r^t$ (the inflows $if_{th}^t$ and $if_{r}^t$ for each cell are trivially derived).

Once the new values of all properties are determined, a new evolutionary step of the landslide is simulated.

Evidence in parallel implementations of the CAN SCIDDICA version were given in [5], where both data parallelism and mixed data and control parallelism were exploited using a multi-threading implementation [5]. The results obtained showed no significant differences in the two approaches in terms of speedups obtained by the ratio of the sequential execution time with the CAN multi-threaded execution time. This was an expected result due to the low number of network branches present in the considered application. As a consequence, in the considered application the data parallelism, depending on the size of the grids (that is fixed for the given application), is predominant if compared to the task parallelism coming from the presence of only two branches of computationally independent set of automata in the network. It should be noted that in the considered application the data parallelism available is predominant since the network has only two branches. Nevertheless performances obtained exploiting both types of parallelism do not degrade. This is an encouraging result since we adopted a simply scheduling strategy that is static; so we foresee better results designing a dynamic scheduling strategy that could take into account the run. This issue regarding convenience of data and/or task parallelism exploitation in CAN paradigm was studied in [4].

4.2. Superconductive devices

In this Subsection we apply the CANv2 approach to build up a qualitative model of a topologically protected qubit [2], physically realized with a fully frustrated Josephson junction ladder (J JL) arranged in an annular geometry [13].
In its simplest version such a device consists of a ladder with \(N\) plaquettes closed in a ring with a half flux quantum \((\frac{1}{2} \phi_0)\) threading each plaquette. The number of plaquettes must correspond to the number of junctions on the vertical links of the ladder, so that each plaquette contains two junctions on the left and right link respectively. If we consider a ladder with, say, \(N = 10\) plaquettes, the corresponding ground state is twofold degenerate and maps into a linear antiferromagnetic chain of half-integer spins. It can be shown that the tunneling between the two ground states \(|0\rangle, |1\rangle\) corresponds to the physical process of creation and annihilation of kink–antikink pairs, which gives rise to a sequence of double flips in the antiferromagnetic pattern of spins. A kink–antikink pair can be produced locally by applying a frustration single sawtooth pulse. Our device has \(N\) degrees of freedom and \(\frac{N}{2}\) double flips are needed to pass from \(|0\rangle\) to \(|1\rangle\). But there are in general \(\binom{N}{2}\) paths along which such a process can occur. Then, switching off the frustration the system relaxes on the new state and the transition is carried out.

Based on such considerations, we construct an elementary NOT gate by setting up a quasi-probabilistic model which implements logical reversibility. In the following we outline the main features of our model [2].

### 4.2.1. The core model

Considering the main device characteristics, see Refs. [2] for details, the formal definition of the CANv2 model for the JJJL device under study follows:

\[
\langle L, X, S, G, P, P_{\text{var}}, f, g \rangle.
\]

- \(L \subset \mathbb{N}\) is the set of integer points in the finite region, the array, where the system evolves; each point identifies a cell. The lattice grid is a linear array with, for example, \(\#L = 10\) cells.
- \(X\) is the neighborhood set \([x - 2, x + 2]\) for each cell \(x\).
- \(S = S_1 \times S_2 \times S_3 \times S_4\) is the set of state values; it is specified by the cartesian product of the finite sets of values of the four following substates:
  1. \(\text{Pseudo}_S\), pseudospin assuming values from \([-1,1]\),
  2. \(\text{Mp}\), magnetic pulse for each cell, fixed and invariant for each macro time step,
  3. \(\text{LABEL}\), cell label in order to identify the cell, corresponding to a monotonic enumeration for all the cells, itself invariant for each time step,
  4. \(\text{FLIP}\) is the flip state in order to register if pseudospin flipping has been taken.
- \(G\) is the set of global variables: \(B_{\text{tot}}\) is the total applied magnetic pulse, and \(\text{Start}_c\) is the number of the corresponding starting cell.
- \(P\) is the finite set of global constant parameters: \(I\), current inside the cell, and \(C\), capacitance
- \(P_{\text{var}}\) is the finite set of the CA component variables: STEP is the step iterator which allows to trigger the evolution.
- \(f : S \rightarrow S\) is the deterministic state transition for the determination of the pseudospin state and values.

**Operator PULSE**

```plaintext
begin
    Btot := 0
    for k = 1 to N
        Btot := Btot + Mp(k)
    end
    if (Btot < N*Mp(i)) then
        call error handler
    end
end
```

**Operator CHOOSER**

```plaintext
begin
    StartC := random(even(N))
end
```

**CA Transition function**

```plaintext
begin
    Step := Step + 1
    if (Mp(i) > 0) then
        if (LABEL(i) = StartC and LABEL(i+1) = StartC+1 and FLIP(i+1) = 0 and FLIP(i-2) = 0 and FLIP(i-1) = 0)
            then
                Pseudo_S(i+1) := Pseudo_S(i)
                Pseudo_S(i) := 1 - Pseudo_S(i+1)
            else
                if (FLIP(i) = 0 and FLIP(i-1) = 1 and FLIP(i-2) = 1 and LABEL(i) mod 2 = 0)
                    then
                        Pseudo_S(i+1) := Pseudo_S(i)
                        Pseudo_S(i) := 1 - Pseudo_S(i+1)
                end if
                LABEL(i) := LABEL(i) + 2
                repeat
                    CALL CA Transition Function
                    until STEP not equal N/2
        end if
end
```

---

Fig. 3. The JJJL system components according to a CANv2 simulation.
• $g: S \rightarrow G$ expresses the global operator which controls the total magnetic pulse applied on the system.
• $g_v: G \rightarrow G$ expresses the global operator which chooses randomly the driving cell.

In view of the implementation of a protected qubit the boundary condition topology is annular. In order to get a transition between the two ground states the magnetic pulse period must be related to the CA time step and it is equal to the pulse period, in order to capture the maximum pulse value. In order to implement the flipping procedure from $|0\rangle$ to $|1\rangle$ our model will select out one particular path (out of $\binom{N}{2}$! paths), so giving rise to a high level description of such tunneling processes: in fact, at this preliminary stage we are interested only in the net result, i.e., the NOT operation. For this reason we choose to use a double step for the CA transition component, with each time step equal to the half of the single sawtooth magnetic pulse period. The CA component has, as initial condition, the pseudospin configuration obtained in the precedent stage since it must obey to an antiferromagnetic arrangement and the flipping state is zero. At the initial time, our device is in a steady state, in one of the two possible ground states. Each parameter is fixed and the LABEL values are fixed for all transition steps, it must obey to an antiferromagnetic arrangement and the flipping state is zero. At the initial time, our device is in a steady state, in one of the two possible ground states. Each parameter is fixed and the LABEL values are fixed for all transition steps, but the variable STEP is initialized to each macro-step $T$. The behavior just described is expressed in pseudocode through the CA transition function shown in Fig. 3.

4.2.2. Adding reversible features

Let us now explain how to introduce reversibility in the core model above outlined. The general transition function takes into account the coupling factor, adding an external frustration, as a single sawtooth magnetic pulse acting on each lattice cell. The system transition is assumed to be given by the possible simultaneous applications of the two global operators followed by the transition function. The evolution of the model obeys to the following function

$$F: S \times G \times P \times \text{Pvar} \rightarrow S \times G \times \text{Pvar}.$$  

The transition function scheme (see Fig. 3) displays two global operators, the pulse application and the random starting cell chooser, and the transition function repeatedly applies the cellular automata components according to the multiplicity related to the double flips. The probabilistic behavior is due to the Operator CHOOSER which randomly chooses at each network time step the driving cell for the flipping procedure. Such an operator plays a crucial role and it must be modified in the following way in order to achieve our reversibility goal: a further global variable must be considered in order to store the sequence of chosen cells at each time step for the whole system evolution. This is enough for our model in order to recover all the states since the starting point and not only the previous one. The Operator CHOOSER is equipped with a global vector (a global variable which is able to store more values) which stores at each time step the starting cell for the evolution.\footnote{In fact, the vector length coincides with the number of iterations.} In this way it is much simpler to recover the previous values as, for each time step directly accessible, the first evolution cell can be recovered.

This approach has two main advantages:

1. we are able to recover all the previous states as each starting cell is stored and, in this case, the previous system state is exactly the negation;
2. we can avoid to store all the cells values, that could be very cumbersome in the case of a device with a high number of cells; instead we restrict to the storage of one value for each time step evolution.

The last point results effective in the case of JJL devices with a large number of cells, or in general with other equivalent devices with a lot of elementary discrete components.

4.3. Forest fires

In this Subsection we deal with a model of forest fires spread that could be useful for the prevention and for the evaluation of environmental impact of forest fires. It generalizes the original core model from which the renamed ABBAMPAU model [6] was developed.

4.3.1. Forest fires original model

According to the forest fire model presented in [6,7], growth is implemented as a two-dimensional CA with hexagonal cells and a radius two neighborhood. The lattice expressed by CAN method remains the same as in the original one. Also the set of parameters $P = \{p_e, p_c, p_v, p_t, p_w, p_{vl}, p_{wd}, p_{nul}\}$, such as (apothem of the cell, time step, type of vegetation and related catch burning, current time, weather condition depending on exposition season and atmospheric parameters and, finally, free wind direction and rate), is the same. The finite set $S$ of the states of the cell is:

$$S = S_{a} \times S_{v} \times S_{t} \times S_{h} \times S_{c} \times S_{d} \times S_{wd} \times S_{wr} \times S_{a}^{g} \times S_{a}^{p} \times S_{a}^{l},$$

where the substrates are

1. In fact, the vector length coincides with the number of iterations.
$S_A$, **altitude**, takes the altitude value of the cell;

$S_V$, **vegetation**, specifies the type of vegetation, relatively to the properties of catching fire and burning (see [6] for the specification of the value);

$S_T$, **temperature**, takes the temperature value of the cell;

$S_H$, **humidity**, takes the relative humidity value of the cell;

$S_C$, **combustion** for the two possible fire types in the cell: **surface fire** and **crown fire**. It takes one of the values **not-inflammable**, **inflammable**, **burning** and **burnt** for each type;

$S_D$, **duration**, takes the value of the duration of the fire in the cell;

$S_{WD}$, **wind direction**, takes the values of the wind directions (the eight directions of the wind rose) at the ground level (that could be different from the free wind direction);

$S_{WR}$, **wind rate**, takes the values of the wind rate (km 0–0) at the ground level (that could be different from the free wind rate);

$S_{PS}$, **fire spread**, accounts for the fire spread from the central cell to the other neighboring cells;

$S_{PR}$, **fire acquire**, is just a renaming of $S_{PS}$ and individuates the fire propagation from the other neighboring cells towards the central cell.

and 18 is the cardinality of the neighbor set.

For the moment let us to postpone the discussion regarding the mapping of substates.

The application of an external influence regards some cells where the fire starts and some cells in which there is the intervention of the fireman. When considering cells, or a set of them, in which the fire starts, this interests a subregion of the entire lattice as $I \in \mathbb{K}$. This is translated in the CAN model as a global operator and, in such a case, the application of the owner rule introduces much more specifications regarding the substate $S_C$. The function

$$
\Gamma = \{\gamma_1, \gamma_2, \gamma_3, \gamma_4\},
$$

which is computed at each step, before the application of the transition function, represents the external influence as follows:

- $\gamma_1$ determines the current time of the CA step;
- $\gamma_2$ supplies the weather conditions related to sun, wind direction and its rate, at each CA step;
- $\gamma_3$: $I \rightarrow S_C$ accounts for external setting fire to the cells of $I$ at prefixed steps;
- $\gamma_4$: $K \rightarrow S_C$ accounts for firemen intervention at prefixed steps.

The transition function is defined as:

$$\sigma = \sigma_{I_4} \otimes \sigma_{I_3} \otimes \sigma_{I_2} \otimes \sigma_{I_1},$$

where $T_1, I_1, I_2$ and $I_3$ are the elementary processes, components of the entire phenomenon, respectively as internal transformation (denoted by $T$) and the other local interaction (denoted by $I$). As a first step the following internal transformation, concerning the effects of combustion in surface and crown fire inside the cell, is computed:

$$\sigma_{I_4} : S_T \times S_T \times S_H \times S_C \times S_D \times S_{WD} \times S_{WR} \rightarrow S_T \times S_H \times S_C \times S_D.$$

Then the following local interactions are applied:

- $\sigma_{I_1} : (S_A)^{19} \rightarrow S_{WD} \times S_{WR}$
- $\sigma_{I_2} : (S_{SA})^{18} \times S_C \rightarrow S_C$
- $\sigma_{I_3} : (S_A)^{19} \times S_V \times S_C \times S_H \times S_T \times S_D \times S_{WD} \times S_{WR} \rightarrow (S_{PS})^{18}$,

where $\sigma_{I_1}$ computes the wind direction and rate at the cell altitude, $\sigma_{I_2}$ computes the change of combustion conditions in the current cell and $\sigma_{I_3}$ computes the fire spread toward the neighboring cells.

**Internal transformation** $\sigma_{I_4}$ **function** acts when substate $S_D$ does not change if the substate $S_C$ is not burning, while the substates $S_T$ and $S_H$ vary their previous values on the basis of the weather change ($p_w$), the day hour ($p_t$), the wind ($S_{WD}$ and $S_{WR}$) and the vegetation type ($S_V$ and $p_v$). When the substate $S_C$ is burning, then the substates $S_T, S_H, S_C$ and $S_D$ depend on the previous values of $S_H, S_D$ and $S_T$. The value 0 for $S_D$ determines the change of $S_C$ to burnt. The conditions for the ignition from surface fire to crown fire are applied in $T_1$.

**Local interaction** $\sigma_{I_1}$ **function** computes the substates $S_{WD}$ and $S_{WR}$, wind direction and rate depends on $p_{wd}$ and $p_{wdr}$ that represent the values of the free wind. The new values of $S_{WD}$ and $S_{WR}$ are obtained by adding the altitude vector to the corrective vector [6].

**Local interaction** $\sigma_{I_3}$ **function** tests if the fire is spreading toward the central cell starting from the other cells of the neighborhood. If the combustion substate $S_C$ is inflammable, then it changes to burning.
Local interaction $\sigma_3$ function computes the spread of fire acting if the state $S_C$ is burning; the following computation steps, depending on the substates $S_V, S_H, S_T, S_D, S_{WD}, S_{WR}$ and the set of parameters $P_v$, are considered according to the maximum spread vector, determined as the sum of the slope effect and the wind effect, computing maximum spread ($R_{max}$), and considering an ellipse with a focus in the center of the cell whose area depends on $R_{max}$ and such that the fire can propagate towards the neighboring cells inside the ellipse, i.e., $S_{FS}$ (which is an alias of $S_{FA}$) takes the value true.

4.3.2. Mapping ABBAMPAU into a CANv2 model

Starting from ABBAMPAU let us now realize the mapping into a CANv2 model by following, step by step, the procedure developed in Section 3. As a first step all the substates of the forest fire CA become properties of the CAN network model, but dealing with the substate $S_C$ requires much more attention because, according to the owner rule in CAN, it violates this principle; indeed the substate, i.e., the property, is written by the external transformations $c_3$ and $c_4$ and the internal transformation and the local interaction $r_I$. At this stage subprocesses must be considered. Regarding $c_1$ that determines the current time of the CA step, in our case the global operator $G_{O1}$ determines the macro time step for all global operators and cellular automata components. On the basis of the same considerations, $G_{O2}$ supplies for the weather global condition for each macro step.

When considering $c_3$, it is possible to introduce the property $S_B$ burning as boolean values as it should be assumed as the state is burning or not and it results modified as

$$G_{O3} : I \rightarrow S_B.$$ 

According to $c_4$, it is possible to introduce the property $S_{off}$ Fire off as boolean values because the state is fire switched off or not and it results modified as

$$G_{O4} : I \rightarrow S_{off}.$$ 

As a first result, all global operators could be computed simultaneously because they do not violate any access in the memory as in the original model for the substates, i.e., the memory location of $S_C$.

By means of the mapping four automata are determined, denoted, respectively with $A_1, A_2, A_3, A_4$.

The automaton $A_1$ has the following transition function:

$$f_1 : (S_V \times S_T \times S_H \times S_D \times S_{WD} \times S_{WR} \times S_C \times S_B \times S_{SB} \times S_{off} \rightarrow S_T \times S_H \times S_B \times S_D),$$

where for us $S_C$, the property combustion, takes the possible values: surface fire and crown fire; it takes one of the values not-inflammable, inflammable.

The local interaction does not change as it maps exactly to the automaton $A_2$ with the transition function corresponding exactly to

$$\sigma_1 : f_2 : (S_{FA})^{18} \times S_C \rightarrow S_C.$$ 

Regarding the local interaction $\sigma_3$, a further substate, for us a property, $S_{SB}$ sensible to burn, should be introduced when we consider: if the cell is burning or not, or, if the fire was switched off, the weather conditions, combustible type and fire duration; all that establishes if the cell is sensible to burn according to a threshold turn-on. Then $A_3$ has the following transition function:

$$f_3 : (S_{FA})^{18} \times S_B \times S_C \rightarrow S_C.$$ 

Finally $A_4$ has the following transition function:

$$f_4 : (S_A)^{19} \times S_V \times S_C \times S_H \times S_D \times S_{WD} \times S_B \times S_{SB} \times S_{off} \times S_{WR} \rightarrow (S_{FS})^{18}.$$
The entire process, resulting in the graph depicted in the Fig. 4, could be summarized in three main stages following the corresponding order (where a component belonging to the same stage can be computed at the same time):

- **Global influence** \((G_{15}, G_{25}, G_{35}, G_{45})\),
- **spreading condition** \((A_2, A_3)\),
- **burning condition change** \(A_1\), and
- **Fire spreading toward their neighborhood** \(A_4\).

Now let us discuss Fig. 4. According our view, black circles represent synchronization points, when the macro step starts \(G_{15}, G_{25}, G_{35}, G_{45}\) as they do not have any dependence between, we could execute at the same time. At this point since CA components \(A_2, A_3\), as they do not have any data dependence, can be executed synchronously before executing \(A_4\) then the net macro time step restart.

5. Conclusions and perspectives

In this paper we show the effectiveness of the CANv2 methodology in simulating the interplay between micro and macro dynamics, which characterizes the evolution of many complex systems within the realm of physical and natural sciences. Our approach is a kind of hierarchical modeling and, as such, is a powerful means to handle model complexity [10], where hierarchies define a relation in which the components are grouped into levels and this can imply the double source of parallelism to be exploited. It encompasses the abstraction by reduction: the complexity of the lower level is reduced with a simpler formalism defined by abstraction. Furthermore the individuation of the lower level coincides with the components essentially by a phase and its properties/substrates. Once defined the elementary processes and a suitable sequence of their execution, the optimal length of the entire sequence is chosen and the simulation can start. When macroscopic phenomena are taken into account, joining the Di Gregorio and Serra empirical method [1] and the CANv2 method allow to model also in parallel vision. Different is the case when microscopic phenomena are approached in which the core model can be further specified tacking into account other system features. Using such an approach has the disadvantage that, when modeling, the definition of processes and sub-processes to be introduced in order to obey to the CAN owner rule for the properties components, results much more complex. Advantages arise because a first computational and load balancing scheme is already adopted at the modeling stage, as at run-time the resulting application can exploit much more parallelism resources than in a classical CA scheme model. The next research step will focus on implementation aspects, especially for the forest fire application, and demonstrate real advantages in the parallel execution.

Acknowledgements

The part of research concerning the model of forest fires was partially funded by ASI (Italian Space Agency) SARFIRE: Spaceborn SAR imagery and environmental data fusion for the dynamical evaluation of land regions susceptibility to fire – Agreement: 1/049/09/00 - Project ID: 2288.

References