Round-based Super-Individuals – Balancing Speed and Accuracy

Citation for published version:

Digital Object Identifier (DOI):
10.1145/3316480.3322894

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and/or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Round-based Super-Individuals – Balancing Speed and Accuracy

Pia Wilsdorf
University of Rostock
Rostock, Germany
pia.wilsdorf@uni-rostock.de

Maria E. Pierce
Thuemen Institute of Baltic Sea Fisheries
Rostock, Germany
maria.pierce@thuenen.de

Jane Hillston
University of Edinburgh
Edinburgh, UK
jane.hillston@ed.ac.uk

Adelinde M. Uhrmacher
University of Rostock
Rostock, Germany
adelinde.uhrmacher@uni-rostock.de

ABSTRACT

Agent-based models which are based on a continuous-time Markov chain semantics are increasingly receiving attention in simulation. To reduce computational cost, model aggregation techniques based on Markov chain lumping can be leveraged. However, for models with nested, attributed agents, and arbitrary functions determining their dynamics it is not trivial to find a partition that satisfies the lumpability conditions. Thus, we exploit the potential of the so-called super-individual approaches where sub-populations of agents are approximated by representatives based on some criteria for similarity, and propose a round-based execution scheme to balance speed and accuracy of the simulations. For realization we use an expressive rule-based modeling and simulation framework, evaluate the performance using a fish habitat model, and discuss open questions for future research.

CCS CONCEPTS
• Computing methodologies → Agent / discrete models; Modeling methodologies; Discrete-event simulation; Multiscale systems.

KEYWORDS
Chemical Reaction Networks, Multi-level Modeling, Model Reduction

1 INTRODUCTION AND MOTIVATION

Agent-based simulation has become a well established approach in many application areas, since it allows one to analyze systems as heterogeneous populations of autonomously interacting entities. Agent-based models may be compactly described as (chemical) reaction networks [8], and are typically executed in a stochastic discrete-event simulation by interpreting them as a continuous-time Markov chain (CTMC) with exponentially-distributed sojourn times. However, these simulations are known to be computationally expensive, and thus their efficient execution poses methodological challenges. This is even aggravated when the model exhibits complex dynamics depending on attributes or nested agents, and thus has dynamically changing sets of species and reactions. To speed up the execution of agent-based models a plethora of approaches has been developed. Beside approaches that, e.g., exploit parallelization, another possibility is to reduce the overall complexity of the model itself. Reduction methods for (chemical reaction network) models typically aggregate species and rules by exploiting equivalences in the model, thereby producing a simplified network which preserves the original dynamics [2, 3, 13]. This may be an option for models that rely on populations of comparatively simple-structured individuals, and if the individuals are not distinguished by the model’s dynamics [3]. However, if an agent’s behavior depends on continuous attributes or is hierarchically and dynamically nested, i.e., contains other individuals, finding a partitioning of the model that satisfies the lumpability conditions is difficult, or such a partitioning may not even exist. Another approach aimed at aggregating individual-based models is the so-called super-individual approach which starts with the individuals and some notion of similarity, e.g., based on the agent’s attributes, to approximate a group of individuals by a representative, i.e., the super-individual. The approach, which trades accuracy for speed to cope with the execution of a large number of agents efficiently, has been suggested in the 1990s [12]. Since then the idea has been applied in various implementations in the area of ecology and beyond [10, 14]. However, when concerned with complex agent-based models static schemes are not applicable, especially when the model is observed over multiple generations with agents frequently leaving and entering the system, or when the model exhibits a changing hierarchical structure. Some authors discussed dynamic aspects of super-individuals [14], however, the problem of when to form and when to split super-individuals has not been addressed in depth. Therefore, in this work we (1) adopt the idea of super-individuals and explore its potential for handling more complex agent-based models, and (2) propose a round-based execution scheme which dynamically switches between simulating non-aggregated individuals and aggregated super-individuals. Our
approach allows for reorganization of super-individuals during the simulation to prevent excessive homogenization within the subpopulations and to compensate for the induced approximation error. Using a case study we demonstrate the benefit of the round-based scheme and discuss interesting research questions for future work.

2 RULE- AND AGENT-BASED MODELS

Our round-based model aggregation approach focuses on rule- and agent-based models with optional hierarchies. Agents are categorized into agent types. All agents belonging to one type share the same attributes and attribute types. Further, some types of agents can contain other agents and are called compartments. However, the concrete attribute values and contents of agents may vary among agents of the same type. The interactions and activity of agents are governed by behavioral rules which rely on arbitrary functions for accessing and updating their attributes and contents. Rules may be expressed in a reaction-style manner, \textit{Reactants} $\rightarrow$ \textit{Products} @ \textit{rate}, defining patterns that have to be matched to actual agents at runtime. For example the rule

\[
\text{Cell}(\text{vol},'S')[\text{Particle}:p + s?]
\rightarrow \text{Cell}(\text{vol} \text{+ norm}(1,0.1),'S')[s?]
\text{\@ \#p * k1 / vol}
\]

describes the elimination of Particle entities that are in Cell[... ] compartments in the cell cycle phase 'S'. The volume of the Cell is represented by the variable vol, i.e., the rule applies for any volume. The reaction rate is influenced by the cell volume, the number of particles \#p in the cell, and a reaction constant k1. Further, the volume of the Cell is increased by a sample of a Normal distribution with mean 1 and variance 0.1. In addition, the reaction rule requires a special variable \#s? that binds all subentities of Cell that shall not be modified.

The above reaction rule emphasizes the importance of attributes and of the possibility to define arbitrary functions to access and determine attribute values and contents. Attribute-based interactions are a powerful tool in modeling languages for open, concurrent systems [1]. For example in cell biology, the possibility to constrain behavior based on attributes allows an easy integration of different abstraction levels, whether these are genetics beyond mass action [7] or modeling spatial dynamics in a non-spatial approach [5,9]. Consequently over the last decade, several modeling languages from various application areas have adopted similar design choices to support the modeling of more complex agents [6,8]. For our case study we will employ ML-Rules which is an expressive rule-based modeling language, that supports multi-level modeling and arbitrary attributes. Its syntax is fairly similar to the syntax introduced above, and its semantics is based on a CTMC semantics [5].

3 ROUND-BASED SUPER-INDIVIDUALS

As a first proof of concept and evaluation for dynamic super-individuals with configurable degree of aggregation, our approach will use fixed, user-defined time slots of aggregated and non-aggregated execution, i.e., alternating rounds of fusion and fission. Thereby, the fusion and fission steps themselves mark the transitions between the two phases, spent either in individual mode or super-individual mode. Each phase has a certain length in units of simulation time, further denoted by $\Delta t_i$ and $\Delta t_s$. As an important aspect of this approach, fusion and fission steps are always performed on the complete set of entities, taking effect instantaneously before the simulation time advances. The phase progression is depicted schematically in Figure 1. There, starting in the individual mode, similar entities, represented by the same color, are aggregated and dissolved over consecutive rounds. After a fixed time interval $\Delta t_i$ the individuals are aggregated. Then, the simulation proceeds in the super-individual mode until $t + \Delta t_s$, i.e., the time of the next fission event, etc. The two modes have complementary purposes. Whereas the super-individual mode is characterized by reduced computational effort, the individual mode grants error compensation through diversification. The two user-defined parameters $\Delta t_i$ and $\Delta t_s$ control the phase lengths, and thus the proportion of super-individual influence. They can therefore be used to adjust the compromise between simulation speed and accuracy, in particular if $\Delta t_s \gg \Delta t_i$ more simulation time is spent in the super-individual mode than in the individual mode, and thus more runtime can be saved.

The round-based procedure may be implemented as an additional step for different types of simulators, e.g., the Gillespie stochastic simulation algorithm [4], but also hybrid or tau-leaping implementations. In each iteration of the simulation loop, the simulator checks whether the next phase transition is due. Depending on the current phase, either the fusion method (if in the individual phase) or the fission method (if in the super-individual phase) is initiated. The fusion method sequentially visits every compartment of the multi-level model and sorts all subentities of the current compartment into groups and aggregates them into super-individuals by approximating the subpopulations with probability distributions. Analogously, the fission method also visits all compartments, and for each compartment it dissolves the contained super-individuals into single individuals by sampling from the respective probability distributions. To work, the approach needs as inputs similarity classes on the attributes, as well as probability distributions to sample from the defined classes. For all continuous attributes we assume (truncated) Normal distributions over intervals $[\text{min}, \text{max}]$ described by mean, lower bound $\text{min}$, upper bound $\text{max}$, and variance = $\text{max} - \text{min}$, whereas for discrete attributes the distribution is defined by value counters. Finally, an additional set of rules has to be generated that functions on the super-individuals instead of single individuals, and preserves firing rates and species counts during the super-individual phases [10,12]. The rule transformations are based on previously defined classes of rules, e.g., elimination of content, change in attribute values, etc.
4 CASE STUDY

To evaluate the performance of the round-based scheme in terms of accuracy and speed, we implemented it within the Java-based ML-Rules stochastic simulator [5], and executed simulations with an ML-Rules model of a fish habitat. This model is a simplification of a model developed to further understand the physiology and ecology of Eastern Baltic Cod [11], and incorporates as main features of interest hierarchically nested individuals with partly continuous attributes, and complex behavioral rules that determine the behavior of these individuals. Two types of agents, i.e., fish and food (typically smaller fish), share the common habitat. The fish are characterized by their mass, health, and sex. The more food they consume the more they grow. The food is assumed to regenerate at a certain rate in the habitat it shares with the fish, however, once consumed it is successively degraded in the process of the fish growing. Over time the fish’s health decreases which ultimately causes them to die. Changes in the habitat conditions, for example changes in water temperature, or in the abundance of food influence the fish’s development, in particular how fast they grow. The initial state of our model contains one habitat with 100 fish (50 females plus 50 males), 2000 food entities, and initial water temperature of 20. Each fish is initialized with a mass of 20 and a health of 1000. The case study model is given in the following:

```java
//Habitat (temperature)
Habitat(num[]);
//Fish (mass, health, sex)
GF(num, num, string[]);
//Fish food
FF();

//Declining health
GF(m, h, s) [gf?] -> GF(m, h-1, s)[gf?] @ 0.5 ;

//Ingesting food
Habitat(t)[GF(m, h, s)[ gf?] + FF:ff + s?] ->
Habitat(t)[GF(m, h, s)[FF + gf?] + s7] @ 0.02 * #ff;

//Growing -- Digesting food and increasing mass
Habitat(t)[GF(m, h, s)[FF + gf?] + s?] ->
Habitat(t)[GF(m*(norm(1,0.5)), h, s)[ gf?] + s?] @ 0.005 * t * m;

//Removing a fish
GF(m, h, s) -> @ if (h<400) then (1/h) else 0.001;

//Adding a new fish
Habitat(t)[s?] -> Habitat(t)[GF(20, 1000, unif("F","M")) + s?]
@ if (count(s?, 'GF')>0) then 0.05 else @;

//Regeneration of FF
Habitat(t)[s?] -> Habitat(t)[s? + FF] @ r;

//Temperature
Habitat(t) -> Habitat((norm(10,2)))
@ EACH 360.0 .001;
```

The round-based simulator was configured to cluster the fish of our habitat model according to their sex (F or M), and six exemplary mass classes of equal size [25, 125], [125, 225], [225, 325], [325, 425], [425, 525], [525, 625], which gives us a total of twelve similarity classes and thus twelve potential super-individuals during the simulation. We then analyzed 16 configurations of the parameters $\Delta t_1$ and $\Delta t_2$. In general, there are two ways for steering the simulator: (1) setting the period of the round-based execution, i.e., $p = \Delta t_1 + \Delta t_2$ and (2) setting the ratio $\Delta t_1:\Delta t_2$, e.g., 1:1 for phases with equal length, or 1:2 to let the super-individual phase be twice as long as the individual phase. The first configuration, called the standard configuration, corresponds to a regular SSA execution without any phase changes. For a fixed simulation duration of 2500.0 for all experiments, this results in $\Delta t_1 = 2500.0$ and $\Delta t_2 = 0.0$. In the other configurations three periods ($p = 2, 100, 200$) and for each of them five different ratios of unclustered and clustered execution (1:1, 1:2, 1:3, 1:4, 1:5) were evaluated. The five configurations for $p=100$ are listed below; all other configurations can be derived analogously.

1. Configuration standard — $\Delta t_1$ is 2500.0 and $\Delta t_2$ is 0.0
2. Configuration 100-1:1 — $\Delta t_1$ is 50.0 and $\Delta t_2$ is 50.0
3. Configuration 100-1:2 — $\Delta t_1$ is 33.333 and $\Delta t_2$ is 66.667
4. Configuration 100-1:3 — $\Delta t_1$ is 25.0 and $\Delta t_2$ is 75.0
5. Configuration 100-1:4 — $\Delta t_1$ is 20.0 and $\Delta t_2$ is 80.0
6. Configuration 100-1:5 — $\Delta t_1$ is 16.667 and $\Delta t_2$ is 83.333

The experiments were carried out on a machine with 64-bit Windows 10, Intel®Core™i7-7820HQ CPU (4 cores, 2.9 GHz), 16 GB RAM, and Java Version 1.8. For each of the chosen configurations 35 replications were executed in batches of seven. All results were obtained at simulation time point 2500 and include the fish with their three attributes, mass, health, and sex. Runtime measurements were carried out using the Java method `System.currentTimeMillis()`.

Figure 2 shows that primarily the ratio is responsible for the speedup. For the 1:1-ratios the runtime could be improved by approx. factor 1.5-1.6, whereas the 1:5-ratios led to an overall speedup of approx. 2.4-3.0. However, there is a saturation, i.e., ratios > 1.5 are becoming less and less beneficial with respect to the traded accuracy. Figure 3 compares the simulation results of different parameter ratios with $p = 100$ for the attributes health and mass. The results for $p = 2$, and $p = 200$ were omitted for brevity, as they did not differ significantly. We observe similar frequency distributions for all configurations, however with increasing ratio the data becomes more sparse, especially for “low-health-high-mass” fish.

![Figure 2: Runtimes for the different periods and ratios.](image-url)
We have introduced an approach for the reduction of agent-based models by fusing agents into super-individuals and splitting super-individuals into agents dynamically during simulation. Thereby the simulation proceeds in alternating rounds of stochastic execution without aggregation and execution based on aggregated super-individuals. The results of the first case study presented here demonstrated that the ratio of super-individual execution and individual execution can indeed be used to steer the balance between runtime and accuracy of the simulation. In contrast, the period length, i.e., the concrete time points or frequency at which fusion and fission steps take place, played only a minor role for the simulation results. However, for other models that exhibit more complex or irregular behaviors, this parameter might be relevant. Therefore, in future work we will explore the impact of (round-based) super-individuals on other, more complex models, e.g., the cod model of Pierce et al. [11], with the goal of discovering patterns in model behavior that can be used to automatically and adaptively configure the simulator, and to make the method less dependent on domain knowledge, i.e., to adaptively find suitable similarity classes, as well as the time points for merging or splitting the super-individuals. Furthermore, we aim to study the relationship with other approaches like CTMC lumping in a formal way, and in particular look at formal error bounds.

REFERENCES


