Abstraction of agent interaction processes: Towards large-scale multi-agent models

Abbas Sarraf Shirazi, Sebastian von Mammen and Christian Jacob

SIMULATION 2013 89: 524 originally published online 5 March 2013
DOI: 10.1177/0037549712470733

Simulation

Simulation: Transactions of the Society for Modeling and Simulation International
89(4) 524–538

DOI: 10.1177/0037549712470733
sim.sagepub.com
SAGE SCS

Abstraction of agent interaction processes: Towards large-scale multi-agent models

Abbas Sarraf Shirazi[1], Sebastian von Mammen[1] and Christian Jacob[1,2]

Abstract

The typically large numbers of interactions in agent-based simulations come at considerable computational costs. In this article, we present an approach to reduce the number of interactions based on behavioural patterns that recur during runtime. We employ machine learning techniques to abstract the behaviour of groups of agents to cut down computational complexity while preserving the inherent flexibility of agent-based models. The learned abstractions, which subsume the underlying model agents' interactions, are constantly tested for their validity: after all, the dynamics of a system may change over time to such an extent that previously learned patterns would not reoccur. An invalid abstraction is, therefore, removed again from the system. The creation and removal of abstractions continues throughout the course of a simulation in order to ensure an adequate adaptation to the system dynamics. Experimental results on biological agent-based simulations show that our proposed approach can successfully reduce the computational complexity during the simulation while maintaining the freedom of arbitrary interactions.

Keywords

agent-based simulation, collective behaviour, abstraction, optimization, online learning

1. Introduction

Phase transitions in complex systems cannot be inferred from the properties of the underlying parts. Rather they occur due to the interactions of the involved variables.[1] The agent-based modelling approach is a well-suited means to model complex systems, as it provides each part of the system with the ability to change its own state and to interact with other parts. Agent-based computational models have also gained great popularity as they can address heterogenous populations, noise, spatial and temporal relationships.[2–5]

The flexibility of agent-based models renders their simulation computationally inefficient.[6] As each agent could potentially interact with all of the other n agents, merely identifying who interacts with whom becomes a computationally expensive task: $O(n^2)$ in the worst case. To overcome this problem, agent-based simulations are often limited to fixed neighbourhoods in discrete lattice spaces as implemented by cellular automata.[7–9] However, the ability of a model to continuously change the interaction topology among the agents is crucial to trace, for instance, the dynamics of transportation effects[10] or developmental processes.[11]

In this article, we present an approach to apply machine learning techniques such as evolutionary algorithms, neural networks, and clustering in order to reduce the computational costs of an agent-based simulation while preserving its inherent flexibility. In particular, we show how groups of agents that exhibit behavioural patterns can be reduced to single agents with (computationally) simplified interaction rules. In order to identify a group of agents that can be substituted by a single agent, either neighbouring agents form a group or, more generically, an *observer agent* monitors arbitrary groups of agents and substitutes them based on their exhibited behavioural patterns. As the agents' interactions may vary over time, the learned behavioural patterns may lose their validity. Therefore, confidence values determine the lifespan of the learned

[1]Department of Computer Science, Faculty of Science, University of Calgary, Canada
[2]Department of Biochemistry and Molecular Biology, Faculty of Medicine, University of Calgary, Canada

Corresponding author:
Christian Jacob, University of Calgary, 2500 University Drive NW, Calgary, AB, Canada T2N 1N4.
Email: cjacob@ucalgary.ca

behavioural abstractions. Continuous re-evaluation of these confidence values allows for a self-organized optimization process in which the substitutions are adaptively created and revoked.

The remainder of this paper is organized as follows. Section 2 reviews related works in multi-agent modelling and abstraction. In Section 3, we first show how we can use artificial neural networks (ANNs) to learn the collective behaviour of agent groups. Next, we present an approach that relies on genetic programming (GP) and manages agent hierarchies dynamically, i.e. it does not destroy the learned abstractions completely should their confidence values drop, but only incrementally, as needed. In this context, we also elucidate the algorithm that ensures the validity of any learned patterns. Section 4 further refines the introduced approaches to consider arbitrary types of agent interactions including collisions and state changes. In order to demonstrate the effectiveness of this refined approach, we apply it to an agent-based blood coagulation simulation. Finally, concluding remarks are presented in Section 5.

2. Related work

Abstract knowledge represents higher-order patterns that occur in lower-level concepts. It bears the essence of a system and ignores unnecessary details.[12,13] Higher-order patterns emerge from the interactions of the parts, or agents, of a system.[4] In natural systems the formation of higher-order patterns happens across several scales of time and space, which renders their complete description impossible. However, it has been suggested that one could approximate the multiple scales of natural systems and their interdependencies by means of computational models that incorporate hierarchies of agents. High-level agents in such hierarchies correspond to high degrees of abstraction of the system processes. In this section, we briefly describe some of the related works that motivated or addressed this concept.

2.1 From bottom-up to abstract models

Artificial chemistries[14] and computational developmental systems, such as L-systems,[15] relational growth grammars,[16] or swarm grammars,[17] explicitly, often visually trace the emergence of high-level structures based on simple constituents. These constituents may be represented as formal symbols or as entities in physics simulations. Complex interaction patterns can emerge from even the most simple interactions. Autocatalytic networks, for example, denote patterns of chemical reactions that nurture one another.[18] Stable interaction networks may even exhibit the property of self-replication.[19] As a result, the formation of intertwined entities is promoted and hierarchies of increasing complexity emerge in nature.[20]

Rasmussen et al. designed a computational model based on artificial chemistries, in which structures are formed with an increase in complexity and with different functionalities: from monomers to polymers to micelles.[21] Although these experiments clearly retrace the formation of patterns at several levels of scale, Dorin and McCormack claim that such phenomena are not surprising given the model's simplicity. Dorin and McCormack argue that it takes considerably more effort to determine the novelties at higher levels in the hierarchy.[22]

A first step toward the identification of high-level patterns is to gain clarity about the abstractions inherent in an agent model to begin with. Bosse et al. propose the classification of types and levels of abstraction of agent-based models based on the following dimensions:[23]

The process abstraction dimension deals with the behaviour representation of an agent, e.g. whether an agent is modelled by its inputs and outputs, whether other variables like beliefs or desires are also considered, or whether even lower level properties of an agent are modelled.

The temporal dimension deals with the definition of the agents' behaviours over smaller or longer periods of time.

The agent cluster dimension specifies the granularity of the agent-based model, i.e. whether an individual agent represents an entity or a cluster of entities.

Ralambondrainy et al. identify the complex task of observing a simulation, for which they propose a separate multi-agent system.[24] They describe an ontology to facilitate the communications of the agents in the second system. The observation agents have three main tasks, namely (1) acquisition of observational elements, (2) processing of simulation results, and (3) presentation of the results to human actors. Although the second system does not affect the original simulation, the notion of a separate system with the ability to present higher-level, abstract knowledge emphasizes the necessity to have external observers in the simulation.

Several approaches rely on a priori definitions to identify emergent patterns in agent-based simulations. Servat et al. acknowledge that simulation states can provide clues for the introduction and configuration of high-order agents.[25] However, they insist on the necessity to predefine the behaviours of high-level agents. The same is true for Chen et al.'s formalism which they specifically use for validating predicted behaviours.[3,26]

In order to capture emergent phenomena, Dessalles and Phan foresaw a system in which detectors would identify emergent patterns and subsume the activity of the respective lower level objects.[1] Similarly, Denzinger and Hamdan introduce a modelling agent that observes perceivable

behaviours of other agents and maps them to a predefined stereotype.[27] However, Denzinger and Hamdan also present a novel aspect: The periodic re-evaluation of the agents' behaviours gives the modelling agent the opportunity to adjust the mappings in accordance with the dynamics of the system. Not only might the local interaction patterns change over time, but high-level phenomena might also influence the underlying layers. Lavelle et al. use the term *immergence*, or downward causation, to describe the impact of high-level organizations on entities at lower scales.[28] They postulate that explicit functions must be defined to bridge between micro and macro levels.

Cardon proposes three organizational levels to control the behaviour of a multi-agent system.[29] The constituent agents are defined in the aspectual level. A geometrical mapping of aspectual agents forms the second level called the morphological level. Using a simplified, higher-level representation of agents in the morphological level, analysis agents in the evocation level identify the current state of the simulation and control the agents in the aspectual level by tampering their behaviour. This approach provides self-adaptability in the system while enforcing a degree of control on the behaviour of the system as a whole.

von Mammen et al. introduced the concept of *self-organized middle-out abstraction (SOMO)*, where observer agents monitor the interaction history of sets of agents, use motif discovery to detect recurrent patterns, and create hierarchies of high-level agents that subsume the lower interacting agencies.[30] Although they do not exclude the possibility of a relationship between learned high-order patterns and emergent phenomena found in nature, SOMO primarily targets an increase of efficiency by repeatedly substituting groups of agents by individual high-level instances that work at lower computational costs.

The authors of the present article have previously demonstrated that high-level agent substitution indeed results in a reduction of computational cost.[31,32] In particular, we deployed ANNs and GP, two established inductive learning methods, to learn agent abstractions in a model of a biological signaling pathway. Clusters of biological substrates and their corresponding activation patterns were substituted by meta-agents. We recently extended our earlier implementation by introducing observer agents that are able to abstract arbitrary patterns of groups of agents.[6]

2.2 Toward a framework for multi-scale modelling

As technology advances the design of multi-scale models becomes more prominent. As long as these approaches merely connect models of different scales and feed back and forth the computed results as variable parameters, the challenge can be addressed with the right level of domain knowledge and software engineering skills.[33,34] As discussed in Section 2.1, there are only few concepts that address the issue of automatic identification and abstraction of emergent patterns, which is crucial for a system that would identify new levels as a result of the computational process.

Martins et al. review different multi-scale models (from biomolecules to cells, tissues and organs) and conclude that despite the lack of a quantitative model of a cell, such models may help understand cancer growth and its therapy.[35] Erson and Cavusoglu propose a software framework for multi-scale model integration and simulation;[36] however, no specific modelling technique is described. There are a few physical multi-scale models, e.g. CPM[37] and Synergetics.[38] However, as of yet, there is no universally adopted computational framework for the assembly of multi-scale biological models.[39]

Bassingthwaighte et al. identify a systems approach for developing multi-scale models which includes six steps:[40] (1) the definition of the model at its highest level of resolution, (2) the abstraction of patterns (''reduced-form modules''), (3) the identification of valid parameter ranges of these abstractions, (4) the observation of the variables of the system, (5) replacement of higher-resolution models with abstractions, and (6) the validation of the performance of the multi-scale model against available real-world data. The authors further discuss open challenges of their approach such as parameter identification in closed-loop systems and the identification of input–output delays.

3. Abstraction in the MAPK signaling pathways

A signaling pathway describes how information travels from the receptors of a cell to an inside target.[41] Typically, the information ripples through a cascade of biochemical reactions that are carried out by enzymes. The mitogen-activated protein kinase (MAPK) pathway plays a key role in the cell cycle and is documented extensively. It is responsible for responses to extracellular stimuli and regulates cellular activities, such as gene expression, mitosis and differentiation.[8] In the MAPK signaling pathway proposed by Huang and Ferrell,[42] a hypothetical enzyme E1 stimulates the cell and results in an increase in production of the MAPK-PP enzyme (Figure 1(a)). In another model,[43] a negative feedback loop causes sustained oscillations in the production of MAPK-PP (Figure 1(b)).

The diagram in Figure 1 describes the interaction topology of substrates. Numerical differential equation solvers are used to calculate their concentration updates over the course of time. For example, the update formula for the MAPK-PP concentration is given as follows:

$$d[MAPK-PP]/dt = v_8 - v_9 \quad (1)$$

$$v_8 = \frac{k_8 \cdot [MKK-PP] \cdot [MAPK-P]}{K_8 + [MAPK-P]} \quad (2)$$

$$v_9 = \frac{V_9 \cdot [MAPK - PP]}{K_9 + [MAPK - PP]} \quad (3)$$

where k_8, K_8, V_9, and K_9 are constants and $[X]$ is the current concentration of substrate X. The complete set of update equations can be found in Huang and Ferrell[42] and Kholodenko.[43]

Amigoni and Schiaffonati present three approaches to multi-agent simulations of the MAPK pathway.[41] In the first approach, each chemical reaction is represented as an agent.[8] The second approach translates each intracellular component into an agent that uses a blackboard mechanism[44] to interact with other agents in the system.[45] In the third model, each molecular entity acts as an agent.[46] For our model, we follow the last approach and consider each substrate a loosely defined, independent agent. Their behaviours are determined by the interaction graphs shown in Figure 2 and the update formulas given in Equations (1)–(3).

3.1 Creating meta-agents

In our system, an agent maintains a list of all of its neighbours and it logs their respective interactions in so-called interaction histories. It weighs the relationships with its neighbours based on its *correlation coefficient*. A correlation coefficient between two statistical variables indicates their linear dependency. A correlation coefficient of zero implies that two variables are independent, whereas ± 1 indicates highly correlated variables. The greater the correlation between two variables, the more similar is their function. Given a series of n measurements of agents s and

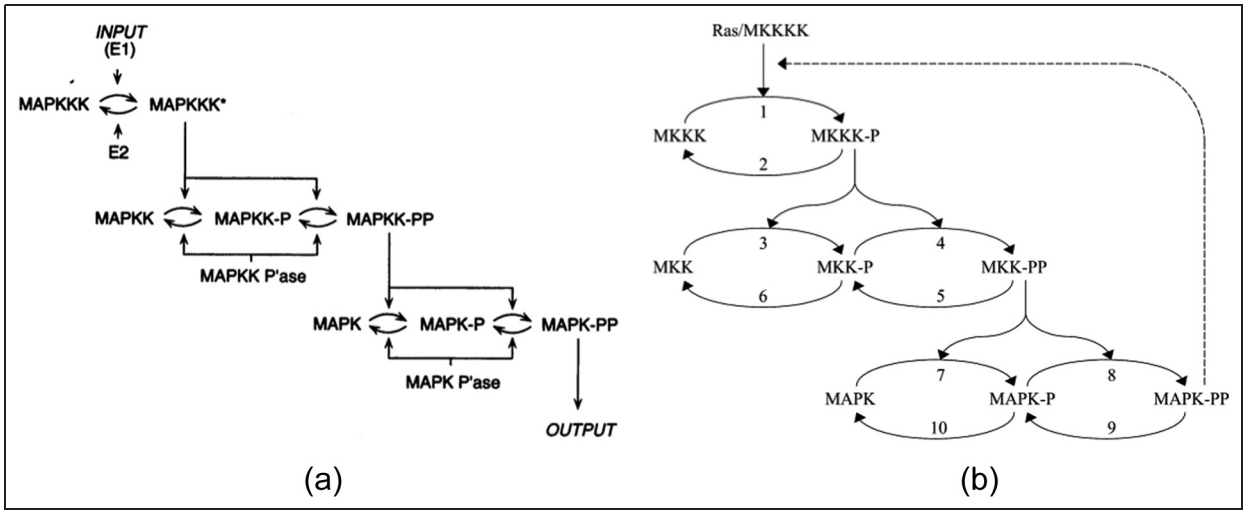

Figure 1. (a) The MAPK signaling pathway[42] and (b) The MAPK signaling pathway with a negative feedback.[43]

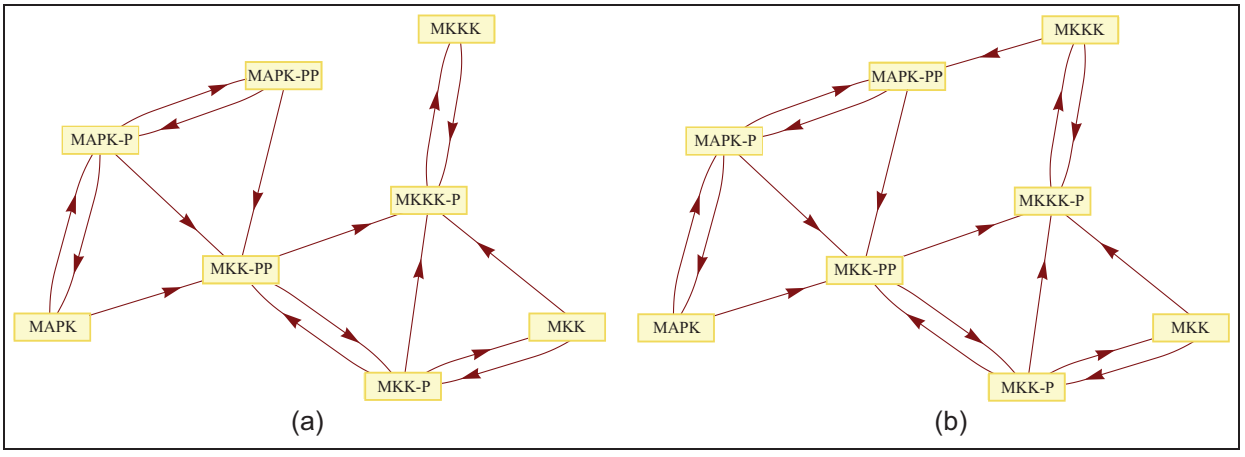

Figure 2. Agent interaction graphs for the MAPK signaling pathways of Figure 1.

t in the form of s_i and t_i, where $i = 1, 2, \ldots, n$, their correlation coefficient (ρ_{st}) is defined as follows:

$$\rho_{st} = \frac{\sum_{i=1}^{n} (s_i - \bar{s})(t_i - \bar{t})}{(n-1)\sigma_s \sigma_t} \quad (4)$$

where $\bar{s}$ and $\bar{t}$ are the mean values, and σ_s and σ_t are standard deviations of s and t, respectively.

Algorithm 1 Meta-agent creation.

```
m = current_agent;
Agent new_agent;
Queue q;
q.Enqueue(m);
new_agent.Add(m);
while !q.empty() do
  Agent head = q.Dequeue();
  for all Agent s in head do
    for all Agent t in s.Neighbours() do
      if |ρ_st| ≥ τ_edge then
        new_agent.Add(t);
        q.Enqueue(t);
      end if
    end for
  end for
end while
return new_agent;
```

Algorithm 2 Validity monitoring.

```
m = current_agent;
needToBreak = false;

for all Agent s in m do
  for all Agent t in s.Neighbours() do
    if |ρ_st − ρ'_st| > τ_valid then
      needToBreak = true;
      break;
    end if
  end for
end for

if needToBreak then
  simulation.remove(m);
  for all Agent s in m do
    simulation.add(s);
  end for
end if
```

Each agent periodically checks whether its correlation coefficient with each neighbour is greater than some threshold τ_{edge}. If this is the case, they form an initial meta-agent. This heuristic process is repeated in order to identify a cluster of agents that are highly correlated (Algorithm 1). Figure 3 shows an example in which *Agent A* finds *Agent C* and *Agent E*, and they form a meta-agent. The set of new neighbours is the union of all neighbours of the underlying nodes.

3.2 Learning the group behaviour

A new meta-agent replaces its underlying agents and interacts on their behalf. In order to approximate the subsumed agents' group behaviour, a learning algorithm such as ANNs, evolutionary algorithms, or motif search in time series can be deployed. The learning algorithm extracts the group behaviour from the interaction histories that are locally stored with each agent.

3.3 Monitoring the validity of modules

Owing to changes in the overall system, meta-agents might exhibit invalid behaviours at some point. Therefore, we check the validity of each meta-agent periodically by comparing its deployed behaviour with its expected behaviour. The correlation coefficients of the underlying agents serve as a heuristic indicator as they triggered the formation of the meta-agent (ρ'_{st}). According to Algorithm 2, we compare the current correlation coefficients of the meta-agent to previous values for each individual agent: if the difference is larger than some threshold, we consider the meta-agent invalid. As a consequence, we break down its hierarchy and set its underlying agents free.

3.4 Results

To validate the performance of our approach, we conducted a series of experiments on both MAPK models. The experiments are determined by the following five parameters (Table 1): we let the system run for some time t_{wait} and then start looking for meta-agents within a given time interval Δ_{find}. The waiting time t_{wait} is important as the system has to reach a rather stable condition before the abstraction algorithm starts to work. We keep monitoring the system in predefined intervals, $\Delta_{monitor}$. In order to integrate agents and to form meta-agents, the correlation coefficient between two agents, or the value of an edge in the interaction graph, should be greater than some threshold τ_{edge}. Finally, a meta-agent is valid as long as its correlation coefficients with its neighbours do not exceed the original correlation coefficients by a threshold τ_{valid}. Working values for τ_{valid} and τ_{edge} have been found through trial and error (Table 1).

3.4.1 ANN learning. First, we present an experiment that utilizes feed-forward ANNs with the back-propagation learning algorithm[47] to train meta-agents. The structure of

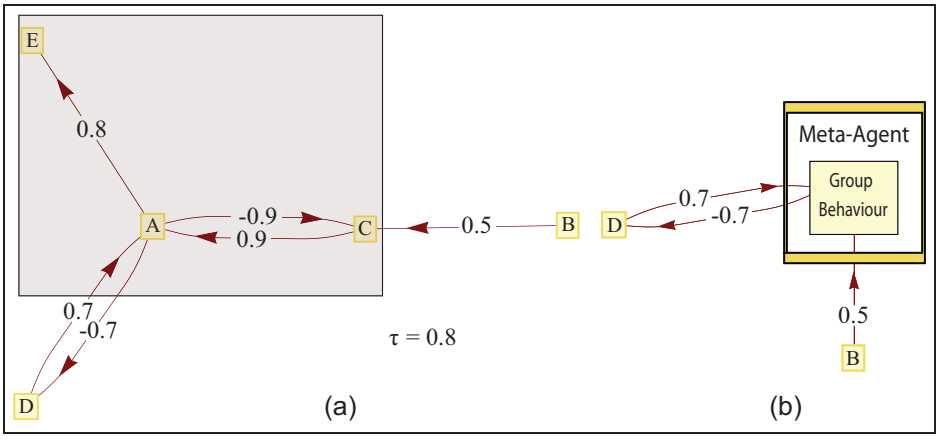

Figure 3. Example of an interaction graph. The edges denote the correlation coefficients. (a) *Agent A, Agent C*, and *Agent E* form a meta-agent. (b) The new neighbours of this meta-agent are *Agent B* and *Agent D*.

Table 1 . Model parameters.

Parameter name	Symbol	Value in the first MAPK model	Value in the second MAPK model
Delay before finding meta-agents	t_{wait}	1200	1500
Meta-agent finding interval	Δ_{find}	300	300
Monitoring interval	$\Delta_{monitor}$	20	20
Validity threshold	τ_{valid}	0.1	0.1
Edge threshold	τ_{edge}	0.95	0.7

an ANN is determined by its inputs and outputs as well as the number of nodes in the hidden layer. Since agents in our model are not aware of their dependent agents (they only know about their outgoing edges in the interaction graph), the output of the network should simply be all of the underlying agents. In the example shown in Figure 3, outputs are *Agents A, C* and *E*. The input nodes of the network are composed of all of the internal and their externally connected nodes (*Agents A, C, D* and *E* in Figure 3). As for the number of nodes in the hidden layer, we follow a simple rule-of-thumb and set it to the number of $inputs + 2$.

Figure 4(a) shows the result of applying our approach to the first MAPK model in terms of the number of agents. Initially, there are eight model agents in the system. We use the term ''model agent'' to emphasize their role in the original model, as opposed to meta-agents that are introduced as part of the abstraction process. The identification of meta-agents starts at $t_{wait} = 1200$. The resulting pattern of periodic creation and destruction of meta-agents (Figure 4(a)) stems from the fact that a meta-agent's probability to become invalid increases with its number of subsumed model agents. In general, a meta-agent becomes invalid even if one of its subsumed agents becomes invalid. Therefore, after the system is reduced to a single meta-agent, it breaks and releases all of the eight model agents.

Figure 4(b) shows that the concentration computed by the agent-based pathway model successfully resembles that of the partial differential equation (PDE) solver. Figure 4(c) shows the result of the same algorithm for the second MAPK pathway. Since this model is periodic, the algorithm successively finds, trains and breaks meta-agents over time. The great number of spikes in Figure 4(c) implies that the meta-agents are only valid for a short period of time.

3.4.2 GP learning and dynamic hierarchies. In a second experiment, we utilize GP to find the function that approximates the group behaviour subsumed by a meta-agent. We include four mathematical operations (+, -, *, /) in the function set of the GP algorithm, whereas the internal nodes of the interaction graph serve as the available terminals. Using a heuristic learning algorithm such as GP enables us to control the speed of learning and to perform a distributed search for good solutions.

The qualitative difference of this second approach compared with the presented ANN approach is the introduction of dynamic agent hierarchies. Previously, the destruction of a meta-agent set free all of the associated model agents. Now, meta-agents store references to their underlying model agents only in the first instance of the learning process. Meta-agents that subsume lower-level meta-agents store those instead, which results in a hierarchy of meta-agents with the original model agents as its leaves. When a meta-agent becomes invalid and is destroyed, its

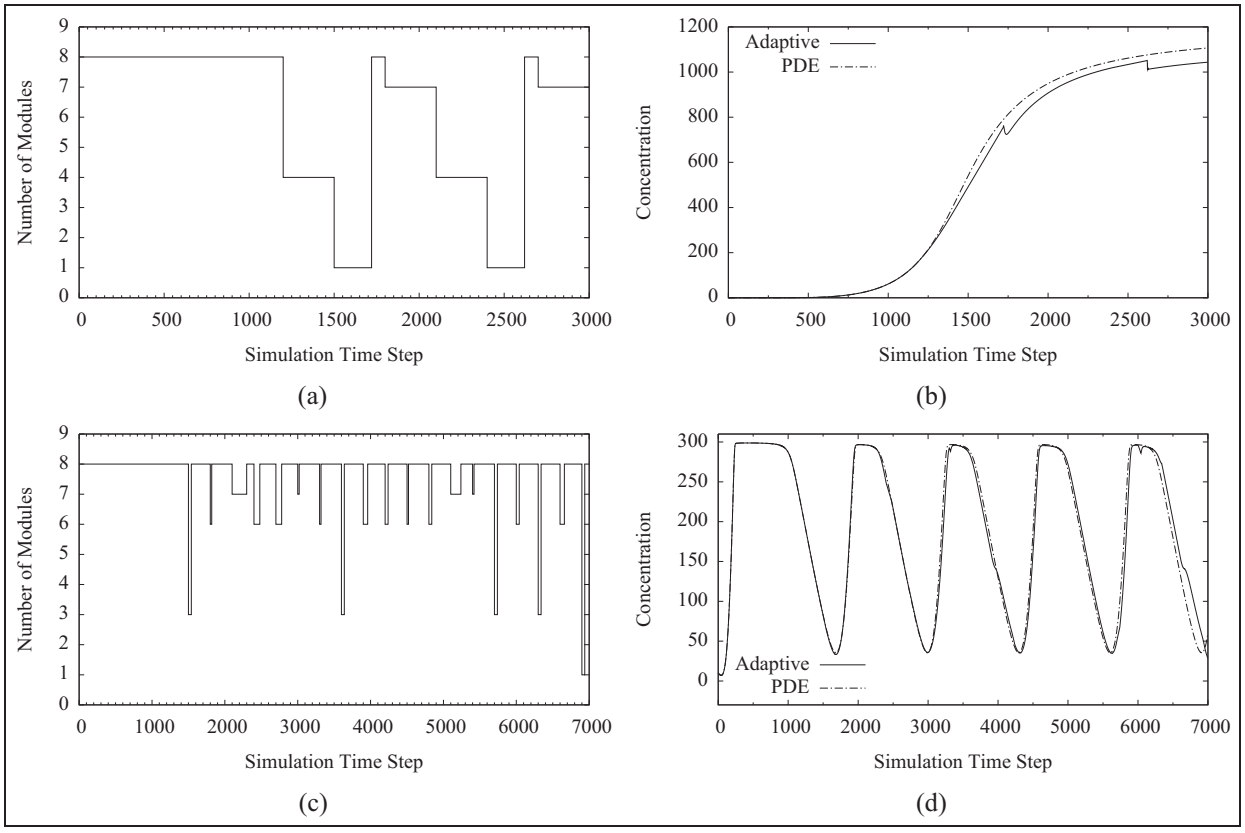

Figure 4. Adaptive modularization results for the MAPK pathway models of Figure 1: (a), (c) number of agents; (b), (d) concentration of MAPK-PP.

underlying agents, whether meta-agents or model agents, are released back into the simulation (Figure 5).

Figure 6(a) compares the performance of the hierarchical and the previously presented non-hierarchical approach. After $t_{wait} = 1200$, the non-hierarchical approach reduces the number of agents faster, but it cannot maintain any of the abstractions once the meta-agent becomes invalid at around $t = 1700$ and $t = 2500$. When the hierarchical meta-agent becomes invalid, its underlying hierarchy is restored: a single meta-agent breaks down at $t = 2200$ and releases 4 meta-agents back into the system (compared with 8 model agents). In both experiments, a meta-agent subsuming the behaviour of a larger number of agents becomes invalid very fast. This explains why an all-encompassing meta-agent does not stay long in the system ($2100 < t < 2200$ in Figure 6(a)). Figure 6(b) shows the MAPK concentration over time produced by the hierarchical approach and compared to the results of the PDE solver.

As Figure 6(c) shows, both experiments performed similarly on the second MAPK pathway. The number of spikes in both approaches suggests that neither learning method makes a significant difference in case of periodicity. We reason that the short period of validity in both

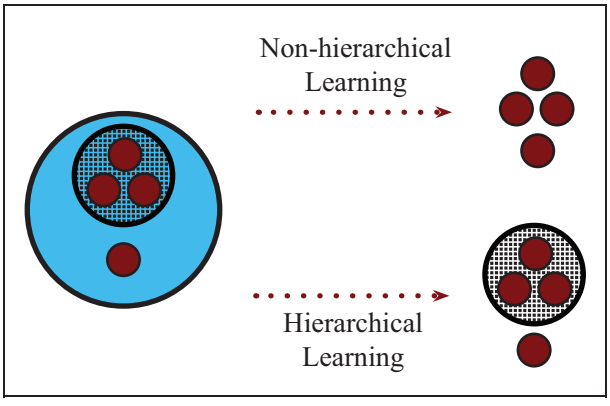

Figure 5. Difference between the non-hierarchical and the hierarchical approach to agent abstraction: when a non-hierarchical agent is destroyed, all of the associated model agents are released back into the simulation (shown at the top). In the other case (at the bottom), the hierarchical configuration stored with a meta-agent is restored resulting in one meta-agent and one model agent.

presented approaches is the result of using the correlation coefficient to measure how closely two agents work together. Since the correlation coefficient varies from -1

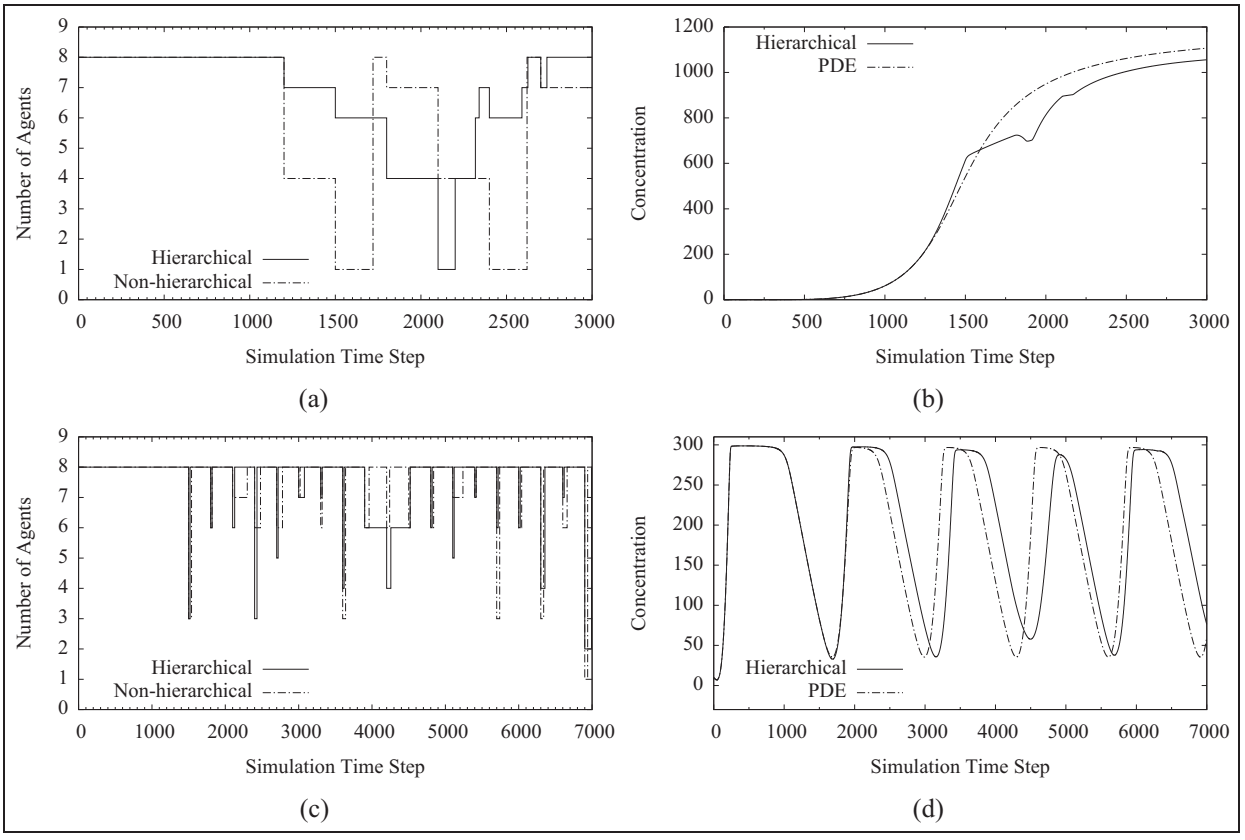

Figure 6. Results for the MAPK pathway model of Figure 1: (a), (c) number of agents (solid line: our hierarchical approach; dashed line: non-hierarchical approach proposed by Sarraf Shirazi[31]); (b), (d) concentration of MAPK-PP.

to $+1$ over a periodic signal, it fails to capture the similarity of two agents in a periodic system. This result suggests that we have to look for other indicators when dealing with a periodic system.

4. Self-organized middle-out learning and abstraction

In the third approach, we introduce *observer agents*, or *observers*, that coexist alongside of the model agents in the simulation space (Figure 7). The simulation framework treats both kinds of agents equally, i.e. each of these agent types is considered for interactions at each simulation step. Instead of an external algorithm (Section 3.4.2), the *observer agents* now handle the creation and management of abstraction hierarchies based on the interaction processes performed by model agents.

Once an *observer* successfully identifies an interaction pattern, it acts as a meta-agent that replaces the individual behaviours previously maintained by the model agents that led to the identified pattern. Acting as a meta-agent, the *observer* itself becomes subjected to observation. In order to verify their validity, *observers* would check whether the deployment of the subsumed individual behaviours would yield an outcome different from the predictions of the learned pattern. If the discrepancy between these two outcomes exceeds a given threshold, the *observer* omits its learned pattern and restores the subsumed individual behaviours.

The success of the abstraction system depends on the configuration of the deployed *observer agents*. In the following paragraphs, we explain one way how they can replace the individual behaviours with a group behaviour and how the *observer agents* can validate, maintain, or abandon the learned patterns throughout the course of a simulation.

4.1 Observer configuration

Like any other agent in a multi-agent system, an *observer* can be defined as $ag = (Sit, Act, Dat, f_{ag})$, a 4-tuple composed of a set Sit of situations, a set Act of actions, a set Dat of internal data, and a decision function f_{ag}[48]. At any point in time, the agent decides to perform an action based on its situation and internal data. This decision is captured in a decision function $f_{ag} : Sit \times Dat \rightarrow Act$. In rule-based agent architectures, Dat can be re-written as Intvar $\times$ RS, where Intvar is a set of values for internal variables and RS is a set of interaction rules:

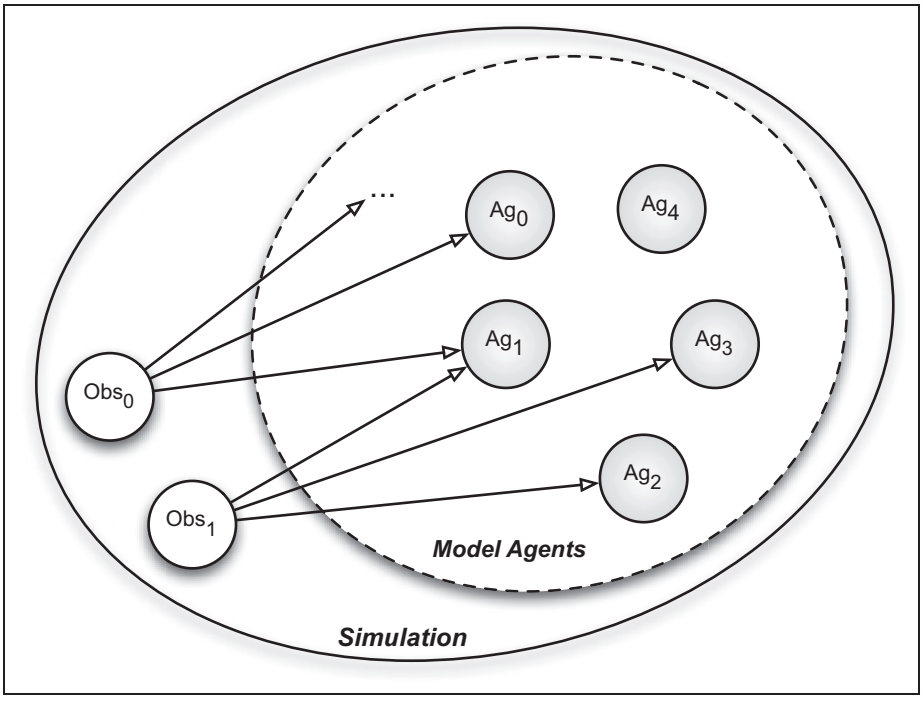

Figure 7. *Observers* Obs_0 and Obs_1 inside the simulation space monitor a subset of agents and log necessary information based on their configuration.

if condition **then** execute *act*,

where $act \in Act$, and condition is a statement about the situation the agent is in and the actual values of the variables in Intvar. Both condition and *act* might involve other agents called interaction partners.

Observers are configured to log the interactions of model agents in their interaction histories: IH_{Exec} is used to log executed interactions, whereas IH_{NExec} logs the numbers of considered but not executed actions (Tables 2 and 3). An IH_{Exec} entry may contain any information related to an observed interaction. For instance, an *observer* may store that the model agent A executed an action $act \in Act$ with time stamp t along with the set of interaction partners $\mathcal{A}$.

An *observer* extracts group behaviours from the logged data by applying a pattern recognition algorithm. In this section, we present results based on clustering, which will be explained next.

4.2 Learning and abstraction

In our prototype, an *observer* logs interaction partners in combination with the time of the interaction. Once the interaction history IH_{Exec} has grown beyond a certain threshold, the *observer* applies a *k*-means clustering algorithm[49] to find a large cluster C of overlapping interaction partners. The similarity between two interactions is calculated based on the number of overlapping interaction partners. When the *observer* finds such a cluster, it infers a generalized group behaviour from the clustered individual interactions by combining their features. The first feature is the set of overlapping interaction partners that are constant for the learned action. Second, the *observer* needs to know when and at which rate it should execute the learned action.

The *observer* first finds the time range $[t_{min}, t_{max}]$ of the executed action from all of the individual interactions in C. Two cases might happen here: (1) an interaction only occurs within a bound time range; (2) an interaction continuously occurs over time or the *observer* is uncertain whether it has had enough time to determine an upper bound t_{max} of the time range. In order to address the latter case, the *observer* compares the two most recent time stamps an interaction occurs in IH_{Exec}. If the difference exceeds the observation time, the *observer* sets t_{max} to ∞.

Next, the *observer* extracts the rate of execution defined as the number of executed interactions divided by the number of total computations of the interaction:

$$p_{exec} = \frac{|IH_{Exec}|}{|IH_{Exec}| + |ihn|}, \quad ihn \in IH_{NExec} \text{ and } ihn.t \in [t_{min}, t_{max}] \tag{5}$$

where IH_{Exec} is the set of executed interactions, ihn is the set of considered but not executed interactions whose timestamp is in $[t_{min}, t_{max}]$, and $|\cdot|$ denotes the size of a set.

For example, all of the IH_{Exec} records in Tables 2 and 3 constitute a cluster in which $[t_{min}, t_{max}]$ is inferred from the first column. The second column (*ags*) is discarded and regarded as wildcard, the set of interaction partners is fixed to $\mathcal{A}_2$, and p_{exec} is calculated as described above.

Finally, the *observer* removes the action *act* from the model agent. From now on, it performs the action on the model agent's behalf. For instance, an *observer* may learn that action *act* of an agent A occurs at $A.t \in [t_{min}, t_{max}]$, and executes it on A's behalf with an according probability p_{exec}. Since the *observer* also learns the interaction partners an action depends on, the computational resources to identify those are saved as well.

4.3 Validation of the learned behaviours

After some time, a learned behaviour may no longer be valid. In order to monitor the reliability of a learned behaviour, it is initialized with an unbiased confidence value $conf_{initial} = 50\%$. At regular time intervals, the *observer* lets some model agents execute their original interactions. The confidence value is regulated based on the difference between the actual behaviour of model agents compared with the behaviour expected by the *observer* (Figure 8). In our prototype, we only consider the difference in interaction partners. The time at which an individual interaction occurs or the rate at which model agents execute their interactions could also be incorporated into the comparison. A confidence measure below a given threshold indicates that a learned group behaviour is not valid any longer and that the *observer* has to restore the model agents' original behaviours.

Table 2. Interaction history of executed actions (IH_{Exec}) inside an *observer*.

Time	Agent	Action	Interaction partners
t_0	ag_0	Activate	$\mathcal{A}_2$
t_1	ag_0	Activate	$\mathcal{A}_2$
⋮	⋮	⋮	⋮
t_1	ag_2	Activate	$\mathcal{A}_2$
t_2	ag_7	Activate	$\mathcal{A}_2$

Table 3. Interaction history of computed but unexecuted actions (IH_{Nxec}) inside an *observer*.

Time	Action	Count
t_0	Activate	n_0
t_3	Activate	n_3
⋮	⋮	⋮
t_{15}	Activate	n_{15}
t_{23}	Activate	n_{23}
t_{32}	Activate	n_{32}

4.4 Experiments

The outlined self-organized optimization method can be employed in arbitrary agent simulations. Biological simulations are particularly suitable applications as biological entities will be directly modelled as agents. When simulating biological systems at the level of inter-cellular and inter-molecular interactions, actions are mostly triggered by collisions or internal agent states. We applied our proposed method to an agent-based simulation of blood coagulation described in this section. All of the experiments were repeated 10 times to ensure that a particular experiment did not bias the results.

4.4.1 Model Setup. Blood coagulates at wound sites because of the interplay of various bio-agents, e.g. platelets, fibrinogens and serotonins. If a collagen protein collides with a platelet, the platelet becomes activated. In case an activated platelet collides with the wound site, it secretes several chemicals which in turn activate more platelets in the blood vessel. Gradually, a network of fibrins together with a platelet plug form a clot around the wound site (Figure 9). We modelled 12 blood factors as agents whose behaviours are expressed as a set of interaction rules. There are 10 different interactions which fall into two categories: (1) state-dependent interactions and (2) collision-dependent interactions. The actions themselves introduce local state changes of the agents (represented as internal variables), or they produce or remove agents in the simulation. The simulation starts with 10 agents and ends up with nearly 140 interacting agents (Figure 10).

4.4.2 Observer setup. Each interaction is monitored by an *observer* that records only the interaction partners. Table 4 lists all the important parameters in our system. Once an *observer* monitors an interaction long enough (t_{wait}), it applies a k-means clustering algorithm to create k clusters. The centroid of the largest cluster is considered to be the learned group behaviour for which $[t_{min}, t_{max}]$ and p_{exec} are inferred. The *observer* subsumes the learned interaction by executing it on behalf of the model agents. In predefined intervals, $V_{interval}$, the *observer* randomly chooses a subset of the subsumed behaviours and allows the model agents to execute their original interactions. The size of this subset is determined by V_{ratio}. After some time, V_{length}, the *observer* subsumes this subset again and validates its abstractions based on the resulting interaction compared to the expected result. The confidence of the learned pattern is regulated accordingly. If the confidence of a pattern is less than some threshold τ_{conf}, the learned pattern will be removed from the simulation.

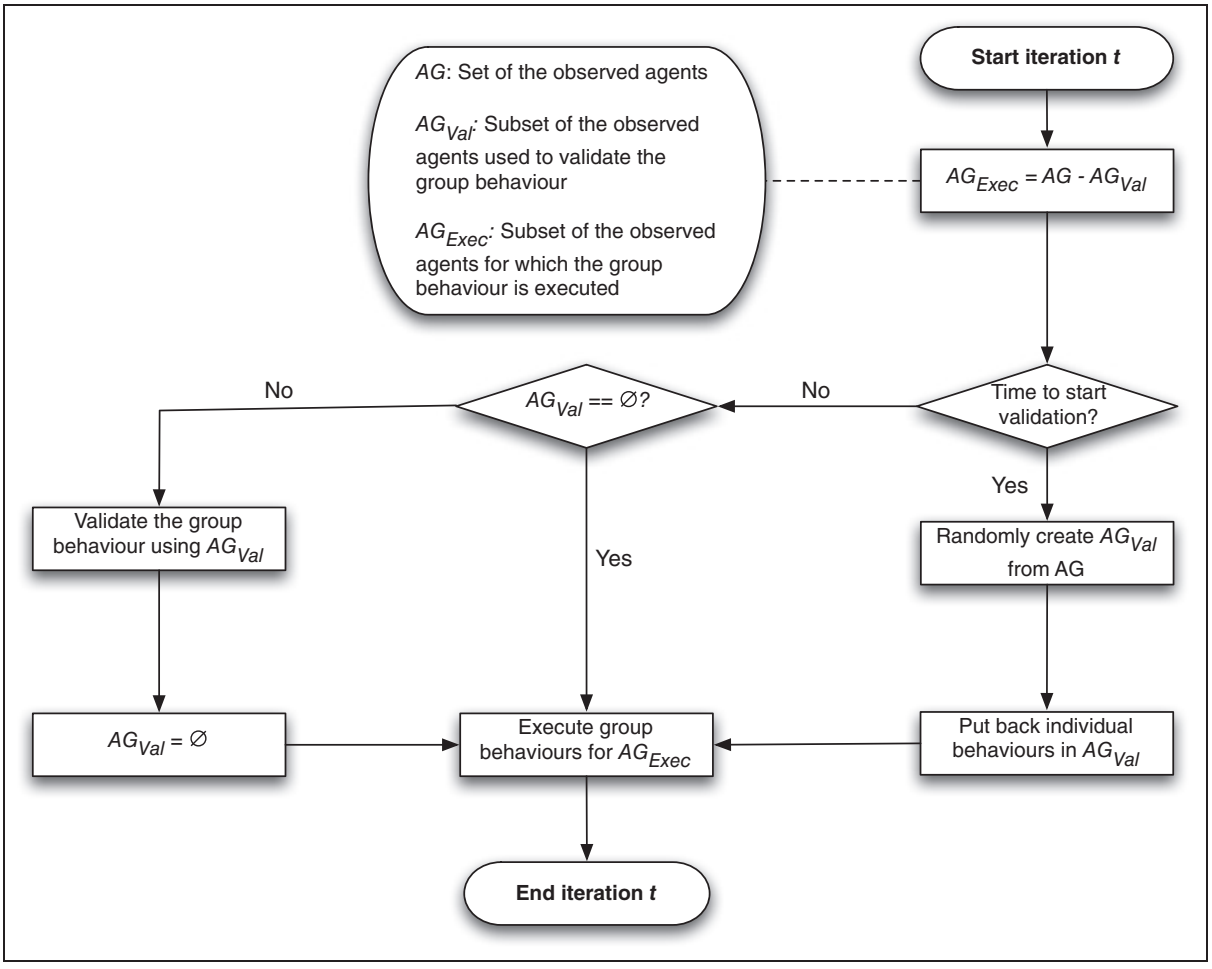

Figure 8. Flow chart of the validation step. At some interval, the *observer* selects a random subset of the observed agents and restores their individual behaviours. The result of their interactions is evaluated in the next iteration to regulate the confidence value. The *observer* continues to execute the group behaviour for all other agents.

4.4.3 Results. The presented prototype implementation successfully identified several group behaviours within the simulation. For example, ''Random walk'' is a self-triggering action found to be executed with probability $p_{exec} = 100\%$ and $t \in [0, \infty)$. ''Adhere'' is an interaction executed in $t \in [172, \infty)$ with probability $p_{exec} = 65\%$. ''Self-activation'' is another collision-based example with $p_{exec} = 2.8\%$ and $t \in [173, 190]$.

Figure 11(a) shows the actual run-time of the simulation at each time step. When there is no *observer*, the simulation slows down as it proceeds, as simulating the interactions among the increasing number of agents requires more computations. When the *observers* are present in the simulation logging interaction data ($0 < t < 350$), they add a little overhead to the run-time of the whole simulation. At $t = 350$ when the learning happens, there is a peak in the run-time. However, once successfully deployed, the *observers* reduce the run-time drastically by executing group behaviours instead of individual behaviours. The validation cycle is triggered every $V_{interval} = 70$ time steps, therefore there is a fairly high peak at this interval. It continues for $V_{length} = 10$ time steps before the learned pattern is evaluated. After this time, the simulation runs fast again until the next validation cycle.

Figure 11(b) depicts the cumulative run-time of the simulation comparing a normal run against a run with *observers* . The overhead of having *observers* clearly pays off at $t > 390$, when the cumulative run-time of a normal run exceeds that of a run with *observers* . On average, a normal run takes 107 seconds to complete 1000 simulation time steps, almost twice as long as the run with the *observers*, which takes 56 seconds.

Figure 12 shows the change of confidence for one of the learned patterns. Since there is no learned pattern before $t = 350$, the confidence value is also 0. However, after the *observer* abstracts an individual behaviour, the confidence value is initially set to 0.5. As all the abstractions work correctly, the confidence values continuously increase over time.

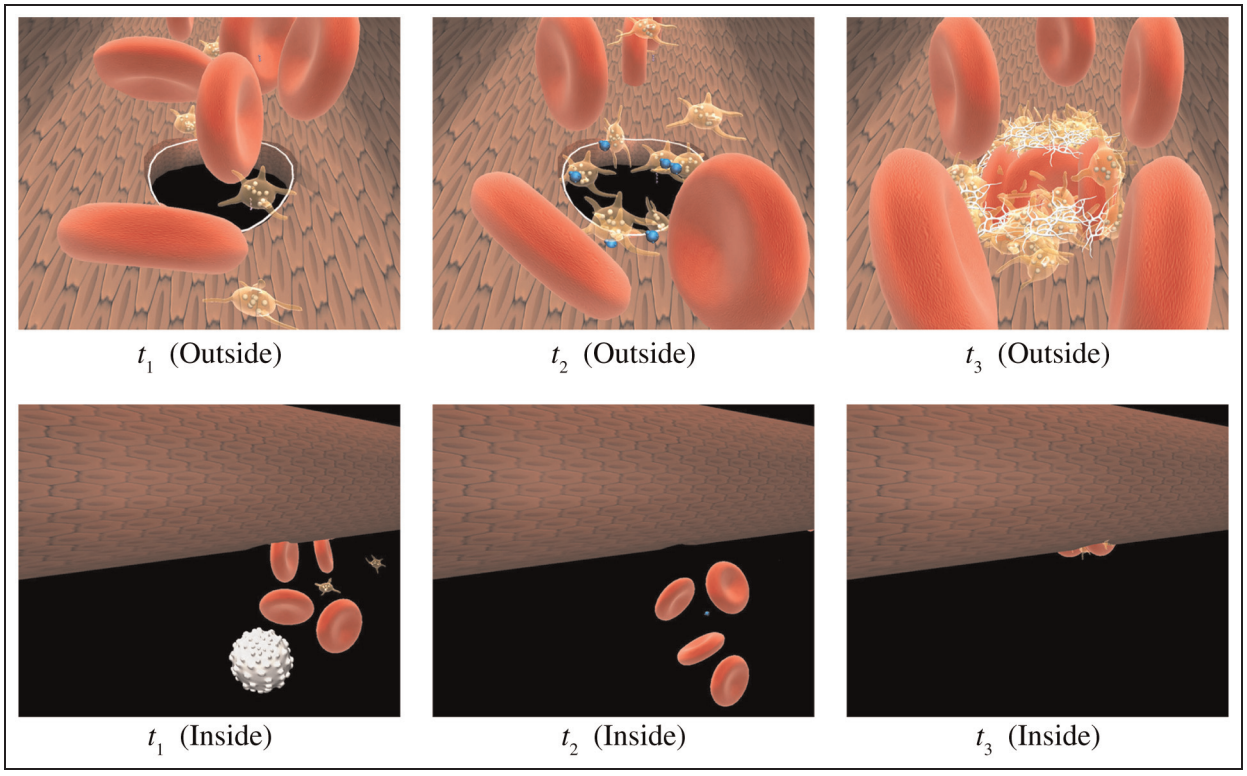

Figure 9. The blood coagulation simulation at different time steps ($t_1 < t_2 < t_3$). The process is observed from two different perspectives: inside and outside of the vessel.

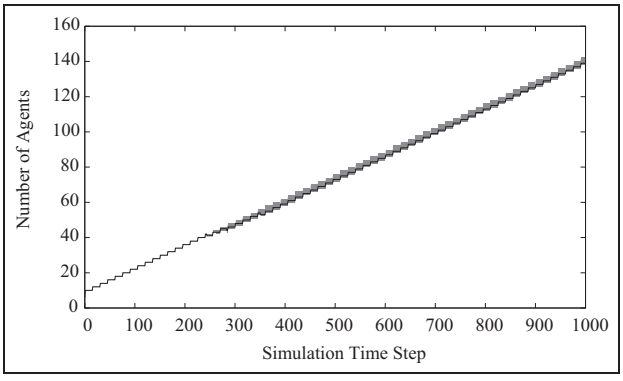

Figure 10. Blood coagulation simulation: number of agents over time.

Table 4. System parameters.

Parameter name	Symbol	Value
Delay before learning	t_{wait}	350
Validation interval	$V_{interval}$	70
Validation length	V_{length}	10
Validation ratio	V_{ratio}	30%
Confidence threshold	τ_{conf}	0.3
Number of clusters in *k*-means	k	10

5. Conclusion and future work

We introduced a concept for the reduction of computational complexity in agent-based models by means of learning behavioural patterns over the course of a simulation. The abstractions would be expressed as meta-agents that subsume lower-level agents and be seamlessly integrated into the agent models.

We presented and evaluated three implementations. (1) The first utilized ANNs to learn collective processes in the flux of concentrations of the MAPK signalling pathway. Here, the learned abstractions were constantly updated to consider a growing number of agents. As a result, the abstractions lost their validity at some point, they were removed from the simulation and relearned. (2) In the second implementation, which relied on GP for learning collective behaviours, the abstractions were not completely revoked when becoming invalid, but they were restored to their previous states. (3) In the third implementation, *observer* agents detected group behaviours and managed the resulting abstractions. We demonstrated the effectiveness of this implementation in the context of a blood coagulation model. We proposed two algorithms to monitor the validity of abstractions by comparing the expected group interactions to the interactions of the actual individuals at regular intervals.

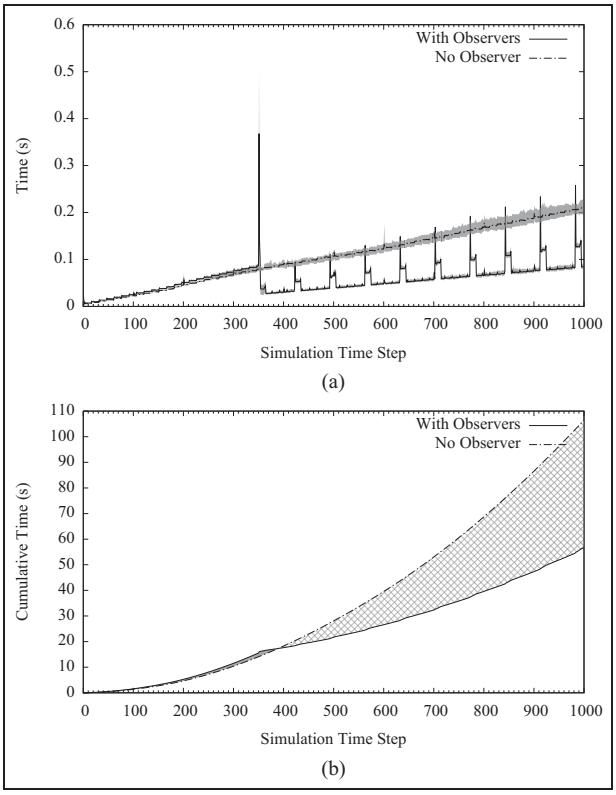

Figure 11. Blood coagulation simulation: run-time with and without *observers* : (a) run-time per simulation time step; (b) cumulative run-time.

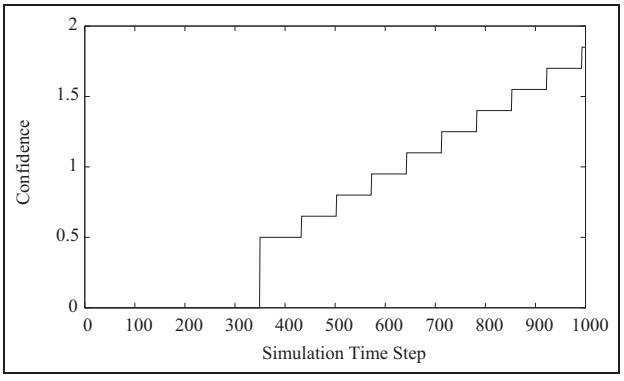

Figure 12. The confidence value over time shown for an exemplarily learned pattern.

In order to further our approach, we suggest the automatic proliferation of a diverse set of *observer agents* based on their workload. An evolution of agents that are primed to identify frequently occurring patterns could be implemented, yielding a self-organized learning system that adapts to specific model domains and even to niches inside of simulation spaces.

The relation between group behaviours and emergent phenomena is another promising area to be investigated in the given context. The possibility to incorporate predefined high-level patterns should be considered. If patterns are described at different scales, multi-scale modelling can be restated as finding transitions from low-level to higher-level patterns.

Acknowledgment

This work has been partially supported by the Natural Sciences and Engineering Research Council of Canada.

References

1. Dessalles JL and Phan D. Emergence in multi-agent systems: cognitive hierarchy, detection, and complexity reduction part I: methodological issues. *Artificial Econ* 2006; 564: 147–159.
2. Jacob C and Burleigh I. Biomolecular swarms: an agent-based model of the lactose operon. *Natural Comput* 2004; 3: 361–376.
3. Chen CC, Nagl SB and Clack CD. Identifying multi-level emergent behaviors in agent-directed simulations using complex event type specifications. *Simulation* 2010; 86: 41–51.
4. Corning PA. The re-emergence of ''emergence'': A venerable concept in search of a theory. *Complexity* 2002; 7(6): 18–30.
5. Bonabeau E. Agent-based modeling: methods and techniques for simulating human systems. *Proc Natl Acad Sci USA* 2002; 99(Suppl. 3): 7280–7287.
6. Shirazi AS, von Mammen S, Yazdanbod I and Jacob C. Self-organized learning of collective behaviours in agent-based simulations. In: Sayama H, Minai AA, Braha D and Bar-Yam Y (eds), *ICCS 2011 - 8th International Conference on Complex Systems*. Boston, MA: NECSI Knowledge Press, 2011.
7. Zhang L, Wang Z, Sagotsky JA and Deisboeck TS. Multiscale agent-based cancer modeling. *J Math Biol* 2009; 58: 545–559.
8. Desmeulles G, Querrec G, Redou P, Kerdelo S, Misery L, Rodin V, et al. The virtual reality applied to biology understanding: The in virtuo experimentation. *Expert Syst Applications* 2006; 30: 82–92.
9. Bandini S, Manzoni S and Vizzari G. Multi-agent modeling of the immune system: The situated cellular agents approach. *Multiagent Grid Syst* 2007; 3: 173–182.
10. Vicsek T and Zafiris A. Collective motion. *Phys Rep* 2012; 517: 71–140.
11. Salazar-Ciudad I. Tooth morphogenesis in vivo, in vitro, and in silico. *Curr Top Dev Biol* 2008; 81: 342.
12. Zucker JD. A grounded theory of abstraction in artificial intelligence. *Phil Trans R Soc Lond Ser B Biol Sci* 2003; 358: 1293–1309.
13. Goldstone RL and Barsalou LW. Reuniting perception and conception. *Cognition* 1998; 65: 231–262.
14. Banzhaf W. Artificial chemistries - towards constructive dynamical systems. *Solid State Phenomena* 2004; 97–98: 43–50.

15. Prusinkiewicz P and Lindenmayer A. *The Algorithmic Beauty of Plants*. Berlin: Springer-Verlag, 1996.
16. Kniemeyer O, Barczik G, Hemmerling R and Kurth W. Relational growth grammars—a parallel graph transformation approach with applications in biology and architecture. In: *Applications of Graph Transformations with Industrial Relevance: Third International Symposium, AGTIVE 2007*, Kassel, Germany, 10–12 October 2007, Revised Selected and Invited Papers. Berlin: Springer-Verlag, 2008; pp. 152–167.
17. von Mammen S and Jacob C. The evolution of swarm grammars: growing trees, crafting art and bottom-up design. *IEEE Computat Intell Mag* 2009; 4(3): 10–19.
18. Kauffman S. *At Home in the Universe: The Search for the Laws of Self-Organization and Complexity*. Oxford: Oxford University Press, 1995.
19. Eigen M and Schuster P. The hypercycle. *Naturwissenschaften* 1978; 65: 7–41.
20. Schuster P. How does complexity arise in evolution. *Complex* 1996; 2: 22–30.
21. Rasmussen S, Baas NA, Mayer B, Nilsson M and Olesen MW. Ansatz for dynamical hierarchies. *Artificial Life* 2002; 7: 329–353.
22. Dorin A and McCormack J. Self-assembling dynamical hierarchies. *Artificial Life* 2003; 8: 423.
23. Bosse T, Hoogendoorn M, Klein MCA and Treur J. A three-dimensional abstraction framework to compare multi-agent system models. In: *Proceedings of the Second International Conference on Computational Collective Intelligence: Technologies and Applications*. Heidelberg: Springer-Verlag, 2010, pp. 306–319.
24. Ralambondrainy T, Courdier R and Payet D. An ontology for observation of multiagent based simulation. In: *IAT Workshops*. Los Alamitos, CA: IEEE Computer Society, 2006, pp. 351–354. Available at: http://dblp.uni-trier.de/db/conf/iat/iatw2006.html#RalambondrainyCP06.
25. Servat D, Perrier E, Treuil JP and Drogoul A. When agents emerge from agents: introducing multi-scale viewpoints in multi-agent simulations. In: *Proceedings of the First International Workshop on Multi-Agent Systems and Agent-Based Simulation*. London: Springer-Verlag, 1998, pp. 183–198.
26. Chen CC, Nagl SB and Clack CD. Multi-level behaviours in agent-based simulation: colonic crypt cell populations. In: *Proceedings of the Seventh International Conference on Complex Systems*, 2008; available at: http://necsi.org/events/iccs7/papers/74bfc021da489126748d7ed5185e.pdf.
27. Denzinger J and Hamdan J. Improving observation-based modeling of other agents using tentative stereotyping and compactification through kd-tree structuring. *Web Intell Agent Syst* 2006; 4: 255–270.
28. Lavelle C, Berry H, Beslon G, Ginelli F, Giavitto JLJL, Kapoula Z, et al. From Molecules to organisms: towards multiscale integrated models of biological systems. *Theor Biol Insights* 2008; 1: 13–22.
29. Cardon A. *Design and Behavior of a Massive Organization of Agents* (*Design of Intelligent Multi-Agent Systems*, vol. 162). Berlin: Springer, 2004, pp. 133–190.
30. von Mammen S, Steghöfer JP, Denzinger J, Jacob C and Stehöfer JP. Self-organized middle-out abstraction. In: Bettstetter C and Gershenson C (eds), *IWSOS 2011: 5th International Workshop on Self-Organizing Systems*. Berlin: Springer, 2011, pp. 26–31.
31. Sarraf Shirazi A, von Mammen S, Jacob C. Adaptive modularization of the MAPK signaling pathway using the multiagent paradigm. In: *PPSN'10: Proceedings of the 11th International Conference on Parallel Problem Solving from Nature: Part II* (*Lecture Notes in Computer Science*, vol. 6239). Berlin: Springer, 2010, pp. 401–410.
32. Sarraf Shirazi A, von Mammen S, Jacob C. Hierarchical self-organized learning in agent-based modeling of the MAPK signaling pathway. In: *2011 IEEE Congress on Evolutionary Computation (CEC)*, 2011, pp. 2245–2251.
33. Li J and Kwauk M. Exploring complex systems in chemical engineering - the multi-scale methodology. *Chem Eng Sci* 2003; 58: 521–535.
34. Eissing T, Kuepfer L, Becker C, Block M, Coboeken K, Gaub T, et al. A computational systems biology software platform for multiscale modeling and simulation: integrating whole-body physiology, disease biology, and molecular reaction networks. *Front Physiol* 2011; 2: 1–10.
35. Martins ML, Ferreira SC Jr and Vilela MJ. Multiscale models for biological systems. *Curr Opin Colloid Interface Sci* 2010; 15: 18–23.
36. Erson EZ and Cavusoglu MC. A software framework for multiscale and multilevel physiological model integration and simulation. In: *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (2008 EMBS)*, 2008, pp. 5449–5453.
37. Merks RMH and Glazier JA. A cell-centered approach to developmental biology. *Physica A* 2005; 352: 113–130.
38. Haken H. *Synergetics: An Introduction. Nonequilibrium Phase Transitions and Self-organization in Physics, Chemistry and Biology*. Berlin: Springer-Verlag, 1977.
39. Walker DC and Southgate J. The virtual cell - a candidate co-ordinator for 'middle-out' modelling of biological systems. *Brief Bioinform* 2009; 10: 450–461.
40. Bassingthwaighte JB, Chizeck HJ and Atlas LE. Strategies and tactics in multiscale modeling of cell-to-organ systems. *Proc IEEE* 2006; 94: 819–831.
41. Amigoni F and Schiaffonati V. Multiagent-based simulation in biology: a critical analysis. *Model-Based Reasoning Sci Technol Med* 2007; 64: 179–191.
42. Huang CY and Ferrell JE. Ultrasensitivity in the mitogen-activated protein kinase cascade. *Proc Natl Acad Sci USA* 1996; 93: 10078–10083.
43. Kholodenko BN. Negative feedback and ultrasensitivity can bring about oscillations in the mitogen-activated protein kinase cascades. *Eur J Biochem* 2000; 267: 1583–1588.
44. Nii HP. *Blackboard Systems*. Techincal Report, Department of Computer Science, Stanford University, 1986.
45. González PP, Cárdenas M, Camacho D, Franyuti A, Rosas O and Lagúnez-Otero J. Cellulat: an agent-based intracellular signalling model. *Biosystems* 2003; 68: 171–185.
46. Khan S, Makkena R, McGeary F, Decker K, Gillis W and Schmidt C. A multi-agent system for the quantitative

simulation of biological networks. In: *AAMAS '03: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*. New York: ACM Press, 2003, pp. 385–392.
47. Haykin S. *Neural Networks: A Comprehensive Foundation*. New York: Macmillan, 1994.
48. Denzinger J and Kordt M. Evolutionary on-line learning of cooperative behavior with situation–action pairs. In: *Proceedings of the Fourth International Conference on MultiAgent Systems (ICMAS-2000)*. Washington, DC: IEEE Computer Society, 2000, pp. 103–110.
49. MacQueen JB. Some methods for classification and analysis of multivariate observations. In: Cam LML and Neyman J (eds), *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1. University of California Press, 1967, pp. 281–297.

Author biographies

Abbas Sarraf Shirazi received his MSc in Artificial Intelligence from Amirkabir University of Technology, Iran, where he studied evolutionary algorithms, statistical pattern recognition techniques, and artificial neural networks to detect patterns in noisy environments. In September 2009, he started his PhD studies in the Evolutionary and Swarm Design research group in the Department of Computer Science at the University of Calgary. His research includes learning and abstraction in complex agent-based simulations. As a researcher in the LINDSAY - Virtual Human, his abstraction mechanisms both speed up the simulation time and also provide a big picture understanding of biological simulations.

Sebastian von Mammen graduated from the University of Erlangen-Nuremberg, Germany, with a Diploma Degree in Computer Science (Diplom Informatik Univ.) and a Minor in Education in Fine Arts in 2005. From 2006 to 2009, he pursued his PhD studies at the University of Calgary, Canada, developing and formalizing Swarm Grammars, a swarm-inspired, multi-agent representation for modelling complex, self-organizing, spatial interaction networks and developmental processes. Until 2012, as a postdoctoral fellow and the project manager of the LINDSAY - Virtual Human, his research focused on application-oriented modelling and simulation of self-organized systems. He continues to pursue these interests as a consultant of the Lindsay Virtual Human project and as a research fellow at the University of Augsburg, Germany.

Christian Jacob received his PhD in Computer Science from the University of Erlangen-Nuremberg, Germany. In July 1999, he joined the Department of Computer Science (Faculty of Science) at the University of Calgary. Since August 2003, he also holds a joint appointment with the Department of Biochemistry and Molecular Biology (Faculty of Medicine), where he is the Director of Bioinformatics in the Bachelor of Health Sciences Program. He leads the Evolutionary and Swarm Design research group of the Artificial Intelligence Laboratory in the Department of Computer Science. He is also the lead investigator in the LINDSAY - Virtual Human project, a collaboration with Undergraduate Medical Education (UME) in the Faculty of Medicine. LINDSAY is an interactive, three-dimensional computer model of male and female anatomy and physiology (http://lindsayvirtualhuman.org).