Self-Organized Middle-Out Abstraction

Sebastian von Mammen[1], Jan-Philipp Steghöfer[2], Jörg Denzinger[1], and Christian Jacob[1,3]

[1] Department of Computer Science, University of Calgary, Canada
[2] Institute of Software & Systems Engineering, Augsburg University, Germany
[3] Department of Biochemistry and Molecular Biology, University of Calgary, Canada
s.vonmammen@ucalgary.ca, steghoefer@informatik.uni-augsburg.de,
denzinge@cpsc.ucalgary.ca, cjacob@ucalgary.ca

Abstract. In this position paper we present a concept to automatically simplify computational processes in large-scale self-organizing multi-agent simulations. The fundamental idea is that groups of agents that exhibit predictable interaction patterns are temporarily subsumed by higher order agents with behaviours of lower computational costs. In this manner, hierarchies of meta-agents automatically abstract large-scale systems involving agents with in-depth behavioural descriptions, rendering the process of upfront simplification obsolete that is usually necessary in numerical approaches. Abstraction hierarchies are broken down again as soon as they become invalid, so that the loss of valuable process information due to simplification is minimized. We describe the algorithm and the representation, we argue for its general applicability and potential power and we underline the challenges that will need to be overcome.

Keywords: Abstraction, multi-agent systems, middle-out modelling, simulation, motif detection, confidence estimation.

1 Introduction

Systems that comprise only few variables can exhibit complex behaviours [18] and due to the multi-facetted interaction networks in natural systems [17] their computation tends to be inefficient. Abstraction is the means to render computations feasible by capturing the essence of what is important on the one hand, and by ignoring those aspects that are seemingly unrelated on the other hand. Thus, the process of abstraction yields either *generalized* or *specialized* models which results in either rather general or very specific results, respectively [9].

However, in models that try to capture the complexity and the nonlinearity of natural systems seemingly unimportant variables can have a major impact under certain circumstances [15]. Hence, contrary to the need for computational efficiency, all available model data should be integrated into one simulation. The solution to this dilemma is a method that automatically adjusts the degree of abstraction of a computational model depending on its expected need of comprehensive calculations. In this position paper, we present the concept of a self-organizing middle-out (SOMO) abstraction system that realizes this idea by means of agents that dynamically establish and break down hierarchical relationships.

After a hint at the vast body of work of related ideas (Section 2), we describe the representation and the algorithm that drive our concept (Section 3). In Section 4, we outline the local agent behaviours for pattern-dependent construction and destruction of hierarchies. We summarize our proposed approach and point out open challenges that will need to be addressed in Section 5.

2 Related Work

At the heart of the presented work lies the idea of emergence, i.e. that a group of interacting units, or agents, produces effects that cannot be inferred from the properties and the behaviour of any of the individual agents itself [5]. There are two ways to approach emergent phenomena: either the investigator starts with a simple model of a system's components and their interactions and he wonders what global phenomena might emerge from them, or a global phenomenon has been observed and now the underlying factors need to be unearthed (*inverse problem*). These two cases are similar to bottom-up and top-down design processes: In a bottom-up approach, basic building blocks are combined to result in a higher order design complexity, whereas the top-down approach starts with the desired product of great complexity and follows its stepwise reduction into numerous parts of lesser complexity.

In the context of simulating a 'virtual heart', Sydney Brenner coined the term middle-out [10] as an approach for consistent computation of processes across several levels of detail. So, a middle-out approach works both ways: Locally interacting agents interact in accordance with a given model with the potential to produce emergent phenomena (bottom-up), and the expected phenomena determine the behaviours of the underlying units (top-down). Finding both these approaches in one system, however, is rarely seen. Instead, the idea of building structural and functional complexity from sets of interacting local agents has been attracting some attention. Peter Schuster, for instance, argues that if some agents' interactions nurture themselves, like chemical reactions in hypercycles, such mergers could drive the evolution of complexity [13]. Steen Rasmussen et al. designed a computational model in which, based on interactions in an artificial chemistry, structures form with an increase in structural complexity and with different functionalities, from monomers to polymers to micelles [12]. Alan Dorin and Jon McCormack argued, however, that such phenomena are not surprising given the model's simplicity. In fact, they argue that it takes considerably more effort to determine the novelties brought about by a novel layer in a hierarchy [2].

In general, synthesizing dynamical hierarchies at all scales is a difficult challenge, especially if real emergent, i.e. unforeseeable, features change the behaviours of higher order agents [7]. But even without considering the emergence of global properties or behaviours, the idea of bottom-up learning can prove useful. Abbas Shirazi et al., for instance, have shown that the effects of interacting agent groups can be learned by means of artificial neural networks and subsumed by according high-order agents to cut computational costs [14].

3 The SOMO Concept

We want to take Shirazi et al.'s approach several steps further. In particular, we envision a multi-agent simulation in which self-organizing individuals form and dissolve abstraction hierarchies (middle-out), incorporating and releasing other agents, based on observed interaction patterns. In general, groups of agents are subsumed by higher-order agents that emulate the groups' interactions. This computational simplification—not all possible interactions are computed—can become invalid at some point in time, as the learned interaction pattern might simply not apply any longer. In this case, the learned hierarchy has to be dissolved, i.e. the adopted agents are released again. Both adoption and release of agents happen recursively, yielding ever-changing, potentially asymmetrical hierarchical structures. Hence, the computational costs over the course of a simulation are only reduced, if the overhead of changing the hierarchy is less costly than the costs for the pruned agent interactions. Therefore, a reduction of computational costs can only be guaranteed, if the learned abstractions have a reasonably long lifetime which is captured in a pattern's *confidence*. Based on the outlined ideas, we term this approach a self-organized middle-out abstraction system, or simply SOMO learner.

3.1 Representation & Algorithm

We choose simple *situation-action pairs* for describing agent behaviours across various levels of scale. Of particular interest in the context of SOMO are *hierarchical operators*, i.e. predicates and actions for coping with hierarchical relationships between agents. In order to establish a hierarchical relationship, an agent might `enter` another agent. Alternatively, it might be `adopted` by another agent. Both actions yield corresponding parent-child relationships between the two agents. Such a parent-child relationship is reverted by `raising` a child in the hierarchy.

We assume that primarily the agents that are root nodes in the hierarchy are considered for execution. Children of a node are only recursively executed, if they are flagged as *active*, which implies the existence of two actions `activate` and `deactivate` to mark an agent accordingly. Deactivated child nodes make sense, for instance, if their parent nodes subsume their behaviours but need to maintain them for state updates and, potentially, their later release and re-activation.

A parent might `cover` its children, rendering them invisible to its environment, or `expose` them. In the latter case, the children are perceivable to other agents at the hierarchical level of their parents and might trigger interactions. The other way round, whenever an active child node is executed, its set of potential interaction partners is limited to its siblings and those agents it is exposed to.

3.2 Interaction Patterns

SOMO agents log the behavioural rules that were activated over a certain period of time in individual *interaction histories.* The entries of the sequential log contain information about the facts that triggered the rule, its effects, and its interaction partners. Similar to [11], we use the interaction histories as databases for finding patterns in the agents' interaction behaviours. Previously unknown patterns, or *motifs*, can be identified in time series relying on various advanced computing techniques such as learning *partial* periodic patterns [4], applying *efficient*, heuristic search [1], *online* motif search [3], and even the identification of patterns of *multiple resolutions* [16]. Motif detection is adapted to interaction histories by assigning symbols, e.g. A or B, to specific log entries and finding patterns in the resulting strings, e.g. $BBABCCBBABDA$. In the given example $BBAB$ is a motif candidate.

The cost of executing agent behaviours can be reduced according to the amount of information provided by the motif. For instance, a motif which relies on a subset of the agent's rules would make it superfluous to consider the remaining rules—the meta-agent would only consider the reduced rule set. Ideally, a motif provides comprehensive information about the interaction partners and the actual interactions, which allows to rewrite the agent rules as efficient sequences of unconditional instructions, with source and target agents readily in place.

SOMO agents can `observe` their own interaction history, as well as those of others, for finding patterns as the basis for building abstraction hierarchies. When `observing` a group of agents, motif detection is applied to their merged interaction history.

4 SOMO Behaviour

By means of the presented representation, the recursive execution of agent hierarchies, and the introduced hierarchical operators, the agents are capable of building and dissolving hierarchies as part of their behaviours. When combined with motif detection in interaction histories, behaviours can be designed to implement the SOMO learner.

4.1 Changing the Hierarchy

In the simplest case, an agent `observes` its own interaction history, `creates` a new agent, `assigns` its own abstracted behaviour, `enters` this new agent and `deactivates` itself. The newly created higher order agent `raises` and `activates` its children and `removes` itself from the simulation, as soon as its confidence has dropped below a certain threshold (see Section 4.2).

More excitingly, as motivated in Section 2, an agent `observes`, `adopts`, `deactivates` a whole group of other agents and `assigns` itself their simplified interaction behaviour. Repeated applications of these learning rules yield continuously growing hierarchies with increasingly simplified behaviours. At the same time, hierarchies are dissolved when no longer appropriate.

4.2 Confidence Estimation

When is it suitable for an `observing` agent to `adopt` another one to build up hierarchies? When will a hierarchy have to be dissolved by `raising` one's children? The key to these questions is *confidence estimation*. There is a large body of work around confidence in statistics [6] and its effective standardization for use in the natural sciences is a vivid research area [8]. The general idea is to estimate the probability that a pattern occurs based on its preceding frequency over a given period of time.

In SOMO, a sufficiently great confidence value leads to abstraction. The confidence value also determines the abstraction's lifespan. Too generous confidence metrics, i.e. too long abstraction lifespans, diminish the accuracy of a simulation. An abstraction agent can be validated by comparing its behaviour to the effects of its children's interaction either at the end of its lifespan or based on heuristics such as the degree of activity in its local environment. In case of miscalculations, the simulation could be reset to a previous simulation state, adjusted and partially recomputed. This additional overhead might make it hard to reach a gain in efficiency. On the other hand, if confidence is assigned too cautiously to motifs, abstraction hierarchies do not get a chance to form in the first place.

5 Discussion and Challenges

We have proposed the algorithmic concept and representation for SOMO, a self-organizing middle-out learner that automatically adjusts process abstractions in multi-agent based simulations. It relies on hierarchical relationships among agents and the fact that agents can change those relationships themselves. Hierarchies of higher order agents are recursively built that subsume and simplify the interaction processes of their children. The actual learning process is performed by motif detection algorithms that work on the agents' locally maintained interaction histories. Each motif is associated with a confidence value that determines the lifespan of a higher level agent.

After the integration of the outlined modules, e.g. motif detection and hierarchical operators, an extensive and systematic investigation into the impact of various system parameters on the accuracy and the efficiency of scientific simulations needs to be started. Following the self-organization paradigm, we aim at an adjustable agent behaviour for building and destroying hierarchies to optimally serve different simulation conditions. For instance, certain subspaces in a simulation might be subjected to fundamental and fast changes, which would result in a loss of efficiency due to the necessary hierarchy management. Other subspaces, however, might be rarely affected by change and abstracting costly agent dependencies could yield a significant gain in performance.

Another challenge is the application of locally learned patterns across the whole simulation space. This could tremendously cut costs for repeatedly observing and learning similar processes. Of course, this idea also emphasizes the more fundamental challenge of how general a motif should be interpreted. Again, this might depend on the volatility of the simulation (sub-)space.

Once the first successful SOMO systems have been explored, it will be important to investigate correlations between the learned abstractions and features and behaviours we find in higher order emergent phenomena. Whether there will be striking similarity or whether these are two completely different aspects of complex systems remains an exciting open question at this point in time.

References

1. Bill Chiu, Eamonn Keogh, and Stefano Lonardi. Probabilistic discovery of time series motifs. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 493–498. ACM, 2003.
2. Alan Dorin and Jon McCormack. Self-assembling dynamical hierarchies. *Artificial life eight*, page 423, 2003.
3. Erich Fuchs, Thiemo Gruber, Jiri Nitschke, and Bernhard Sick. On-line motif detection in time series with swiftmotif. *Pattern Recognition*, 42(11):3015 – 3031, 2009.
4. Jiawei Han, Guozhu Dong, and Yiwen Yin. Efficient mining of partial periodic patterns in time series database. In *Proceedings of the International Conference on Data Engineering*, pages 106–115. Citeseer, 1999.
5. Steven Johnson. *Emergence: The Connected Lives of Ants, Brains, Cities, and Software.* Scribner, New York, 2001.
6. Jack Kiefer. Conditional confidence statements and confidence estimators. *Journal of the American Statistical Association*, 72(360):789–808, 1977.
7. Tom Lenaerts, Dominique Chu, and Richard Watson. Dynamical hierarchies. *Artificial Life*, 11(4):403–405, 2005.
8. Thomas A. Louis and Scott L. Zeger. Effective communication of standard errors and confidence intervals. *Biostatistics*, 10(1):1, 2009.
9. Tom Mitchell. *Introduction to Machine Learning.* McGraw Hill, Boston, Massachusettes, 1997.
10. Denis Noble. *The music of life.* Oxford University Press Oxford, 2006.
11. S.D. Ramchurn, N.R. Jennings, C. Sierra, and L. Godo. Devising a trust model for multi-agent interactions using confidence and reputation. *Applied Artificial Intelligence*, 18(9):833–852, 2004.
12. Steen Rasmussen, Nils A. Baas, Bernd Mayer, Martin Nilsson, and Michael W. Olesen. Ansatz for dynamical hierarchies. *Artificial Life*, 7(4):329–353, 2001.
13. Peter Schuster. How does complexity arise in evolution. *Complex.*, 2(1):22–30, 1996.
14. Abbas S. Shirazi, Sebastian von Mammen, and Christian Jacob. Adaptive modularization of the mapk signaling pathway using the multiagent paradigm. In *Parallel Problem Solving in Nature (PPSN)*, Lecture Notes in Computer Science. Springer, 2010.
15. Steven H. Strogatz. *Nonlinear dynamics and chaos: With applications to physics, biology, chemistry, and engineering.* Westview Press, 2000.
16. Qiang Wang, Vasileios Megalooikonomou, and Christos Faloutsos. Time series analysis with multiple resolutions. *Information Systems*, 35(1):56 – 74, 2010.
17. Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.
18. Steven Wolfram. *A new kind of science.* Wolfram Media Inc., Champaign, Ilinois, US, United States, 2002.