A Tool for the Modelling and Simulation of Ecological Systems Based on Grid Systems

Suryana Setiawan[1,3], Antonio Cerone[1,2(✉)], and Paolo Milazzo[1]

[1] University of Pisa, Pisa, Italy
{setiawan,cerone,milazzo}@di.unipi.it, setiawan@cs.ui.ac.id
[2] IMT Institute for Advanced Studies, Lucca, Italy
antonio.cerone@imtlucca.it
[3] University of Indonesia, Jakarta, Indonesia

Abstract. Grid Systems is a formalism for modelling population and ecosystem dynamics that combines features of membrane computing, such as rewrite rules and maximal parallelism, with a representation of space similar to that of Cellular Automata. Moreover, Grid Systems include features for the description of environmental events and of events that can be associated with frequencies and durations that can be either deterministic or stochastic. The combination of all of these features makes Grid Systems a comprehensive formalism for the modelling and analysis of ecosystems.

This tool paper describes the implementation and the features of a simulator for Grid Systems. The simulator is equipped with a graphical user interface for defining and editing models of populations, and for simulating population dynamics and movement. The aim of this tool is to allow modellers to construct and analyse models based on a comprehensive and rigorous formalism such as Grid Systems with a friendly interface.

1 Introduction

In order to better understand how to preserve highly endangered species and plan actions of biodiversity conservation in complex ecological communities, social scientist need the support of effective tools, equipped with graphical user interfaces (GUI) that facilitate visual animation and visual presentation of the output [9]. Tools would help in modelling the link between local and global processes, simulating density dependence [7] and dealing with several other challenges of ecology.

Ecologists emphasised the importance of modelling demographic and environmental stochasticity in metapopulation dynamics [8], investigated fluctuations affecting the densities of populations in communities as a consequence of environmental variability [18], and analysed the effects of random perturbations on cyclic population dynamics [12]. We developed a simulator aiming to address these important needs by implementing the Grid Systems, a formal notation for modelling population dynamics, which was inspired by the concepts

D. Bianculli et al. (Eds.): SEFM 2015 Workshops, LNCS 9509, pp. 198–212, 2015.
DOI: 10.1007/978-3-662-49224-6_17

of membrane computing, as in P systems [17], and spatiality dynamics, as in Cellular Automata (CA) [3] and spatial P systems [5,6].

A Grid System consists of an envelop compartment (the outer membrane) containing a grid of adjacent inner compartments (inner membranes, also called cells). Each membrane, characterised by its position in the grid, may represent a distinct part of the environment with specific parameters and behavioural rules. Rules can move objects across membranes, thus describing the migration of resources or individuals.

Our simulator was developed using the Java Netbeans IDE. The latest version of the tool was updated to Java 7.6. A GUI was developed using JSwing and, for the graphical output, AWT. The models created by the user are stored in XML format. Details on the XML encoding can be found in the simulator documentation [20]. The simulator was tested on two case studies: population dynamics of a species of mosquitoes (*Aedes albopictus*) [4], and seasonal migration of a wildbeest species in the Serengeti National Park [21]. Such case studies will be used in this paper to describe the tool functionalities and to include screenshots of the main tool windows.

As regards related (and competitor) tools, we mention CoSBiLab LIME [11], Ecopath with Ecosim (EwE) [15], DISPAS [16], and more general tools such as NetLogo [1] and tools for system dynamics [2]. All of these tools can be used to model and simulate ecosystems, with different levels of detail on the description of the individuals and of the environment. The Grid Systems simulator presented in this paper is a prototype tool based on a formalism that aims at making the description of events that may happen in the ecosystem unambiguous and easy. Moreover, the modelling language is designed with the aim of allowing many aspects of the ecosystem dynamics to be dealt with, such as, in particular, environemental events and spatial dynamics.

The rest of this paper is organised as follows. Section 2 introduces Grid Systems [4,19], the formal notation on which the simulator is based. Section 3 describes the implementation by sketching the evolution algorithm that represents the engine of the tool and briefly presents the software architecture. Section 4 briefly introduces the two case studies used to show the tool features in action. Section 5 illustrates the features of the GUI, which include model definition and editing windows, presentation of the simulation output, visual animation and debugging. Section 6 concludes the paper.

2 Grid Systems

A grid system models space as a two dimensional grid of cells adopting the idea of cells in CA. Each cell, also called membrane, is addressed by an ordered pair of natural numbers: row number and column number. The term "membrane" comes from P Systems and the term "cell" comes from CA. Objects are represented by a set of unique symbols Σ and the state of a cell is represented by a multiset over Σ. Similar to CA and P Systems, the behaviour of Grid Systems is described by reaction rules, which are defined as in P Systems, with the additional features of

having a duration and modelling spatially dynamic behaviour. In order to model spatially dynamic behaviour, the applicability of the rules is defined by means of a set of associations of rules with membranes.

Definition 1. *A Grid System* $G = (\Sigma, R, A, C^{(0)})$ *is defined as follows:*

- G *is the Grid System name;*
- Σ *is a finite set of symbols representing the alphabet of object types;*
- R *is a finite set of reaction rules (see Definition 2);*
- $A = \{(\rho, \gamma) \mid \rho \in R,\ \gamma \in \{G_{i,j} \mid i, j \geq 0\} \cup \{G_E\}\}$ *is the set of associations of the rules with the membranes, where:*
  - $G_{i,j}$ *denotes the cell at position* (i, j)*, called local membrane;*
  - G_E *is the global membrane surrounding the cells;*
- $C^{(0)}$ *is the initial configuration of the Grid System (see Definition 3).*

2.1 Reaction Rules

The behaviour of a Grid System is defined using reaction rules that rewrite a given multiset of objects into a new multiset of objects.

Definition 2. *Let* Σ *be the alphabet in a Grid System. A rule* ρ *is a relation of multiset* α *giving multiset* β *under conditions given by parameters* ψ*,* χ*,* c*,* d*,* X*, and written as*

$$\rho : \alpha \xrightarrow[d,X]{c} \beta \ [\psi \mid \chi]$$

where

- ρ *is the unique identifier of the rule;*
- α *is a non-empty multiset of reactants,* α *is a multiset over* Σ *and* $\alpha \neq \lambda$*;*
- β*,* ψ *and* χ *are multisets over* Σ *of products, promoters and inhibitors, respectively. Elements of these multistes are possibly associated with coordinates of other membranes*
- $c \in \mathbb{R}^+$ *is the rate with which the rule may be applied to perform a reaction;*
- $d \in \mathbb{R}^+$ *is the (exact or mean) duration of the reaction;*
- $X \in \{$*'D'*, *'M'*$\}$ *is a marking to the rule; 'D' indicates that the duration of the reaction will take exactly* d *time units when it is applied; and 'M' indicates that the duration time is an exponentially distributed random variable that has mean value* d *time units.*

2.2 Configurations and Evolution Algorithm

A configuration $C^{(t)}$ is the state of the system at time point t. It consists of two parts: current objects and current ongoing reactions. The current objects are represented as the multisets of existing objects in each membrane. The ongoing reactions are the reactions that have been applied but due to their durations they have not been accomplished. For its semantic purpose, the list of ongoing reactions is sorted according to reactions' termination time.

Definition 3. *A Configuration* $C^{(t)}$ *is a pair*

$$(\{C^{(t,m)} \mid m \in \{G_{i,j} \mid i,j \geq 0\} \cup \{G_E\}\},\ \Omega^{(t)})$$

where

- $C^{(t,m)}$ *is the multiset over objects that exist inside membrane* m *at time* t*;*
- $\Omega^{(t)} = \{(r_k, t_k, m_k) \mid k = 1, ..., n \ \ and \ \ t \leq t_1 \leq ... \leq t_n\}$ *is the set of ongoing reactions where each* (r_k, t_k, m_k)*,* $k = 1, ..., n$*, denotes ongoing reactions that instantiate rule* r_k *in membrane* m_k *and will terminate at time* t_k*.*

The semantics of Grid Systems is described through an algorithm called Evolution Algorithm of Grid Systems. The algorithm starts from a given initial configuration $C^{(0)}$ in which $\Omega^{(0)}$ is assumed to be an empty set. The configuration at time point t_k is denoted by $C^{(t_k)}$. It contains all the objects and the membranes where they are located, and a list of on-going reactions: $C^{(t_k,m)}$ represents the objects in membrane m and Ω^{t_k} represents the list of ongoing reactions. Moreover, configuration $C^{(t_k,m)}$ contains two multisets over objects: $Avail^{(t,m)}$ and $Committed^{(t,m)}$. The former represents the objects that are available for the next reactions and the latter represents the objects that are already involved in some ongoing reactions. In addition to these dynamic aspects, the configuration also contains some static entities: the list of objects Σ, the list of reactions rules R and the association list A.

2.3 Links

Living species have been given by nature the ability to sense and follow the pathways for movements. Pathways can either result from physical perceptions (salmons sense the geomagnetic field [14] and sperm cells sense chemotaxes to locate the ovum [13]) or cognitively created by the individual (wood ants memorize snapshot views and landmarks [10]). In Grid Systems pathways are modelled using links. A link is defined as a special object that carries pointers. A pointer is a piece of information that provides a dynamic addressing of a destination membrane. The pointers can be used by rules in referring to the objects in another cell. Different pointers carried by a link introduce further non-determinism into the system. In order to resolve this form of non-determinism, a decision is made stochastically based on the weight of each pointer. Weights are real numbers between 0 and 1.0, and the total weight in the same cell is 1.0. Like ordinary objects, the number of links in a cell can be increased or decreased by applying its related rule.

Definition 4. *A pointer is an ordered pair of integers. There are two types of pointers: relative pointers and absolute pointers. For the relative pointer the pair of integers is marked by curved brackets, as "*(a, b)*", where* $a, b \in \mathbb{Z}$*. For the absolute pointer the pair of integers is marked by squared parentheses, as "*$[r, c]$*", where* $r, c \in \mathbb{N}$*.*

Links are definitely objects in Grid Systems. Objects are called links when they carry at least one pointer. When they carry several pointers, the pointers are equipped with weights that represent a form of priority.

Definition 5. *G is a Grid System extended with links when any object in Σ can carry pointers. A link is an object that carries at least one pointer.*

The use of links requires rule replication, that is, each rule has to be instantiated by an actual rule in which the link is replaced by its pointer. When a link of the rule has more than one pointer, the rule is instantiated by one actual rule for each pointer. Furthermore, one rule may have several different links. Replications for such a rule are based on the combinations of the pointers of each link.

3 Implementation

The semantics of Grid Systems is described operationally in terms of the Evolution Algorithm of Grid Systems. Such operational semantics has been the basis for our implementation. However, since the operational semantics is given as a recursive mathematical function, we need to identify the key aspects that have to be reworked to implement it using an imperative programming language. First, we need to categorise Grid Systems in terms of the types of reaction rules used:

– *L0 (Stepwise Rule) Grid Systems*, whose reaction rules are in the form

$$\rho : \alpha \underset{D}{\longrightarrow} \beta \ [\psi \mid \chi].$$

The absence of values for c and d on the arrow denotes that rate and duration take default value 1.

– *L1 (0-1 Rule) Grid Systems* include L0 Grid Systems and the ones whose rules are in forms

$$\rho : \alpha \xrightarrow[1,D]{0} \beta \ [\psi \mid \chi].$$

Each rule has duration of either 0 or 1 time unit.

– *L2 (Stochastic) Grid Systems* have all possible types of reaction rules as in Definition 2.

Second, we have to distinguish between *Grid Systems without links* and *Grid Systems with links*.

Third, although Grid Systems are unbounded, as for Definition 2, in our implementation we consider the dimension of Grid Systems bounded to $N \times M$ cells. Therefore, we will use *bounded Grid Systems*

$$G = (N, M, \Sigma, R, A, C^{(0)}),$$

where $N, M \in \mathbb{N}$, such that the number of membranes is $N \times M$ and the association set is $A = \{(\rho, \gamma) \mid \rho \in R,\ \gamma \in \{G_{i,j} \mid 0 \leq i < N,\ 0 \leq j < M\} \cup \{G_E\}\}$.

Finally multisets of the configurations are represented as tables. For example, Ω is represented by Ω-table.

In Sect. 3.1 we sketch the implementation of the evolution algorithm. In Sect. 3.2 we outline the software architecture of the simulator. Full details of the implementation are available in the first author's PhD thesis [19].

3.1 Evolution Algorithm Implementation

Let $C^{(t_k)}$ be the current configuration where $Avail^{(t_k,m)}$ is the multiset of the objects that are available for the next reactions and $Committed^{(t_k,m)}$ is the multiset of the objects that are already involved in some ongoing reactions. The following steps are performed iteratively.

1. Find Reaction Candidates

For each membrane m, if $Avail^{(t_k,m)} = \emptyset$, then the algorithm continues on the next membrane. Otherwise, the multiset $Cand^{(t_k,m)}$ of the candidate reactions is calculated as follows.

1. for each rule r, calculate the number of times $\text{mult}(r, m)$ that rule r is applicable in m with maximum parallelism;
2. $C0 := \{(r, \text{mult}(r, m)) \mid r \in \text{assoc}(m) \text{ and } \text{mult}(r, m) > 0\}$;
3. partition $C0$ into multiset Cn of the rules that are involved in non-determinism and multiset Cd of the other rules;
4. calculate multiset Cs of the selected rules by resolving non-determinism in Cn, using a stochastic rule selection adapted from Gillespie's Stochastic Simulation Algorithm;
5. $Cand^{(t_k,m)} := Cd \cup Cs$.

Grid Systems with links require some additional calculations.

2. Add New Ongoing Reactions to Ω table

For each rule r and membrane m such that $(r, \text{mult}(r, m)) \in Cand^{(t_k,m)}$, a number $\text{mult}(r, m)$ of elapse times $elapse(r, m, i)$, with $i = 1, ..., \text{mult}(r, m)$, is calculated as follows:

- for *L0 Grid Systems*, $\text{elapse}(r, m, i) := 1$;
- for *L1 Grid Systems*, $\text{elapse}(r, m, i)$ equals the duration of rule r (0 or 1);
- for *L2 Grid Systems*, if r is a deterministic duration time rule, then $\text{elapse}(r, m, i)$ returns the duration of r, otherwise (r has an exponentially distributed duration time)

$$\text{elapse}(r, m, i) = -\text{duration}(r) \ln X_i$$

where $X_i \sim U[0, 1]$ (X is a uniformly distributed random variable).

For each $(r, n) \in Cand^{(t_k,m)}$ and $i = 1, ..., n$, build the multiset Ω^+ of triples $(r, t_k + \text{elapse}(r, m, i), m)$. Update the table of ongoing reactions Ω-table by inserting the triples from Ω^+.

For *Grid Systems with links*, rules are replicated as described in Sect. 2.3.

3. Determine the Earliest Reactions to Remove from Ω-table

For *L0 Grid Systems*, $t_{k+1} := t_k$.

For *L1 Grid System*, if there is a new reaction with duration 0, then $t_{k+1} := t_k$, otherwise $t_{k+1} := t_k + 1$.

For *L2 Grid System*, t_{k+1} is the minimum t such that (r, t, m) has been added to Ω-table.

Calculate the multiset Ω^- of the triple (r, t, m) in Ω-table such that $t \leq t_{k+1}$.

4. Generate Products, Remove Reactants and Terminated Reactions

For each membrane m, calculate

- the new multiset $Avail^{(t_{k+1},m)}$ from $Avail^{(t_k,m)}$ by removing all reactants of reactions in Ω^+ and adding all products of reactions in Ω^-;
- the new multiset $Committed^{(t_{k+1},m)}$ from $Committed^{(t_k,m)}$ by removing all reactants of reactions in Ω^- and adding all reactants of reactions in Ω^+

Update the table of ongoing reactions Ω-table by removing the triples in Ω^-. If Ω-table is empty and, for each membrane m, no rule is applicable in $C^{(t_{k+1})}$, then the algorithm terminates. Otherwise, the next iteration of the algorithm starts from **Step 1**.

In terms of efficiency, for *L2 Grid Systems*, the Ω-table is implemented by using a priority queue (heap tree) data structure, which supports insertion and removal in logarithmic time. In this way the performance is still acceptable in presence of a large number of ongoing rules.

3.2 Software Architecture

Models are encoded in XML packages. The simulator also uses two internal data representations: raw data tables, for editing functionalities, and actual data table, for running the simulation. The software architecture of the simulator consists of three modules:

XML Data Handler Module. It is the layer that accesses XML packages. It provides methods for loading the data tables from the XML structures, updating the structures and load from or save to files.

Editor Module. It contains editing interfaces and methods for verifying the input. In general, each interface is specialised to handle the editing of a specific part of the model. Then, when the interface is closed and the update is accepted, the editor module accesses the XML data handler for updating that part. Finally, it updates related tables and sends the update to the XML data handler module. The interfaces implemented by this module define the four editing modes described in Sect. 5.1

Simulation Module. It implements the Evolution Algorithm of Grid Systems and visualises the results. There are three simulation modes, which are described in Sect. 5.2

4 Case Studies

In order to describe, in Sect. 5, the model definition and simulation functionalities, we consider two case studies of population dynamics that have been modelled by means of Grid Systems in our previous work [4,21].

The first case study [4] consists in modelling the dynamics of a population of a species of mosquitoes, *Aedes albopictus*. The model includes objects that represent population individuals at different development phases: egg (E), immature (I), i.e. pupa/larva, and adult (A). Development takes place in small

water containers, and it is influenced by water level and environmental temperature. Hence, the model considers three types of external events, temperature change, rainfall, and desiccation, which change the behaviour of the species either directly or indirectly. The ecosystem is modelled as a 3×3 grid. Temperature is represented by the number of objects T in the global membrane, while the water level in containers is modelled by the number of objects W in the central cell of the grid. Reaction rules of this model include rules for the movement of adults into adjacent cells, rules for oviposition, rules for development (E into I, and I into A) and rules for death.

The second case study [21] consists in the modelling of the seasonal migration of a wildebeest species in the Serengeti National Park, Tanzania. The area of the park is modelled as a 50×50 grid. In order to precisely describe the shape of the territory, dummy objects Z are inserted in grid cells that are not included in the representation of the park. In the other cells, the instances of object G represent the quantity of available grass, objects $A1, \ldots, A9$ and $B1, \ldots, B9$ represent wildebeests at different stages and in different states (movable/resting). Instances of object C are used (for modelling purposes) to maintain the total number of individuals in the population.

The model relies on the observations that wildebeest migration is driven by the search for grazing areas and water resources, and individuals tend to follow movements of other individuals. Assuming the existence of dynamic guiding paths, which could be representations of individual or communal memory of wildbeests, or physical tracks marking the land, we model movement by rewritings between adjacent cells driven by conditions in the origin and destination cells. As conditions we consider number of individuals, grass available, and dynamic paths. Paths are intialised with the patterns of movements observed in reality, but dynamically change depending on variation of movement caused by other conditions. In the grid systems model, paths are represented by objects *path* carrying links that are dynamically created by reaction rules, and that are used by other rules to determine the movement of wildebeests. Reaction rules of the model include rules for birth and death of individuals, for grass growth (that uses axiliary objects R and H to describe grass growth phases), for feeding of individuals, for change of stage and state of individuals and for movement according to grass availability and paths. Moreover, reaction rules for the update of paths are present.

5 Features of the GUI

5.1 Model Definition and Editing

In this section we briefly illustrate how to define and edit a model by showing screenshots that refer to the *Aedes albopictus* case study [4]. Full details are available in the simulator documentation [19]. There are four editing modes that allow the user to define and modify a model.

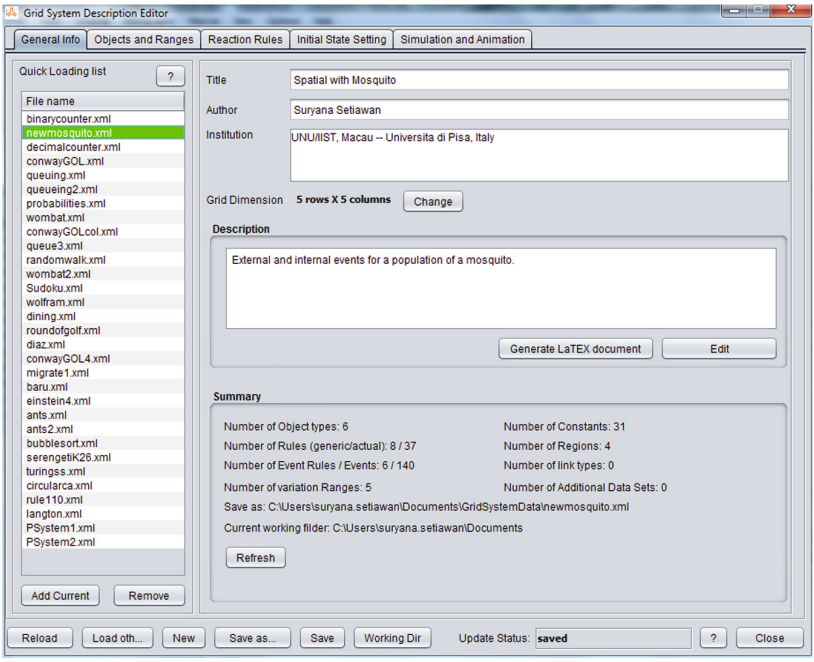

Fig. 1. The Dashboard: "General Info" Mode

"General Info" Mode. This mode allows the user to create a new model or select the model to be analysed from a list of models. The user can access and edit the model general information. As shown in Fig. 1, the list of models is on the left side while the editable information about the selected model is visualised on the right side. It is also possible to generate a LaTeX description of the model.

"Objects and Ranges" Mode. This mode allows the user to define the objects and the topology of the model. In particular, the following entities can be created and edited:

object which defines a species or a developmental stage of a species (e.g. Adult Mosquito, Egg, Pupa/larva), an environmental condition (e.g. Temperature level and Water level) or a movement constraint (e.g. Mosquito movement blocker);

constant which describes fixed values such as thresholds and can be an expression containing other defined constants and the values associated with the grid, such as number of rows (NumRows) and number of columns (NumCol);

region which is interactively defined by means of a dialog box by selecting cells and incorporating already defined regions over the canvas that represents the grid;

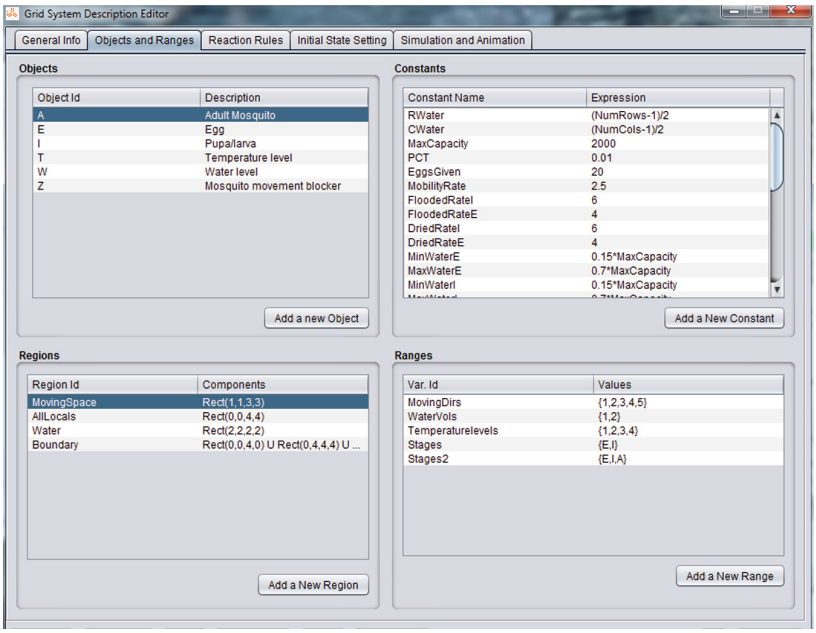

Fig. 2. The Dashboard: "Objects and Ranges" Mode

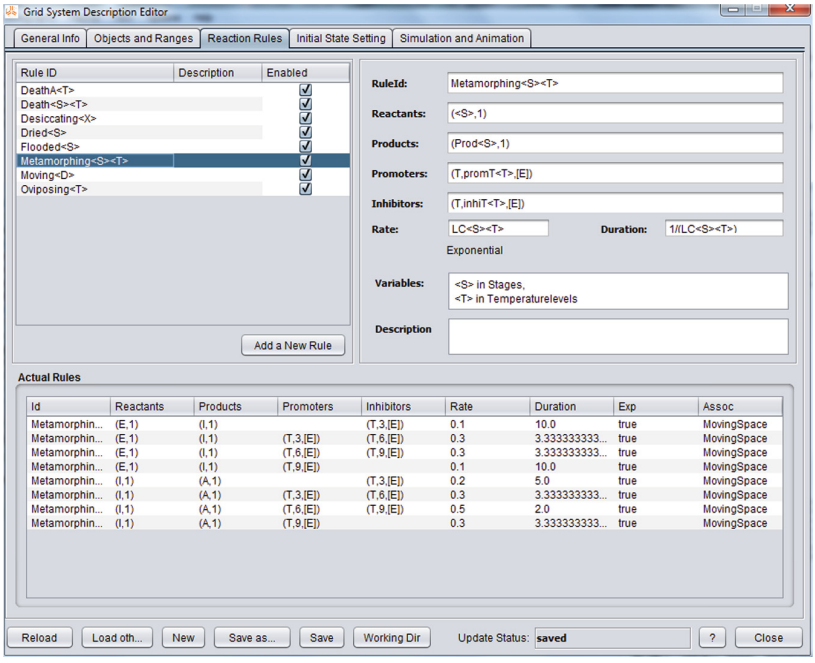

Fig. 3. The Dashboard: "Reaction Rules" Mode

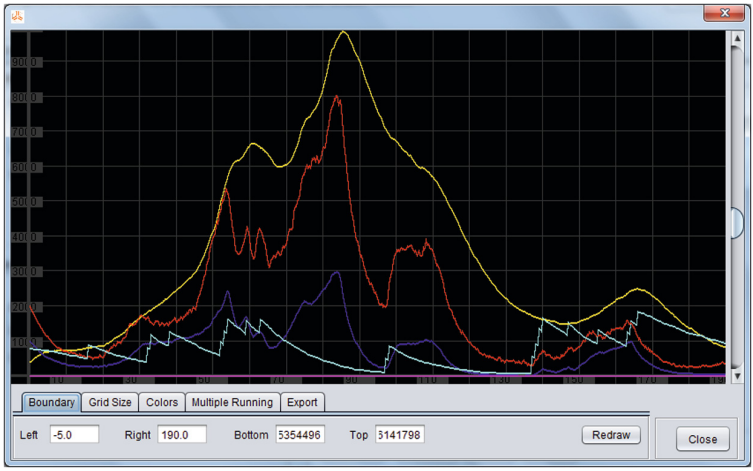

Fig. 4. Output as Chart

range which is defined as a set of discrete values associated to variables. The dashboard of this mode is shown in Fig. 2.

"Reaction Rules" Mode. This mode allows the user to define reaction rules. As shown in Fig. 3 the dashboard supports the definition of the rules according to Definition 2.1, the selection of the rules to be used in the simulation and visualises, in a tabular form, all actual rules corresponding to a highlighted template rule.

"Initial State Setting" Mode. This mode allows the user to define the list of datasets for different initial objects, links, external events and rules for events.

5.2 Simulation and Animation

Simulation can be performed using three different modes as follows.

"Growth" Mode. This mode is intended for observing how the configuration changes during simulation. There are two different ways to observe such changes: (1) as textual output, and (2) as a chart. The chart presents a graphical output using different colours for different objects as shown in Fig. 4 for a mosquito population (adults in yellow, immatures in dark blue and eggs in red) and its environment (water level in light blue) [4].

"Animation" Mode. Animating the simulation is essential to observe spatial aspect. This mode supports the visualisation of object changes in each membrane. It is illustrated on a model of the seasonal migration of a wildebeest species in the Serengeti National Park, Tanzania [21]. Figure 5 shows a map of the Serengeti National Park, with the white part, consisting of tiny dots, showing the density of the individuals of the species.

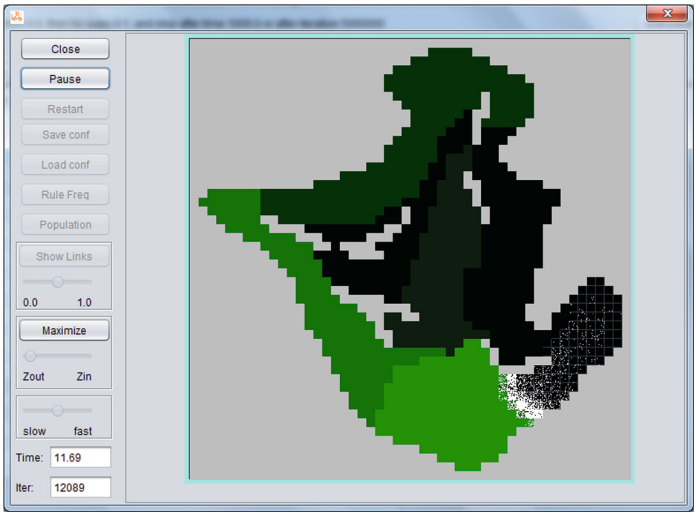

Fig. 5. Animation

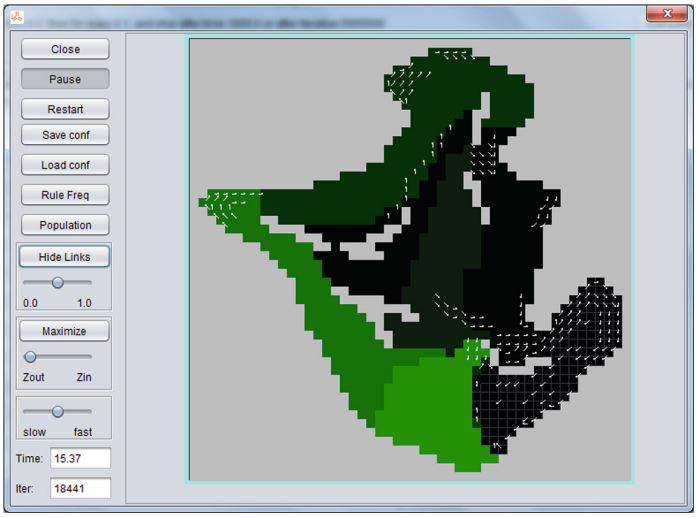

Fig. 6. Displaying the Links

The animation can be paused with button "Pause" (as in Fig. 6) to observe an instantaneous state, possibly saving the current configuration in a file (button "Save conf") in order to continue at a later stage (button "Load conf"), and can be restarted from the initial configuration (button "Restart"). When the animation is paused it is possible to inspect rule frequency (for both template and actual rules) with button "Rule Freq" and object numbers (available, committed and total) with button "Population".

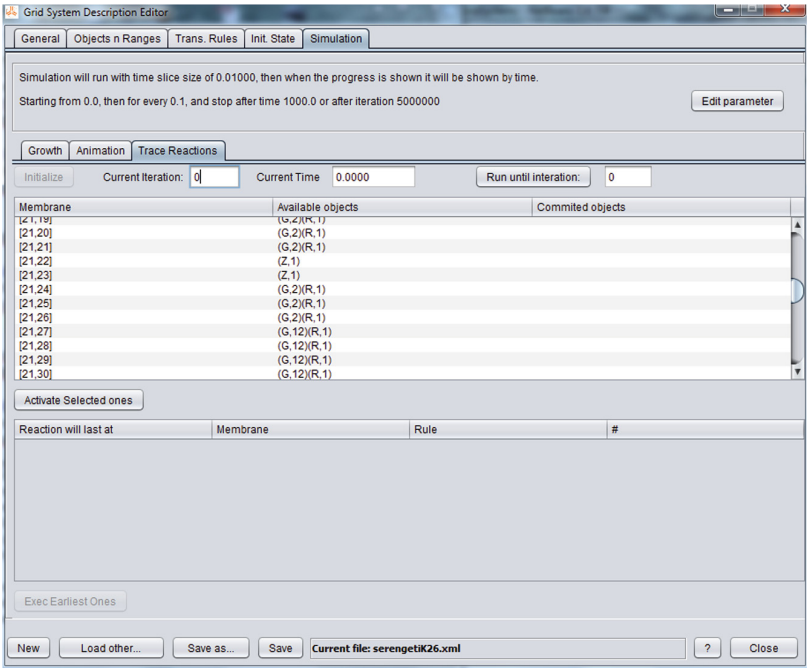

Fig. 7. The Dashboard: "Trace Reactions" Mode

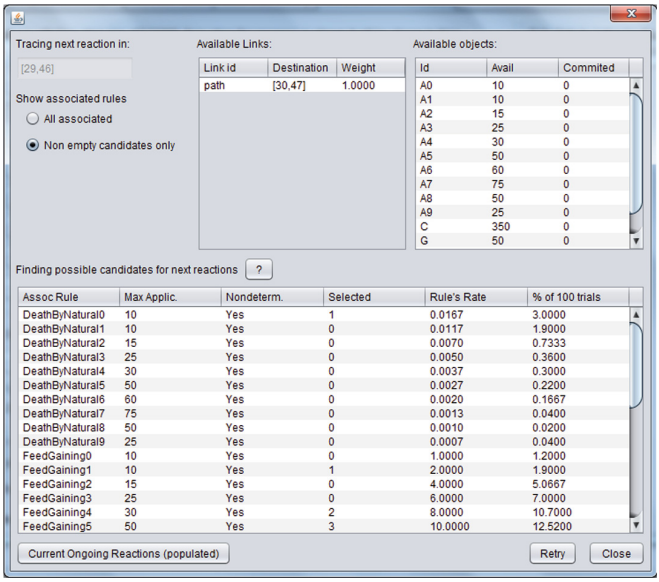

Fig. 8. State of a Membrane

It is also possible to visualise the current links as arrows on the canvas (buttons "Show Links" and "Hide Links"), possibly with the objects hidden (button "Only Links", which appears after pushing "Show Links"). Since links have different directions and each direction has its own weight, the interface provides a sliding bar to set the weight limits so that only directions with weights above the limits (high priority links) are visualised. Visualisation of links with objects hidden is shown in Fig. 6 for the wildebeest case study. Finally, frames of animation can be captured and stored in files.

"Trace Reactions" Mode. This debugging mode supports the modeller in observing the behaviour of the rules step-by-step during the simulation. As shown in Fig. 7, in this mode, the initial dashboard shows, for each membrane, the available and committed objects it contains. After selecting a membrane, it is possible to inspect its state, represented in textual format organised in tables. The state of a membrane (for rules with available reactants, as shown in Fig. 8, or for all associated reaction rules) includes:

- the available and committed objects;
- the current paths, including their destinations and weights;
- for each considered rule associated with the membrane;
  - the identifier of the rule for that reaction;
  - the maximum number of times the rule can be applied;
  - the existence of other rules competing for the same reaction;
  - the number of reactions resulting from the stochastic selection process applied to the rule;
  - the rule rate;
  - the percentage of trials, that is, the empirical likelihood that the rule is selected out of 100 trials;

6 Conclusion and Future Work

We have presented a simulator for ecological systems that is equipped with a GUI that supports visual animation and visual presentation of the output. We have illustrated the modelling, editing and analytical functionalities of the tool presenting screenshots from two case studies: the dynamics of a population of a species of mosquitoes, *Aedes albopictus*, and the seasonal migration of a wildebeest species in the Serengeti National Park, Tanzania. Simulator and documentation can be downloaded at http://www.di.unipi.it/msvbio/wiki/GridSystems/.

In our future work we plan to apply the simulator to other case studies in ecology and aim to extend it with further analytical functionalities based on model checking and statistical model checking.

References

1. NetLogo web site. https://ccl.northwestern.edu/netlogo/
2. Tools for system dynamics web site. http://tools.systemdynamics.org/
3. Adamatzky, A.: Identification of Cellular Automata. Taylor & Francis, London (1994)
4. Barbuti, R., Cerone, A., Maggiolo-Schettini, A., Milazzo, P., Setiawan, S.: Modelling population dynamics using grid systems. In: Cerone, A., Persico, D., Fernandes, S., Garcia-Perez, A., Katsaros, P., Ahmed Shaikh, S., Stamelos, I. (eds.) SEFM 2012 Satellite Events. LNCS, vol. 7991, pp. 172–189. Springer, Heidelberg (2014)
5. Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., Pardini, G.: Simulation of spatial P system models. Theor. Comp. Sci. **529**, 11–45 (2014)
6. Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., Pardini, G., Tesei, L.: Spatial P systems. Nat. Comput. **10**(1), 3–16 (2011)
7. Björnstad, O.N., Fromentin, J.M., Stenseth, N.C., Gjøsæter, J.: Cycles and trends in cod populations. Proc. Nat. Acad. Sci. U.S.A **96**, 5066–5071 (2009)
8. Bonsall, M.B., Hastings, A.: Demographic and environmental stochasticity in predator-prey metapopulation dynamics. J. Anim. Ecol. **73**, 1043–1055 (2004)
9. Cerone, A., Scotti, M.: Research challenges in modelling ecosystems. In: Canal, C., Idani, A. (eds.) SEFM 2014 Workshops. LNCS, vol. 8938, pp. 276–293. Springer, Heidelberg (2015)
10. Durier, V., Graham, P., Collett, T.S.: Snapshot memories and landmark guidance in wood ants. Curr. Biol. **13**, 1614–1618 (2003). Elsevier Science Ltd
11. Kahramanoğulları, O., Jordán, F., Lynch, J.: Cosbilab lime: a language interface for stochastic dynamical modelling in ecology. Environ. Model. Softw. **26**(5), 685–687 (2011)
12. Kaitala, V., Ranta, E., Lindstroem, J.: Cyclic population dynamics and random perturbations. J. Anim. Ecol. **65**, 249–251 (1996)
13. Kaupp, U.B., Kashikar, N.D., Weyand, I.: Mechanism of sperm chemotaxis. Annu. Rev. Physiol. **70**, 93–117 (2008)
14. Lohmann, K.J., Putman, N.F., Lohmann, C.M.F.: Geomagnetic imprinting: a unifying hypothesis of long-distance natal homing in salmon and sea turtles. Proc. Nat. Acad. Sci. **105**(49), 19096–19101 (2008)
15. Pauly, D., Christensen, V., Walters, C.: Ecopath, ecosim, and ecospace as tools for evaluating ecosystem impact of fisheries. ICES J. Mar. Sci. **57**(3), 697 (2000)
16. Penna, P., Paoletti, N., Scarcella, G., Tesei, L., Marini, M., Merelli, E.: DISPAS: an agent-based tool for the management of fishing effort. In: Counsell, S., Núñez, M. (eds.) SEFM 2013. LNCS, vol. 8368, pp. 362–367. Springer, Heidelberg (2014)
17. Păun, G.: Computing with membranes. J. Comput. Syst. Sci. **61**(1), 108–143 (2000)
18. Ripa, J., Ives, A.R.: Food web dynamics in correlated and autocorrelated environments. Theor. Popul. Biol. **64**, 369–384 (2003)
19. Setiawan, S.: Formal modelling for population dynamics. Ph.D thesis, Department of Computer Science, University of Pisa, May 2015
20. Setiawan, S.: The grid systems simulator (version date: 3 June, 2015) 2015. http://www.di.unipi.it/msvbio/wiki/GridSystems/
21. Setiawan, S., Cerone, A.: Stochastic modelling of seasonal migration using rewriting systems with spatiality. In: Counsell, S., Núñez, M. (eds.) SEFM 2013 Workshops. LNCS, vol. 8368, pp. 313–328. Springer, Heidelberg (2014)