ABSTRACT

Multiagent systems (MASs) are seemingly a promising way to model and simulate biological systems and networks, going toward a system-level understanding of such complex systems, as promoted by Systems Biology. In this paper we propose an agent-based framework based on the notion of agents and artifacts, implemented on top of the TuCSoN agent coordination model and infrastructure. The approach promotes the engineering of distributed simulations, where biological networks are modelled as multiagent systems. There, agents—representing active biological components such as proteins and enzymes—interact by means of tuple centres—representing the environment that enables and mediates the interaction of such components.

INTRODUCTION

Multiagent systems (MASs) have been widely recognised as a good approach for modelling and simulating complex systems (Epstein and Axtell 1996). As such, they can be seen as a promising choice also for modelling and simulating biological complex systems, in particular those scenarios in which traditional approaches—such as the one based on Ordinary Differential Equation (ODE)—fail. Typically, in such scenarios interaction inside the system and between the system and its environment plays a fundamental role and must be explicitly taken into account in order to understand the overall system behaviour, typically emergent, along with related effects concerning non-linearity, stochastic phenomena, feedbacks, and so on.

Several research studies coming from different disciplines are pointing out the importance of such aspects for understanding living systems. A foremost and recent example is given by Systems Biology (Kitano 2002), which collects approaches that aim at understanding biology at the system level. Agent-based approaches provide a natural way to model such aspects. In literature, we can identify two basic families for such approaches: either (1) MASs with relatively complex, large-grained agents, typically interacting through direct communication and without a notion of agent environment, or (2) agent-based approach based on very simple homogeneous agents—similar to cellular automata—with an explicit notion of computational environment, with which they explicitly interact.

In this paper we introduce an agent-based approach to model and simulate biological networks, characterised by large-grained agents as in the first case, but with an explicit notion of environment, as in the second case. The approach is based on the notion of artifact, used to model agent environments, implemented on top of TuCSoN, a coordination model (and infrastructure) exploited in the context of multiagent system coordination.

Our general objectives are:

- From a scientific point of view, defining a computational model for Systems Biology based on agent and artifact concepts, with artifacts realised in particular on top of TuCSoN tuple centres. The idea is to exploit the peculiar features of such a model to set up virtual experiments in which scientists can (i) simulate the behaviour of biological systems by executing the corresponding MAS, and (ii) both observe and influence system evolution, with also the chance of interacting with the system while it is running.

- From the engineering point of view, defining a general-purpose distributed platform to be reused for different kinds of experiments, capable of running distributed simulations, exploiting multiple computing machines linked through a network.

The remainder of the paper is organised as follows. In the next section, we introduce some key points of Systems Biology, starting from the main aspects that characterise biological systems as complex systems and of biological networks in particular; Thereafter, we first introduce a modelling framework for biological networks based on agents and artifacts, then we describe a computational model based on TuCSoN, taken as a concrete model to implement artifacts. Finally, we conclude the paper by briefly discussing ongoing and future works.
SYSTEMS BIOLOGY

Biological Systems as Complex Systems

Generally speaking, biological systems exhibit a number of key properties that characterise them as complex systems, namely:

- hierarchy — structures and processes in biological systems are organised in hierarchical highly-structured levels, each one exhibiting a different kind of complexity (Dhar et al. 2004); sequences, molecules, pathways (such as metabolic or signalling), networks as collection of cross-interacting pathways, cells, organs;

- interaction — each level is characterised by the interaction of a set of components inside some kind of environment. The interaction can occur both directly among the components, and between the components and their environment. Such interactions involve some kind of information exchange, where the information can be in the form of energy and material, frequently creating both negative and positive feedback loops;

- emergent behaviours — each level exhibits forms of self-organisation among the components, resulting in the development of emergent behaviours, whose existence does not depend, cannot be predicted and cannot be explained from the knowledge about the individual components. In other words, the emergent properties cannot be readily explained in terms of the properties of individual components, just as systems are not the mere sum of their individual components (Dhar et al. 2004);

- stochastic phenomena — stochastic effects play a crucial role at various levels of the hierarchy, in particular at those in which even the overall macro-behaviour can be influenced and determined by the interaction of a relatively small number of components. It is the case, for instance, of signal transduction pathways, involving a small number of particles (Chee Meng et al. 2004);

- non-linearity and chaotic behaviour — the type of interaction among the components often results in systems exhibiting a non-linear behaviour, i.e. systems that are able to react in quite different manners to similar condition or small variations occurring in their environment. According to several researchers, this is an essential and distinguishing feature of living systems. Such interactions can even result in chaotic behaviours, which, according to several studies, seems to be essential in several contexts for keeping a good functioning of the overall system.

These aspects indeed make the analysis and comprehension of biological system dynamics an incredibly challenging task, for which traditional mathematical modelling approaches proved to be inadequate.

The Systems Biology Approach

The study of biological systems as complex systems, with a strong focus on the system level and on the interaction dimension, is among the key points of Systems Biology (SB), recently raised in bio-informatics contexts as innovative way of doing biology. SB promotes a more systemic and informatics-oriented way to conceive models, methodologies, and simulations of biological systems. Following Kitano (Kitano 2002), the aim of SB is understanding biology at the system level, moving from the understanding of the individual cellular components—which is the main aim of traditional approaches, described as reductionist—to the understanding of how these components interact as systems to produce the observed emergent behaviour, and how all the pieces are integrated and assembled together so as to get a systems view of biology.

This objective leads to focus more on the dynamics of cellular and organismal function rather than on the characteristic of the isolated parts of a cell or organism. For the purpose, we need to (a) obtain a massive amounts of data about whole biological systems via high through-put experiments; (b) build, with such data, a science of the principles of operation of biological systems, based on the interactions between components (Cardelli 2005).

The main investigating tool at the core of the SB approach—and of computational Systems Biology (CSB) in particular—are informatics systems (including network of computing and software systems), which provide the necessary means to create computational models of biological systems and of their environment (in-silico biology), and to observe and control the dynamic emergent behaviours of the systems through simulations.

Such computational models typically promotes bottom-up methodologies and techniques for modelling and simulating systems, in contrast to top-down ones provided by traditional approaches. Top-down approaches are based on mathematical models created starting from the knowledge that scientists have about the overall system: how it behaves, what phenomena are linked to its behaviour, what are the general laws that rule such behaviour. Models are typically based on the identification of a basic set of variables representing key properties of the system behaviour—for instance: the concentrations of some type of molecules—and then by a set of ordinary differential equations (ODE) that describes how properties change along time and space dimensions. Such approaches typically require some strong continuity hypotheses to be feasible, which often are not satisfied in the biological systems under investigation. As a main example, they assume that the concentrations would change in a continuous and deterministic way, along quite long periods of time. This assumption is feasible only if we consider systems with a very high number of components, interacting in quite large environments, conditions that are not satisfied in most of
Involving some different kind of interacting entities: biological systems can be identified constituting a hierarchy, each describing essentially in terms of the laws that locally characterise component interactions, with a level of abstraction that makes the model more adherent to what happens in nature.

**Biological Networks**

By adopting the SB point of view, different levels of biological systems can be identified constituting a hierarchy, each involving some different kind of interacting entities:

- interaction between molecules inside a *compartment*, i.e. a functionally distinct region of a cell enclosed by a membrane;
- interaction between compartments in a cell
- interaction between cells of a system (e.g. immune system)
- interaction between systems (immune, nervous, endocrine)

In this paper we focus on the first one, i.e. intra-cellular interaction and processes, which can be categorised in two main kinds of pathways or networks: *metabolic pathways* and *signalling pathways* (Dhar et al. 2004).

Metabolic pathways contain a series of biochemical reactions in which the product of one reaction serves as the substrate of the next reaction. These pathways control the production, transformation and consumption of energy in cells, and are usually self-sustained and self-sufficient.

Signalling pathways are the biochemical pathways regulating the flow of information within and between cells, including the ability to propagate information across different time scales and to function as switches and oscillators. Thanks to these pathways the cell is allowed to respond and to adapt to an ever-changing environment. In fact, context-dependency is actually the key feature of signalling pathways.

If traditional mathematical approaches (such as flux balance analysis) have proved to be quite effective in modelling and simulating metabolic pathways, the same does not hold for signalling pathways. The mechanisms underlying these complex behaviours involve many interacting components and cannot be understood by experiments alone. No adequate mathematical models are known for analysing such pathways.

**AGENTS & ARTIFACTS FOR SYSTEMS BIOLOGY**

As pointed out by Systems Biology, the development of new computational models of biochemical networks, integrating existing experimental data, will be fundamental in next years (and perhaps decades) to understand, model, control and re-create biological systems (Finkelstein et al. 2004). Several innovative computational approaches have been developed recently in research literature, both from researchers belonging to biology contexts—such as Shimizu and Bray (2001)—and from researchers in disciplines more related to computer science and engineering, and bioinformatics—examples are approaches based on process algebra (Regev et al. 2001, Curti et al. 2004, Regev et al. 2004), Petri Nets (Peleg et al. 2002), reactive systems (Fisher et al. 2005), and even agents and MASs (Khan et al. 2003, Querrec et al. 2003, Pogson et al. 2004, Gonzalez et al. 2003, Corradini et al. 2005, Borghoff et al. 1996). In the following we introduce an agent-based computational model recently developed in the context of complex software systems engineering, which we consider effective for modelling some crucial aspects of biological networks and systems in general.

**Agent and Artifact Abstractions for Modelling Complex Systems**

Agent-based and multiagent systems have already been recognised as a promising approaches for modelling and simulating biological systems. Generally speaking, existing approaches can be subdivided in two main categories:

- MAS-based models that adopt intelligent or cognitive agents, interacting through direct communication, typically based on speech act-like models. An example is given by Khan et al. (2003), where a MAS based on large-grained agents is used for the quantitative simulation of biological networks. These approaches draw their inspiration or directly adopt models and systems used for engineering real software systems.
- Agent-based approach based on very simple homogeneous agents interacting in grid-like environments (Epstein and Axtell 1996).

These approaches have been created specifically in the context of simulation-based systems, and share several aspects with other related models, such as cellular automata.

The approach proposed in this paper, on the one side is based on a notion of agent as depicted for the first category. Accordingly, agents can be defined as pro-active entities situated in some kind of environment, where they autonomously execute some kind of activity toward the achievement of some goal. MASs introduce a social dimensions, modelling systems as set of set of agents situated in a shared environment interacting according to various kinds of social models.
(coordination, cooperation, competition and so on). Such an approach is based on a full distribution and encapsulation of the control: each agent encapsulates a state, a behaviour and the control of the behaviour, that is, of its activities. Different computational models can be adopted for defining agent behaviour, according to the complexity of its activities: from simple reactive models to cognitive models, adopted by the so-called intelligent agents (Wooldridge and Jennings 1995).

Then, differently from existing agent-based approaches of the first category—and more similarly to approaches found in artificial like contexts—we introduce a computational model in which also the agent environment is represented as a first class entity, and such an entity is fundamental for enabling forms of interaction among agents different from direct communication, such as mediated interaction, which can play a key role in the overall system coordination. We exploit the notion of artifact as basic abstraction to model and structure agent environment (Omicini et al. 2006, Ricci et al. 2006). Artifacts can be generally defined as objects—as synonym of entities—encapsulating some kind of function, which agents either individually or collectively exploit in the course of their activities, analogously to artifacts as found in human society (where humans play the role of agents). Artifacts can be used to represent either the target of agent activities—agents constructing, manipulating artifacts—or “tools” that agents use to help and sustain their activities. A simple example of artifact is given by a blackboard, as a kind of coordination artifact, designed to provide coordination functionality that agents can exploit in their collective activities. An agent interacts with an artifact through its interface, as a set of operations conceived by the artifact designer to let agents access and exploit artifact functionalities.

Artifacts then can be used as the basic abstraction to model such parts of a system that are not suitably captured by the agent abstraction: a primary example is given by the agent computational environment itself, in particular those parts that are specified and under control of MAS designers.

Modelling and Simulating Biological Networks

The general idea of this paper is to model and simulate biological networks and systems as multiagent systems exploiting the agent and artifact abstractions to represent different aspect such networks.

On the one side, we use agents to model any macromolecular component with a complex structure and behaviour, such as proteins, referred here as active bio-components. Agents are designed and programmed so as to simulate the behaviour of such components, modelled as autonomous stateful entities interacting with the bio-chemical environment where they are situated, acting in parallel, asynchronously. The behaviour of the active bio-components is then modelled in terms of agent activities or tasks, composed by set of actions with which the agents perceive and influence environment by consuming and producing molecules, and then changing their internal state. For instance, the computational behaviour of an agent representing a protein models how the protein changes its state according to the interaction with other molecules, and the interactive behaviour of protein sites—binding and unbinding molecules—as forms of actions and perceptions. So, the chemical reactions that characterise such bio-component can be used to design agent behaviour, modelling also important aspects such as reaction speed. Sensors and effectors of the agent represent the sites of the bio-components, through which it interacts with its environment, in our case with artifacts.

On the other side, artifacts are used to represent the biochemical environment (or compartment) that makes it possible for agents to survive and interact, both providing the operations to produce / consume molecules, and encapsulating and applying the rules that define and characterise such interaction. Such rules are not under the control of the individual interacting entities but are defined at a system level, and are essential to establish the overall system evolution. Among the responsibilities charged upon artifacts we count: (a) collecting and managing the set of molecules inside a compartment, as the resources that are shared and concurrently produced / consumed by the active bio-components as agents; (b) establishing dynamically which interactions can take place, i.e. which bio-components can consume / produce which molecules, according to what is known from the theoretical models, and taking into the account stochastic aspects. Actually, the artifact can be used also to embed and apply those chemical reactions that are not in charge to any specific agent.

A FRAMEWORK BASED ON TuCSoN

Given the conceptual background based on agents and artifacts, we exploit the TuCSoN coordination model and infrastructure to setup a general purpose framework for creating agent-based in-vitro distributed experiments, in which biological systems / networks are modelled and simulated, starting from simple and well-known cases.

The TuCSoN Model

An example of concrete model and technology supporting the agents and artifact conceptual model is given by the TuCSoN agent coordination infrastructure (Omicini and Zambonelli 1999).

TuCSoN (Tuple Centres Spread over the Network) has been ideated in the software engineering context, as a coordination model and infrastructure to be exploited for engineering complex agent-based software systems. TuCSoN provides tuple centres as coordination artifacts that agent can use to
Roughly speaking, a tuple centre can be figured out as a sort of reactive blackboard, a shared place where agents can put and retrieve in a pattern-driven manner pieces of structured data called tuples, with basic synchronisation facilities ruling such insertions and associative retrieval (see Fig. 1). The blackboard is reactive in that it can be programmed to react to data insertion / retrieval and communication events in general, applying rules that manipulate the tuples so as to implement some kind of coordinating functionalities. Among the operations listed of the interface of the tuple centre as artifact we cite here \( \text{out}(T) \) to insert a tuple \( T \), \( \text{in}(TT) \) to remove or just read a tuple matching the template \( TT \) and \( \text{r} \text{d}(TT) \) to read a tuple without removing it from the tuple centre. Technically, a tuple centre is a \textit{programmable tuple space}, that is a tuple space whose interactive behaviour can be dynamically programmed by means of a set of reactions encoded in a language called \textit{ReSpecT} (Omicini and Denti 2001). Both the tuples and the \textit{ReSpecT} language are logic-based: tuples—called logic tuples—are encoded as Prolog facts of the kind \( \text{reaction(Event,Body)} \), which specify a sequence \textit{Body} of basic operations (manipulating the tuple set or observing the properties of the Event) that must be executed whenever the specified communication event Event occurs. An example of simple reaction is the following:
\[
\text{reaction(out(pp(X,Y)),}
\text{(in\_r(pp(X,Y)),out\_r(p(X)),out\_r(p(Y))))}.
\]
The reaction is triggered by the insertion of any tuple matching the template \( pp(X,Y) \), and has the effect of removing the tuple just inserted, and of inserting two new tuples, \( p(Y) \) and \( p(Y) \). The detailed description of the tuple centre model and of \textit{ReSpecT} can be found in Omicini and Denti (2001).

\textbf{TuCSoN infrastructure} makes it possible to exploit tuple centres in a distributed and network environment, by providing services to access and use tuple centres that are spread over the nodes of the network (Internet)—wherever the \textit{TuCSoN} infrastructure has been installed. A \textit{TuCSoN} node can contain any number of tuple centres, each one identified by a logic name. A software system engineered upon \textit{TuCSoN} consists then by a MAS—adopting eventually different models an technologies for implementing the individual agents—distributed among some nodes and exploiting tuple centres, possibly located in different nodes, to communicate and coordinate.

In this paper we exploit \textit{TuCSoN} on the one side as source of abstractions for defining the computational model for biological networks, and on the other side as an infrastructure and a technology for engineering distributed simulations, based on a general purpose framework for creating agent-based \textit{in-vitro} distributed experiments.

\section*{A Simple Computational Model based on TuCSoN}

By exploiting \textit{TuCSoN}, we can keep the separation of concerns discussed in previous section, using artifacts—tuple centres in this case—to model the compartment and agents for representing active bio-components.

A compartment is modelled by a single tuple centre, in which: (a) we keep track of the the set of molecules inside the compartment as logic tuples contained by the tuple centre; (b) agents (active bio-components) can produce / consume (bind) molecules by executing suitable \textit{out} and \textit{in} operations on the tuple centre; (c) the rules establishing which interactions can take place are encapsulated in the tuple centre behaviour and encoded then in the \textit{ReSpecT} language.

In particular, as far as point (a) is concerned, we use tuples of the kind 
\[
molecule\text{(Type,Number)}
\]
A tuple of such a type means that the compartment contains a number of molecules of type \textit{Type}. \textit{Type} is a simple name, used to identify the kind of molecules involved in the system: for example, \textit{atp, glucose}. For instance a tuple 
\[
molecule\text{(atp,1000)}
\]
indicates that in the compartment there are 1000 molecules of ATP.

As far as point (b) is concerned, by using \textit{TuCSoN} agent sensor and effectors—i.e. the sites of the bio-components—are realised through \textit{in} and \textit{out} operation, which agents invoke on the tuple centre. In particular, an agent binds molecules to its current set of free sites by executing an \textit{in} operation of the kind
\[
in\text{(molecules(MolTypes,Affinities,Result))}
\]
where \textit{MolTypes} is the list of the types of molecules that the agents can consume, one for each free site; \textit{Affinities}...
is the list of the affinity coefficients associated to each molecules type—so one type for each site—, representing a measure of the probability of binding such molecule type: 

Result is a list representing the number of molecules consumed for each type (site) specified in MolTypes, as result of the interaction. As an example, an agent executing an operation

```plaintext
in(molecules([atp,glucose],[0.5,0.8],Result))
```

has two free sites, which can be bound respectively to an ATP molecule and a glucose molecule, with affinity respectively of 0.5 for ATP and 0.8 for glucose. The operation succeeds if (when) one molecule is actually consumed from the compartment and bound to one of the site. As an example of tuple returned by the operation

```plaintext
molecules([atp,glucose],[0.5,0.8],[1,0])
```

indicating that one molecule of ATP has been bound to the first site.

Analogously, the production of molecules is realised by means of a simple out operation of the kind

```plaintext
out(molecules(MolTypes,Quantities))
```

where MolTypes has the same structure as before, and Quantities is a list of integers indicating the number of the molecules of the specified type must be produced. For instance, by executing a

```plaintext
out(molecules([glu,adp],[1,1]))
```

the agent produces in the compartment a molecule of glucose and a molecule of ADP.

Finally, as far as point (c) is concerned, the tuple centre is programmed so as to update the set of molecules according to agent operation requests on the one side, and to select the interactions that will occur on the other side. That is, given the overall set of agent requests (molecules that can be consumed), the tuple centre establishes which ones will be served, from time to time. For the purpose, different kinds of models can be experimented, implementing strategies of different complexity: for instance, the approach currently adopted takes into account the concentrations of the molecules and the affinity specified by agents to select—using stochastic laws—which interactions will occur—that is which molecules will be bound by agents. From the overall MAS point of view, such strategies represent the laws that rule agent cooperation and competition in consuming and producing molecules.

An Example: Glycolysis

First experiments have been developed by considering glycolysis, a well-studied metabolic pathway occurring in citosol compartment.

The overall process counts ten steps in cascade, in which the product of a stage is the substrate needed for the next stage: the net result is the transformation of 1 molecule of glucose into 2 molecules of pyruvate, consuming two ATP during the process and producing, at the end, 4 ATP and 2 molecules of NADH. Each stage is carried on by some kind of enzymes, who mainly play the role of catalysts, consuming substrates and generating products. Actually, three of them—namely hexokinase (HK), phosphofructokinase-1 (PFK-1) and pyruvate kinase (PK)—play also another crucial role, as controllers of the overall process. Through allosteric regulating mechanisms, they act as valves of the glycolytic process, guaranteeing that globally on the one side there would not be an excess of products (ATP and pyruvate), on the other side that the speed of overall process would be adequate to produce the energy (ATP) needed for the cell to work.

By adopting the modelling approach presented in the paper, enzymes—controllers in particular—are modelled as agents and the citosol, the environment where the enzymes act, is represented by a tuple centre. Here we focus on the modelling of the PFK-1, represented by the EnzPFK agent: the others agent are analogous. The enzyme activity, as a catalyst, is to execute the following reaction:

\[ \text{F6P} + \text{ATP} \rightarrow \text{FBP} + \text{ADP} \]

where F6P is fructose-6-phosphate and FBP is fructose-1,6-biphosphatate.

As a valve, such activity changes according to its state. It has four different states—ready, active, inhibited, and reactive—, which changes according to the interaction that the enzyme has with its environment. In each state—which corresponds to a different physical structure and spatial configuration—the enzyme exposes different sites for binding molecules. In particular, the enzyme has five sites of three different kinds: two sites for binding substrate molecules—fructose-6-phosphate and ATP—, two sites for binding molecules that act as activators—both ADP—, and one site for binding molecules that act as inhibitors, that are ATP.

The overall behaviour of the agent is briefly described as follows. In the ready state, the enzyme has all the sites free to bind molecules, resulting in an in of the type:

```plaintext
in(molecules([f6p, atp, atp, adp, adp], [0.5,0.5,0.1,1,1],Result))
```

If it binds both substrate molecules, it becomes reactive; if it binds ATP in the inhibitor state, it becomes inhibited; if it binds ADP, it becomes active.

In the active state the enzyme has a greater affinity with substrate and then a greater probability to bind such molecules. The free sites are the ones for binding substrate molecules and for inhibition. The resulting in is

```plaintext
in(molecules([f6p, atp, atp], [1,1,0.1],Result))
```

As in the previous case, if it binds both substrate molecules, it becomes reactive; if it binds ATP in the inhibitor state, it becomes inhibited.
In the reactive state, the enzyme transforms the substrate molecules in products, taking a certain amount of time (properly scaled) computed according to the speed of the reaction and stochastic effects, and then it releases the products in the citosol by means of an out of the kind:

\[
\text{out}(\text{molecules}([\text{fbp}, \text{adp}], [1,1]))
\]

In the inhibited state, the agent interacts with the environment only by means of sites to be reactivated. The resulting is:

\[
\text{in}(\text{molecules}([\text{adp}, \text{adp}], [1,1]), \text{Result})
\]

When both the sites have bound an ADP molecule, then it becomes ready again. It is easy to see that the more the concentration of ADP is high, the more the probability to bind ADP is high and the agent can start again to play its catalysing activity.

The tuple centre playing the role of citosol environment is programmed so as to behave as follows: (a) It updates the set of molecules (represented by the tuples molecule) accordingly to the action of the agents, consuming and producing molecules. In particular, it reacts to

\[
\text{out}(\text{molecules}(\text{MolTypes}, \text{Quantities}))
\]

actions and updates all the tuples molecule\((T,Q)\) for which \(T\) is in the MolTypes list, adding the value in Quantities to \(Q\). (b) It keeps track of the agent requests (done with

\[
\text{in}(\text{molecules}(\text{Mol}, \text{Aff}, \text{Res})\) actions) and generates the tuples that satisfy such requests according to the availability of the molecules requested, the affinity specified in the request and stochastic effects in general. Such a behaviour is encoded as a set of reactions in ReSpecT language.

**DISCUSSION**

The adoption of the agent and artifact abstractions to model biological systems—supported by an infrastructure such as TuCSoN—makes it possible to scale quite well with the complexity of the systems, from both the conceptual and engineering point of view.

From a conceptual point of view, by means of the agent abstraction we can model interacting entities specifying a behaviour that can range from simple state-based stochastic behaviour (as in the case of PFK-1 enzyme in the example) up to entities with an “intelligent behaviour”, encapsulating more complex strategies for choosing actions and reacting to environment stimuli. This is particularly useful when—depending on the specific biological system—it is more natural for researchers and scientists to model the behaviour of the components starting from the knowledge they have about what is the role (objective or goal) of such components inside the systems, instead of specifying how they achieve such objectives (in terms of detailed interactions).

From an engineering point of view, TuCSoN naturally supports distributed simulations, in which agents (interacting through the same tuple centre) can be distributed over multiple machines, as well as tuple centres, representing distinct compartments.

By virtue of such a flexibility, it will be interesting to explore the approach for modelling and simulating other biological networks at a different level, such as artificial immune systems, comparing the results with existing approaches (Stepney et al. 2005, Celada and Seiden 1992).

**ONGOING & FUTURE WORK**

Starting from the computational model presented in this paper, ongoing works are focusing on setting up a suitable application environment for flexibly executing distributed simulations, testing the validity of the model. Such application environment is based on the TuCSoN technology, available on Java platform as an open source project.\(^1\)

As previously mentioned, the objective is to create a virtual laboratory distributed on multiple machines, on top of which executing distributed simulations engineered as MASs, with agents interacting through one or multiple tuple centres, according to the specific experiments. The idea is to create an environment not only for executing the simulations and observing dynamically the results from the simulation run, but also to execute online experiments, in which scientists—supported by suitable tools—could interact with the ongoing systems, for instance changing the structure of the systems, both by introducing / removing agents (active components, such as proteins, enzymes) or by changing the content of the tuple centres (that is, changing the molecules quantities of the compartments).

As test cases, we will consider some well-studied biological pathways, such as the glycolysis metabolic pathway as briefly discussed in the paper, and the MAPK (Mithogen-Activated Protein Kinase) transduction pathway, which has a key role in cell growth and cell cycle.

**REFERENCES**


\(^1\)[http://tucson.alice.unibo.it](http://tucson.alice.unibo.it)


**Acknowledgment**

We would like to thank the reviewers for their precious comments, which have been very helpful to identify further issues to investigate, and our colleagues bioengineers and biochemists at DEIS for their support and cooperation.