

Proceedings of the 2014 Workshop on
Complex Systems Modelling and Simulation

CoSMoS 2014

Susan Stepney, Paul S. Andrews
Editors

CoSMoS 2014



Luniver Press
2014

Published by Luniver Press
Frome BA11 6TT United Kingdom

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

CoSMoS 2014

Copyright © Luniver Press 2014

All rights reserved. This book, or parts thereof, may not be reproduced in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval system, without permission in writing from the copyright holder.

ISBN-10: 1-905986-41-6
ISBN-13: 978-1-905986-41-5

While every attempt is made to ensure that the information in this publication is correct, no liability can be accepted by the authors or publishers for loss, damage or injury caused by any errors in, or omission from, the information given.

Preface

The CoSMoS workshops series has been organised to disseminate best practice in complex systems modelling and simulation, with its genesis in the similarly-named CoSMoS research project, a four year EPSRC funded research project at the Universities of York and Kent in the UK. Funding for the CoSMoS project has now completed, but we have continued to run the workshop series as a forum for research examining all aspects of the modelling and simulation of complex systems. To allow authors the space to describe their systems in depth we put no stringent page limit on the submissions.

We are pleased to be running the seventh CoSMoS workshop as a satellite event at the 14th International Conference on the Synthesis and Simulation of Living Systems (ALIFE 14), New York, NY, USA. ALIFE is the leading international conference on artificially constructed living systems, a highly interdisciplinary research area rich in complexity, which provides a natural complement to the issues addressed by the CoSMoS workshop.

The main session of the workshop is based on four full paper and two extended abstract submissions:

Afshar Dodson et al. apply the CoSMoS approach to analyse and re-engineer Schelling’s Bounded Neighbourhood Model, highlighting the importance of formalising a model for clarity and reproducibility in simulation studies.

Youssef and Rizk study node-heterogeneity in complex networks, proposing a new model for generating various types of complex networks by varying model parameters.

Banda et al. introduce a web-based chemistry simulation framework that provides an intuitive user interface, access to a computational grid and reliable database storage.

Andrews and Stepney show how the CoSMoS process and patterns can be used to reverse engineer a domain model of an existing simulation, Aevol, from the simulation code and associated research literature.

Leijnen and Dormans present a case study for designing emergence in games, showing how dynamical feedback loops in game mechanics creates fun and interesting gameplay experiences.

Mavelli et al. describe a computational platform for studying the role of randomness in a minimal cell model, highlighting how random fluctuations can play an important role in determining timing behaviour.

Our thanks go to all the contributors for their hard work in getting these submissions prepared and revised. All submissions received multiple reviews, and we thank the programme committee for their prompt, extensive and in-depth reviews. We would also like to extend a special thanks to the organising committee of ALIFE 14 for enabling our workshop to be co-located with this conference. We hope that readers will enjoy this set of papers, and come away with insight on the state of the art, and some understanding of current progress in complex systems modelling and simulation.

Programme Committee

Kieran Alden, University of York, UK

Paul Andrews, University of York, UK

Jim Bown, University of Abertay, Dundee, UK

Philip Garnett, Durham University, UK

Fiona Polack, University of York, UK

Mark Read, University of Sydney, Australia

Adam Sampson, University of Abertay, Dundee, UK

Susan Stepney, University of York, UK

Chris Timperley, University of York, UK

Table of Contents

CoSMoS 2014

Using the CoSMoS approach to study Schelling's Bounded Neighbourhood Model	1
<i>Ali Afshar Dodson, Susan Stepney, Emma Uprichard, Leo Caves</i>	
Effect of Arriving Nodes Connection-Standards on Models for the Generation of Heterogeneous Complex Networks	13
<i>Bassant E. Youssef, Mohamed R. M. Rizk</i>	
COEL: A Web-based Chemistry Simulation Framework	35
<i>Peter Banda, Drew Blount, Christof Teuscher</i>	
Using CoSMoS to Reverse Engineer a Domain Model for Aevol . .	61
<i>Paul S. Andrews, Susan Stepney</i>	
Order of Battle: A Case Study for Designing Emergent Structure in Games	81
<i>Stefan Leijnen, Joris Dormans</i>	
<i>In Silico</i> Minimal Cell Model Systems	85
<i>Fabio Mavelli, Emiliano Altamura, Pasquale Stano</i>	

Using the CoSMoS approach to study Schelling’s Bounded Neighbourhood Model

Ali Afshar Dodson^{1,2}, Susan Stepney^{1,2}, Emma Uprichard³,
and Leo Caves^{1,4}

¹ York Centre for Complex Systems Analysis, University of York,
YO10 5DD, UK

² Department of Computer Science, University of York, YO10 5DD, UK

³ Centre for Interdisciplinary Methodologies, University of Warwick,
CV4 7AL, UK

⁴ Department of Biology, University of York, YO10 5DD, UK

Abstract. The basic CoSMoS process concerns the design, implementation, and use of a simulation built from scratch. However, the CoSMoS approach may be tailored and adapted for other styles of use. Here we describe how it has been applied to analyse and re-engineer an existing simulation, that of Schelling’s Bounded Neighbourhood Model. We find that using a principled approach to the analysis of an existing simulation facilitates formalisation of the model and reimplementing of the simulation. In the process, several ambiguities in implementing a simulation from the model were revealed. This highlights the importance of formalising a model for clarity and reproducibility in simulation studies, and also for providing new avenues for exploration of, and insight into, the factors influencing its emergent behaviour.

1 Introduction

The Complex System Modelling and Simulation (CoSMoS) approach [1] has been designed for the purpose of developing and using a simulation as a scientific instrument [2]. It provides a guide for modelling and simulating complex systems, and incorporates verification and validation throughout. It provides a structure for the development and use of simulations in an interdisciplinary endeavour between scientists who study a particular domain (the *domain scientists*), and software engineers who construct simulations to facilitate the study of that domain (the *simulation engineers*).

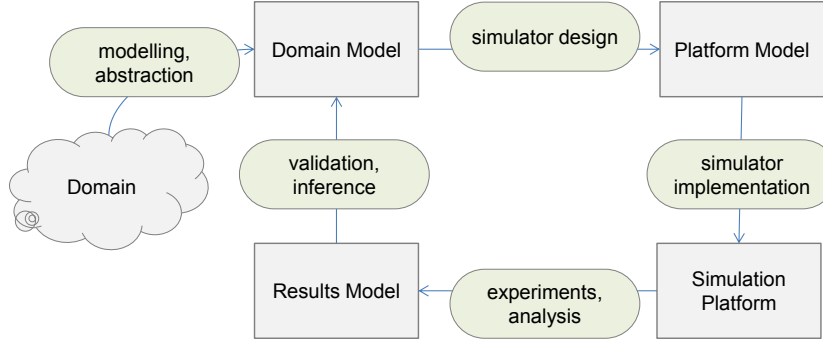


Fig. 1. The generic Complex Systems Modelling and Simulation (CoSMoS) approach. Products are shown in rectangles; activities are shown in rounded boxes; the entire approach (products and activities) takes place within a specified research context.

The CoSMoS approach has already been successfully used in a number of studies including auxin transport canalisation [3], environment orientation [4], immunology [5] and cancer systems biology [6]. Despite this specific use, CoSMoS can also be tailored and adapted for other uses. For example, it can be used to design bio-inspired algorithms [7], and to reverse engineer models from implementations [8]. Here we describe how it has been used to formalise, reimplement, and analyse an existing third party model and simulation: Schelling’s Bounded Neighbourhood Model [9] of segregation.

The structure of the rest of the paper is as follows:

- §2 summarises the relevant parts of the CoSMoS approach.
- §3 introduces Schelling’s Bounded Neighbourhood Model.
- §4 uses CoSMoS concepts to analyse Schelling’s model, formalise it, and then reimplement it using the standard CoSMoS approach.

2 The CoSMoS approach

The CoSMoS process is described in detail in [1], and summarised in figure 1. It comprises a set of products (models, software, arguments), including [1, p13]:

Research Context : captures the overall scientific research context of the simulation development project, including the motivation for the research, the questions to be addressed by the Simulation Platform, and the requirements for validation and evaluation.

Domain : the domain scientist's view of the subject of simulation; for example, a real-world system that is the subject of scientific research, or an engineered system that is the subject of engineering research and design.

Domain Model : distils appropriate aspects of the Domain into explicit domain understanding. The Domain Model focuses on the scientific understanding; no simulation implementation details are considered.

Platform Model : comprises specification, design and implementation models for the Simulation Platform, based on the Domain Model and research context. Any emergent properties captured in the Domain Model that are hypothesised to arise from lower level interactions are removed from the Platform Model, to ensure that the desired result is not explicitly coded into the simulations.

Simulation Platform : encodes the Platform Model into a software and hardware platform with which simulation experiments can be performed.

Results Model : encapsulates the understanding that results from simulation experiments: the Simulation Platform behaviour, results of data collection and observations of simulation runs. Note that the way that the Domain Model captures the relevant understanding of the domain, via experiments, observation, and theory, is mirrored by the way that the Results Model captures understanding of the simulation experiments.

By following the CoSMoS process an important distinction is made, not only between the model and the software that implements the model in the simulation, but also between the results from the simulation and any results from the 'real world' domain.

3 Schelling's Bounded Neighbourhood Model

Schelling's 1971 paper "Dynamic Models of Segregation" [9] is a seminal work in the field of Computational Social Science and is often used as a classic example of what agent based modelling (ABM) can offer to social sciences and urban studies. The models of segregation presented therein have been the focus of a number of studies, using a variety of different techniques. Schelling used agent based models to explore ideas of segregation in heterogeneous populations. He developed models to examine levels of segregation between two populations, using simplified ideas of social interaction. His work has been described as the first ABM [10]. It has also been called the first simulation of an *artificial* society [11].

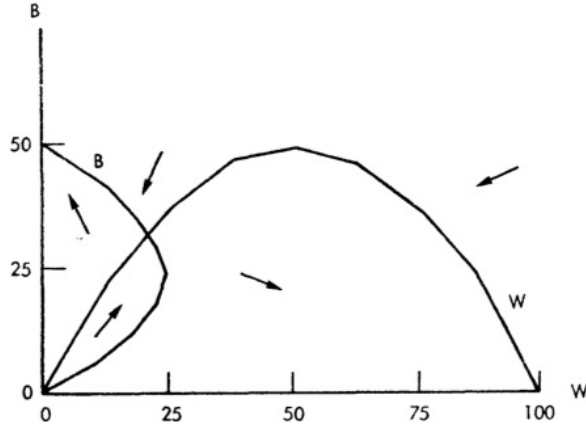


Fig. 2. Schelling's results (from [9])

Schelling describes two distinct models, a Spatial Proximity model [9, p149] and an aspatial Bounded Neighbourhood model [9, p167]. In both models, agents have a general preference for their own type (*like*) over the other type (*unlike*), and move to maintain a favourable ratio. Schelling found that mixed populations are impossible to maintain, with populations eventually segregating, even when the desire for like over unlike is quite weak.

The vast majority of the work after Schelling focuses on the grid-based Spatial Proximity Model in both one [12] and two [13] dimensions. Here we focus on the less well studied aspatial Bounded Neighbourhood Model [9].

The Bounded Neighbourhood Model has a heterogeneous population of two agent types, with some agents inside and some outside an aspatial *neighbourhood*. It considers the flow of agents into, and out of, the neighbourhood. Each agent calculates its *happiness*, based on the current ratio of the number of the other type, compared to the number of its own type, inside the neighbourhood. This ratio is compared to the agent's individual fixed *tolerance* of such a ratio. If its tolerance exceeds the ratio, it is deemed *happy*; if its tolerance is below the ratio, it is deemed *sad*. Sad agents inside the neighbourhood leave, while happy agents outside enter.

Figure 2 shows Schelling's analysis of the resulting population flows, given a total population of 100 *W*s and 50 *B*s, where the agents of a given type have a uniform distribution of tolerances, from 2 (willing to tolerate twice as many unlike agents as like agents) down to 0 (unwilling

to tolerate any unlike agents). The change of population numbers inside the neighbourhood for any given current number is shown by the arrows. In some regions both W and B agents leave, in some both enter, and in some one type enters and one leaves. The resulting equilibrium state is tipped to either all B or all W . This is despite the fact that half of each population is actually willing to tolerate a (limited) majority of unlike agents: the migration of the least tolerant changes the ratios enough that even the most tolerant find themselves sufficiently outnumbered that they too leave. The same result is found with different population numbers, and different tolerance schedules.

The popularity of Schelling's ideas led to criticisms, most notably Yinger [14], who argues that there is no consideration of a number of factors such as economics and social mobility. Similarly, Massey [15] argues that not accounting for environmental/spatial factors and using a homogenous environment means the models are too simplistic to have any relation to reality. If the model were being developed today, this could be a fair criticism, since the input of experts in the field is an important part of the modelling process. However, it is important to recognise the purpose and scope of Schelling's simple model. Schelling demonstrated that a simple desire for non-minority status in a population could lead to highly segregated systems.

Recent work has nevertheless attempted to apply Schelling's models to 'real-world' situations [16]. Such attempts, without acknowledging the limitations of the models, could lead to invalid conclusions. We apply the CoSMoS approach retrospectively to Schelling's Bounded Neighbourhood Model [9, p167], in order to formalise and re-implement it, and to analyse the consequences of its assumptions through simulation.

Whilst our work is an attempt to gain a better understanding of Schelling's model, his interpretation is necessarily retained. *Because of this, and the inherent difficulty of relating models to reality, we make no attempt to relate the model back to reality.* Instead, results from the simulation are used to formulate questions about Schelling's model.

4 Schelling's development through a CoSMoS lens

Here we analyse Schelling's process in building his model in terms of the CoSMoS approach (figure 3). There are points where Schelling's approach differ from the CoSMoS approach. We need to analyse these differences to determine their implications for his model, the simulation results, and subsequent interpretations. Any shortcomings are not, necessarily, the fault of Schelling; the work presented here is an attempt to 'lift' Schelling's model to current computational modelling standards.

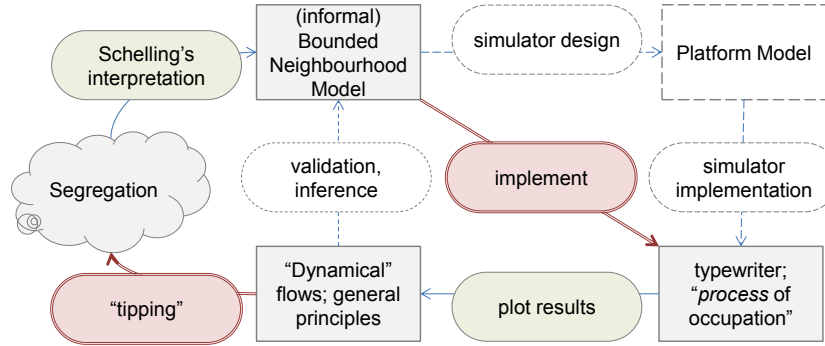


Fig. 3. Schelling’s process in developing the Bounded Neighbourhood Model. The dashed boxes show the CoSMoS products and activities not present in Schelling’s work, whilst the double edged boxes and lines are alternative steps employed.

Firstly, and most importantly, Schelling’s first step from the domain (Segregation) to the Domain Model (informal Bounded Neighbourhood Model) is Schelling’s personal interpretation [9, p143], rather than being based on observational and experimental data from the Domain. Whilst this may be acceptable (but not advised) for models in which the simulation engineer is also a domain expert, Schelling’s background was mathematical, rather than sociological. CoSMoS requires the Domain Model to be built in collaboration with domain scientists, or, at the very least, validated by them.

Secondly, Schelling’s presentation of the Domain Model is informal, and contains ambiguities that require resolution before a Platform Model or Simulation can be developed. For example, when talking of the movement of sad agents, Schelling states “some will move” in order of tolerance, but does not specify more precisely which ones (which type?), or how many (one? all?). Which agents move at each step needs to be formalised before a computational simulation can be developed, and the decisions published, for clarity and reproducibility.

Next, Schelling proceeds directly from this informal Domain Model to an simulation, without passing through an equivalent of the CoSMoS Platform Model. In this case the Domain Model is very simple, and it might be thought that no Platform Model is necessary. However, some of the ambiguities identified in the Domain Model became clear only by attempting to formalise them within a Platform Model (order of execution of operations, for example). If proceeding directly to imple-

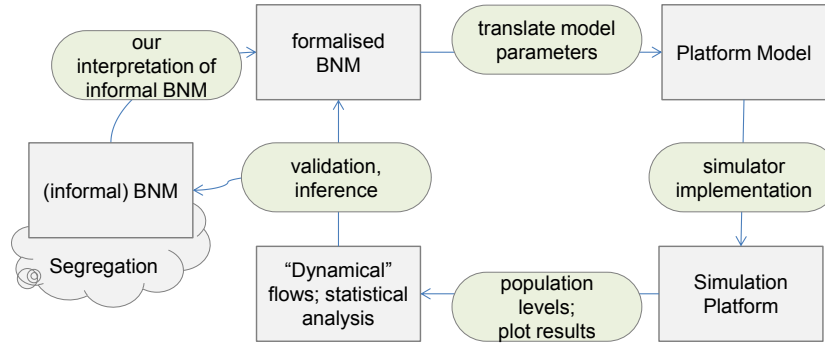


Fig. 4. An application of the CoSMoS process in developing a simulation of Schelling’s Bounded Neighbourhood Model. Note the outputs of the process feed back into the interpretation, rather than the real world.

mentation, such ambiguities would be resolved only in the code, leaving their resolution (and existence) opaque to the general reader. Note that in Schelling’s case, there was no computer-based simulation: the work was done with pencil and paper, or on a typewriter. Hence these ambiguities were resolved in Schelling’s head, leaving their resolution even more opaque.

Finally, Schelling takes his results and then attempts to apply them to a real world event (neighbourhood tipping) [9, p181]. Because of the extreme distance of his Domain Model from the real world Domain of population segregation, and lack of any validation from social scientists, it is difficult to defend attempts to relate his results back to reality. This argument is used by a number of authors who attack his models as missing essential components of reality [14], [15].

5 Reimplementing the Bounded Neighbourhood Model using CoSMoS

Having analysed Schelling’s model, and his (assumed) development approach, we are now in a position to reimplement the model, in the form of a simulation, suitable for further experimentation. We use the basic CoSMoS approach to do so, modified to accommodate the fact that we are using only literature, not domain experts, in order to build the models and simulation.

Algorithm 1: Agent movement rule, one timestep

if $\exists a_n \in E^* \wedge \tau(a_n) \geq R_n$ (there is a happy agent outside) **then**
 move a_n with max τ to E (move happiest agent inside)
else if $\exists a_n \in E \wedge \tau(a_n) \leq R_n$ (there is a sad agent inside) **then**
 move a_n with min τ to E^* (move saddest agent outside)

5.1 Domain Model

Following the CoSMoS approach, we refined Schelling’s formulation into a formalised Bounded Neighbourhood Model, specifying mathematically what is meant by terms such as ‘environment’, ‘agent location’, ‘agent type’, ‘ratio’, ‘tolerance’, ‘happiness’, and agent movement. This activity highlighted the key parameters of the model and made them explicit, removing ambiguities that could arise from the language of the model [Afshar Dodson *et al.*, forthcoming *a*].

Note here that we effectively took Schelling’s informal model as our Domain, and formalised it by examining the text in detail, and by using social science knowledge (of author EU).

5.2 Platform Model and Simulation Platform

Once explicitly defined, the parameters were encoded in the Platform Model. This process highlighted a few more ambiguities, this time in the actual algorithm used to iterate through generations of movement in and out of the neighbourhood. These were resolved by specifying the details in high-level pseudo-code (for example, algorithm 1), using the mathematical notation defined in the Domain Model. Here we had to make specific choices (such as, the happiest agent moves); highlighting the choice allows us to make other choices (such as, a random happy agent moves) to investigate the effect of the choice on the behaviour of the system [Afshar Dodson *et al.*, forthcoming *b*].

The Platform Model was used as the specification for a NetLogo ABM implementation.

5.3 Simulation Experiment Results

Given the re-implemented Simulation Platform, it was possible to conduct a number of experiments on the model.

In the first iteration, the validity of the formalised model was tested against the original model results. We initialised the model to each of the possible starting conditions (number of agents of each type) and plotted the resulting flows on a 2D plane (figure 5). The comparison shows

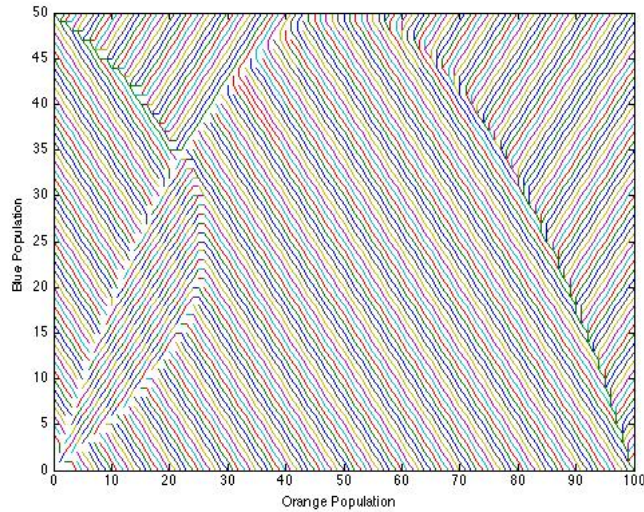


Fig. 5. To validate the simulation, its output and Schelling's results (figure 2) were compared. For each initial condition the resulting population flow is plotted. The simulation results clearly show Schelling's boundaries, with the flows following Schelling's results.

the simulation reproduces Schelling's flows (figure 2), providing evidence that it can be considered a valid simulation of Schelling's Bounded Neighbourhood Model.

Given a valid formalisation and Simulation Platform, we could then perform further simulation experiments, and to explore the assumptions underlying the original model, and determine how the resulting behaviour depends on those assumptions. We have discovered that the model is surprisingly robust to quite drastic changes in the underlying assumptions: it is very difficult to change find a model that does not inevitably result in segregation [Afshar Dodson *et al*, forthcoming *b*].

6 Summary and Conclusions

We have applied the basic CoSMoS approach in order to develop a formalised model and validated simulation of Schelling's Bounded Neighbourhood Model, suitable for use as the basis for further simulation

experiments to gain a better understanding of the factors underlying its emergent properties.

This work demonstrates that the concepts from the CoSMoS approach can be employed to interrogate a previously developed model and simulation. We have analysed Schelling's original model through the CoSMoS approach, identified products and activities missing or glossed over, and explained why these are necessary for a reproducible simulator, even in such a seemingly simple model as this. Our intention is not to criticise Schelling's approach; his work represents the very earliest steps in agent-based social simulation, and he was running experiments on a typewriter or using pencil and paper, resolving ambiguities on the fly. However, it is telling that few if any subsequent authors implementing his model in a computational setting have noted the many ambiguities in Schelling's original description.

One of the key advantages to using the CoSMoS approach is that it breaks down the making and shaping of a simulation into distinct parts of the modelling process. Importantly, it does this without losing the depth and breadth of expertise that may be required at different parts of the process, whilst at the same time allowing for the model to be developed iteratively, and thereby refined and tested over time. By focusing on Schelling's Bounded Neighbourhood Model, we have shown how the CoSMoS approach can be usefully deployed to re-develop past models.

We have deliberately made no attempt to relate this particular model to reality, yet the CoSMoS approach may be especially interesting for modelling social phenomena more generally. Social behaviours are emergent properties of an underlying complex system. The crux in the interpretation of social simulation is what factors influence the emergent features. Such factors span issues of approximation, implementation and potential coding errors. Thus, it is essential that models and their implementations are formalised and carefully described, in order that their results and interpretation may be fully reproduced and tested. These steps are necessary to put simulation studies (in general, not just in the social domain) into the realm of science. Principled model construction approaches, such as CoSMoS, offer a promising way forward.

In redeveloping Schelling's model, we have also, by implication, made a first step towards an ambitious longer term project that involves using CoSMoS as a way of rethinking how we might methodically and systematically (re)explore causal mechanisms from which the social emerges.

References

- [1] P. S. Andrews, F. A. C. Polack, A. T. Sampson, S. Stepney, and J. Timmis, "The CoSMoS Process, Version 0.1: A Process for the Modelling and Simulation of Complex Systems," Tech. Rep. YCS-2010-453, University of York, 2010.
- [2] P. S. Andrews, S. Stepney, and J. Timmis, "Simulation as a scientific instrument," in *Proceedings of the 2012 Workshop on Complex Systems Modelling and Simulation, Orleans, France, September 2012*, pp. 1–10, Luniver Press, 2012.
- [3] P. Garnett, S. Stepney, F. Day, and O. Leyser, "Using the CoSMoS Process to enhance an executable model of auxin transport canalisation," in *Proceedings of the 2010 Workshop on Complex Systems Modelling and Simulation, Odense, Denmark, August 2010*, pp. 9–32, Luniver Press, 2010.
- [4] T. Hoverd and S. Stepney, "Environment orientation: an architecture for simulating complex systems," in *Proceedings of the 2009 Workshop on Complex Systems Modelling and Simulation, York, UK, August 2009*, pp. 67–82, Luniver Press, 2009.
- [5] K. Alden, J. Timmis, P. S. Andrews, H. Veiga-Fernandes, and M. C. Coles, "Pairing experimentation and computational modeling to understand the role of tissue inducer cells in the development of lymphoid organs," *Frontiers in immunology*, vol. 3, no. 172, 2012.
- [6] J. Bown, P. S. Andrews, Y. Denni, A. Goltsov, M. Idowu, F. A. C. Polack, A. T. Sampson, M. Shovman, and S. Stepney, "Engineering Simulations for Cancer Systems Biology," *Current Drug Targets*, vol. 13, no. 12, pp. 1560–1574, 2012.
- [7] P. S. Andrews, S. Stepney, T. Hoverd, F. A. C. Polack, A. T. Sampson, and J. Timmis, "CoSMoS process, models, and metamodels," in *Proceedings of the 2011 Workshop on Complex Systems Modelling and Simulation, Paris, France, August 2011*, pp. 1–13, Luniver Press, 2011.
- [8] P. S. Andrews and S. Stepney, "Using CoSMoS to reverse engineer a Domain Model for Aevol," in *Proceedings of the 2014 Workshop on Complex Systems Modelling and Simulation, New York, USA, July 2014*, pp. 61–80, Luniver Press, 2014.
- [9] T. C. Schelling, "Dynamic Models of Segregation," *Journal of Mathematical Sociology*, vol. 1, pp. 143–186, 1971.
- [10] J. M. Epstein, "Why model?," *Journal of Artificial Societies and Social Simulation*, vol. 11, no. 4, p. 12, 2008.
- [11] R. Sawyer, *Social Emergence, Societies as Complex Systems*. Cambridge University Press, 2008.
- [12] R. Pans and N. J. Vriend, "Schelling's spatial proximity model of segregation revisited," *Journal of Public Economics*, vol. 91, pp. 1–24, 2006.
- [13] M. Fossett, "Ethnic preferences, social distance dynamics and residential segregation," *Presented at the Annual Meetings of the American Sociological Association*, March 1999.

- [14] J. Yinger, *Closed doors, opportunities lost: the continuing costs of housing discrimination*. Russel Sage Foundation, 1997.
- [15] D. S. Massey and N. A. Denton, *American apartheid: segregation and the making of the underclass*. Harvard University Press, 1993.
- [16] E. Hatna and I. Benenson, "The Schelling model of ethnic residential dynamics: Beyond the integrated - segregated dichotomy of patterns," *Journal of Artificial Societies and Social Simulation*, vol. 15, no. 1, p. 6, 2012.

Effect of Arriving Nodes Connection-Standards on Models for the Generation of Heterogeneous Complex Networks

Bassant E. Youssef¹ and Mohamed R. M. Rizk²

¹ Bradley Department of Electrical and Computer Engineering,
Virginia Tech, VA, USA

² Alexandria University, Egypt

Abstract. World Wide Web (WWW), social networks, food web (or food chain) networks are examples of complex networks. Complex networks are characterized by having a scale-free power-law degree distribution, a small average path length (small world phenomenon), a high average clustering coefficient, and showing the emergence of community structure. Mathematical models were used as a technique for generating graphs with particular statistical properties. Most proposed complex networks models have not incorporated all of these four statistical properties of complex networks. Additionally, models have also neglected incorporating the heterogeneous nature of network nodes. Moreover, even proposed heterogeneous complex network models were not general for the generation of different complex networks. Here, we define a new aspect of node-heterogeneity that was never previously considered which is the node connection standard heterogeneity. Thus, we have two heterogeneity aspects which are the heterogeneity of node characteristics and heterogeneity of node connection-standards. Finding a faithful general complex network model that preserves real complex network statistical properties and incorporate these two heterogeneity aspects is still an open research question. In this paper, we propose a generation model for heterogeneous complex networks. We introduce our novel model “settling node adaptive model” SNAM. SNAM reflects the heterogeneous nature of nodes’ connection-standard requirements. Such novel nodes’ connection standard criterion was not included in any previous network generation models. SNAM excels in its capability to generate various types of complex networks by varying model parameters. Additionally, SNAM was successful in preserving

the power law degree distribution, the small world phenomenon and the high clustering coefficient of complex networks.

1 Introduction

Complex networks are ubiquitous in many areas. The Internet, the World Wide Web (WWW), social networks, food web (or food chain) networks and many other networks are complex networks [1]. Researchers studied and analyzed data extracted from complex networks which led to the discovery of their distinct features and behavioral patterns. Solid awareness of these features can lead to an improved understanding of the network's structure and dynamics. Devising a mathematical model for complex networks can aid in making decisions about complex networks management and help allocating their resources. It can be used to answer research questions such as discovering the mediator for disease transmission in sexual networks, predicting future connections between websites in the WWW, identifying critical nodes or links in power grid networks; etc. Therefore, finding a sufficiently detailed mathematical model that is capable of mimicking the structure, dynamics and evolution of complex networks is paramount. Researchers used advanced computer capabilities to analyze real large databases to identify essential properties for modeling complex networks in the process of creating such mathematical models [2, 3]. Complex networks represented as graphs have been shown to exhibit several common statistical properties, including degree distribution, average path length, clustering coefficient, and community structure.

Degree Distribution The degree of a vertex in a network represents the number of connections that the vertex has. The degree of a vertex j in an undirected graph is the total number of edges connected to that vertex and it is expressed as k_j . However, in a directed graph, edges are classified as ending at a vertex or as originating from a vertex. The in-degree of a vertex j is the total number of edges ending at j , while the out-degree of a vertex j is the total number of edges originating from j . The in-degree and out-degree of a vertex j are expressed as $k_{j.in}$ and $k_{j.out}$, respectively. Thus, the total degree of a vertex j in a directed graph will be expressed as $k_j = k_{j.in} + k_{j.out}$. $P(k)$ represents the fraction of vertices in the network with degree k and it denotes the probability that if a vertex v is picked uniformly at random it will have degree k . Degree distributions in complex networks can follow an exponential, Poisson or power law distribution according to the network's nature [3].

Average Path Length The path length between a pair of vertices is equal to the number of links or hops that form the path that connects the two vertices [3]. There may be different paths connecting a pair of vertices. The shortest path, referred to as geodesic distance, is the connecting path that has the lowest number of links. The average path length in a network is defined as the average number of links along the shortest paths for all possible connected pairs of vertices in the network [3]. For an undirected network having n vertices, the average path length l is the mean of the geodesic (the shortest) distance between all vertex pairs in the network and it is defined as: $l = \frac{1}{\frac{1}{2}n(n-1)} \sum_{i>j} d_{ij}$, where d_{ij} is the shortest path (or geodesic distance) between any two vertices i and j [3].

Clustering Coefficient A node's clustering coefficient is defined as the average fraction of pairs of neighbors of a node that are also neighbors of each other [2]. Generally, the clustering coefficient is used to assess transitivity of real world networks. Transitivity means that if vertex i is connected to vertex j , and vertex j is connected to vertex k , then there is a high probability that vertex i will also be connected to vertex k . The value of the average network clustering-coefficient, C , ranges between 0 and 1, and can be defined in any of the following ways [3]:

1. $C = 3 \times \frac{\text{number of triangles in the network}}{\text{number of connected triples of vertices}}$, where a triangle contains three interconnected vertices and a connected triple is a single vertex with its two edges running to an unordered pair of vertices [3].
2. $C = 6 \times \frac{\text{number of triangles in the network}}{\text{number of directed paths of length 2}}$, where triangles are as defined above and a directed path of length 2 refers to a directed path of length 2 starting from a specified vertex [3].
3. Watts and Strogatz [4, 5] calculated the network's average clustering coefficient as the average of the individual clustering coefficients of network vertices C_i s. The clustering coefficient for node i is given by: Node i clustering coefficient = $C_i = \frac{\text{number of triangles connected to node } i}{\text{number of triples centered on } i}$, where a triangle has one of its vertices at node i , while a triple is a two-sided incomplete triangle with its vertex at i .

Community Structure A community is a group of vertices having high density of edges within the group (the community) and a lower density of edges to vertices of other groups (other communities) [5]. Some networks show the presence of communities or a "community structure". This can be accurately evaluated by using community identification techniques. Networks having a community structure are sometimes referred to as networks with high clustering coeffi-

icients. However, according to present definitions, the two properties are not considered equivalent [5].

1.1 Statistical properties of complex networks

Statistical properties of complex networks were identified as: small world effect, high average clustering coefficient, scale free power law degree distributions, and the emergence of community structure [3, 4].

Small world effect means that for a certain fixed value of the nodes' mean degree, the value of the average path length scales logarithmically, or slower, with network size. The average clustering coefficients in real complex network tend to have high values. Community structure emerges when nodes in a community have denser connections within themselves than to vertices of different communities [5]. In other words, community structure exists when sub-regions of relatively higher interconnectivity form are separated by regions of lower interconnectivity in the network. Degree distribution follows a scale free power law distribution in real complex networks. Scale free power law distributions, $P(k) \sim k^{-\gamma}$, have a power law (PL) exponent γ independent of the size of the network and its values are in the range of $1 < \gamma < \infty$ [3, 4].

Various models tried to find a faithful model for complex networks. The most influential models in the complex-network modeling field are: Erdős and Rényi (ER), Watts and Strogatz (WS), and Barabási and Albert (BA). Networks generated according to the ER random graph model have small average path length but they have Poisson degree distributions and are characterized by having clustering coefficients lower than that found in real complex networks [3, 4]. Networks generated by the WS small-world network model have a short average path length and a high clustering coefficient. However, it lacks modeling the scale free property for the networks' degree distribution [2–4]. Thus, the scale-free power-law degree distribution of real complex networks was not represented in the ER or the WS models, rendering both models to be insufficient in modeling the four characteristics of real complex-networks. This motivated Barabási and Albert to induce the scale free property for node-degree distribution in their highly acclaimed model [2]. The BA model uses a Preferential Attachment (PA) connection algorithm that reflects the belief that nodes usually prefer to connect to higher-degree structurally-popular nodes [2]. BA model succeeded in preserving the PL degree distribution and small world phenomenon of real complex-networks. Networks generated by the BA model show a power-law heavy-tail degree distribution, if and only if, the model has the following two properties; growth (where new nodes are continuously added to the network) and preferential attachment (PA). The BA model starts with a

Table 1. BA model vs. WS and ER

	Barabási Albert	Erdős Rényi (ER) & Watts Strogatz
Degree distribution	Power Law	Poisson
Number of nodes (N)	Growing	Constant
Connection probability	Preferential attachment	Random

small number of nodes (m_0), which is referred to as the seed network. A new node is added at each time step. The new node preferentially attach to other m nodes, (where $m \leq m_0$) using a connection function based on the old nodes' normalized degrees. Thus, new node i connects preferentially to an old node j having degree D_j using a connection function (CF) based on the normalized degree of the node d_j , where $d_j = \frac{D_j}{\sum_j D_j}$.

Networks generated using the BA model have a scale-free power-law degree-distribution and their average path lengths exhibit the small world phenomenon. However, the BA model generates networks with a constant PL exponent value of $\gamma = 3$, unlike real networks where the exponent values differ according to the network type and ranges between $1 < \gamma < \infty$. Additionally, the BA modeled network average clustering coefficient is lower than that observed in real complex networks of the same size [3–5]. A comparison of the Erdős and Rényi (ER), Watts and Strogatz (WS), and Barabási and Albert (BA) models is shown in table 1.

The BA model was still inaccurate in representing all four properties observed in real complex-networks. This motivated many researchers to introduce modifications to the BA model in an attempt to remedy the model's shortcomings. Accordingly, devising a model that can represent all four properties of complex-network is still an ongoing research [5]. Additionally, most models have assumed that nodes have the same properties and neglected incorporating the heterogeneous nature of network nodes. Moreover, even proposed heterogeneous complex network models did not integrate it with other structural properties of the network in the analysis and growth algorithms of such networks. Also, models were not general for different types of complex networks [6–8]. Therefore, finding a faithful general heterogeneous complex network model that preserves real complex network statistical properties is still a challenge.

In this paper we aim to devise a mathematical model that preserves the statistical properties of complex-networks. Additionally, we include a factor that, we claim, was undermined in most contemporary complex-network models which is the node heterogeneity, [4, 5]. We identify two

types of node-heterogeneity; node characteristics heterogeneity and node connection standard heterogeneity. Node characteristics heterogeneity reflects the different properties or attributes that network-nodes have. Node connection standard heterogeneity reflects the difference in each node's requirements to make a connection. The contribution in this paper can be summarized as:

1. Accounting for node heterogeneity in the graph-theory by incorporating node-attributes as one of the elements defining a network graph. Accordingly, our model defines the network graph, G as a set of three elements: $G = \{V, E, A\}$, where V is the number of nodes in the network, E is the number of edges and A is the set of attributes assigned to each network node.
2. Based on 1. we propose the Settling Node Adaptive Model (SNAM) for generating complex-networks. SNAM acknowledges the heterogeneous nature of nodes by integrating the attribute-similarity with the structural popularity measure within the CFs.
3. SNAM departs from the BA algorithm while acknowledging the node heterogeneity. SNAM introduces the idea of heterogeneous node connection-requirements as a criterion for connecting nodes.

Our proposed models will be validated using Matlab simulation [9]. The success of each proposed model to mimic real complex networks will be verified by examining the generated network statistical properties, namely the average path length, clustering coefficient, and degree distribution.

The rest of the paper is organized as follows: section two presents the related work, section three presents our proposed models and their simulation results, and section four is the conclusion and future work.

2 Related Work

Several researchers have proposed mathematical models that address the heterogeneous nature of the nodes composing a network. The authors of this work examined the success of these models in generating networks that mimic real complex-networks by observing the statistical properties of these networks. This section will review a subset of these attempts.

Bianconi and Barabási in [6] were interested in WWW networks. They introduced the term node fitness to represent nodes' different abilities to attain connections. Their work was motivated by the observation that the nodes' abilities to attract connections do not depend only on their degrees (based on the nodes' ages). WWW nodes that provide good content are likely to acquire more connections irrespective of their ages.

In citation networks, a new paper with a breakthrough is likely to have more connections than older papers. Thus each node should be assigned a parameter that describes its competitive nature to attain connections. In their model, node j upon birth is assigned a fitness factor η_j , following some distribution $\rho(\eta)$, which represents its intrinsic ability to attain connections. Bianconi and Barabasi model followed the BA PA connection algorithm with a modified PA function. The model has the PA function value for connecting an old node j to a new added node i depending on the old-node degree D_j , and its fitness value η_j . When $\rho(\eta)$ follows a uniform distribution, the degree distribution is a generalized power law, with an inverse logarithmic correction. The average clustering coefficient and average path length values of networks generated by this model were not calculated in the presented work.

Shaohua et al. in [7] observed that nodes with common traits or interests tend to interact. They introduced an evolving model based on attribute-similarity between the nodes. Each of the network nodes has an attribute set. Node-attributes can be described by a true or false function as in fuzzy logic. Shaohua et al. used fuzzy similarity rules to define a similarity function between attribute sets of two nodes. A connection is established between two nodes if their attributes similarities fall within a certain sector. Despite that this model satisfies the small world property; its degree distribution follows a Poisson distribution and does not follow a power law.

Yixiao Li et al. in [8] argued that every vertex is identified with a social identity represented by a vector whose elements represent distinctive social features. The new node added at each time step connects with probability p to the group closest to its social identity and to the other groups with probability $(1 - p)$. The higher degree node is attached to the new node within a group using PA. Random linking to neighbors of the previously attached old node is repeated until the new node establishes its m links. Their generated network follows power-law degree distribution and used triad formation to produce high average clustering coefficients. The authors claimed that using triad formation produced high average clustering but they did not present values for it and they did not measure their generated networks' average path length. Additionally, the model did not increase the length of the attribute vector to more than one which is unrealistic as nodes usually possess more than one attributes. Increasing the attribute vector length would be useful in investigating the impact of including different attributes in the model on the generated network.

While [6–8] based their connection algorithm on the PA attachment algorithm, some authors experimented with models that were not based

on the BA PA algorithm such as those presented in [10] and [11]. Kleinberg et al. in [10] used a copying mechanism which entails randomly choosing a node then connecting its m links to neighbors of other randomly chosen nodes. The model was found to preserve power-law distributions using heuristics only. They argued that analytical tools were unable to prove this conclusion, because the copying mechanism generated dependencies between random variables. Krapivsky et al. [11] argued that an author, in a citation network, citing a paper is most likely going to cite one of its references as well. In their model, when a new node i is added to the network, its edge attaches to a randomly chosen node j with probability $(1 - r)$. Then with probability r this edge from the new node i is redirected to the ancestor node o of the previous randomly chosen node j . The rate equations of the model show that it has a power-law degree distribution with degree exponent decreasing with the increase of the probability r value. Other statistical properties were not studied. These models were able to generate networks having a power-law degree distribution without using the PA algorithm of BA. However, they are not applicable to all complex-networks as they assume that the arriving node decision to establish connections is affected by decisions made by other nodes (the one it will copy from the connections or its neighbor). Whether the node is copying its connections from a random node or connecting to the ancestor of a node previously connected to it, is not applicable for some types of complex networks. Additionally, the choice of the nodes from which the links are copied or the choice of the ancestors of the node is made randomly without regards to nodes-heterogeneous characteristics or their heterogeneous connection-standards.

Our previous paper [12] introduced the integrated attribute similarity models "IASM". IASM is a growing network model. It uses a preferential attachment algorithm to connect the nodes. The CF in IASM depends on the attribute similarity between newly arriving nodes and old network nodes as well as the structural popularity of old nodes. Two different structural popularity measures are used in IASM simulation. In IASM_A, a nodes structural popularity is based on the number of connections that the node has, i.e. the node-degree, while in IASM_B, the structural popularity is based on the nodes Eigen vector centrality. IASM preserved the power law degree distribution and the small world phenomenon but it did not reflect the high average clustering coefficient and the emergence of community structure. We enhance the IASM by adding a triad formation step (TFS) by having the arriving node make an additional connection to a neighbor of its previous neighbor. The TFS results in increasing the clustering coefficient values.

This motivated us to find a mathematical model that does not necessarily use the PA connection algorithm but at the same time generates networks with PL degree distribution. Our model should also be able to generate different types of complex networks. Additionally, the model should generate networks with the before mentioned statistical properties. The generated model should also preserve the concept of heterogeneity of nodes properties and their connection standards.

3 Settling Node Adaptive Model (SNAM)

3.1 Introduction

Nodes, users or entities, in real complex-networks have different profiles and characteristics. Connections between nodes affect the network dynamics, and their future evolution. We argue that nodes having different characteristics influence the density and the pattern of connections within a network. The notion of node-attributes is used to highlight the node-distinct characteristics. The attribute set is extracted from the characteristics or profiles of the network node. In our models, nodes are assigned their attributes upon their arrival to the network. Accordingly, the network graph G is now defined by a three-element set $G = \{V, E, A\}$, where V is the number of nodes in the network, E is the number of edges and A is the set of attributes defining the profiles/characteristics of all the network nodes. SNAM is a growing generation model with nodes constantly being added to the network during its evolution.

SNAM's connection algorithm uses attribute-similarity between the newly added node and the old node attribute(s) in the connection function (CF). Including the attribute similarity in CF makes it dependent on the attributes of both of the newly added node and the old node rather than having the CF dependent only on the old node's fitness/degree. SNAM integrates the attribute-similarity between new node and old nodes with the structural popularity of old nodes in the CF. The node structural popularity is a measure of the node's popularity based on its network position and connections. SNAM uses the node-degree as the structural popularity measure. The CF can be dependent on the normalized degree of the old node i D_i or on the normalized degree of the old node node i D_i multiplied by or added to the attribute similarities (A_{ij}) for both node i and new node j . Thus, the CF is given by: $CF = \alpha \frac{D_j A_{ij}}{\sum_j D_j A_{ij}} + \beta \frac{D_j}{\sum_j D_j} + w \frac{A_{ij}}{\sum_j A_{ij}}$, where $\alpha + w + \beta = 1.0$, $0 \leq \alpha \leq 1$, $0 \leq w \leq 1$, and $0 \leq \beta \leq 1$.

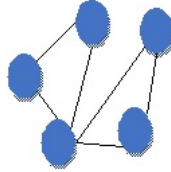


Fig. 1. Seed network, $m_o = 5$

SNAM departs from the classic PA connection algorithm presented in BA. SNAM reflects the idea that nodes are not only differentiated by their attributes but also according to their connection-standard requirements. Connection-standard requirements for the nodes represent the minimum CF values that a node find satisfactory to connect with other nodes. Thus, an arriving node x calculates its connection function value with an existing node z (CF1). If CF1 is equal to or higher than the arriving node's x connection standard, then node x makes a connection to node z . Another node y calculates its connection function with the same test node z (CF2) and finds CF2 lower than its connection standard. Thus, node y refuses to connect to the same existing node z . The CF used in our SNAM model depends on the structural popularity of the tested existing node and its attribute similarity with the new node.

To evaluate our models, we generate networks based on each model using MATLAB simulation. For each of the generated networks, values for the power law exponent, the average path length and the average clustering coefficients were measured and assessed against values reported for a variety of real complex-networks [3, 4]. These statistical properties are the three metrics that validate that the three features of real complex networks are preserved in our models.

In SNAM, each new network-node upon birth possesses its own distinct attribute-set (attribute vector of length L) that represents the interests or engagements of the node in the network's L interests or activities. The CF does not depend solely on a specific characteristic of the old node but on the characteristics of both the new and the old nodes. SNAM is a growing network model and starts with a seed network of size m_o shown in figure 1. Then at each time step a new node is added with m edges to be connected to it, where $m \leq m_o$.

Each node is assigned an attribute vector having L elements. Each element takes binary values of '1' or '0' representing the presence or absence of an attribute in the attribute-vector respectively. Our proposed attribute similarity is equal to the normalized summation of the inner

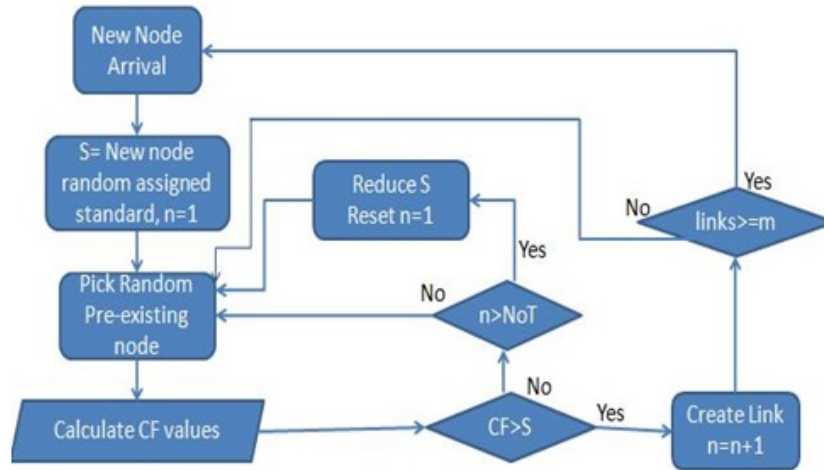


Fig. 2. Flowchart of SNAM algorithm

product of the new-node and old-node attribute vectors. The algorithm of SNAM is shown in the flowchart in figure 2.

To the extent of our knowledge, all previously proposed models assumed that all newly arriving nodes have the same requirements when connecting to old nodes. In reality, nodes may have different views of the same value of a connection-function (CF) calculated based on attribute similarity and/or structural popularity. A network node may have high connection standard and does not settle for the CF value offered by the tested old node, thus rejecting the connection. Another new node may have lower standards and considers the same CF value acceptable. To reflect this, we assign a characteristic that reflects the node's standards. This characteristic represents the minimum acceptable value of the CF for each node. All old pre-existing nodes whose CF values with the new node are below the newly arriving node standard will not be attached to that new node. The arriving node must then test other old pre-existing nodes to find the ones that satisfy its connection standard.

Thus, in SNAM, each arriving node, upon birth will be assigned a value representing its own connection standard value which is derived from a uniform distribution. Arriving node will calculate its CF values with old nodes. Hence, the CF obtained values will not be used to deploy the preferential attachment algorithm but will be used to examine if

the randomly chosen old nodes will meet the arriving node's standards. A newly arriving node will calculate the CF corresponding to random chosen nodes. The new node will establish connections with the old nodes whose CF values are equal to or higher than its connection-standard. The used CF depends on the normalized degree values and/or attribute similarity.

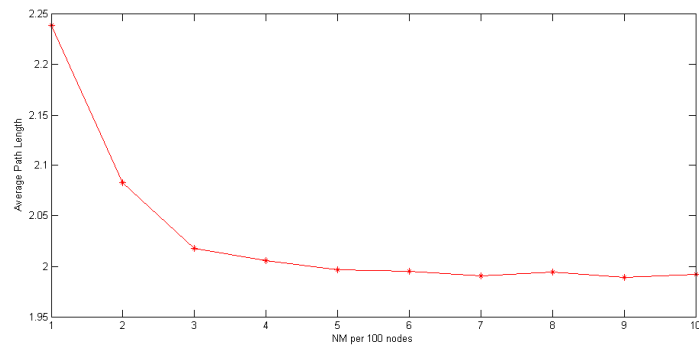
3.2 Results

The network starts with a seed network m_o . A new node arrives at each time step and each new node i is assigned a random connection-standard value S_i , where $0 < S_i \leq 1$. If for a chosen pre-existing old node j the CF value exceeds S_i then i will establish a connection to j , otherwise i rejects the connection to j and another old node is tested. This testing of other existing nodes continues until the new node achieves its predefined m connections or reaches its maximum number of tests, NoT. If node i reaches its maximum number of tests, NoT, and it still did not make its m connections, then arriving node i reduces its connection standard by a certain percentage and the testing of randomly chosen existing nodes is resumed. The reduced standard-connection value, S_i , reduced, is determined as follows: $S_{i \text{ reduced}} = S_i(1 - \varepsilon)$, where $\varepsilon < 1.0$

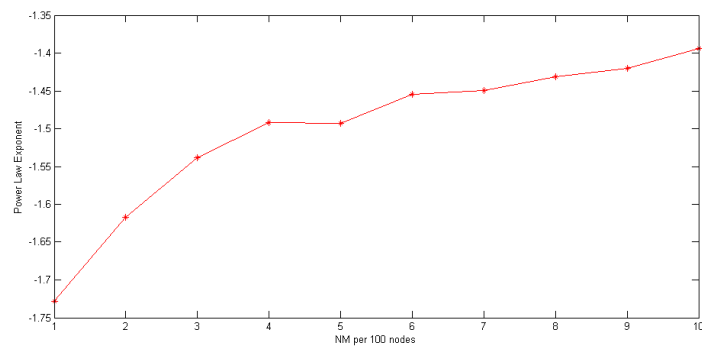
In the present model, we experiment with the maximum number of tests, NoT, required for the arriving node i to lower its standard if i has not already established its m connections during the NoT tests. NoT is initially taken as the integer value of half the seed size m_o . NoT value is dependent on the current size (CS) of the network. NoT is increased by one whenever the CS of network reaches certain predefined milestones. The number of milestones occurring during the arrival of every 100 nodes to the network is varied between 1 and 10. Thus, a number of milestones (NM) value of 5 means that the NoT is increased by one 5 times during the arrival of 100 nodes to the network (i.e NoT increased by one each time 20 new nodes arrive to the network).

The value of these milestones is dependent on the final size of the network (N) and the number of milestones (NM) occurring during network evolution. The number of milestones has two extreme values. The smallest number of milestones is one which is reached when the network reaches its final size. The largest number of milestones occurs when we consider the arrival of each to the network as a milestone. Thus, NM ranges between 1 and N. Thus, NoT is increased by one whenever the CS of the network reaches the milestones.. The higher the value of NM, the more rapid is the increase in NoT.

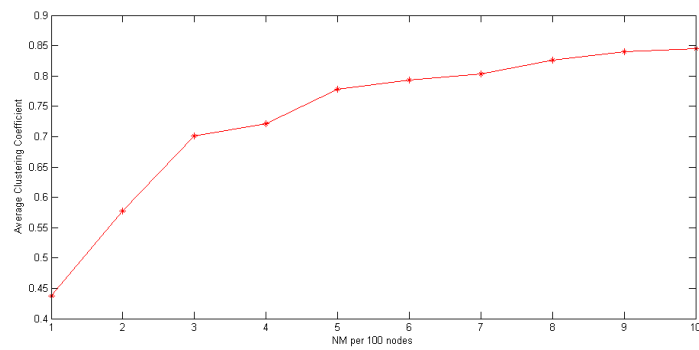
Our experimentation with NoT parameter indicated that rapid increase of NoT with network growth resulted in the presence of irregular-



(a) Average Path length

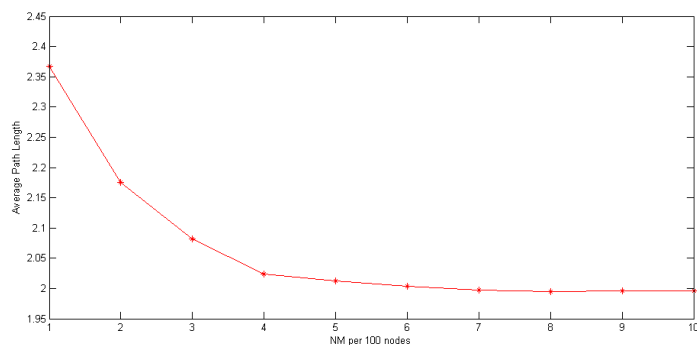


(b) Power law Exponent of Degree distribution

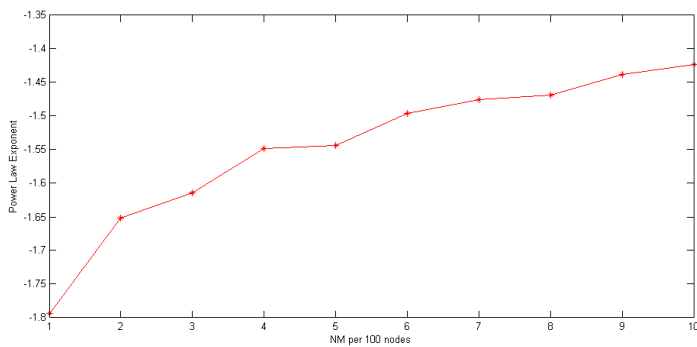


(c) Average clustering coefficients

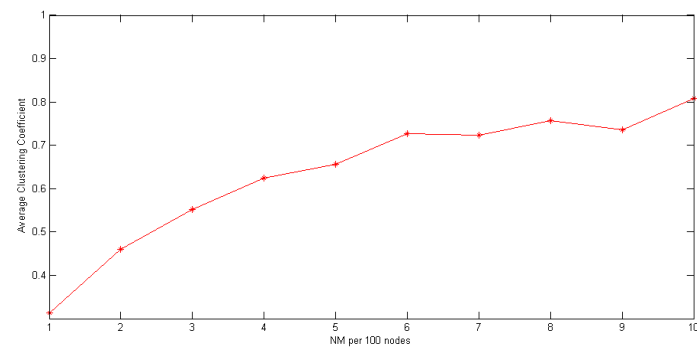
Fig. 3. SNAM algorithm with a normalized degree CF ($\beta = 1$)



(a) Average Path length

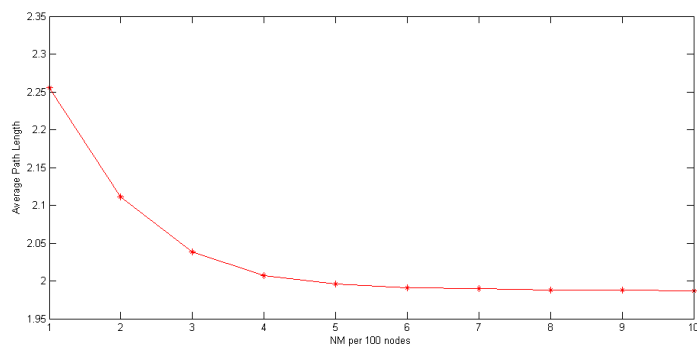


(b) Power law Exponent of Degree distribution

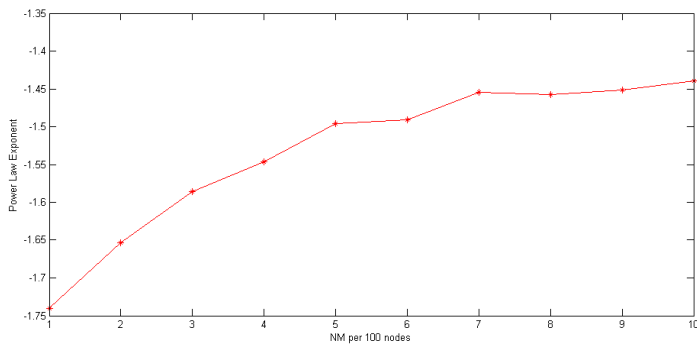


(c) Average clustering coefficients

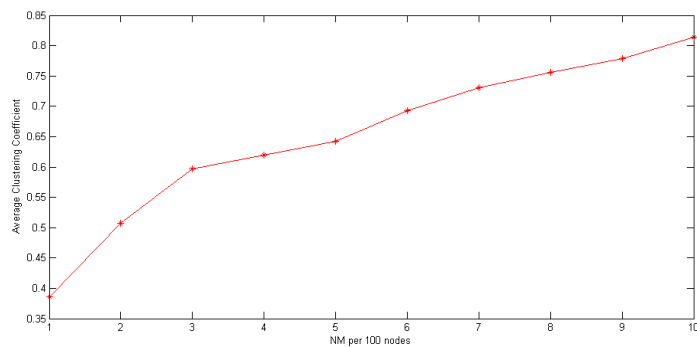
Fig. 4. SNAM algorithm with a normalized degree with added attribute similarity CF ($w = \beta = 0.5$)



(a) Average Path length



(b) Power law Exponent of Degree distribution



(c) Average clustering coefficients

Fig. 5. SNAM algorithm with a normalized degree with multiplied attribute similarity CF ($\alpha = 1$)

Table 2. Simulation Parameter Values

m_o	m	L	N
5	5	10	500

ities in the statistical characteristics of the generated network. Here, we consider the arrival the maximum value of NM is 10 . This choice was made to avoid irregular statistical properties, which appears when $NM > 10$, and has proved to give satisfactory results as shown in figures 3, 4, 5, 6. The connection function CF is dependent on the old node i degree D_i and attribute similarities (A_{ij}) for both node i and new node j , namely: $CF = \alpha \frac{D_j A_{ij}}{\sum_j D_j A_{ij}} + \beta \frac{D_j}{\sum_j D_j} + w \frac{A_{ij}}{\sum_j A_{ij}}$, where $\alpha + w + \beta = 1.0$, $0 \leq \alpha \leq 1$, $0 \leq w \leq 1$, and $0 \leq \beta \leq 1$. α , w and β are weighting coefficients used to give different weights to the combined structural popularity and the attribute similarity in the CF terms to test their influence . Simulation of SNAM starts with a seed network of size $m_o = 5$. The network size grows as new nodes arrive to the network, until reaching a predetermined final size N. In our simulation N=500. Each newly arriving node has to establish m links with the preexisting network nodes, where $m = m_o = 5$. Each new node in the network is randomly assigned an attribute vector of length L=10, whose elements are derived from a uniform distribution. Simulation parameter values used are summarized in table 2.

Matlab [9] simulations were performed for different combinations of the CFs' coefficients for both models. The simulation results show the average of 10 experiments with different random-seed generator values. CFs used can be based on normalized degree only ($\beta = 1, \alpha = w = 0$), on degree with added attribute similarity ($\alpha = 0$ and $w = 1 - \beta$ where $0 \leq \beta \leq 1$), and on degree multiplied by the attribute similarity. Simulation results for the Average Clustering Coefficient (Av_CC), the Average Path length (Av_PL), and the Exponent of PL (Exp_PL) for three combinations of the coefficients α , β and w are shown in figures 3, 4, 5.

Figures 3(a), 4(a), 5(a) for the average path length indicate that small world effect is preserved for three combinations of α , β and w as its value is $\leq \log N$, where $N = 500$. Average path length decreases with the increase of NM. Figures 3(b), 4(b), 5(b) show that the magnitude of PL exponents for the three variations remains in the range of $1.35 \leq \gamma \leq 1.75$ which is consistent with values found in real networks [1, 3, 4]. Additionally, the magnitudes of PL exponent saturates at values close to $\gamma \cong 1.35$ with the increase of NM.

The average clustering coefficient values increase with the increase of NM for the three variations as seen in figures 3(c), 4(c), 5(c). Average clustering coefficient reach high values compared to those of BA model. The clustering coefficients in figure 3(c) corresponding to degree only achieves higher values than those of figures 4(c) and 5(c) using additive attribute similarities and multiplicative attribute similarity CF respectively.

Examples of the generated networks using SNAM are shown in figure 6 for the three combinations of the coefficients α , β and w . The simulation parameters values are $m = m_o = 5$, $N = 500$, $L = 10$, and $NM = 10$. The generated networks are plotted using Pajek [13].

As previously mentioned, the connection-standard is reduced by a step of value ε . When increasing ε , the connection standard is reduced at a faster rate. Thus, we experiment with the effect that changing the connection-standard reducing step would have on the generated networks statistical properties. A 500 nodes network with CF coefficient depending on normalized degree ($\beta = 1$) is generated with ε taking the values of 0.1, 0.25, 0.5 and 0.9. From figure 7, we can conclude that at $\varepsilon = 0.9$, the presence of the connection-standard becomes less effective. The average path length values increases with increased ε . The average clustering coefficient decreases with the increase of ε . The power law exponents becomes unstable at $\varepsilon = 0.9$. High ε values would make the nodes reduce their standard more rapidly. This will decrease the effect of the presence of the node's connection-standard on the generated network. Thus, when ε approaches the value of one, the generated network approaches a network generated with no connection standards.

Thus, the SNAM generation model has preserved the PL degree distribution, has a small average path length, and has high clustering coefficient values. Parameters NM and ε values can be used to generate a variety of complex networks with specific values of the clustering coefficient, the average path length and the PL exponent.

4 Conclusion

This paper took into consideration that complex networks mathematical models should incorporate their statistical properties and should also reflect the heterogeneous nature of network nodes. In this paper, we propose several mathematical models that pave our path to find a final mathematical model that can successfully mimic real complex networks. The proposed models have heterogeneous network nodes with assigned distinct attributes. Our work is the first to assign more than one attribute to each node. SNAM integrates the attribute similarity measure

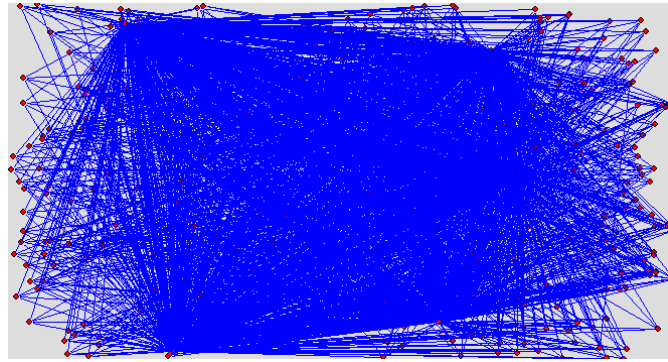
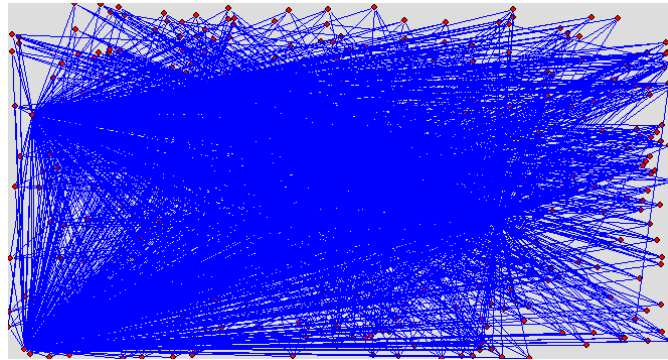
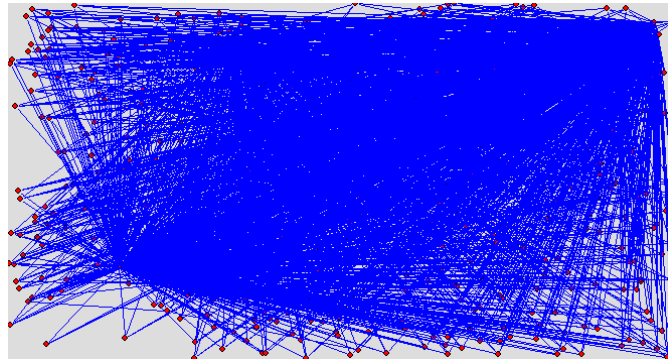
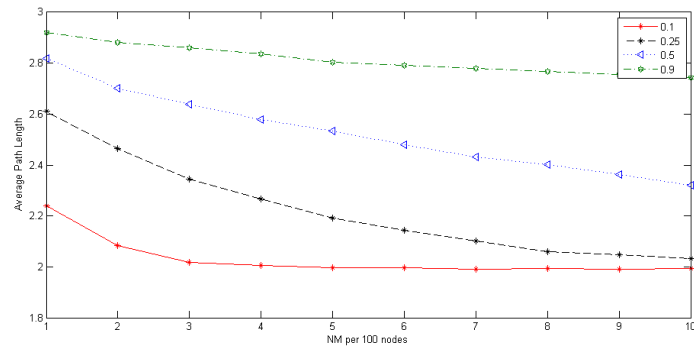
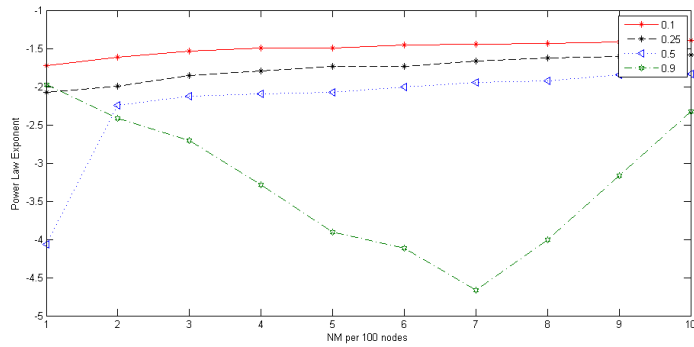
(a) Normalized Degree CF ($\beta = 1$)(b) Degree with added attribute similarity ($w = \beta = 0.5$)(c) Degree with multiplied attribute similarity ($\alpha = 1$)

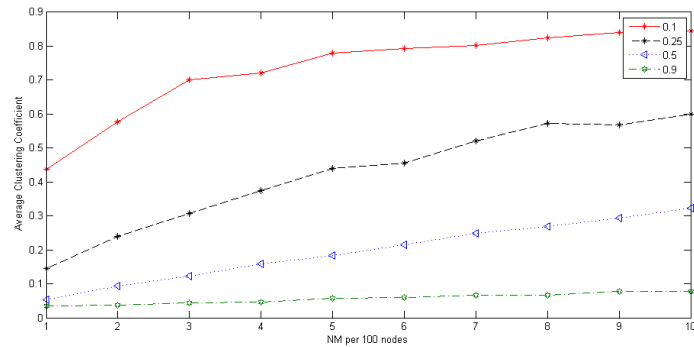
Fig. 6. Networks generated using SNAM algorithm with varying CF coefficient values



(a) Average Path length



(b) Power law Exponent of Degree distribution



(c) Average clustering coefficients

Fig. 7. SNAM algorithm with normalized degree CF and varying $\epsilon = 0.1, 0.25, 0.5$ and 0.9

within the CF. SNAM uses the CF values to connect the nodes. The CF depends on the old node degree simultaneously with the attribute similarity between new node and old node. SNAM is the first model that has new arriving nodes having different connection-standard requirements. SNAM proved to be very promising as it generated a network that had a PL degree distribution, small average path length and high clustering coefficient values. Simulation parameters as NM and ε can be tuned to generate a variety of complex networks with specific values of the clustering coefficient, the average path length, and the PL exponent. The effect of using Eigen vector centrality instead of degree centrality on the emergence of community structure in SNAM is still to be examined in the future work. We are also working on implementing an algorithm to SNAM that would result in the emergence of community structure. Implementing an analytical model for SNAM is also part of our future work. SNAM is general and hence can be used to generate any type of complex networks. As a proof of concept, we will consider a case study where we will apply these models to online social networks. Our choice of online social networks is mainly due to their wide spread and their currently excessive applications in fields such as marketing, information diffusion, recommendation, and trust analysis. We believe that our model will be useful in studying online social networks and mimicking their structure and dynamics.

References

- [1] X. F. Wang and G. Chen, "Complex networks: small-world, scale-free and beyond," *Circuits and Systems Magazine, IEEE*, vol. 3, no. 1, pp. 6–20, 2003.
- [2] A. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, pp. 509–512, 1999.
- [3] M. E. J. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, pp. 167–256, 2003.
- [4] R. Albert and A. Barabási, "Statistical mechanics of complex networks," *Reviews of Modern Physics*, vol. 74, pp. 47–97, 2002.
- [5] E. Ferrara, *Mining and Analysis of Online Social Networks*. PhD thesis, Department of Mathematics, University of Messina, 2012.
- [6] G. Bianconi and A. Barabási, "Competition and multiscaling in evolving networks," *Europhysics Letters*, vol. 54, pp. 436–442, 2001.
- [7] S. Tao and X. Yue, "The attributes similar-degree of complex networks," vol. 3, pp. 531–535, 2010.
- [8] Y. Li, X. Jin, F. Kong, and J. Li, "Linking via social similarity: The emergence of community structure in scale-free network," in *1st IEEE Symposium on Web Society (SWS09)*, pp. 23–24, 2009.
- [9] <http://www.mathworks.com/products/matlab/>.

- [10] J. M. Kleinberg, S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, “The web as a graph: Measurements, models and methods,” in *Lecture Notes in Computer Science 1627: Proceedings of the International Conference on Combinatorics and Computing*, pp. 1–18, Springer, 1999.
- [11] P. L. Krapivsky and S. Redner, “Organization of growing random networks,” *Physical Review E*, vol. 63, p. 066123, 2001.
- [12] B. Youssef and H. Hassan, “IASM: An integrated attribute similarity for complex networks generation,” in *Proceedings of the International Conference on Information Networking (ICOIN)*, 2014.
- [13] <http://vlado.fmf.uni-lj.si/pub/networks/pajek/>.

COEL: A Web-based Chemistry Simulation Framework

Peter Banda¹, Drew Blount², and Christof Teuscher³

¹ Department of Computer Science, Portland State University,
OR, USA, banda@pdx.edu

² Artificial Life Lab, Reed College, OR, USA
dblount@reed.edu

³ Department of Electrical and Computer Engineering,
Portland State University, OR, USA, teuscher@pdx.edu

Abstract. The chemical reaction network (CRN) is a widely used formalism to describe macroscopic behavior of chemical systems. Available tools for CRN modelling and simulation require local access, installation, and often involve local file storage, which is susceptible to loss, lacks searchable structure, and does not support concurrency. Furthermore, simulations are often single-threaded, and user interfaces are non-trivial to use. Therefore there are significant hurdles to conducting efficient and collaborative chemical research.

In this paper, we introduce a new enterprise chemistry simulation framework, COEL, which addresses these issues. COEL is the first web-based framework of its kind. A visually pleasing and intuitive user interface, simulations that run on a large computational grid, reliable database storage, and transactional services make COEL ideal for collaborative research and education.

COEL's most prominent features include ODE-based simulations of chemical reaction networks and multicompartment reaction networks, with rich options for user interactions with those networks. COEL provides DNA-strand displacement transformations and visualization (and is to our knowledge the first CRN framework to do so), GA optimization of rate constants, expression validation, an application-wide plotting engine, and SBML/Octave/Matlab export. We also present an overview of the underlying software and technologies employed and describe the main architectural decisions driving our development. COEL is available at coel-sim.org for selected research teams only. We plan to provide a part of COEL's functionality to the general public in the near future.

1 Introduction

The main motivation behind the development of the COEL framework is the often monotonous and low-level management of scientific models. Further, running simulations on multiple threads and CPUs requires non-trivial effort. Research avenues built on solid theoretical ideas often run into trouble because of a lack of appropriate tools and software, leading to unnecessary delays, implementation of proprietary (home-made) solutions for basic tasks and reinventions of standard design patterns. As is true with most desktop applications, most existing tools provide access to only a single user on a local machine, requiring version-management software to enable collaboration, and general usability and visual appeal are usually low priorities. We argue that the way we work and conduct research must dramatically change to keep pace with the amount of data produced by simulations, to provide immediate and integrated visualization, and to enable geographically dispersed teams to work together on a single platform.

In this paper we introduce the *Collective cELLular computing* (COEL) framework, the first web-based simulation framework for modeling and simulating chemical reaction networks (CRNs). COEL's web client is immediately accessible without any installation or download. The computational load of simulations is handled by COEL's grid rather than the client's machine. Remote teams can share and manipulate chemical models in real time. Data is stored remotely and safely in COEL's database, which is backed up daily. In developing COEL we emphasized platform-wide visualization, providing quick and embedded insight for users.

It is important to emphasize the significance of COEL's database storage. Even though raw file storage (as opposed to structured databases) has been obsolete in industry for more than two decades, the scientific community still widely practices this approach. Storing data in files is not only ineffective, but its textual representation requires cumbersome parsing and tedious serialization for later structured searches or data mining. More so, files are inherently local, and without proper back-up, it is not uncommon that scientific data are lost. A recent study by Vines et al. in *Current Biology* [1] found that 80% of scientific data are lost within two decades, disappearing into old email addresses and obsolete storage devices. Alarming, the authors found that the average rate of data loss is 17% each year. Furthermore, because of private and local storing only 11% of the academic research in the literature was reproducible by the original research groups, as reported in *Nature* [2]. This is intuitively more prevalent in experimental science, but computer-based research is affected as well. We suggest that with current scientific ap-

proaches this problem will only worsen in the age of big data. We argue that storing all (even intermediate) models and results remotely and in a reliable long-term fashion, and making them accessible to the general scientific community should become the new standard. With remote data storage and a convenient web client, users do not have to deal with version-compatibility of data structures, as it is the case with traditional approaches. Since a new application release is deployed together with a central migration of the database, version updates are worry-free for users.

Accessibility has two important consequences: collaboration and transparency. Using COEL, as with so-called ‘cloud-based’ web applications, individuals can work on different facets of the same project and see each other’s modifications in real-time. This has allowed the authors of this paper, for example, to study the same system, run parameter evolutions and performance evaluations, modify simulation dynamics and so on from separate campuses.

COEL has been developed as a part of the NSF project “Computing with Biomolecules”. We have successfully applied COEL as a sole tool to model and evaluate various types of chemical perceptrons [3–5], chemical delay lines and time-series learners [6, 7], and random DNA circuits [8].

In this paper we first discuss the state-of-the-art in chemistry simulation frameworks (Section 2), then present COEL’s functionality (Section 3) and technical architecture (Section 4). We conclude with a discussion of COEL’s place in the ecosystem of chemistry simulation frameworks, and the future of COEL (Section 5).

2 Related Work

COEL is not the first software made to simulate chemical reaction networks. There are already many programs which do so, and together the field of CRN simulators [9–13] offers a huge set of technical features, e.g., simulation options and statistical tools. Our goal with COEL was not (so much) to introduce new simulation algorithms or methods of analysis, but to include the most common and useful tools among CRN simulators in an intuitive and modern web-based package. This makes the tools of systems biology more accessible, and the research done with them more transparent, collaborative, and replicable.

COPASI [9] is arguably the most advanced and widely used tool. In a nutshell, COPASI simulates a variety of chemical objects and allows for freedom in experiment design and statistical analysis. COPASI is quite feature rich, and could be considered the gold standard of CRN simulation frameworks. There are others worth mentioning, of course, such

as those in the MATLAB Systems Biology Toolbox [11], and CellDesigner [12], which is a modeling tool for biochemical networks. Most of these tools share support for the SBML language for describing chemical systems [13], which as a standard has been a great boon to the field, enabling cross-platform migration.

Along with SBML support, most simulation environments share a core set of capabilities. Beyond basic deterministic ODE integration of CRNs (and stochastic reactions, a feature which COEL notably does not have), it is common to offer parameter optimization to help in the design of the networks themselves. Programs such as COPASI and CellDesigner can simulate a number of other biochemical objects of interest, such as cellular compartments. It is common to allow for various kinetic models of chemical interactions, such as Michaelis-Menten [14] and mass action [15].

In many kinds of frameworks, there is some tension between the depth of features and the features' accessibility, especially for highly technical applications such as CRN simulators. In addition to offering rich design capabilities, many developers of CRN simulators have the explicit motivation of reaching a large audience: The authors of COPASI said, "... the software needs to be available for the majority of scientists ..." (p. 3069, [9]). The authors of CellDesigner felt similarly, saying that they wish to "confer benefits to as many users as possible" (p. 1255, [12]). COEL automatically runs on any operating system with a web browser, including smartphones or tablets, so it is accessible anywhere in the world without any installation. Further, COEL's computational grid centrally runs any difficult tasks which might run slowly on clients' computers. We strongly believe that there is no more accessible paradigm for research tools than a web-based interface with computation performed in the cloud.

3 Features and Functionality

COEL provides a unified web environment for the definition, manipulation, and simulation of chemical reaction networks. In this section, we will discuss COEL's functionality and application-wide features in detail.

3.1 Chemical Reaction Network Definition

At its most basic level, a chemical reaction network (CRN) consists of a finite set of chemicals and reactions. A CRN represents an unstructured macroscopic simulated chemistry, hence the species labeled with symbols

are not assigned a molecular structure. The state of a CRN is represented by a vector of chemical species concentrations.

Each reaction is of the form $a_1X_1 + \dots + a_nX_n \rightarrow b_1Y_1 + \dots + b_mY_m$, where species X_i are reactants and Y_i products. Constants a_i and b_i are stoichiometric factors, i.e., positive integers describing how many copies of each molecule are involved in the reaction. For instance the reaction $A + B \rightarrow C$ describes species A and B binding together to form species C . Reactions can also involve catalysts or inhibitors, which speed up or slow down the reaction, but are not consumed.

Note that a legal reaction could have no reactants or no products. For that purpose we include a special no-species symbol λ to represent a formal annihilation $A + B \rightarrow \lambda$ or a decay $A \rightarrow \lambda$. Mass conservation states that matter cannot be destroyed nor created, i.e., in a closed system the matter consumed and produced by each reaction is the same. Annihilation and decay as we defined them seem to violate that, however, in the chemical analogy, λ does not signify a disappearance of matter but simply an inert species, effectively absent from the system of chemical interactions. Similarly we interpret a reaction $\lambda \rightarrow A$ as an influx of A rather than a creation of a molecule A from nothing.

Reaction rates define the strength or speed of reactions, as prescribed by kinetic laws—Michaelis-Menten [16] kinetics for catalytic reactions, and mass action kinetics [17] otherwise. The rate of an ordinary reaction $a_1S_1 + a_2S_2 \rightarrow P$ is defined by the mass-action law as

$$r = \frac{d[P]}{dt} = -\frac{1}{a_1} \frac{d[S_1]}{dt} = -\frac{1}{a_2} \frac{d[S_2]}{dt} = k[S_1]^{a_1}[S_2]^{a_2},$$

where $k \in \mathbb{R}^+$ is a reaction rate constant, a_1 and a_2 are stoichiometric constants, $[S_1]$ and $[S_2]$ are concentrations of reactants (substrates) S_1 and S_2 , and $[P]$ is a concentration of product P . The rate of a catalytic reaction $S \xrightarrow{E} P$, where a substrate S transforms to a product P with a catalyst E , whose concentration increases the reaction rate, is given by Michaelis-Menten kinetics as

$$r = \frac{d[P]}{dt} = \frac{k_{cat}[E][S]}{K_m + [S]},$$

where $k_{cat}, K_m \in \mathbb{R}^+$ are rate constants.

COEL is consistent with these general CRN formalisms; next, we will describe details particular to COEL's implementation. COEL automatically computes appropriate rate functions once given numeric rate constants, yet it also allows users to define arbitrary rate functions using custom expressions over species labels, giving the user full freedom over

Species																		
X	X1C	X1signal	X2signal	Y_aux	B	Sin	Sout	W+	W-	W0	W1	W1-	W2	W2-	X1	X2	Y	+
Reactions																		
Do	Label	Group	Reaction	Forward Rate	Catalysts	Inhibitors				+ New reaction								
	DL01		$X \rightarrow X1 + X1C$	$0.0225 * X1signal * X / (0.0020 + X)$	X1signal													
	DL02		$X1C \rightarrow X2$	$2.0000 * X2signal * X1C / (0.0706 + X1C)$	X2signal													
	DL03		$X2signal \rightarrow X1signal$	$1.3648 * X2signal$														
	DL04		$X1signal \rightarrow$	$0.0039 * X1signal$														
	R01	RG 1	$Sin + Y \rightarrow$	$0.4584 * Sin * Y$														
	R02	RG 2	$Sin \rightarrow Y$	$0.4459 * W0 * Sin / (1.8066 + Sin)$	W0													
	R03	RG 3	$X1 + Y \rightarrow$	$0.0203 * Y * X1$														
	R04	RG 4	$X1 \rightarrow Y$	$0.0378 * W1 * X1 / (2.5665 + X1)$	W1													
	R05	RG 3	$X2 + Y \rightarrow$	$0.0203 * Y * X2$														
	R06	RG 4	$X2 \rightarrow Y$	$0.0378 * W2 * X2 / (2.5665 + X2)$	W2													

Fig. 1. A partial description of a chemical reaction network in COEL. Species are listed at the top, and their reactions are presented in tabular form. The reactants and products are described in the third column, the forward reaction rates are in the fourth column, and any catalysts are in the fifth.

the system's dynamics. Reactions can be uni- or bidirectional, and bidirectional reactions can have independent forward and backward rates.

Both species sets and reaction sets are extensible, in that new sets can be defined as expansions of old ones. This promotes reuse and modular design. Further, two CRNs can be merged combining their reactions and species into one network.

Figure 1 shows an example CRN in COEL, a memory-enabled chemical perceptron [6]. The CRN's species, reactions, and reaction rates are presented in a unified view from which any of these objects can be easily edited in a few steps. Also, users can export CRNs in Matlab, Octave, or SMBL formats if they wish to study their systems using different tools. It is also possible to import an SBML-defined CRN into COEL.

In imitation of biochemical cells or membranes, CRNs in COEL support hierarchical tree-like compartmentalization. Each compartment hosts an independent reaction set and vector of chemical concentrations. Compartments communicate with each other through permeation, formalized in what we call 'channels.' A channel works just like an ordinary reaction, except the reactant and product species reside in adjacent compartments. Among other things, this allows for modular design of chemical systems, where connected modules reside in nested compartments, as shown in Figures 2 and 3.

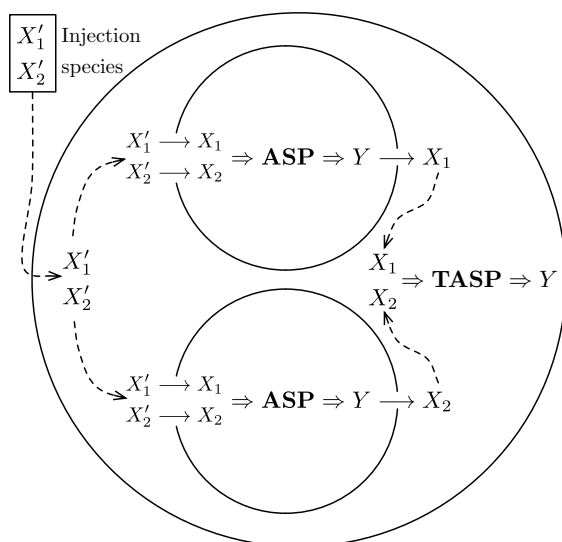


Fig. 2. Schematic of permeation in a simple 2-1 multicompartment system from one of the authors' current projects. The 'tagged' input species X'_1 and X'_2 are injected into the outer compartment. They permeate into the inner compartments via channels which transform them into regular, untagged input species X_1 and X_2 . The inner compartments' ASPs (Asymmetric Signal Perceptrons [4], each of which is a large CRN) process the input species into the output Y . Each compartment has a unique outgoing channel to transform Y into one of the input species, which are then processed in the outer compartment.

Order	Id	Label	Channels	Permeability	Disassociate?
0	3574	Example subcompartment 1	<ul style="list-style-type: none"> • 1164 : $X_1 \leftarrow X'_1$ • 1165 : $X_2 \leftarrow X'_2$ • 1169 : $Y \rightarrow X_1$ 	<ul style="list-style-type: none"> • 0.3388 • 0.3388 • 0.4387 	→
1	3575	Example subcompartment 2	<ul style="list-style-type: none"> • 1171 : $X_1 \leftarrow X'_1$ • 1172 : $X_2 \leftarrow X'_2$ • 1176 : $Y \rightarrow X_2$ 	<ul style="list-style-type: none"> • 0.3388 • 0.3388 • 0.4387 	→

Fig. 3. COEL's representation of the permeation schema depicted in Figure 2.













Do	Start Time	Time Length	Cache Write/ Species Action
  	0	0	<ul style="list-style-type: none"> • $3 \rightarrow IN$ • $W0 \leftarrow \text{random}(0.5, 1.5)$ • $W1 \leftarrow \text{random}(0.5, 1.5)$ • $W2 \leftarrow \text{random}(0.5, 1.5)$
  	100	0	<ul style="list-style-type: none"> • $(\text{rand}() < 0.5) * IN \rightarrow X1_inj$ • $(\text{rand}() < 0.5) * IN \rightarrow X2_inj$ • $Sin' \leftarrow IN$ • $X1' \leftarrow X1_inj$ • $X2' \leftarrow X2_inj$ • $Y \leftarrow 0$
  	120	0	<ul style="list-style-type: none"> • $Sin \leftarrow 1.5$
  	300	0	<ul style="list-style-type: none"> • $B \leftarrow 0.5 * (Y > 0.5 \text{ \&not; } (X1_inj \text{ \&not; } 0 \text{ \&or; } X2_inj \text{ \&not; } 0))$

Fig. 4. The details of a COEL interaction series. Left arrows denote the setting of species concentrations, and right arrows indicate assignments of user-defined variables. The interaction at time 100 does the following (note that at time 0 the variable IN is set to 3): first, the variables $X1_{inj}$ and $X2_{inj}$ are randomly set to 0 or 3 with equal probability. The concentration of Sin' is set to 3, then the concentrations of $X1'$ and $X2'$ are set equal to their respective injection variables. Finally, Y is flushed from the system—its concentration is set to 0.

3.2 Chemical Reaction Network Simulation and Interaction Series

A major feature of COEL, in that it has been crucial to its early users and their work, is so-called interaction series. An interaction series allows the user to directly manipulate concentrations of species in the CRN. This feature is analogous to, though more capable than, automatic chemical injections into a reaction chamber. For compartment-extended CRNs, interaction series can be identically hierarchical, allowing for precise interaction with each component of the network.

Concentrations can be modified multiple times, not just initially. E.g., for iterative processes it is useful to define a set of periodic interactions. In specifying interactions, a user can define custom concentration-setting expressions, as well as custom variables for use in those expressions. For example, the bottommost interaction in Figure 4 injects species B (here a ‘penalty species’) at concentration 0.5 if the output species Y does not match **AND** of the original input concentrations, $X1_{inj}$ and $X2_{inj}$. The COEL Interaction Series API, as we call it, is then a scripted language

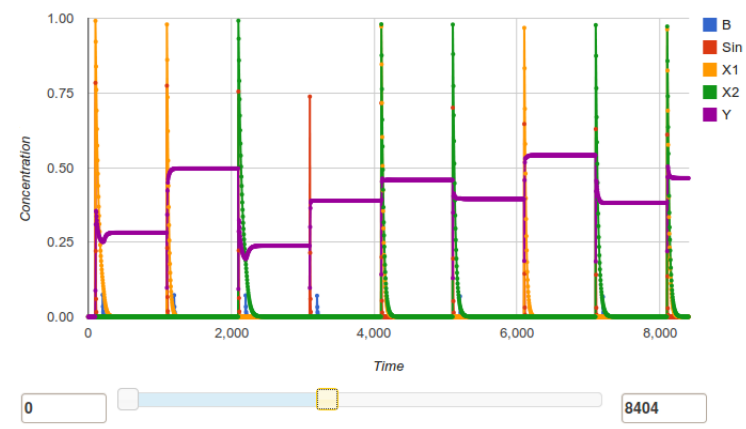


Fig. 5. A chart showing concentration traces of 5 chemical species over time in COEL. In this case, an interaction series injects a random combination of $X1$ and $X2$ at concentration 1, every 1000 time steps.

that can describe a variety of complicated experimental scenarios without touching the underlying simulation-framework code. Thus end users have the freedom to manipulate the chemical system in a dynamic and safe way (basic expression validation is provided).

To actually simulate a CRN, a user runs a defined reaction network with a selected interaction series (which might be as simple as setting initial concentrations). Users can choose from a number of non-adaptive and adaptive deterministic ODE solvers to integrate their system. Upon running such a simulation, the user is by default shown an embedded chart of species concentrations over time (Figure 5). If further post-processing is required, full or filtered data could be easily exported into a CSV file.

Note that since ODE solvers are deterministic, two simulations using the same CRN and interaction series will always produce the same concentration traces if the interaction series is deterministic. That is, however, not the case for the interaction series in Figure 4, which uses random weight setting and randomly injects binary inputs at concentration 0 or 3. COEL does not currently have a feature to save random number seeds to exactly replicate simulations such as these.

3.3 Performance Evaluation and Dynamics Analysis

COEL provides a core set of tools for analyzing and modifying CRNs, enabling statistical record-keeping as well as the design of complex net-

works whose precise architecture is initially unknown to the user. COEL’s basic interpretive tool is the “translation series,” defined by the user in a similar manner to interaction series, described above. A single translation is a straightforward function of the current concentrations and any predefined constants, and can be Boolean or numeric in its output.

One can simply plot the output of a translation series to see the CRN’s behavior through a certain lens, or use the series as the basis of evaluation and optimization. Because many CRNs involve a random component, especially in (but not limited to) their interaction series, COEL allows the user to run large batches of simulations and collect statistics based on these translation series.

Because it is usually difficult to precisely translate simulated chemistries into wet ones, COEL also offers perturbation analysis. Users can evaluate the performance of the CRN if a defined set of rates are randomly perturbed according to set parameters. This is useful in measuring the robustness of a chemical system.

COEL also offers dynamics analyses with a detailed statistical view of an individual CRN simulation. This includes Lyapunov exponents, Derrida stability, time and spatial nonlinearity errors, and more; along with reports about the simulation itself, like how many species concentrations reached fixed points for given tolerance.

To allow maximum freedom in analysis, COEL offers CSV export of any raw data a user might produce. Every chart and data visualization in COEL is accompanied by a CSV export function, allowing the user to export either the data currently displayed on-screen (to replicate a chart or precisely modify its appearance) or the entire raw dataset, as shown in Figure 6.

3.4 Rate Constant Optimization

With defined evaluation criteria, a user can optimize CRN’s parameters with COEL’s flexible genetic algorithm tool. Users define the space to be optimized by selecting which reaction and channel permeation rates are to be modified, in what ranges, and under what constraints (e.g. several reaction rates can be fixed to each other). Chromosomes are then vectors of rate constants.

The parameters of COEL’s GAs are easily modified, allowing for different rates of mutation, rules of reproduction, initial populations, and so on. Chromosomes can be selected to reproduce either deterministically with elite selection, or probabilistically relative the measured fitness of each chromosome. Reproduction can be sexual or asexual. In the former case, crossover between two chromosomes can be either one-point (i.e., in chromosomes of length n , the child’s first $p \leq n$ genes are from one parent

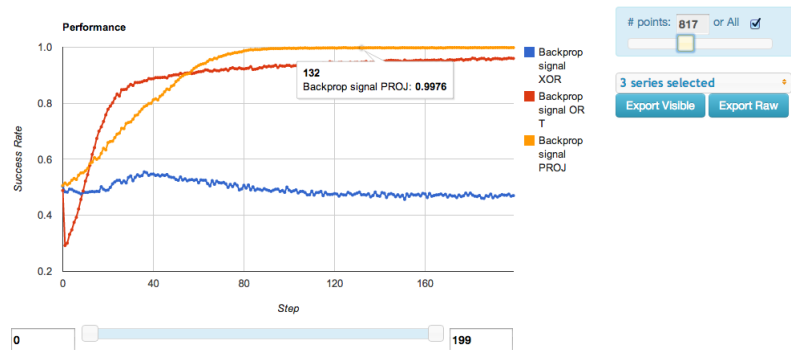


Fig. 6. A chart of three separate performance evaluations, each one showing the performance of a binary chemical perceptron averaged over 10,000 repetitions for given interaction series representing desired binary function (XOR, OR, PROJ). Note the data export options on the right.

and the last $n-p$ are from the other), or a probabilistic shuffle. Supported mutation types are one-bit, two-bit, exchange and per-bit, with content replacement and perturbation options. COEL's GAs also support fitness renormalization, and selection of maximization or minimization of the target function (fitness vs. error).

3.5 DNA Strand Visualization and Displacement Reactions

COEL has a convenient web interface for visualizing DNA strands specified by the Microsoft Visual DSD syntax [18], which decomposes single and (full or partial) double DNA strands into labeled subsequences called domains. Domains are classified as either long or short, also called toe-holds. These DNA-strand images can be exported in the svg format, appropriate for publications and educational purposes alike. Note that the Microsoft Visual DSD web tool (unlike COEL) requires an installation of Microsoft Silverlight, whose support on Linux is problematic.

Furthermore, COEL can transform any CRN based on mass-action kinetics into a DNA strand-displacement circuit using the methods of Soloveichik et al. [19]. In strand displacement systems, populations of these species are typically represented by the populations of single stranded DNA molecules. These interact with double-stranded gate complexes which mediate transformations between free signals. In a nutshell, the mass-action reaction $X_1 + X_2 \rightarrow X_3$ is translated to three displacement reactions $X_1 + L \rightleftharpoons H + B$ (a single strand X_1 displaces an upper strand B from the complex L), $X_2 + H \rightarrow O + W_1$ (a single strand X_2 displaces

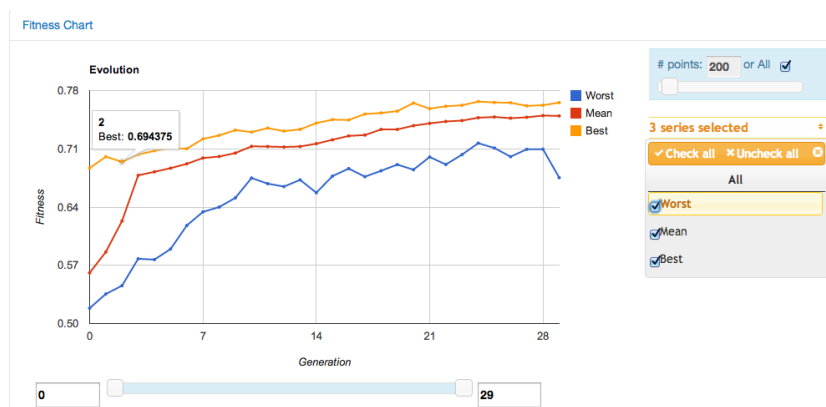


Fig. 7. A chart of a population's fitness over time in a run of a particular GA. This plot displays several features shared by all plots in COEL, enabling modification of the plot without refreshing the web page: an x-axis slider to specify the plot's domain, a drop-down menu to select which series to display, and a slider to select the plot's resolution relative the data set.

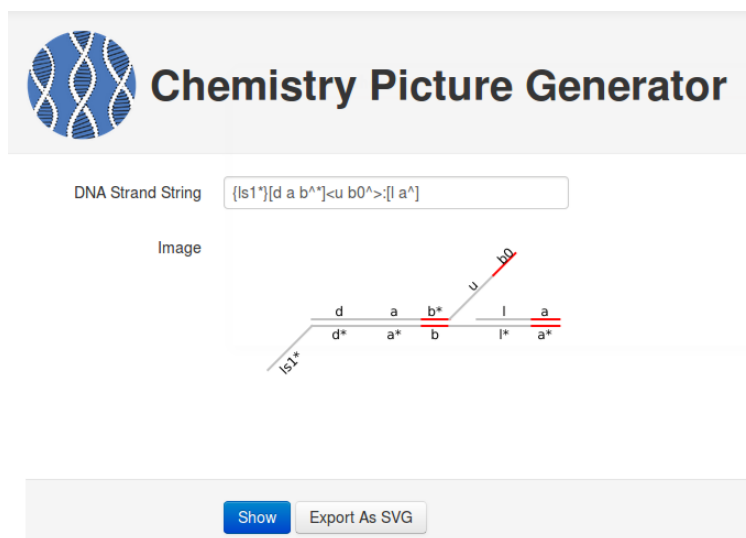


Fig. 8. COEL's tool for visualizing DNA strands specified in Visual DSD. Red lines represent toeholds, and gray lines are long domains.

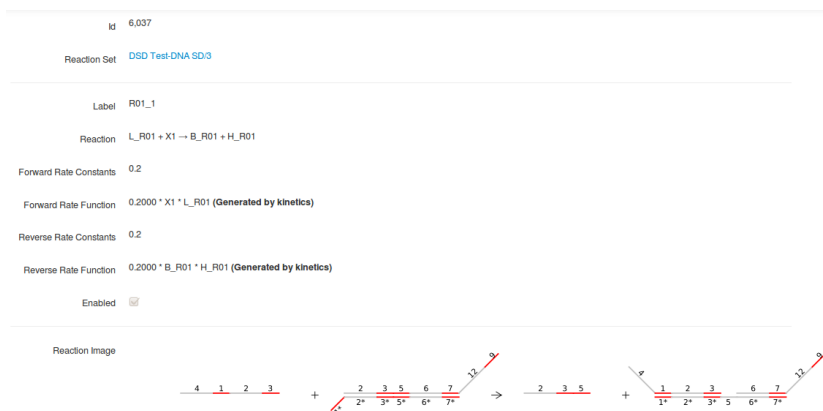


Fig. 9. A DNA strand displacement reaction obtained by COEL’s transformation of arbitrary CRNs into strand displacement circuits.

an upper strand O from the complex H), and finally $O + T \rightarrow X_3 + W_2$ (a single strand O displaces an upper strand X_3 from the complex T), where L, H, B, O, T, H are auxiliary fuel species, and W_1 and W_2 are waste products.

Once applied to a reaction set, the transformation produces a CRN with new intermediate species and reactions, describing displacements of single strands from partial or full double strands. Besides new reactions, COEL also specifies the DNA structure of each species in terms of numerically-labeled domains, the output of which is shown in Figure 9. This is a powerful tool for automatic translation of so-called *in silico* systems to feasible wet chemistries in a user-friendly way. The authors are not aware of any other CRN simulation framework that includes DNA strand displacement transformations as a part of their application toolbox.

3.6 Random Chemical Reaction Network

COEL offers functionality to quickly make a random chemical reaction network with set specifications. User-defined parameters include the number of species, the number of reactions, the number of reactants and products in each reactions, and a random distribution of reaction constants; COEL meets all of these constraints with combinatorial design. For open systems the user can also specify influx and efflux constraints.

Furthermore, COEL also supports generation of random DNA-stand circuits [8] using single, full double, and partial double strands. Parame-

ters for this function include number of single strands, ratio of upper to lower strands, ratio of upper strands with complements, (positive) normal distribution of partial double strands per upper strand, (positive) normal distribution of rate constants, ratio of influxes and effluxes, and distribution of rate constants. Based on a randomly generated ordering, DNA strands with higher order take precedence over lower-order strands in DNA-strand displacement reactions (Section 3.5). Also, note that the maximum number of strands that could bind together is two, which is justified by assuming that a single strand does not bind to partial double strand, but always displace its upper or lower part. We assume wet synthesis of these networks is possible by standard DNA sequence design [20].

3.7 Platform-wide Features

Numerous features of COEL are omnipresent throughout the platform, creating a familiar look-and-feel as well as providing intuitive access to common features. Throughout COEL, users input mathematical functions in the straightforward syntax of the Java Expression Parser (displayed in Figure 4), and those expressions are always validated by COEL before being input into any simulation. Views, such as COEL's list of reaction sets or interaction series, have a common search and filter feature, allowing for easy navigation through huge sets of objects.

All charts in COEL are made with the Google Charts API, and include sliders for domain selection and data filtering (see Figure 7), as well as CSV export options (see Figure 6). Finally, COEL has rudimentary user privacy protocols, where each user account is either a 'user' who can see only his/her own projects, or an 'admin' who can see every project on COEL. In order to share a project, a group of users currently have to have admin rights. We plan to expand privacy features in later versions.

4 Architecture and Technology

COEL's architecture is highly modular with strict separation of business logic and technological aspects. Nowadays, the main challenge of enterprise application development is not programming per se but rather the integration of diverse technologies and libraries which each address different application needs. The absence of strict inter-modular / inter-layer dependencies enables quick and easy customization and replacement of technologies and providers.

Table 1. A list of the acronyms used in this section.

Acronym	Description
JVM	Java Virtual Machine
ORM	Object-Relational Mapping
POJO	Plain Java Object
DAO	Data-Access Object
IoC	Inversion of Control
JEP	Java Expression Parser
JMS	Java Message Service
REST	Representational State Transfer
HPC	High Performance Computing
JDBC	Java Database Connectivity
SQL	Structured Query Language
PLSQL	Procedural Language/Structured Query Language
HQL	Hibernate Query Language

At this level of abstraction only the domain objects, the data holders of business data, implemented as POJOs (Plain Java Objects), are shared among all application parts and layers. Figure 10 presents a high-level overview of COEL’s architecture with call (request) pathways. On the very top we have two clients representing the only entry points to the application: the web client backed by Grails [21], jQuery [22] and Bootstrap [23] frameworks (discussed in Section 4.4), and the plain console client implemented in standard Java for “headless” scripting.

Based on user’s requests, the clients call the services such as `ChemistryService`, `EvolutionService`, and `UserManagementService` (Section 4.2) maintained by the Spring application container (Section 4.1), which then redirects either to a computational grid implemented on the top of GridGain HPC technology [24] (Section 4.3) for distributed task execution, or to the persistence layer with DAOs (Data-Access Objects) and ORM (Object-Relation Mapping) provided by Hibernate [25] (Section 4.5). In addition, the web client controllers have a direct link to the persistence layer, which is beneficial especially for basic CRUD (Create, Read, Update, Delete) operations. At the very bottom a PostgreSQL [26] database stores and provides data on the demand of the persistence layer.

The business logic such as chemistry simulation and GA optimization is implemented mainly in the Scala language, leveraging both object-oriented and functional programming approaches. All technologies and libraries integrated into COEL are either open-source or free to use.

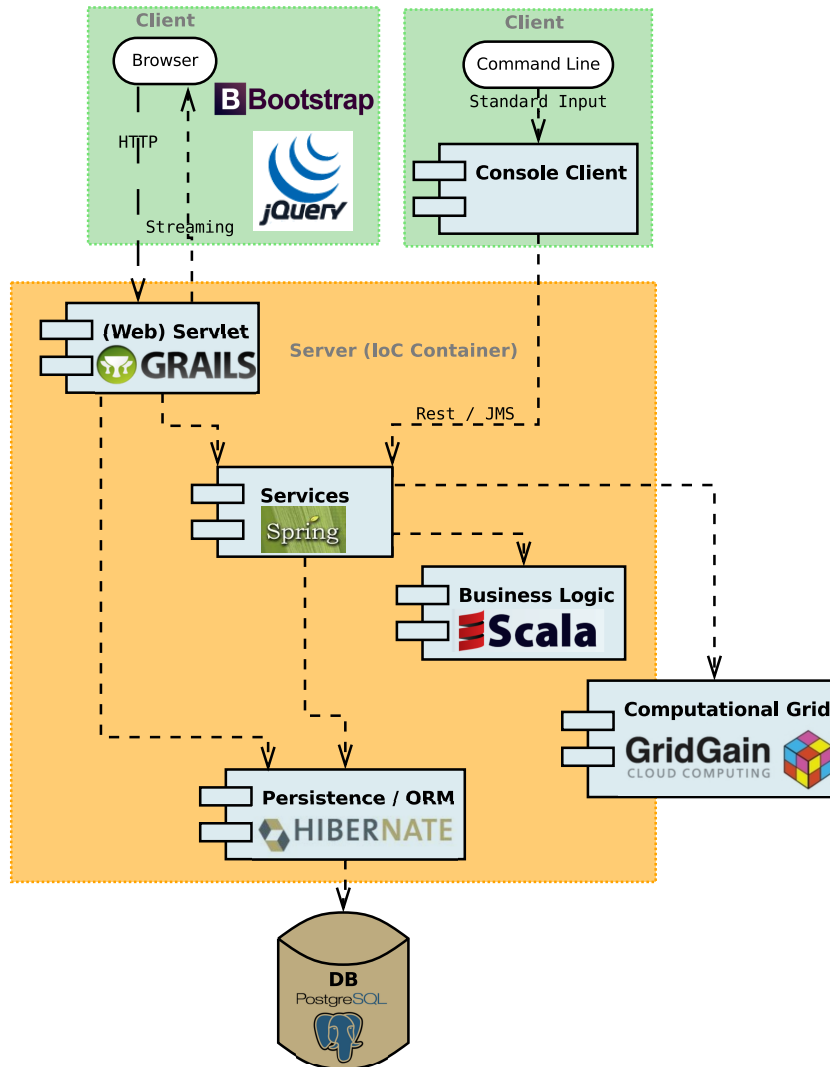


Fig. 10. A high-level overview of COEL's architecture consisting of web and console clients, web servlet, services, business logic, persistence layer, and computational grid. The application (IoC) container holding the server-side of the application is implemented in Spring framework.

4.1 Application Container

The Spring Framework [27, 28] provides the COEL’s core application infrastructure. Spring is a leading enterprise solution for Java maintained by the SpringSource community since 2002. Compared to Enterprise Java Beans, the Spring portfolio is less invasive and more flexible. Spring is not an application server, it is just a set of libraries which can be used and deployed anywhere (like e.g., Tomcat and Jetty). It consists of several sub-projects which can be used separately or together as needed. Spring is a lightweight tool that shows how little is really needed for enterprise application development. It does not have strict dependencies, and it detaches technical and business concerns.

The IoC (Inversion of Control) container is a central part of the Spring Framework. It controls the creation, number of instances (with singleton and prototype scopes), lifecycle, inter-dependencies (loose-coupling or wiring) and general configuration of application components, modules, adapters, specific utility classes or in general any POJO whose creation and use should be maintained in the application context. Spring IoC is a simple and transparent glue or integrator of various components and frameworks which are provided either by Spring Portfolio itself or other parties.

The IoC container encourages the best practices of programming with interfaces, i.e., each bean (POJO object in the IoC container) should consist of an interface and implementation class. Therefore, each bean knows that it can talk to a different bean that does something specific, but not which type of object, how its functionality is implemented, nor how the call is carried out. The IoC container injects the dependencies into POJOs at the runtime, and so beans take care only about their business purpose, not creation (and maintenance) of their relationships.

This approach is superior to the factory design pattern because all dependencies get injected and configured through the application container (annotations and/or XML), however beans are not aware of the container’s existence, i.e., unlike the factory pattern they do not need to call the application container in order to get their dependencies. The application code of Spring beans has little dependency on Spring itself. As a matter of fact, IoC is often described with the Hollywood principle: “Don’t call us, we call you.” Besides Spring, other popular IoC containers include GUICE and Pico.

IoC abstraction results in modular, lightweight and layered architecture with loose-coupled pluggable components. Programmers are also encouraged to implement beans as thread-safe and stateless if possible, so several callers could safely query the same component without worrying about timing and/or call history.

Last but not least, Spring IoC enables COEL to become a truly test-driven project. Because of loose-coupling and dependency injections, our JUnit tests could switch to test (rather than production) application context and substitute for instance implementation classes that require remote access to production systems with mock objects.

4.2 Services

The service layer is the actual gateway to the business/functional part of the application. Services are callable functions provided to the clients (or outside world). COEL is divided into five functional modules, each exposed by a separate service interface (facade): `ChemistryService`, `EvolutionService`, `NetworkService`, `AnalysisService` and `UserManagementService`.

One of the most compelling reasons to use Spring for service management is its comprehensive transaction support. Spring provides a consistent abstraction for transaction management that integrates very well with various data access abstractions. For remote access, the service interfaces can be easily injected by appropriate stubs. Spring supports for example Remote Method Invocation (RMI), Spring's HTTP invoker, JAX-RPC, JAX-WS or JMS.

Since the web client runs as a part of the application context, i.e., it lives inside the same server-side JVM (Java Virtual Machine) as Spring, all service calls are local. On the other side, the console client runs as a separate process and its calls are remote. More precisely, console clients requests are carried out by RESTful Web Services and alternatively by JMS. In the future we might consider exposing a portion of services to 3rd parties, possibly other universities or teams, through REST.

4.3 Cloud Computing

COEL's computational grid has been built on top of the GridGain In-Memory Computing Platform [24]. The GridGain HPC (High Performance Computing) library implements a scalable low-latency zero-deployment computational grid, which fits seamlessly into our Spring-backed IoC container (Section 4.1).

COEL's grid currently consists of 19 nodes with around 500 cores. All nodes are hosted on Portland State University hardware, though the technology allows us to add any geographically remote resource, since the communication is carried out by TCP/IP protocol with optimized marshaling (serialization) of exchanged data. We plan to utilize existing grid technology to pool the resources with other geographically dispersed teams.

COEL's grid acts transparently, as a single computing resource. GridGain enables COEL's users to be more productive by eliminating the complexity of distributed computing. Regardless of a user's geographic location, they can add tasks to the grid from the COEL web page without much effort. When a user submits a task, after the chain of calls the request is ultimately received by the grid master node running within the application context. The task splits into many partial jobs, which are then distributed over the grid.

GridGain provides zero-deployment technology, so a new (slave) node could be added to the grid on-the-fly by registering with the master node identified by the IP address or domain name. Therefore the grid's topology might change freely during its lifetime. COEL's grid supports several enterprise features contributing to effective and robust execution of jobs. The grid keeps track of various node statistics such as CPU performance, execution time, and availability, which are constantly updated and utilized for adaptive job distribution such that high performing nodes obtain more jobs. Also, if a node disconnects from the grid, the exception is noted by a periodic heartbeat, and disconnected node's jobs are redistributed across the grid. Moreover, if a node finishes its execution sooner than expected and so it sits idle (its wait queue is empty), it steals jobs from other nodes.

Due to the communication and task initialization overhead we execute only nontrivial tasks on the grid, with compute times that can last seconds, hours, or days. The main grid tasks include chemical ODE simulations, dynamics analyses, and evolutionary optimizations of rate constants.

4.4 Web Client

COEL's web client is implemented in Grails [21], which is a powerful web 2.0 framework using the Groovy dynamic language for the Java Virtual Machine. JVM compatibility means that Java, Groovy, and Scala source compiles into Java byte code, hence these three languages are natively inter-callable. Grails follows the "Convention over Configuration" approach, which emphasizes standard (conventional) naming, binding and data flow, so the structure of the application is simply implied if it is not explicitly configured. This approach is heavily utilized in a function called scaffolding, which based on a domain object structure generates dynamically at runtime the controller with associated web pages, providing basic CRUD operations without any effort. As a matter of fact, we could build a COEL prototype web client just with a few lines of code. Grails internally uses Spring IoC for dependency injection and bean cre-

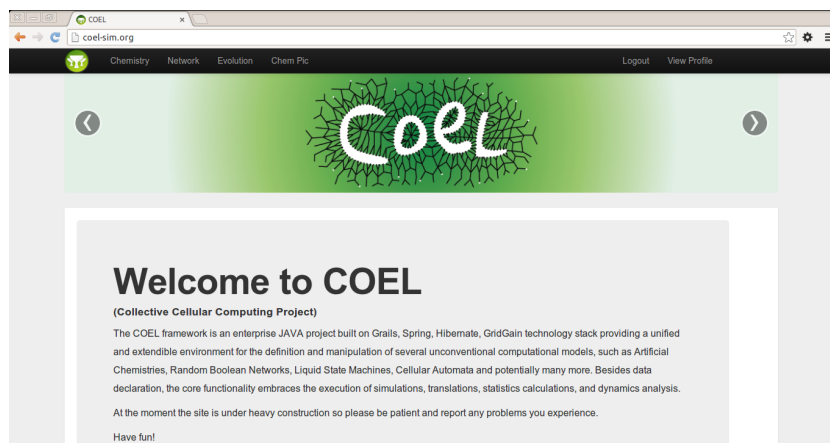


Fig. 11. COEL's home (welcome) page. URL: coel-sim.org.

ation. Furthermore, Grails was officially incorporated into Spring at the end of 2008.

The web front-end relies heavily on Javascript provided by the jQuery library [22], which makes UI interactive and intuitive and moves a part of data processing and visualization directly to the web browser. For instance, although COEL runs all simulations server-side, if a user wishes to see a chart, e.g., of species concentration traces, COEL sends the user raw data which is transformed into a chart by client-side Javascript using Google Charts API. For styling and some widgets we used the Bootstrap library [23] created by Twitter.

4.5 Persistence

The persistence layer consists of DAOs (Data-Access Objects) wrapping storing, retrieving, deleting, and filtering functionality for domain objects. To map an object-oriented domain model to a traditional relational database we use Hibernate [25], an object-relational mapping (ORM) library for the Java language. DAOs and Hibernate are widely supported by Spring, which offers hooks for fast integration.

Hibernate solves Object-Relational impedance mismatch by replacing direct persistence-related database accesses with high-level object handling functions. Hibernate provides declarative strategy for persisting data. We define a mapping of columns, reference metadata and inheritance strategy mapping. Hibernate handles details about persistence implementation, like SQL statements and JDBC connection creation. To

obtain data we use SQL or the Hibernate query language (HQL). The actual translation from the POJO to JDBC result set is automatic. Hibernate also uses various optimization strategies, such as cache and DB access optimization.

We believe that it is imperative to store data in a structured database, enabling prompt retrieval, searching and post-processing. PostgreSQL [26] is a mature open source database providing standard SQL/PLSQL language support with numerous additional features. The decision to select PostgreSQL as DB provider was driven mainly by the following factors: a lot of hands-on experience, a comprehensive console as well graphical UI (PgAdmin), an open source license, and support for array data types, useful for storing scientific vector data. The database model currently contains about 90 tables. To assure compatibility for each version of COEL we migrate data by a set of SQL scripts. Also, each day the whole database is dumped (backed-up), so we could restore the state of the DB to a certain date and time very quickly. That means our data is stored safely in structured and indexed format.

4.6 Build, Deploy, and Testing

To build COEL's project and to maintain its library dependencies, we use Apache Maven [29]. For a new application version we run a set of JUnit tests, which guarantee that the core functionality works as expected. After that, COEL is deployed to the Tomcat application server. Figure 12 shows a deployment schematic of COEL's components over several resources (machines), each running some part of the application: the database server, the application server, and the cloud. Due to the extendability of the computational cloud, the number of resources is not bounded. Also, note that the database server and the application server are currently hosted on the same machine.

COEL currently has about 30 users (exclusively from the NSF project "Computing with Biomolecules" and Portland State University), 5 of which are active, i.e., they access COEL on a daily basis. Once COEL will be available to the research community we expect the number of users to grow to hundreds, which would require more resources and more rigorous testing. If the users find a production issue or want to recommend a new feature, they will be able to submit a report through a Jira issue tracking system. More than 60 issues and new feature requests have been reported so far internally. Currently, the development of COEL is largely driven by the authors' research needs.

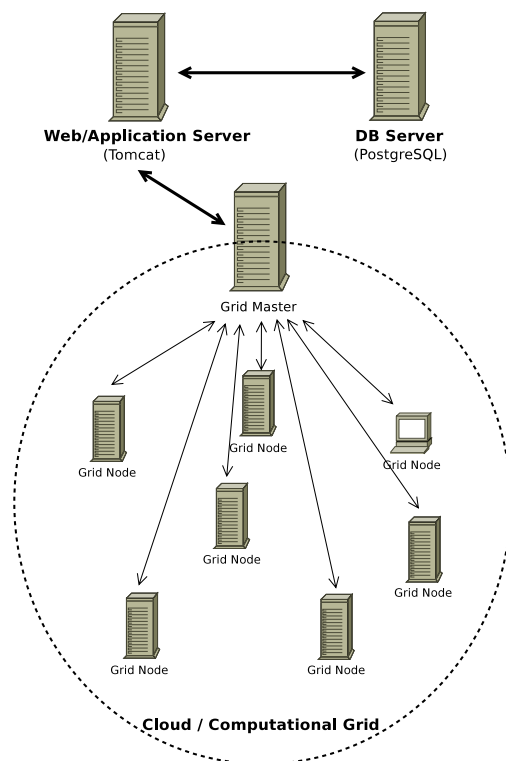


Fig. 12. Diagram showing a physical deployment of COEL's components.

5 Conclusion and Future Work

In this paper, we presented a new web-based chemistry simulation framework, COEL. Its modern layered architecture includes a scalable computational grid, a user-friendly and interactive web UI, and the safe and transactive persistence of chemistry models and simulation results. Its wide range of features primarily target chemistry simulations, GA optimization of rate constants, performance evaluations, and dynamics analysis. We paid particular attention to general usability and lightweight and fluid layout, and embedded data visualization using Google's charting engine.

COEL can be used without any installation, and from any web browser. As such, it is easier to start using and has a larger potential audience than existing desktop-application based frameworks. Keeping COEL in

the cloud allows for easy collaboration and sharing of results, and makes it simple to build upon another's work.

COEL's computational grid utilizes CPU resources only, however, it would be beneficial to extend the grid over GPUs as well. GridGain, our current computational grid library, does not provide native support for GPUs. On the other hand, we argue that reimplementing all tasks and business logic in (J)CUDA or OpenCL and maintaining two code branches would not be feasible. Therefore, we plan to explore transparent compilation mechanism such as Aparapi, where a single Java code compiles to CPU and GPU version transparently and gets executed based on resource availability.

Furthermore, we often face the situations when we want a newly submitted task to be executed as soon as possible, or we want to associate more CPU time to the tasks of a certain user. To achieve that we would like to assign priorities to the tasks based on their type and users' privileges.

As mentioned in Section 4.2 we might consider exposing certain services and routines through RestFul API so 3rd party applications could call, integrate and tailor COEL's functionality for their needs.

To improve the quality of chemistry ODE-based simulations we plan to integrate the standard LSODA solver. Also, to provide an alternative to the deterministic ODE solvers our goal is to introduce a stochastic simulator based on the Gillespie method [30]. The Gillespie method simulates each reaction step stochastically on a molecular level [31, 32]. It is computationally more demanding than ODE integration, however, it is physically more realistic, especially if the number of molecules in the system is low. Therefore, COEL is currently best-suited to simulate systems with large numbers of each chemical species.

Also, we plan to introduce more advanced sharing permissions, so each user could specify with which group or user he wants share the models and results for viewing and editing.

Last but not least, our vision for COEL is to become a common platform for diverse unconventional computing models. One step toward that goal is a new Network module, which will simulate complex spatial, random, or layered networks with configurable node functions and interaction series.

Acknowledgment

This material is based upon work supported by the National Science Foundation under grant no. 1028120. We acknowledge Avi Debnath for

his work on DNA-strand visualization, and GridGain for an academic license to use the GridGain HPC technology.

References

- [1] T. H. Vines, A. Y. K. Albert, R. L. Andrew, F. Débarre, D. G. Bock, M. T. Franklin, K. J. Gilbert, J.-S. Moore, S. Renault, and D. J. Rennison, “The availability of research data declines rapidly with article age.,” *Current biology*, vol. 24, pp. 94–7, Jan. 2014.
- [2] C. G. Begley and L. M. Ellis, “Drug development: Raise standards for preclinical cancer research.,” *Nature*, vol. 483, pp. 531–3, Mar. 2012.
- [3] P. Banda, C. Teuscher, and M. R. Lakin, “Online learning in a chemical perceptron,” *Artificial life*, vol. 19, no. 2, pp. 195–219, 2013.
- [4] P. Banda, C. Teuscher, and D. Stefanovic, “Training an asymmetric signal perceptron through reinforcement in an artificial chemistry,” *Journal of The Royal Society Interface*, vol. 11, no. 93, 2014.
- [5] P. Banda and C. Teuscher, “Learning two-input linear and nonlinear analog functions with a simple chemical system (in press),” in *Unconventional Computing and Natural Computing Conference* (O. H. Ibarra and L. Kari, eds.), vol. 8553 of *Lecture Notes in Computer Science*, pp. 14–26, Springer International Publishing Switzerland, 2014.
- [6] P. Banda and C. Teuscher, “An analog chemical circuit with parallel-accessible delay line learning temporal tasks (accepted),” *ALIFE 14: The Fourteenth International Conference on the Synthesis and Simulation of Living Systems*, 2014.
- [7] J. Moles, P. Banda, and C. Teuscher, “Delay line as a chemical reaction network (accepted),” *Parallel Processing Letters*, 2014.
- [8] P. Banda and C. Teuscher, “Complex dynamics in random DNA strand circuits (extended abstract),” in *The 19th International Conference on DNA Computing and Molecular Programming*, 2013.
- [9] S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, and U. Kummer, “COPASI—a COMplex PATHway SIMulator.,” *Bioinformatics (Oxford, England)*, vol. 22, pp. 3067–74, Dec. 2006.
- [10] A. Castellini and V. Manca, “Metaplab: a computational framework for metabolic p systems,” in *Membrane Computing*, pp. 157–168, Springer, 2009.
- [11] H. Schmidt and M. Jirstrand, “Systems Biology Toolbox for MATLAB: a computational platform for research in systems biology.,” *Bioinformatics (Oxford, England)*, vol. 22, pp. 514–5, Feb. 2006.
- [12] A. Funahashi, Y. Matsuoka, A. Jouraku, M. Morohashi, N. Kikuchi, and H. Kitano, “CellDesigner 3.5: A Versatile Modeling Tool for Biochemical Networks,” *Proceedings of the IEEE*, vol. 96, pp. 1254–1265, Aug. 2008.
- [13] M. Hucka, a. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, a. P. Arkin, B. J. Bornstein, D. Bray, a. Cornish-Bowden, a. a. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley,

- T. C. Hodgman, J.-H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, a. Kremling, U. Kummer, N. Le Novere, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, and J. Wang, "The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models," *Bioinformatics*, vol. 19, pp. 524–531, Mar. 2003.
- [14] L. Michaelis and M. L. Menten, "Die Kinetik der Invertinwirkung," *Biochem. Z.*, vol. 49, pp. 333–369, 1913.
- [15] A. J. Lotka, "Undamped oscillations derived from the law of mass action.," *Journal of the american chemical society*, vol. 42, no. 8, pp. 1595–1599, 1920.
- [16] R. A. Copeland, *Enzymes: A practical introduction to structure, mechanism, and data analysis*. New York, New York: John Wiley & Sons, Inc., second ed., 2002.
- [17] J. Espenson, *Chemical kinetics and reaction mechanisms*. Singapore: McGraw-Hill, 1995.
- [18] M. R. Lakin, S. Youssef, L. Cardelli, and A. Phillips, "Abstractions for DNA circuit design.," *Journal of the Royal Society, Interface*, vol. 9, pp. 470–86, Mar. 2012.
- [19] D. Soloveichik, G. Seelig, and E. Winfree, "DNA as a universal substrate for chemical kinetics.," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 107, pp. 5393–5398, Mar. 2010.
- [20] J. N. Zadeh, C. D. Steenberg, J. S. Bois, B. R. Wolfe, M. B. Pierce, A. R. Khan, R. M. Dirks, and N. A. Pierce, "Software News and Updates NUPACK : Analysis and Design of Nucleic Acid Systems," 2010.
- [21] "Official Grails web site." <http://www.grails.org>.
- [22] "jQuery website." <http://jquery.com/>.
- [23] "Bootstrap website." <http://getbootstrap.com/>.
- [24] "Grid Gain website." <http://www.gridgainsystems.com/wiki/display/GG15UG/GridGain+Book>.
- [25] "Official Hibernate web site." <http://hibernate.org/>.
- [26] "Official PostgreSQL web site." <http://www.postgresql.org>.
- [27] "Official spring source web site." <http://www.springsource.org>.
- [28] C. Walls, *Spring in Action*. Manning Publications, third edition ed., 2010.
- [29] "Official Apache Maven web site." <http://maven.apache.org/>.
- [30] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *The Journal of Physical Chemistry*, vol. 81, pp. 2340–2361, Dec. 1977.
- [31] T. E. Turner, S. Schnell, and K. Burrage, "Stochastic approaches for modelling in vivo reactions," *Computational biology and chemistry*, vol. 28, pp. 165–178, July 2004.
- [32] T. Jahnke and D. Alntan, "Efficient simulation of discrete stochastic reaction systems with a splitting method," *BIT Numerical Mathematics*, vol. 50, pp. 797–822, Sept. 2010.

Using CoSMoS to Reverse Engineer a Domain Model for Aevol

Paul S. Andrews^{1,2} and Susan Stepney^{1,2}

¹ York Centre for Complex Systems Analysis, University of York, UK

² Department of Computer Science, University of York, UK

Abstract. The CoSMoS approach and pattern language has typically been used to guide the entire process of development and use of scientific simulators. The resulting CoSMoS components (domain, domain model, platform model, simulation platform and results models) provide an explicit framework for presenting and reasoning about both the engineering and scientific aspects of the simulator and its results. The flexibility of CoSMoS enables us to use the same patterns to reverse engineer CoSMoS model components from pre-existing simulators and associated research literature. We demonstrate this here by applying the CoSMoS patterns to the Aevol (*artificial evolution*) simulator in order to extract an explicit Aevol domain model.

1 Introduction

The CoSMoS approach [1, 2] provides guidance on how to engineer, and subsequently use, computer simulators for scientific investigations. At the heart of CoSMoS is a collection of **core components**, shown in figure 1, whose construction let us reason about the process of engineering and using a simulator. The process of identifying and constructing these components is governed by a set of CoSMoS **patterns** [3], which each describe a generalised core solution to a task that one might encounter when working with scientific simulations. Appendix A provides a brief summary of the main CoSMoS patterns referred to in this paper.

The CoSMoS components and patterns have been successfully used to assist the engineering of simulators for science [4, 5]. These simulators have followed a typical application of the CoSMoS approach by applying the CoSMoS **Simulation Project** pattern that guides the identification and construction of the core components through the application of three phase patterns, **Discovery Phase**, **Development Phase** and **Exploration Phase**. The flexibility of the CoSMoS products and patterns

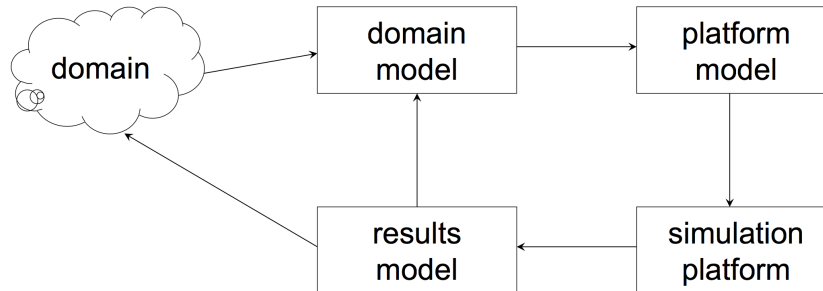


Fig. 1. The CoSMoS core components. Each is captured and referred to by its associated pattern, Domain, Domain Model, Platform Model, Simulation Platform, Results Model.

approach, however, does not dictate that we must always follow the complete CoSMoS Simulation Project pattern. Instead we can apply a subset of the CoSMoS patterns to construct core components in a way that best suits our particular circumstances. Here we illustrate such an example by showing how the CoSMoS patterns can be used to reverse-engineer a Domain Model from pre-existing simulator code (i.e. a simulator that hasn't been created within the CoSMoS approach).

Specifically, we are motivated to reverse engineer a Domain Model of Aevol [6], an *in silico* experimental artificial evolution platform [7] in which populations of digital bacteria are subject to Darwinian-style evolution. Although there are numerous publications that detail aspects of Aevol [7–9], as well as the simulator's source code [6], a single consistent Domain Model (in the CoSMoS sense) does not exist. As a Domain Model provides an explicit representation of the model of the science that underpins a simulator, an Aevol domain model is desirable to help us reason about Aevol simulation results and to help us produce any future extension of the Aevol platform.

We first outline in section 2 how we can apply the CoSMoS approach to develop an Aevol domain model, identifying our starting point and method for reverse engineering the model. We then describe the Aevol domain in section 3, and establish the Aevol platform model in section 4. From this point we outline the Aevol domain model in section 5, and finally discuss the reverse engineering exercise in section 6. Appendix A provides a brief summary of the main CoSMoS patterns referred to in the paper, and appendix B contains the Aevol domain model and Aevol platform model elements.

2 Application of CoSMoS to Aevol

We have as our starting point for reverse engineering an Aevol domain model the Aevol code (freely available to download at [6]) and a corpus of research literature such as [7–9]. This literature contains descriptions of the biology that has inspired Aevol, descriptions and ‘cartoon’ depictions (see figure 1 in [8]) of how Aevol has been implemented, and Aevol simulation results. Our first task is to examine how the Aevol code and literature relate to the CoSMoS core components identified in figure 1. This enables us to then identify how we can approach engineering the Aevol domain model, and which CoSMoS patterns will help us do this.

The CoSMoS core components shown in figure 1 are described in detail in [10], and summarised here:

Domain: a particular view or perspective of the real world system under study.

Domain Model: a model that explicitly captures aspects of the domain, identifying and describing the structures, behaviours and interactions. It is a model based on the science and is free from any simulation implementation details.

Platform Model: a model derived from the domain model that details how the concepts captured in the domain model will be implemented by the simulation platform.

Simulation Platform: encodes the platform model into a software and hardware system upon which simulation experiments are run, which in turn generate the results.

Results Model: a model describing the behaviour of the simulation platform based on the output of simulation experiments, providing the basis for interpretation of what the simulation results show.

These components are framed within a **Research Context** that identifies the scientific context of a simulation-based research project, establishing its scope, purpose and success criteria.

The closest match between the Aevol resources we possess and a CoSMoS component is the Aevol source code and the **Simulation Platform**. In this case the Aevol simulation platform is represented by the Aevol C++ code, relevant programming libraries, and the operating system and computer hardware on which it all runs. Any operating system and computer hardware that has the necessary C++ compiler will be able to run as an Aevol simulation platform, so for the purposes of this paper we just consider the Aevol simulation platform as being the Aevol code.

Elements of the other four components are contained within the Aevol literature, although there is no explicit mapping to any specific core

component. As our task is to construct a **Domain Model**, we ignore trying to establish the Aevol results model for this exercise and instead focus on identifying the Aevol domain model from the details contained within the literature and our Aevol simulation platform. First we use the literature to infer a description of the **Research Context** and **Domain** of Aevol within section 3.

Due to the relationship between the **Domain Model**, **Platform Model** and **Simulation Platform**, the Aevol simulation platform will contain many of the concepts, structures and behaviours present in the Aevol domain model and the Aevol platform model. So, instead of trying to infer the Aevol domain model solely from the Aevol domain and literature we work backwards from the Aevol simulation platform to create the Aevol platform model (section 4). From there we create the Aevol domain model (section 5) by cross-referencing the structures and behaviours in the Aevol platform model with Aevol domain concepts.

3 Aevol Domain

Examining the Aevol research literature, we can use the **Domain Identification** pattern to outline Aevol's domain. In short, this pattern involves: providing an overview description of the **Domain**; using the **Cartoon** pattern to present an informal sketch that identifies system components in domain specific terms; identifying the relevant sources for domain knowledge; applying the **Document Assumptions** pattern; and defining the scientific scope and boundary of the domain.

In normal circumstances we would identify a **Domain Researcher** as our source for domain knowledge, however for this exercise we already have published material on Aevol and we wish to demonstrate how a **Domain** can be extracted from such a body of work. So, the chosen sources for domain knowledge on Aevol are the three publications [7–9], which capture a representative cross-section of the entire body of published work on Aevol (see [6] for a full list of Aevol publications). The domain description now follows.

Aevol is designed to provide insight into the real world dynamic of Darwinian evolution. The Aevol domain, therefore, falls within the areas of evolutionary theory and digital genetics [11] in which populations of artificial organisms evolve within a computer simulation. Specifically, Aevol is focused on the evolutionary dynamics of the size and organisation of bacterial genomes, enabling the user to run digital experiments in an *in silico* laboratory to test evolutionary scenarios [8].

The cartoon in figure 2, adapted from [8], summarises the biological processes that have been modelled by Aevol. Here we see individuals

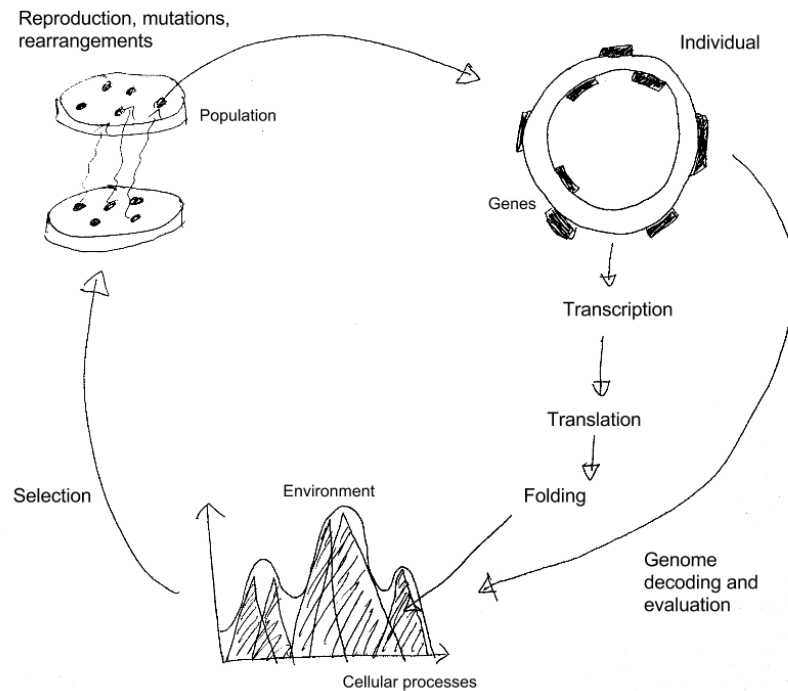


Fig. 2. Aevol domain cartoon, adapted from [8]

have double-stranded circular genomes, which are constructed from a sequence of nucleotides. The nucleotides on the genome are grouped into coding sequences (genes) and non-coding sequences. Genes are decoded via an explicit process of transcription (via messenger RNA), translation and folding, which results in a signature of the cellular processes for that individual. Selection is performed as a function of this signature compared to an environment, with individuals with a closer match to the environment being favoured for selection. Selected individuals are asexually reproduced to form the next generation of the population. Variation is introduced to the population by exposing reproduced individuals to genetic mutations and rearrangements such as point mutations, insertions, deletions, duplications, inversions and translocations. These mutations can create new genes and modify or destroy existing genes.

In addition to the domain description above, we can extract from the background Aevol literature a number of important assumptions (Document Assumptions) and establish the scientific scope and boundary

of the domain. Whilst we don't capture all assumptions, the following should be considered when interpreting outputs from the Aevol simulator:

- Space is not modelled, bacteria exist in a well mixed environment. (Modified versions of Aevol exist which consider a grid-like environment, but this is not considered here).
- Individuals are composed only of their genome and mechanisms to transcribe, translate and fold proteins; there is no cell metabolism.
- The non-linear mapping between genotype and phenotype (cellular processes) is sufficient for the selection process.
- Genetic transfer only occurs vertically during reproduction; there is no horizontal transfer (plasmids) between individuals. (Modified versions of Aevol exist with plasmids, but this is not considered here).
- The environment is static and does not change. (Modified versions of Aevol exist with varying environments, but this is not considered here).
- Complete populations of individuals are replaced per generation.

Elements of the Research Context for Aevol can be identified from the literature as:

- Investigating processes of indirect selection and evolvability
- Insights into circular bacterial genome structures (chromosome length and composition)
- Dynamics of structural changes to genomes only
- Spatial environment effects not taken into account
- Only vertical transfer of genetic material (no plasmids or horizontal transfer)

4 Aevol Platform Model

Although the Platform Modelling pattern (see appendix A) assumes that to develop the Platform Model from the Domain Model, we can still apply it (with caveats) to help us develop the Aevol platform model in the absence of the Aevol domain model. First we apply the Modelling Approach pattern to identify a language to describe the Aevol platform model. Given object-oriented design of the Aevol code, we have chosen the UML's Class Diagrams and Activity Diagrams to express the implemented structures and behaviours respectively. These diagrams are also a natural language in which to express a domain model (see examples [4, 5]), so we keep continuity of modelling language between the Aevol platform model and Aevol domain model.

As previously mentioned, the **Platform Modelling** pattern states that the platform model should be developed from the **Domain Model**. This process considers engineering factors and should result in a **Platform Model** in which some **Domain Model** components have been removed (such as emergent behaviours), components not in the **Domain Model** have been added (such as visualisations), and components differ between the two models (such as number of agents). Here, however, we can extract the platform model components and behaviours directly from the Aevol code (the **Simulation Platform**), and we consider later how the **Platform Model** and as yet undefined **Domain Model** differ.

Extracting the platform model from the Aevol code is essentially a mechanical task with a one-to-one mapping between class structures, behaviours and parameters in the code and **Platform Model**, as the **Simulation Platform** is simply a concrete implementation of the **Platform Model** in a given programming language. The Aevol code is around 20000 lines of native C++ and, other than the standard C++ and system libraries, uses the SIMD-oriented Fast Mersenne Twister (SFMT) library [12] for generating pseudo-random numbers.

The main classes present in the Aevol code, and the relationships between these classes, are shown in figure 5. The methods of the Aevol classes implement the behavioural aspects of the Aevol platform model. Being guided by the Aevol domain description established above, these behaviours have been summarised in three activity diagrams in figures 6, 7, 8 which show the selection process, mutation process, and fitness evaluation process respectively. It is noted that the activities in these diagrams reflect only the gross-level structure of the Aevol code and that more in-depth descriptions of the code could be produced. However, this is infeasible for the many thousands of lines of Aevol code, and the activity diagrams presented provide the necessary level of detail to help us reach our ultimate goal of presenting the Aevol domain model.

Having documented the main structural and behavioural concepts present in the **Platform Model** diagrams, we can identify which elements of the **Platform Model** have been added for instrumentation purposes (to run simulations and record data). The following classes shown in figure 5 are instrumentation classes: *ExperimentManager*, *ExperimentSetup*, and *OutputManager*. These instrumentation concepts may, or may not, be present in the **Domain Model** depending on whether it explicitly captures the domain experiment within. We will revisit this when we consider the Aevol domain model in the next section.

We can identify cases of abstractions that have obviously been made for implementation reasons, and these should be reflected in the **Domain Model**. First, the DNA nucleotide sequence captured by the *DNA* class

is a binary string, therefore there are just two nucleotides, 0 and 1. This has effects throughout the genotype-to-phenotype mapping, namely in the translation and folding processes encoded within the *GeneticUnit*, *RNA* and *Protein* classes. Second, the *Environment* class is implemented as a *FuzzySet*, which represents a continuous function as a discrete set of points.

These last two platform model abstractions highlight two of the main assumptions (applying Document Assumptions) about the platform model, namely:

- A binary string nucleotide representation of the genome is sufficient to reflect the structural organisation on the genome during the evolutionary process
- The genotype-to-phenotype mapping implementation provides an acceptable non-linear mapping between the two to mimic that of real bacteria.

Lastly, the Platform Model and Domain Model differ from what has been removed from the domain model when constructing the domain model, such as emergent behaviours. We identify these in the next section when comparing the Aevol platform model and Aevol domain to infer the Aevol domain model.

5 Aevol Domain Model

Given our identified Aevol domain and Aevol platform model, we can use the Domain Modelling pattern to construct our Aevol domain model. First we apply the Modelling Approach pattern. As mentioned above, UML's class and activity diagrams are natural tools for expressing modelling structures and behaviours and they have been used to capture the Aevol platform model, so we use the same modelling approach here. Next, we apply a four stage process to identifying the Aevol domain model:

1. Extract the domain structures and behaviours from the Aevol platform model diagrams removing those concepts that were added for additional instrumentation and modifying those concepts that were deemed to be implementation abstractions.
2. Identify the behaviours that are not in the Aevol platform model, but are necessary to explain the Aevol domain.
3. Modify the Aevol platform model diagrams as according to 1. and 2. to make the Aevol domain model.
4. Construct the Aevol domain experiment model (not shown here) based on relevant instrumentation present in Aevol platform model.

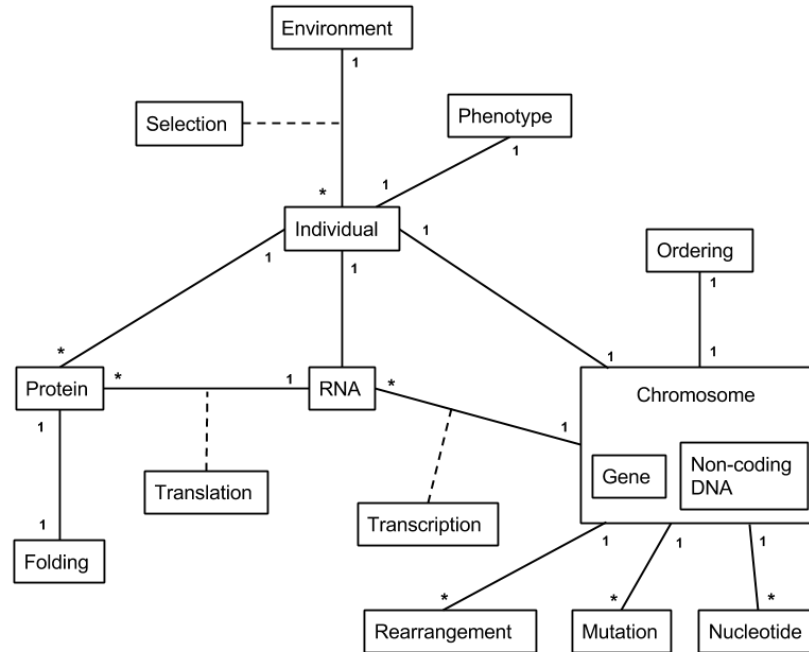


Fig. 3. Aevol domain model class concepts.

The resulting Aevol domain model is summarised in figures 3 and 4. In addition, the Aevol platform model activity diagrams (figures 6, 7 and 8) are also suitable components of the Aevol domain model as they are able to convey the selection, mutation, and fitness evaluation processes present Aevol domain. We may in future wish to create more detailed activity diagrams separately for the Aevol domain model and Aevol platform model, but the current more generic activity diagrams suffice for this exercise.

The Aevol domain model classes in figure 3 differs from the Aevol platform model classes (figure 5) in the following ways:

- We have removed the experiment specific instrumentation classes such as *ExperimentManager* and moved these concepts to the Aevol domain experiment model (not shown here).
- The relevant evolutionary processes (*Folding*, *Translation*, *Transcription*, *Mutation*) are represented explicitly as classes that act upon the structures (*Protein*, *RNA*, *Chromosome*).

- The *Chromosome* is composed of both coding (*Gene*) and *Non-coding DNA*
- The concept of *Ordering* is applied to the *Chromosome* to capture the concept of the **emergent** process that acts up the bacterial genome as a whole. This concept is expanded with the aid of figure 4.

The **Cartoon** in figure 4 captures the **Research Context Diagram** element of the Aevol domain model and complements the concepts provided in class diagram, specifically the emergent ordering of the bacterial genome. This provides an overview of the behaviours being modelled, highlighting the components hypothesised to play a significant role in the real-world phenomena and the system-level behaviours that result from their interactions. This links the domain language with expected observables in the simulation. Here we are highlighting the concepts of **indirect selection** and **evolvability** that occur within the research literature with how we expect these concepts to be revealed in the Aevol simulation platform and subsequent **Results Model** for Aevol: namely the **dynamical changes** in the population chromosome length and make-up of coding and non-coding DNA on that genome over an evolutionary run. We also highlight the components in the Aevol domain that we believe are responsible for these emergent behaviours. This essentially captures the rates at which mutational and rearrangement processes occur, and how these are selected for according to the individual's phenotype.

Finally, we apply **Document Assumptions**, recording the main Aevol domain model assumptions as:

- We have suitably identified the source of emergent behaviours we expect to arise in the Aevol simulation platform
- The mutation, rearrangement and selection dynamics are sufficient to change the structure of the chromosome
- We can measure ordering on the bacterial chromosomes

6 Conclusions

We have shown how we can use the CoSMoS pattern language to take a pre-existing scientific simulation platform and extract an explicit **Domain Model** via a process of constructing the associated **Domain** from the domain literature, and the **Platform Model** from the simulator code. To do this we have used only those CoSMoS patterns that were required. Importantly, this is made possible as the patterns do not impose a strict ordering on *when* things need to be done, only *what* needs to be done.

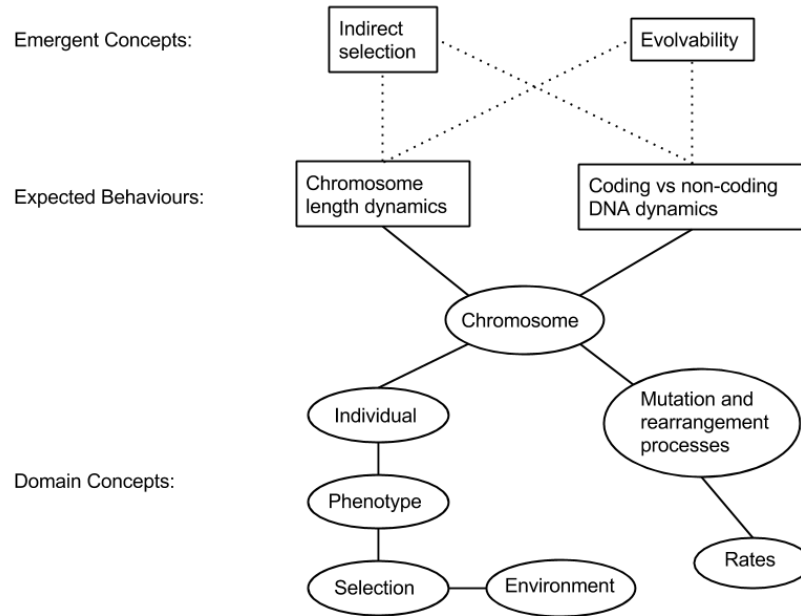


Fig. 4. Aevol research context diagram.

Consequently we have been able to apply the patterns in a different ordering to a typical application of the CoSMoS Simulation Project.

For simplicity we have not followed the argumentation patterns that are associated the other patterns we have used, but there is no reason why these cannot be used for a similar exercise. These patterns help us record and justify assumptions and design decisions. Also, we have not yet employed the Domain Researcher (the developers of Aevol) to check that we have ultimately captured the correct Domain and Domain Model. This is left as a necessary task for the future, and may result in the incremental re-application of the patterns to refine our CoSMoS products.

We have subsequently used the Aevol domain and platform model to create a refactored version of the simulator in the Python programming language. Whilst this does not have the performance of the original C++ version, it is providing to be a very useful prototyping tool to explore extensions of the Aevol simulator. With the explicit Domain Model and Platform Model, these extensions can be done in a principled and transparent manner. Future work (as part of the EvoEvo project [13]) will

apply the same domain model reverse engineering process to a family of simulators based on the “Pearls on a string” (PoaS) formalism that have been used to explore evolutionary dynamics [14, 15]. We can then use the Aevol domain model and the domain model of the PoaS simulators to develop a common metamodel [10]. This metamodel will provide the basis to develop a framework for novel evolutionary algorithms.

Acknowledgments

This work was funded by the EU FP7 project EvoEvo [13], grant number 610427. We would like to thank our colleagues on the project for helpful discussions.

References

- [1] P. S. Andrews, F. A. C. Polack, A. T. Sampson, S. Stepney, and J. Timmis, “The CoSMoS process, version 0.1: A process for the modelling and simulation of complex systems,” Tech. Rep. YCS-2010-453, Department of Computer Science, University of York, Mar. 2010.
- [2] S. Stepney and et al., *Engineering Simulations as Scientific Instruments – A Pattern Language*. Springer, to appear.
- [3] S. Stepney, “A pattern language for scientific simulations,” in *2012 CoS-MoS workshop* (S. Stepney, P. S. Andrews, and M. Read, eds.), pp. 77–103, Luniver Press, 2012.
- [4] M. N. Read, P. S. Andrews, J. Timmis, R. A. Williams, R. B. Greaves, H. Sheng, M. C. Coles, and V. Kumar, “Determining disease intervention strategies using spatially resolved simulations,” *PloS one*, vol. 8, no. 11, 2013.
- [5] K. J. Alden, J. Timmis, P. S. Andrews, H. Veiga-Fernandes, and M. C. Coles, “Pairing experimentation and computational modelling to understand the role of tissue inducer cells in the development of lymphoid organs,” *Frontiers in Immunology*, vol. 3, no. 172, 2012.
- [6] <http://www.aevol.fr/>.
- [7] D. Parsons, *Indirect Selection in Darwinian Evolution: Mechanisms and Implications*. PhD thesis, L’Institut National des Sciences Appliquée de Lyon, 2011.
- [8] B. Batut, D. Parsons, S. Fischer, G. Beslon, and C. Knibbe, “*In silico* experimental evolution: a tool to test evolutionary scenarios,” *BMC Bioinformatics*, vol. 14, no. Suppl 15, p. S11, 2013.
- [9] C. Knibbe, A. Coulon, O. Mazet, J.-M. Fayard, and G. Beslon, “A long-term evolutionary pressure on the amount of noncoding DNA,” *Mol. Biol. Evol.*, vol. 24, no. 10, pp. 2344–2353, 2007.

- [10] P. S. Andrews, S. Stepney, T. Hoverd, F. A. C. Polack, A. T. Sampson, and J. Timmis, “CoSMoS process, models, and metamodels,” in *2011 CoSMoS workshop* (S. Stepney, P. Welch, P. S. Andrews, and C. G. Ritson, eds.), pp. 1–13, Luniver Press, 2011.
- [11] C. Adami, “Digital genetics: unravelling the genetic basis of evolution,” *Nature Reviews Genetics*, vol. 7, pp. 109–118, 2006.
- [12] <http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/SFMT/index.html>.
- [13] <http://evoevo.liris.cnrs.fr/>.
- [14] A. Crombach and P. Hogeweg, “Chromosome rearrangements and the evolution of genome structuring and adaptability,” *Molecular biology and evolution*, vol. 24, no. 5, pp. 1130–1139, 2007.
- [15] A. Crombach and P. Hogeweg, “Evolution of evolvability in gene regulatory networks,” *PLoS Computational Biology*, vol. 4, no. 7, 2008.

A CoSMoS Patterns

The patterns briefly summarised here are in alphabetical order and will appear in full in the forthcoming publication [2].

► Cartoon

Produces an informal sketch that identifies system components in domain specific terms. It provides a step towards more formal modelling.

► CoSMoS Simulation Project

Develop a fit-for-purpose simulation of a scientific or engineering system. Application of this pattern results in the construction of the products shown in figure 1 from the application of the Discovery Phase, Development Phase and Exploration Phase patterns.

► Data Dictionary

Defines the modelling data that will be used to parameterise the Simulation Platform, and relevant real-world data that will form the basis for comparison to data produced from simulation experiments.

► Development Phase

Based on the outputs from Discovery Phase, produces the simulation platform upon which simulation experiments can run. Will require the completion of the Platform Model and Simulation Platform patterns.

► Discovery Phase

Establishes what simulation platform needs to be built. Other patterns required to complete this pattern include the **Research Context**, **Domain** and **Domain Model** patterns detailed below.

► Document Assumptions

Records assumptions appropriately to ensure they are explicit and justified. For each assumption record its nature, criticality, reason for existence and a justification such that its consequences are understood.

► Domain

A particular view or perspective of the real world system under study.

► Domain Experiment Model

A model that explicitly captures the experimental system that is applied to the domain components identified in the **Domain Model**. This model identifies the domain variables and how we manipulate and record them.

► Domain Identification

Identifies the **Domain** (see figure 1), establishing the perspective on the real world system of study used as the basis for simulation. This pattern makes use of the **Cartoon** and **Document Assumptions** patterns.

► Domain Model

A model that explicitly captures aspects of the domain, identifying and describing the structures, behaviours and interactions. It is a model based on the science and is free from any simulation implementation details.

► Domain Modelling

Generates the **Domain Model** (see figure 1) and **Domain Experiment Model**, which forms the scientific description of the identified **Domain**. This pattern makes use of the **Cartoon**, **Modelling Approach**, **Document Assumptions** and **Data Dictionary** patterns.

► Domain Researcher

Identifies the domain researcher who is the point of contact for domain knowledge, providing the interpretation of the domain literature.

► Exploration Phase

Uses the outputs from **Development Phase** to run simulation experiments to investigate the questions identified during **Discovery Phase**.

► Modelling Approach

Identifies an appropriate approach and notation for producing a model. Takes account of suitability and understandability with respect to the modelling to be performed.

► Platform Implementation

Generates the **Simulation Platform** (see figure 1), which incorporates a software and hardware platform capable of running simulations of the implemented **Platform Model**.

► Platform Model

A model derived from the domain model that details how the concepts captured in the domain model will be implemented by the simulation platform.

► Platform Modelling

Generates the **Platform Model** (see figure 1) detailing how the **Domain Model** and **Data Dictionary** concepts will be implemented and analysed in the **Simulation Platform** product. This pattern makes use of the **Modelling Approach** and **Document Assumptions**.

► Research Context

Identifies the scientific context of a simulation-based research project, establishing its scope, purpose and success criteria.

► Research Context Diagram

A Cartoon that provides an overview of the behaviours being modelled, highlighting the components hypothesised to play a significant role in the real-world phenomena and the system-level behaviours that result from their interactions.

► Simulation Platform

Encodes the platform model into a software and hardware system upon which simulation experiments are run, which in turn generate the results.

B Platform Model Diagrams

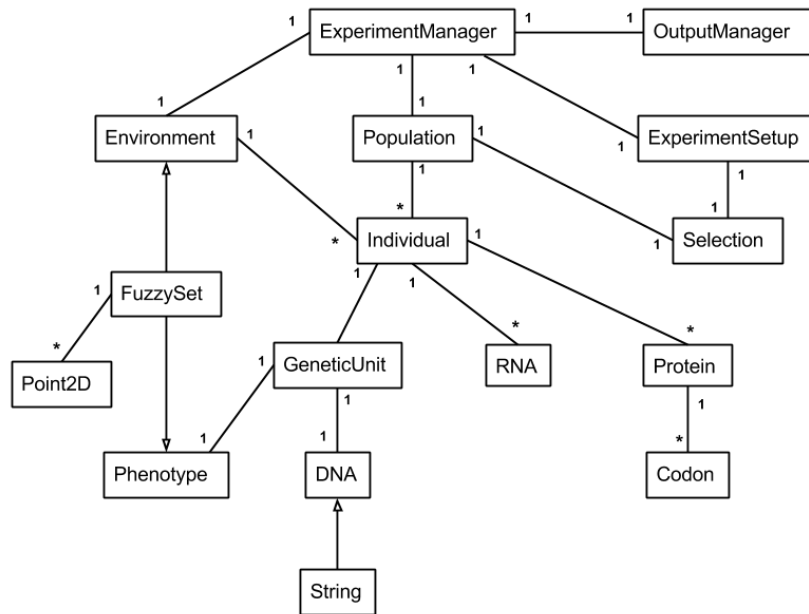


Fig. 5. Aevol platform model classes.

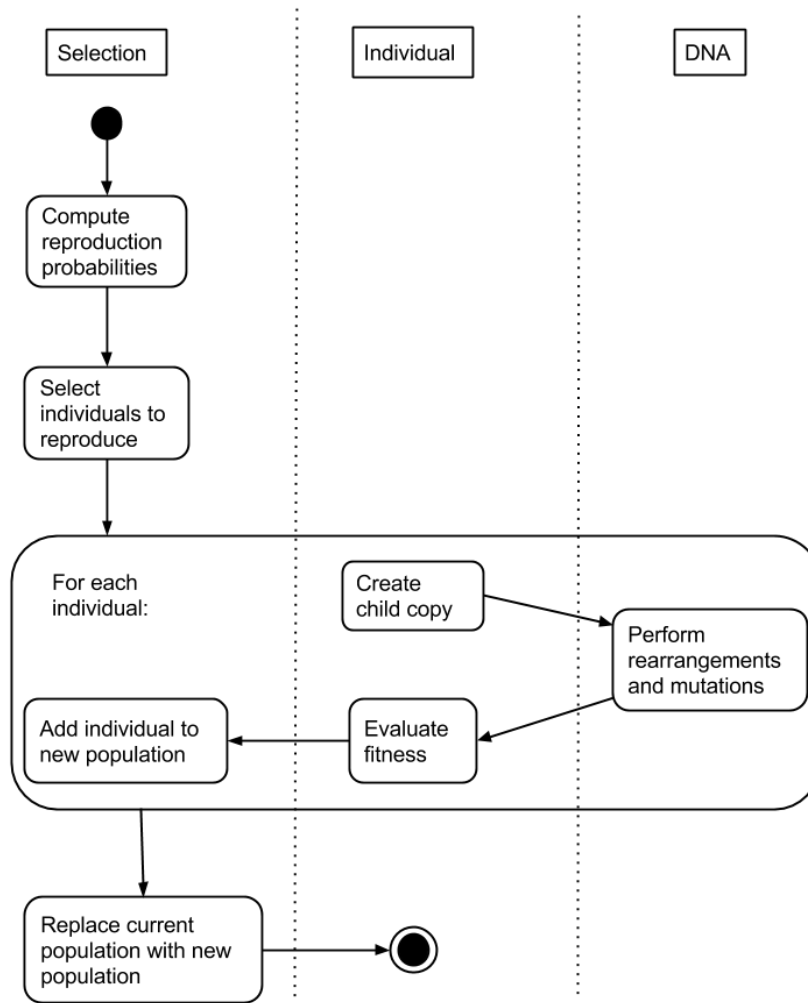


Fig. 6. Aevol platform model selection activity.

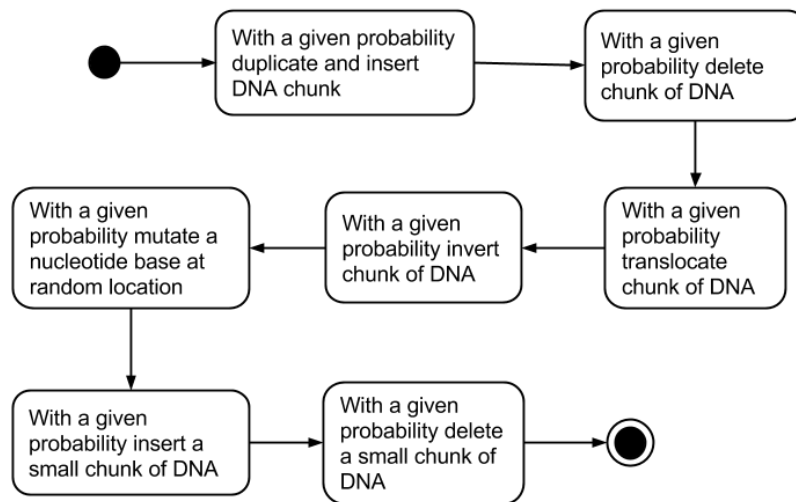


Fig. 7. Aevol platform model mutation activity.

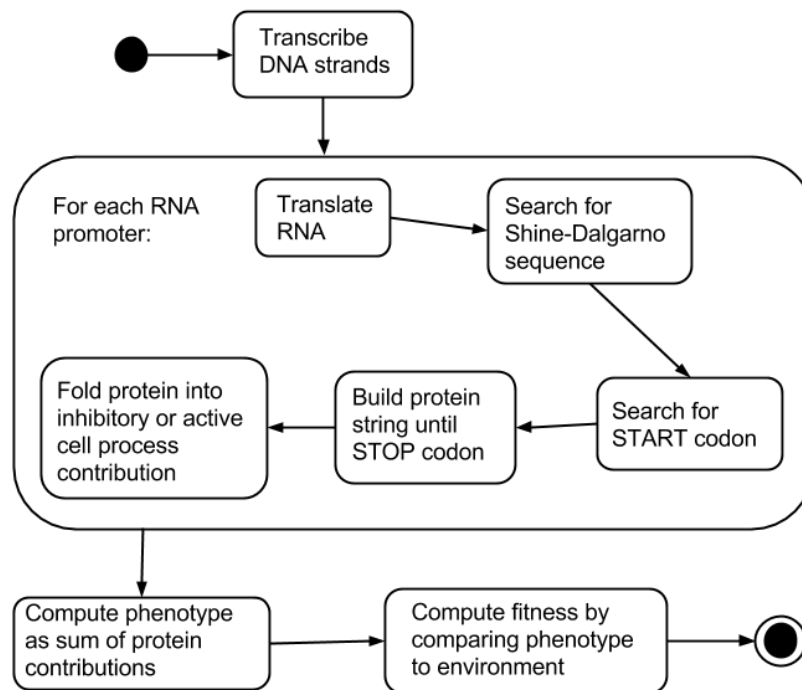


Fig. 8. Aevol platform model evaluation activity.

Order of Battle: A Case Study for Designing Emergent Structure in Games

Stefan Leijnen and Joris Dormans

Amsterdam University of Applied Sciences,
Duivendrechtsekade 36–38, Amsterdam, the Netherlands

Abstract. Modeling dynamical feedback loops in game mechanics as complex systems helps to create fun and interesting gameplay experiences. Including structural aspects of emergence yields a more integrated approach and allows for new gameplay possibilities.

Creating games that are simple enough for a player to learn but that allow for challenging complexities to keep him or her interested is a craft often likened to a form of art. Yet, general rules of thumb that may help guide the design process are being discovered. One such rule is to exploit emergence in the game world by creating a world where a limited number of different entities combine into a nearly unbounded universe of possible interactions. If done right, this allows for elegant designs where the world seems complex and unpredictable for the player but is controllable from a design perspective.

Machinations is a Domain Specific Language modeled after Petri Nets which can be used to formalize game mechanics and produce visual diagrams of the relevant component interactions [1]. It can be used online [2] to produce diagrams and subsequently analyze the dynamics that emerge.

In an experimental game called Order of Battle (OOB), two players each control a commander unit who, locally, increases order in their respective battle ranks (fig. 1, left). As soldiers fire salvos towards their adversaries, their order (represented in the game as discipline or morale) decreases due to enemy fire. A decrease in troop order will negatively effect the power of a salvo which in turn does less damage to the enemy's troop order, similar to historical musket warfare. Troop order also decreases naturally over time.

OOB game mechanics is modeled after Rayleigh-Bénard (RB) convection [3], an emergent process that can occur when a liquid is exposed

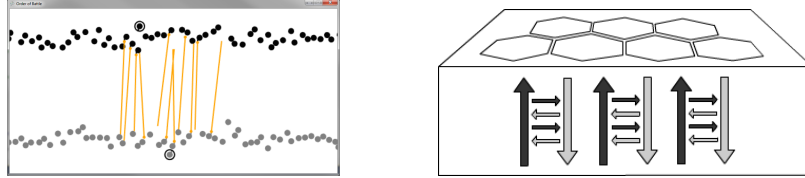


Fig. 1. Commanders (illustrated by circled dots) increase local troop order in OOB, which in turn affects the effectivity of firing salvos (yellow lines) that decrease the enemy troop order (left); a complex feedback loop that is modeled after entropy production maximization in RB convection (right).

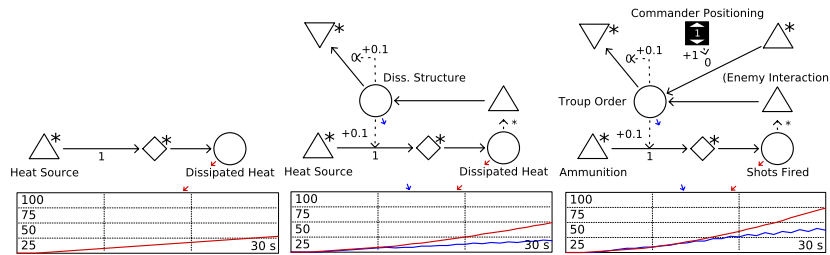


Fig. 2. Machinations diagrams and corresponding graphs of straightforward heat dissipation (left), heat dissipation maximization (red) due to the formation of dissipative structure (blue) in RB convection (center), and the effect of commander positioning on troop order (blue) and firing rate (red) in OOB game mechanics (right).

to a heat source from below. According to the Maximum Entropy Production Principle [4], the structure which will dissipate most energy is bound to arise. This principle explains a phenomenon that was previously thought to be inherently unpredictable: how and where emergent structure arises spontaneously. Thus, as the upward dissipation of heat is obstructed by the downward motion of cooled molecules, dissipative structures (in the form of hexagonal columns known as convection cells) spontaneously take shape and minimize horizontal exchange, thereby maximizing vertical dissipation (fig. 1, right).

Although it increases the complexity of the OOB game mechanics model, including dissipative structure (i.e. troop order) to maximize the firing rate provides an interesting and realistic game dynamic (fig. 2). As next steps, we intend to extend the model with battle commands, multiple commanders, and a hierarchy of structural components [5] that allows for long-term persistence of emergent gameplay structures.

References

- [1] J. Dormans, “Simulating mechanics to study emergence in games,” in *Proceedings of the 2011 AIIDE Workshop, Menlo Park, California, USA*, pp. 2–7, 2011.
- [2] J. Dormans, “The machinations framework.” <http://www.jorisdormans.nl/machinations>.
- [3] A. V. Getling, *Rayleigh-Bénard Convection: Structures and Dynamics*. World Scientific, 1998.
- [4] G. W. Paltridge, “Climate and thermodynamic systems of maximum dissipation,” *Nature*, vol. 279, 1979.
- [5] T. Hoverd and S. Stepney, “Formalising harmony seeking rules of morphogenesis,” in *Proceedings of Twelfth Artificial Life Conference*, pp. 386–393, 2012.

In Silico Minimal Cell Model Systems

Fabio Mavelli¹, Emiliano Altamura¹, and Pasquale Stano²

¹ Chemistry Department, Aldo Moro University, Bari, Italy

² Science Department, Roma Tre University, Rome, Italy

The top-down synthesis of artificial cells and bottom-up self-assembly of protocells are attracting the interest of a growing number of researchers [1–4]. In this framework, the properties of lipid vesicles (e.g., their stability, permeability, growth dynamics, potential to host reactions or undergo division processes...) are being experimentally explored by an increasing number of labs, eliciting an ever increasing interest on their use as compartmentalized chemically reacting systems and plausible models for minimal living systems. Thus, from a theoretical standpoint, it is highly attractive the determination, under a set of assumptions as broad as possible, the conditions under which simple protocells spontaneously settle into a stationary reproducing regime, characterized by a regular growth/division cycle and the maintenance of a certain standard size and chemical composition across generations (figure 1). The occurrence of a stationary regime of protocells reproduction can be expressed in terms of the *growth control coefficient* (γ), based on simple geometrical considerations, and of an explicit deterministic relationship, the osmotic synchronization condition, which can be analytically derived under a set of kinetic simplifications. This general condition constrains different molecular/kinetic parameters and features of the compartmentalized self-producing system (reaction rates, permeability coefficients, metabolite concentrations, system volume) due to the osmotic pressure balance operating across the protocell membrane. Accordingly, this model can be used for predicting and characterizing the stationary regime in terms of protocell size and lifetime.

Furthermore, in compartmentalized reacting systems where the molecular population of the reactants is very low, random fluctuations due to the stochastic nature of reacting events (*intrinsic stochasticity*) can bring an open system towards unexpected time evolutions [5]. Additionally, this effect results to be amplified by the spreading of different initial concentrations of biological molecules encapsulated in lipid compartments, the molecular distribution being dependent on the experimental preparation procedure (*extrinsic stochasticity*). In recent years we developed a computational platform [6, 7] suitable for studying the

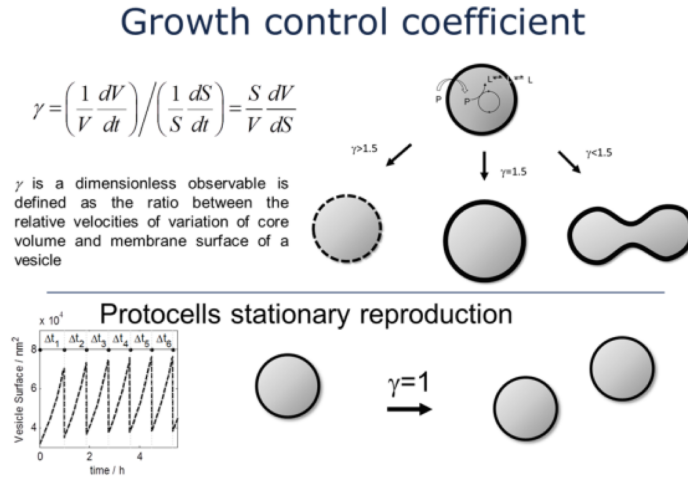


Fig. 1. Stationary reproduction conditions of *in silico* protocells

stochastic time evolution of reacting lipid compartments in order to elucidate the role of randomness [8] in the time behaviour of chemically reacting and self-reproducing lipid compartments, such as vesicles or micro emulsions. This program is a C++ object oriented code that implemented the Gillespie stochastic simulation direct method [9] in order to cope with reacting compartments that can increase in number during the time evolution. Moreover, a MPI version has been also implemented in order to run in parallel both statistically equivalent simulations, in order to get insights on the average time behaviour of the reacting system, and simulations that differ in the kinetic parameters, in order to quickly explore the space of kinetic constants, molecular permeability, initial concentrations, etc..

Two main research lines are then generated from these premises: the first consists in modelling and simulating the structural properties and dynamic behaviour of lipid vesicle populations. This is done by a comparison with experimental data available in literature [8, 10], and gives us the opportunity to test our approach, our simplifying assumptions, and to estimate dynamic and structural parameters by fitting experimental data (for example, kinetic parameters for the exchange of lipids between the vesicle membrane and bulk aqueous solution are obtained). The second line of research explores hypothetical protocell models that keep a relatively low degree of molecular complexity like the ‘minimal

lipid-peptide cell' [11] and the 'Ribocell' [12, 13]. More recently, we also started an investigation on the effect of different molecular distribution laws on the protein expression taking place in giant lipid vesicles [14]. In all these cases, random fluctuations can play an important role in determining the time behaviour of the studied systems.

References

- [1] P. L. Luisi and P. Stano, *The minimal cell. The biophysics of cell compartment and the origin of cell functionality*. Springer, 2011.
- [2] V. Noireaux, Y. T. Maeda, and L. A., "Development of an artificial cell, from self-organization to computation and self-reproduction," *PNAS*, vol. 108, 2011.
- [3] I. Chen and P. Walde, "From self-assembled vesicles to protocells," *Cold Spring Harbor Perspectives in Biology*, vol. 2, p. a002170, 2010.
- [4] R. V. Solé, A. Munteanu, C. Rodriguez-Caso, and J. Macía, "Synthetic protocell biology from reproduction to computation," *Philosophical Transactions Royal Society B*, vol. 362, pp. 1727–39, 2007.
- [5] F. Mavelli and S. Piotto, "Stochastic simulations of homogeneous chemically reacting systems," *Journal of Molecular Structure*, vol. 771, pp. 55–64, 2006.
- [6] F. Mavelli and K. Ruiz-Mirazo, "Stochastic simulations of minimal self-reproducing cellular systems," *Philosophical Transactions Royal Society B*, vol. 362, pp. 1789–1802, 2007.
- [7] F. Mavelli and K. Ruiz-Mirazo, "Theoretical conditions for the stationary reproduction of model protocells," *Integrative Biology*, vol. 5, pp. 324–341, 2013.
- [8] F. Mavelli and K. Ruiz-Mirazo, "ENVIRONMENT: a computational platform to stochastically simulate reacting and self-reproducing lipid compartments," *Physical Biology*, vol. 7, p. 036002, 2011.
- [9] D. Gillespie, "Stochastic simulation of chemical kinetics," *Annual Review of Physical Chemistry*, vol. 58, pp. 35–55, 2007.
- [10] F. Mavelli and P. Stano, "Kinetic models for autopoietic chemical systems role of fluctuations in homeostatic regime," *Physical Biology*, vol. 7, p. 016010, 2010.
- [11] K. Ruiz-Mirazo and F. Mavelli, "On the way towards 'basic autonomous agents' stochastic simulations of minimal lipid-peptide cells," *BioSystems*, vol. 91, pp. 374–387, 2008.
- [12] J. W. Szostak, D. P. Bartel, and P. L. Luisi, "Synthesizing life," *Nature*, vol. 409, pp. 387–390, 2001.
- [13] F. Mavelli, "Stochastic simulations of minimal cells: the ribocell model," *BMC Bioinformatics*, vol. 13, no. 4, p. S10, 2012.
- [14] S. P. Mavelli F., "Experiments and numerical modelling on the capture and concentration of transcription-translation machinery inside vesicles," (in preparation).

