Proceedings of the 2019 Winter Simulation Conference
N. Mustafee, K.-H.G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, eds.

TOWARDS ADAPTIVE ABSTRACTION IN AGENT BASED SIMULATION

Romain Franceschini

University of Corsica Pasquale Paoli
UMR CNRS 6134
Campus Grimaldi
Corte, 20250, FRANCE

Simon Van Mierlo
Hans Vangheluwe

Department of Mathematics and Computer Science
University of Antwerp - Flanders Make
Middelheimlaan 1
Antwerp, 2020, BELGIUM

ABSTRACT

Humans often switch between different levels of abstraction when reasoning about salient properties of systems with complex dynamics. In this paper, we study and compare multiple modelling and simulation techniques for switching between abstractions. This improves insight and explainability as well as simulation performance, while still producing identical answers to questions about properties. Traffic flow modelled using an Agent Based Simulation formalism is used to demonstrate the introduced concepts. The technique requires explicit models (1) of the dynamics of both individual cars and of emergent "jams", (2) of the conditions –often involving complex temporal patterns– under which switching between the levels of abstraction becomes possible/necessary and (3) of the state initialization after a switch. While aggregation is natural when going from detailed to abstract, the opposite direction requires additional state variables in the abstract model.

1 INTRODUCTION

Models typically only capture specific properties of the system under study and result in particular abstractions. By definition, an abstraction is a simpler representation: it is meant to contain carefully selected information, captured in a way that only satisfies the aspects the modeler wants to study. The idea that a model is an incomplete representation of a system, only relevant to answer specific goals is well addressed in general system theory literature (Minsky 1965; Stachowiak 1973).

Thus, before the formulation of a model, an important task is to determine the most appropriate level of abstraction at which to solve the problem (Iwasaki 1993). The study of complex systems may involve many modeling perspectives, eventually with heterogeneous domains and different goals, which suggests multiple abstraction levels to be considered. Multi-level models can be used to solve several modeling problems (Morvan 2013) which are related to the concerns of multi-paradigm modeling: (1) *the modeling of cross-level interaction*, (2) *the coupling of heterogeneous models* and (3), *the dynamic adaptation of the level of abstraction*. For this paper, we mainly focus on *adaptive abstraction*, where levels are automatically switched to other levels according to the simulation context at runtime. This technique has various benefits since it may improve:

- insight and explainability of the system, by automatically detecting emergent behaviors and corner cases, all being enablers for a switch to another abstraction level to happen;
- simulation performances in terms of computational needs, both CPU and memory wise.

However, switching from one level to another during simulation certainly has an impact on results. This paper aims at studying this impact on a different set of results, in order to find the best trade-off between simulation execution time and accuracy. Based on a traffic network model, we compare obtained results with

different variants of adaptivity to the results obtained with the fully microscopic model, where no adaptivity is involved. Agent-based models (ABM) are often used to microscopically describe complex systems, and their simulation can both reveal interesting collective behaviors and require important computational needs. As adaptive abstraction aims at reducing computational needs through pattern recognition, this paradigm is well-adapted to the study of adaptivity.

This paper is organised as follows. In the next section, we provide some background about multi-level modelling and adaptive abstraction. We then present our traffic modelling case study used to compare the results of several adaptive variants of a model. Then, we discuss the results and the remaining research challenges to provide a generic modelling and simulation framework for adaptive abstraction before concluding the paper.

2 BACKGROUND

Multi-level modelling received growing interest, particularly in the agent-based community since the need to simulate large populations as individuals (Scheffer et al. 1995) have been expressed. In this section, we first briefly provide some background about the notion of abstraction as well as adaptivity. We then present related works specifically related to adaptive abstraction in ABM.

2.1 Model abstraction

While humans switch naturally back and forth among abstractions without any pre-defined hierarchy, an explicit representation of levels and of their relations must be defined for a program to switch between levels. In this section, we describe what a level includes, besides the model itself.

An abstraction level is related to the amount of information found in a model (Benjamin et al. 1998): the more information a model encodes, the less abstract it is. Conversely, the less information it contains, the more abstract the model is. This amount of information is tightly coupled to the viewpoint on the system, which includes assumptions, observations, and modelling goals. Such a viewpoint helps decide what information is relevant, in what detail and thus, is essential for an abstraction level to be shaped.

The very idea of abstracting a model is to aggregate information by relaxing one or several dimensions, while the inverse of abstracting is about disaggregation. There are many dimensions along which a model can be abstracted (Iwasaki 1993), namely:

1. the *structural* dimension, which consists of lumping together a group of components that are spatially close;
2. the *functional* dimension, where several components are lumped together to achieve a distinct, higher-level function;
3. the *temporal* dimension, where abstraction involves ignoring a behavior over a given period of time;
4. the *quantitative* dimension, where abstraction involves ignoring relative differences in state variables values.

Switching from one level to another does not necessarily involve only one of these dimensions being abstracted. For example, a structural aggregation may come with a quantitative aggregation: models that are physically close are lumped together, and a meaningful state for the new model is constructed from the more detailed ones. Some formalisms already support hierarchical constructs, providing means to model multiple levels of abstraction. Some ABM platforms which provide dedicated modelling languages such as GAMA (Grignard et al. 2013) or NetLogo (Tisue and Wilensky 2004) offers multi-level capabilities since agents can be nested in other agents, also allowing cross-level interaction between multiple levels.

A switch may cause a dimension to disappear entirely from a level. In such case, the question which arises is one of the particular formalism to express the abstraction. If the abstraction of a spatially-explicit model results in a spaceless model, the formalism used in the former is likely to be inappropriate to represent the latter. This means adaptive abstraction may involve heterogeneous models to be used. For

example, an agent-based model can eventually be abstracted to a macroscopic model, where the population is considered as a whole and expressed in another formalism such as System Dynamics, causing the space dimension to be irrelevant altogether.

To be able to partially order the different abstractions, upward and downward relationships between abstractions should be defined so that a pre-defined hierarchy can be used to determine which abstraction to switch to depending on the context. Another way to infer level hierarchy is by explicitly defining scales and precision units (Iwasaki 1993). Although the author only focused on the temporal dimension (simulation time), we believe this idea can also be applied to structural and quantitative dimensions by defining precision scales for space and states, respectively. However, such an idea requires the model visibility to be a "white box". More generally, visibility is an important concern regarding features to consider for an adaptive abstraction approach. Thus, an abstraction includes the following metadata: (1) the viewpoint (a.k.a experimental frame), (2) the featured dimensions and (3) their scales, (4) the formalism(s) used to express the model, (5) the visibility of the model and the model itself.

2.2 Adaptivity

Adaptive abstraction has been particularly studied for agent-based models (ABM), which resulted in the Multi-Level Agent-based Modeling (ML-ABM) domain. An extensive literature survey on this topic was recently proposed (Morvan 2013), which raise the following challenges to perform adaptive abstraction: (1) the definition of generic representations for aggregated entities, and (2) the detection of emergent phenomena. A section is dedicated to each.

2.2.1 Representing aggregates

Recently, Mathieu et al. (2018) provided four Design Patterns defined after a literature study that can be generally applied to perform the aggregation or the disaggregation of agents. Such a process has different characteristics: whether information transformation is involved, whether the behaviour is transferred to the other level, or if both levels evolve independently. Table 1 lists the patterns and summarise their

Table 1: Aggregation/disaggregation patterns identified for multi-level ABMs.

Pattern	Behavior	Information transformation
ZOOM	transferred	yes
PUPPETEER	transferred	no
VIEW	unchanged	no
COHABITATION	distinct	no

characteristics. The ZOOM pattern is destructive in both ways, meaning behaviour is transferred to the new level and states are transformed: information must be reduced when aggregating and re-constructed when disaggregating. The PUPPETEER pattern is the closest to the ZOOM one since behavior is also transferred to the new abstraction. However, individual states are gathered by the upper-level and kept as-is, meaning there is no information transformation involved. As its name suggests, the VIEW pattern is only here to provide insight to the user. The behaviors and states from the lower level remain as-is; the upper-level is only here to keep track of the individual models without having any specific behavior, as in Caillou and Gil-Quijano (2012). Finally, the COHABITATION pattern features levels with distinct behaviours and states, and with feedback loops between levels.

Previous works either use one of these patterns or a combination of them. For example, Sarraf-Shirazi et al. (2014) approach is mainly based on the PUPPETEER pattern to aggregate agents that are physically close, i.e. along the structural dimension, and delegate their behavior. A relatively small amount of information is added to the aggregate level during aggregation, but all individual states are also kept.

Similarly, Wendel and Dibble (2007) work is also based on this pattern. Both works claim a speedup with adaptivity enabled.

Sharpanskykh and Treur (2011) apply multiple methods on a social diffusion model and compare results both in terms of computational efficiency and of approximation errors. To our knowledge, this comparative study is the closest work found in the literature that allows observing accuracy versus performances, as we aim at. However, disaggregation is not involved and all variants of their adaptive models are based on the ZOOM pattern. Two approaches are considered to build the upper-level state during aggregation: (1) statistical-based (weighted averaging) and, (2) invariant-based (determining equilibrium state).

Navarro et al. (2011) use a hybrid approach which combines the PUPPETEER and the ZOOM patterns to aggregate agents spatially when a cluster is detected. Individual states are kept when an abstraction is first formed, and information is gradually transformed as the abstraction evolves. Bouha et al. (2015) also use two different patterns to switch between heterogeneous models: ZOOM for adaptivity and PUPPETEER for cross-level interaction.

The first three patterns from Table 1 (ZOOM, PUPPETEER and VIEW) are suitable for adaptive abstraction with different purposes, while the last one is more suitable for cross-level interaction goals. For scenarios seeking to improve simulation performances, the ZOOM and PUPPETEER patterns are the most appropriate ones. Since our objective is to study performance gains, we will focus on those two in this study.

In order to support to some extent multiple levels, many platforms and formalisms have been proposed. For example, Steiniger et al. (2012) extends the DEVS formalism to ease the modelling of environments in ABM models, however, the resulting Multi-Level-DEVS formalism can also be used to implement the COHABITATION pattern, eventually in a heterogeneous way. On the tooling side, the GEAMAS-NG agent-oriented platform (David et al. 2012) was specifically developed to provide multiple levels of abstraction. Similarly, the Janus platform (Galland et al. 2017) has builtin support for multiple abstraction levels. It is described as an holonic platform, referring to the concept of holons (Koestler 1967), which is used by the HMAS (Holonic Multi-Agent System) community to build multi-level agent societies with interaction between abstraction levels (COHABITATION pattern).

Nevertheless, broader ABM simulation platforms such as NetLogo (Tisue and Wilensky 2004) or GAMA (Grignard et al. 2013) also supports the implementation of multiple levels in an integrated and homogeneous way: abstractions are agents, as individuals, and they share entities and the environment. In order to implement the PUPPETEER pattern, the underlying modelling language must support some kind of hierarchy between levels. In the case of the two platforms, this can be done by nesting agents within agents through simple references. To implement the ZOOM pattern, the language must support dynamic structure so that models can be removed and added during simulation, which is already a requirement of multi-agent systems.

The switching semantics between levels also requires an "omniscient" process that is capable of checking domain-specific properties about the system. ABM simulation platforms usually allow this by providing an "observer" agent to which a specific behavior can be associated. In the case of adaptive abstraction, this behavior is about detecting the conditions under which a switch to another level should occur.

2.2.2 Detecting emergent behavior

An important aspect to perform adaptive abstraction concern the conditions under which a switch to a particular level is triggered. A trigger can be either manual or automatic. A manual switch occurs as a result of a user-input when a new modelling perspective on the system is considered by the user. An automatic switch however, is much more challenging since specific model properties must be checked during simulation, which involves a process periodically sampling the model to detect over time if a more appropriate level should be instantiated.

Interacting entities such as agents typically generates emerging properties and behaviors that hold for a certain amount of time, either in an ephemeral or in a long-lived way. While humans easily detect patterns from a particular dimension out of a temporal sequence, adaptive abstraction requires such phenomenon to

be detected by explicitly keeping track of interactions. We should emphasize that the notion of temporal sequence is not related to the temporal dimension defined in section 2.1. Both could be combined to detect a particular phenomenon. For example, a periodic peak in the event distribution of a discrete-event simulation is a temporal pattern over a time sequence. If detected, such pattern could be used as a condition to switch to a more appropriate abstraction where discrete time is used instead.

The detection process involves several distinct steps, namely:

1. *state monitoring*, where properties are checked for a snapshot of the states;
2. *aggregation*, where upward abstraction is done if enabling properties holds since a given amount of time;
3. and *disaggregation*, where downward abstraction is done if disabling properties holds since a given amount of time.

The main difficulties are (1) to recognize possible emergent phenomena over a snapshot and (2), to reinforce these by keeping track of them over time.

For recognition, most existing works in ABM explored abstraction through the structural dimension by aggregating close agents together in a mesoscopic level, but some also explored abstraction through the quantitative dimension, e.g. grouping agents with similar states. Most authors use statistical classification to detect emergence through graph-based clustering methods (Moncion et al. 2010) or clustering algorithms (Gil-Quijano et al. 2012). Formal frameworks were also proposed to detect emergence. David and Courdier (2008) formalize the concept of emerging structures along with emergence revelators and laws. However, the notion of temporal sequence is not addressed, and domain-specific properties about the model cannot be expressed. Chen et al. (2009) allows entities to be located in a hyperspace that includes the structural and temporal dimensions and that can also include other dimensions.

To keep track of properties over a temporal sequence, graphs are often used in the literature. Moncion et al. (2010) use a weighted graph where vertices are the agents and edges represents the likelihood for an emergent property to exist between two agents. The graph is updated each time the state monitoring process sample states, and weights are adjusted depending on agent properties. Strong relations between agents indicates long-lived emergent phenomena. Conversely, ephemeral phenomena disappear from the graph when a relationship is not strong enough to be maintained. Groups are formed using a graph clustering algorithm. However, since their work implements the VIEW pattern, nothing is mentioned about disaggregation.

In a similar way, Sarraf-Shirazi et al. (2014) use a weighted graph to keep track of emergent properties over time. However, since the authors implemented the PUPPETEER pattern, vertices can be either agents or meta-agents from upper levels. This way, when a cluster is detected, the graph is also transformed. Vertices corresponding to agents from the group are deleted, a new vertex is inserted and edges are adjusted. As for disaggregation, authors' approach is to assign a confidence value to each group. Groups are periodically and temporarily disaggregated to observe over time if the underlying agents still exhibit the same properties. Confidence is adjusted accordingly and agents are aggregated back if the group still holds. While this technique allows performing disaggregation without checking additional properties, it is less appropriate if used together with the ZOOM pattern. Periodic disaggregation/re-aggregation as a validation process is less costly when individual agents are kept in memory than if they are re-allocated and freed over and over.

3 CASE STUDY: APPLICATION TO TRAFFIC MODELLING

In order to study the trade-off between results accuracy and simulation execution time that adaptive abstraction can offer, we base our case study on an agent-based traffic network model. Road traffic systems feature complex dynamic interactions between vehicles and have self-organisation properties and observable emergent behaviors such as traffic jams. Therefore, traffic modelling is an appropriate candidate to study dynamic adaptation between levels of abstractions.

This section is organised as follows. We first introduce the microscopic model. We then present the different variants of adaptive abstraction we use for this case study. Finally, we present the results of our experiment.

3.1 Traffic network model

Road traffic networks are typically studied across different levels of abstractions, macroscopically or microscopically. The first focuses on macroscopic variables such as flow, mean speed or density and are usually expressed by mathematical relations. Microscopic traffic models focus on drivers behavior and on their interactions with other vehicles, which requires high computational needs. The benefits of introducing intermediate levels in traffic simulation were already stated: previous works also used traffic networks to study ABM multiple level capabilities both for cross-level interaction (Tchappi et al. 2018) and for adaptive abstraction (Bouha et al. 2015) purposes.

We describe a microscopic model which features a simple car-following behavior, where each vehicle tries to maintain its preferred speed and adjust it when a car is ahead. With heterogeneous speeds and no overtaking allowed, traffic jams are likely to occur. Traffic jams can be recognized through space locality, which is used for our adaptive variants to construct aggregates. The different levels of abstraction in our traffic network example are based on spatial clustering. They are expressed in the same formalism and are integrated, meaning they interact and share the same environment and the same entities such as cities and road segments. Such a model can be implemented in existing ABM tools as we mentioned in section 2. Our implementation is based on NetLogo 6.0.4.

Cars and jams, that represent different abstraction levels, are modelled as agent types: $C \subset V$ and $J \subset V$, respectively, with $C \cap J = \emptyset$ and where V is the set of vehicles. Road segments and cities are defined by the R and T sets and form a road network through the graph $G_N = (R, T)$. The road network is a representative example which lacks some details since the model does not feature multiple lane roadways and no overtaking behavior is taken into account.

Let $S = \bigcup_i s_{i \in A}$ be the set of agent states. For all $i \in C$, cars states are defined by the following n-tuple:

$$s_i = (p_i, t_i, v_i, v_{pref_i}, r_i, tt_i)$$

where $p_i \in \mathbb{R}^2$ is the location of the vehicle, $t_i \in T$ is the target city to reach, $v_i \in [0,1]$ is the current speed of the vehicle, $v_{pref_i} \in [0,1]$ is the preferred speed of the driver, $r_i \in R$ is the current road segment and tt_i is the transit time of the vehicle along the current segment.

Based on a given number of cities n_{cities}, a random road network is generated using a preferential attachment process. The network is then laid out in space. For each road segment $i \in R$, the state s_i is defined by the following n-tuple:

$$s_i = (v_{lim_i}, \bar{tt}_i, m_i, m_{arr_i})$$

where $v_{lim_i} \in [0,1]$ is the speed limitation on this road segment, $\bar{tt}_i$ is the mean transit time on this road segment, and m_i is the number of vehicles on the roadway, and m_{arr_i} is the number of cars that passed this segment. The density represents the number of vehicles present on a road segment at a given time. Thus, the density of a road section is given by $k_i(t) = m_i(t)/l_i$, with l_i the length of the road segment. The mean road transit time is a simple moving average and evolves according to the following relation:

$$\bar{tt_i}(t + \Delta t) = \frac{\bar{tt_i}(t).m_{arr_i}(t) + tt_j(t)}{m_{arr_i}(t) + 1} \qquad \text{with } i \in R \text{ and } j \in C$$

As for jams, since the state depends on the variant of the adaptive abstraction used, it is given in the next section. However, each variant $i \in J$ share the following variables: p_i the position, n_i the number of cars stuck in the traffic jam, tt_i the transit time of the jam along the current road segment r_i.

Vehicles are either driving on a road segment or waiting inside cities. If waiting inside a city, cars have a probability of leaving determined by the $\rho_{leave} \in [0,1]$ parameter. If leaving, a car $i \in C$ randomly

choose a final target city $t_i \in T$ and a path towards t_i is computed following the road network G_N. Once they have a target, cars will adjust their speed to move towards the target with a car-following behavior. Each driver will always try to maintain its desired speed v_{pref_i}, with respect to the allowed speed on the current segment $v_{lim_{t_i}}$ (no speeding assumed). If a vehicle in front is present, the driver decelerates to match the other's speed.

The macroscopic output of the ABM that we use to compare the results of the model variants is given by variables such as transit time, speed and density. The average transit time can be calculated from a road segment and is given by $\bar{tt}_{roads}(t)$ while the average vehicle speed is given by $\bar{v}(t)$. Finally, the mean density is calculated with $\bar{k}(t)$ using the number of driving vehicles and the total roadway length. All three are given as follows:

$$\bar{tt}_{roads}(t) = \frac{\sum_{i \in R} \bar{tt}_i(t)}{Card(R)} \qquad \bar{v}(t) = \frac{\sum_{i \in C} v_i(t) + \sum_{i \in J} s_i(t).n_i(t)}{n_{cars}} \qquad \bar{k}(t) = \frac{\sum_{i \in R} m_i}{\sum_{i \in R} l_i}$$

Despite the simplicity of this agent-based traffic model, emergent properties such as traffic jams appear on multiple road segments during simulation. Involved vehicles then exhibit similar behavior, driving at a similar speed until the next city. The next section presents several adaptive variants which take advantages of these traffic jams to reduce the number of simulated agents.

3.2 Adaptive variants

To study different types of adaptivity, we propose several variants of the traffic model based on the PUPPETEER and on the ZOOM patterns (see section 2.2.2). While the former is simply about delegating behavior and involves no state transformation, the latter does introduce additional state variables to represent the aggregates (in a reduced way). The way aggregation and disaggregation reduce and reconstruct information is important since it can eventually increase the error drift from the original model. As the (dis-)aggregation process is domain-specific and requires the modeller to make explicit choices, we explore distinct ways of performing those choices. Sharpanskykh and Treur (2011) explored two ways of abstracting data for the ZOOM: by determining equilibrium states or by using statistics. Since the authors obtain better results in terms of execution time and in terms of errors with the statistical approach, we compare different variants of statistical aggregation. Those are also compared with the PUPPETEER pattern and with the microscopic variant of the model.

For the variants based on the ZOOM pattern, we use more or less naive statistics and probabilities across different dimensions. For quantitative values such as speed or preferred speed, a naive approach is to gather incomplete information over the distribution. For example, only calculating mean values makes the reconstruction of the statistical dispersion arbitrary. Conversely, calculating statistical dispersion adds some overhead but makes the reconstruction of individual states more precise. Dispersion is a key element to automatically detect cases where abstraction is still relevant while it evolves. If we consider space (or any other dimension), a squeezed dispersion reveals a strong space locality. If the distribution stretches over time, it is an indicator that the flock is collapsing.

However, not all vehicle characteristics are meaningfully represented by spatial values. As cars have different target cities while being stuck in the same traffic jam, the question of how to represent destinations in the aggregate arise. Again, this can be made in a more or less destructive way. Final destinations could be ignored altogether during aggregation, but a random city must be assigned to cars during disaggregation. Another solution is to count occurrences and to preserve the most recurrent destination, but all cars in the traffic jam will end up to this destination. These two solutions are very destructive as they change final destinations of the cars and might increase the error on results. In order to keep the final destinations, a solution is to keep a vector of all final destinations. During disaggregation, cars have a similar probability to go to any target from the vector. Finally, the most precise solution is to keep occurrences so weight can be assigned to each destination, and better assign event probabilities.

As there are many possible variants, we had to restrict our study on a few of them. The baseline is based on the microscopic variant described in section 3, where no adaptivity is involved. A second variant does not perform information transformation during aggregation and is based on the PUPPETEER pattern. For the ZOOM pattern, we selected three different variants. The most naive only gather mean values from the quantitative values and keep a vector of unique final destinations. Another variant follows the same principle but only gather the minimum speeds instead of the mean. Finally, a third variant gathers more information: mean values are calculated along with the standard deviation, which is used when disaggregating. The destinations are also based on a vector, but it is weighted so probabilities better reflect the original behavior. As for the location and the length of the "jam", it is calculated based on the number of vehicles and on the minimum allowed distance headway in the model, and thus, is trivial to reconstruct.

Regarding emergence detection, we adopted a graph-based approach similar to Sarraf-Shirazi et al. (2014). The state monitoring process checks agent properties every Δ_{mon} and update the weighted graph (V,L), where L is the set of interaction links that connects the set of vehicles (cars and jams) V according to possible detected emergent phenomena. When a strong interaction is detected between vehicles, an initial weight of Δ_{inc} is associated with the edge if it does not exist already. Otherwise, the weight is incremented with Δ_{inc}. For all existing edges that were not reinforced, associated weights are decremented with Δ_{dec}. An edge is deleted if its weight reaches 0.

The aggregation process happens every Δ_{agg} to analyse the state of the (V,L) graph to detect clusters of strongly interacting agents. A new subgraph is first built from (V,L), where only vertices connected through edges having a weight greater than the θ_w threshold are considered. Clusters are then identified by cutting the graph using a weakly connected components algorithm. Each cluster of vehicles represents emerging collective behaviours and is used to perform an aggregation.

3.3 Results

This section presents the results of the comparative study of each variant of the traffic model. After exposing our methodology in a first subsection, two subsequent ones discuss the model output results and the execution performances, respectively.

3.3.1 Methodology

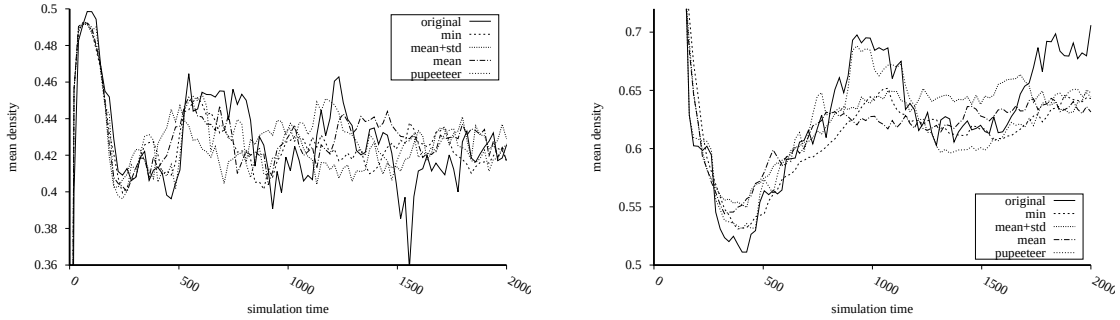
The hardware environment used to run the study is based on an Intel(R) Core(TM) i7-4870HQ CPU @ 2.50GHz, 16 GB (1600 MHz DDR3) of RAM, and an Apple SM0512G SSD hard drive. As for the software stack, NetLogo 6.0.4 is running on the Java HotSpot(TM) 64-Bit Server VM (build 25.0-b70, mixed mode) and relies on macOS 10.14.4. Mean wall clock time of 5 repetitions is measured for each variant of the model. Results used for comparison are the following macroscopic outputs: mean transit time, mean road density, and mean car speed over 2000 time steps. To compare the results on a smaller and on a larger scale, two parameters are varied: the number of vehicles and the number of cities (which impacts the size of the road network). The fist pair of parameters is based on $n_{cars} = 100$ and $n_{cities} = 10$ while the second pair is based on $n_{cars} = 400$ and $n_{cities} = 20$. Due to some noise in the results, density and speed plots (Figure 1 and 2) have been smoothed using a cubic spline interpolation for better clarity.

3.3.2 Model outputs

Across all variants, density results (Figure 1) are the closest from the original model. The maximum relative standard deviation (RSD) from the whole set is $\pm 8.25\%$ for the small scale and $\pm 6.94\%$ for the larger scale. The average RSD is $\pm 2.44\%$ for both scales. However, this is not surprising since the approximation of the spatial distribution is aggregated in a similar way for all variants: jam length is calculated based both on the minimum distance headway and the number of stuck vehicles. Once calculated, the spatial dispersion does not vary with the behaviour of the abstracted traffic jam.

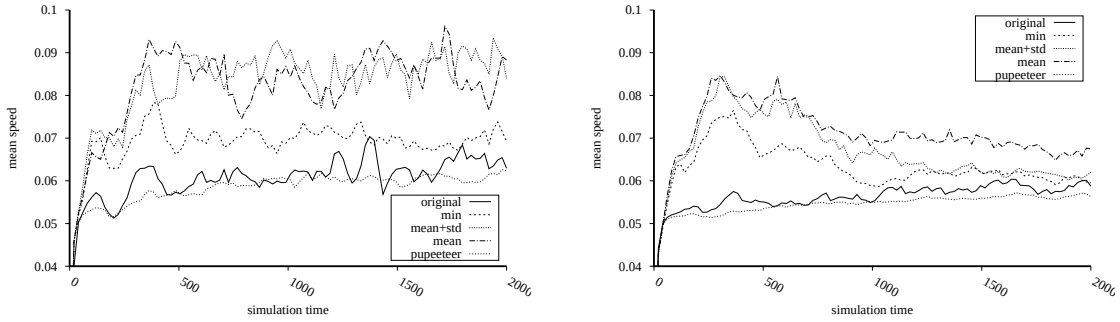
Regarding speed results (Figure 2), overall dispersion across all variants is higher. Here, the maximum and average RSD is $\pm 23.28\%$ and $\pm 16.75\%$, respectively. From the variants implementing the ZOOM pattern, the min one seems to produce the closest results from the original model. However, this makes sense since in the model we described, vehicles do not accelerate once the preferred speed or the maximum allowed speed is reached. All cars in a jam undergo the preferred speed of the front car, which is the minimum preferred speed from all the values. Thus, taking the mean value -with or without dispersion approximation- does not reflect the original behavior, especially since the dispersion is likely to be significant. While this is true for the preferred speed, the aggregation of the actual speed does not follow the same principle. Taking the minimum or the mean value for the actual speed does not make a significant difference since the dispersion is likely to be squeezed. Indeed, during aggregation, all cars already drive at a similar speed. The results between the two mean variants show that having an approximation of the dispersion does not impact the results in this case.

Since the transit times results (Figure 3) are directly influenced by vehicle speeds, similar observations can be made. However, the large scale simulation results (Figure 3b) show that the variant with more detailed information about final destination is more accurate than its counterpart. This can be explained by the fact that road segments have heterogeneous maximum allowed speed. Thus, the roads taken by the vehicles have a direct influence on transit times results.



(a) Mean density ($n_{cars} = 100$, $n_{cities} = 10$) (b) Mean density ($n_{cars} = 400$, $n_{cities} = 20$)

Figure 1: Mean road density results for each variant on a small scale (a) and a larger scale (b).

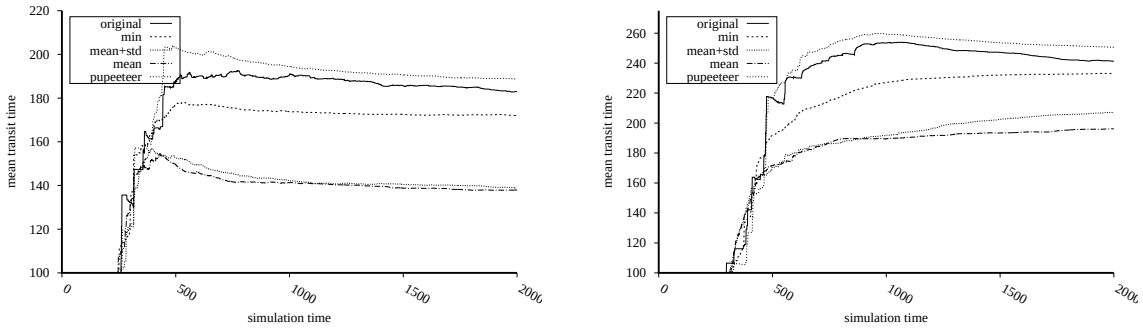


(a) Mean speed ($n_{cars} = 100$, $n_{cities} = 10$) (b) Mean speed ($n_{cars} = 100$, $n_{cities} = 10$)

Figure 2: Vehicle speed results for each variant on a small scale (a) and a larger scale (b).

3.3.3 Execution speed

Table 2 gives the measured wall clock time in seconds along with the standard deviation for each model variant. The PUPPETEER implementation slightly improves runtime performances both for the small scale



(a) Mean transit time ($n_{cars} = 100$, $n_{cities} = 10$) (b) Mean transit time ($n_{cars} = 400$, $n_{cities} = 20$)

Figure 3: Mean road transit times results for each variant on a small scale (a) and a larger scale (b).

and for the large scale experiment (1.12X and 1.4X respectively). However, this is not surprising since previous works (Sarraf-Shirazi et al. 2014) already shown that this approach improves performances. On a small scale, variants based on the ZOOM pattern also improve simulation performance, with a speedup between 1.16 and 1.60. The variant taking mean values with an approximation of the dispersion offers better performance in this case, which can be surprising, since having an extra calculation for the dispersion adds a slight overhead during aggregation. On a larger scale, however, the best performance results are given by the minimum approach variant with a speedup of 6X, outperforming other ZOOM variants, which obtain a speedup of approximately 2.69X.

Table 2: Execution time for each variant and for each configuration, wall clock time (s) measured.

Variant	**Smaller scale**		**Larger scale**	
	Execution time (s)	Std. dev.	Execution time (s)	Std. dev.
Micro	15.8286	0.217334535	129.4378	10.77029241
Min	11.75244444	1.477378092	32.06522222	6.331938976
Mean	13.5938	0.488359192	48.5176	2.904819926
Mean + std	9.845	0.380067099	47.7462	15.27419781
Puppeteer	14.0872	0.542739072	92.44133333	1.280585934

4 DISCUSSION

The traffic network example shows that adaptive abstraction can be used to speedup simulations in a more or less effective way. The result set presented in the previous section suggests that the best trade-off between performance and result accuracy is either given by a conservative approach such as PUPPETEER or by a destructive approach such as ZOOM with aggregation rules that closely match the domain-specific properties of the model (i.e. the minimum variant). In our case, the former has closer results to the original model while the latter has more distant results, but is significantly faster. The domain-specific knowledge encoded to express when and how to switch between abstractions is crucial. Appropriate choices to choose what to keep or re-construct during aggregation and disaggregation have to be made to reduce errors.

An open research question is about expressing the tolerance over errors with regard to certain value properties of the system. This would allow the definition of a set of properties of interest that should be satisfied at all times regardless of the abstractions that are switched to. Explicitly bounding tolerance is a way to allow relative errors while considering the results to be correct with respect to the properties of interest. Such an approach would allow a particular abstraction to be considered substitutable over a more detailed model.

Another area of research is related to multi-formalism adaptive abstraction. To our knowledge, previous works only focused on adaptivity over the same formalism, as we did in this paper. However, the most appropriate formalism for a particular abstraction may be different from the formalism used to express another. If we consider the traffic model example, a dense road segment where a large traffic jam occurs, with a very low car flow, can be a condition to switch to a queuing model of the road segment. Such an approach requires a general framework for monitoring and adapting the network of models, and can be integrated into a multi-paradigm tool or defined as an external process.

5 CONCLUSION

In this paper, several adaptive techniques allowing to switch between different model abstractions during simulation are compared. Particularly, we focus on agent-based simulation (ABS), which often requires important computational needs and exhibits emerging properties. On one hand, this work is an opportunity to give an overview of existing adaptive abstraction techniques in agent-based models. On another hand, it is an opportunity to compare existing approaches and to study their respective impact on results in terms of errors and performances, to find the best trade-off. The different techniques to monitor and detect specific temporal sequences used to recognize emergent properties, to perform aggregation, as well as disaggregation and how to represent abstractions is discussed. A comparative study based on a traffic modelling example first describes the model as well as different adaptive abstraction variants. Performance results are compared to several outputs of the model, which suggest the best trade-off is provided by variants where the associated aggregation function either keeps all individual states or better grasp domain-specific properties of the model. Finally, we discuss open research challenges for a general modelling and simulation adaptive abstraction framework to be defined.

As future work, we will attempt to introduce a modelling and simulation framework for switching between abstractions which address two main concerns: (1) formalize the notion of substitutability between abstractions through the explicit modelling of properties of interest and (2), allowing heterogeneous models to be switched from one formalism to another.

REFERENCES

Benjamin, P., M. Erraguntla, D. Delen, and R. Mayer. 1998. "Simulation modeling at multiple levels of abstraction". In *Proceedings of the 30th Conference on Winter Simulation*, edited by D. Medeiros, E. Watson, J. Carson, and M. Manivannan, 391–398. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Bouha, N., G. Morvan, H. Abouaïssa, and Y. Kubera. 2015. "A First Step Towards Dynamic Hybrid Traffic Modeling". In *Proceedings of the 29th European Conference on Modelling and Simulation*, edited by V. M. Mladenov, G. Spasov, P. Georgieva, and G. Petrova, 64–70: European Council for Modeling and Simulation.

Caillou, P., and J. Gil-Quijano. 2012. "SimAnalyzer: Automated Description of Groups Dynamics in Agent-based Simulations". In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 3*, 1353–1354. Valencia, Spain: International Foundation for Autonomous Agents and Multiagent Systems.

Chen, C.-C., C. D. Clack, and S. B. Nagl. 2009. "Identifying Multi-Level Emergent Behaviors in Agent-Directed Simulations using Complex Event Type Specifications". *SIMULATION* 86(1):41–51.

David, D., and R. Courdier. 2008. "Emergence as metaknowledge: refining simulation models through emergence reification". In *Proceeding of the 2008 European Simulation and Modelling Conference*, edited by C. Bertelle and A. Ayesh, 25–27. Le Havre, France.

David, D., Y. Gangat, D. Payet, and R. Courdier. 2012. "Reification of emergent urban areas in a land-use simulation model in Reunion Island". In *Proceedings of the 1st workshop on Intelligent Agents in Urban Simulation and Smart Cities in the context of the 20th European Conference on Artificial Intelligence*, edited by V. Corruble, F. Carrera, and S. Guerin, 28–32. Montpellier, France.

Galland, S., S. Rodriguez, and N. Gaud. 2017. "Run-time environment for the SARL agent-programming language: the example of the Janus platform". *Future Generation Computer Systems*. DOI: 10.1016/j.future.2017.10.020.

Gil-Quijano, J., T. Louail, and G. Hutzler. 2012. "From Biological to Urban Cells: Lessons from Three Multilevel Agent-Based Models". In *Principles and Practice of Multi-Agent Systems*, edited by N. "Desai, A. Liu, and M. Winikoff, 620–635. Berlin, Heidelberg: Springer.

Grignard, A., P. Taillandier, B. Gaudou, D. A. Vo, N. Q. Huynh, and A. Drogoul. 2013. "GAMA 1.6: Advancing the Art of Complex Agent-Based Modeling and Simulation". In *PRIMA 2013: Principles and Practice of Multi-Agent Systems*, edited by G. Boella, E. Elkind, B. T. R. Savarimuthu, F. Dignum, and M. K. Purvis, 117–131. Berlin, Heidelberg: Springer.

Iwasaki, Y. 1993. "Reasoning with multiple abstraction models". In *Recent Advances in Qualitative Physics*, edited by B. Faltings and P. Strauss, 67–82. Cambridge, MA, USA: MIT Press.

Koestler, A. 1967. *The Ghost in the Machine*. London, UK: Hutchinson & Co.

Mathieu, P., G. Morvan, and S. Picault. 2018. "Multi-Level Agent-based Simulations: Four Design Patterns". *Simulation Modelling Practice and Theory* 83:51–64.

Minsky, M. L. 1965. "Matter, minds, models". In *Proceedings of International Federation for Information Processing*, 45–49.

Moncion, T., P. Amar, and G. Hutzler. 2010. "Automatic Characterization of Emergent Phenomena in Complex Systems". *Journal of Biological Physics and Chemistry* 10(1):16–23.

Morvan, Gildas 2013. "Multi-level Agent-based Modeling - A Literature Survey". https://arxiv.org/abs/1205.0561, accessed 17th August 2019.

Navarro, L., F. Flacher, and V. Corruble. 2011. "Dynamic level of detail for large scale agent-based urban simulations". In *AAMAS '11: The 10th International Conferene on Autonomous Agents and Multiagent Systems - Volume 2*, 701–708. Taipei, Taiwan: International Foundation for Autonomous Agents and Multiagent Systems.

Sarraf-Shirazi, A., T. Davison, S. von Mammen, J. Denzinger, and C. Jacob. 2014. "Adaptive agent abstractions to speed up spatial agent-based simulations". *Simulation Modelling Practice and Theory* 40:144–160.

Scheffer, M., J. M. Baveco, D. L. DeAngelis, K. A. Rose, and E. H. van Nes. 1995. "Super-individuals a Simple Solution for Modelling Large Populations on an Individual Basis". *Ecological Modelling* 80(2-3):161–170.

Sharpanskykh, A., and J. Treur. 2011. "Group Abstraction for Large-Scale Agent-Based Social Diffusion Models with Unaffected Agents". In *Agents in Principle, Agents in Practice*, edited by D. Kinny, J. Y.-j. Hsu, G. Governatori, and A. K. Ghose, 129–142. Berlin, Heidelberg: Springer.

Stachowiak, H. 1973. *Allgemeine Modelltheorie*. Wien, New York: Springer.

Steiniger, A., F. Kruger, and A. M. Uhrmacher. 2012. "Modeling agents and their environment in Multi-Level-DEVS". In *Proceedings of the 2012 Winter Simulation Conference*, edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, 2629–2640. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Tchappi, I. H., S. Galland, V. C. Kamla, and J. C. Kamgang. 2018. "Holonification of Road Traffic Based on Graph Theory". In *Cellular Automata*, edited by G. Mauri, S. El Yacoubi, A. Dennunzio, K. Nishinari, and L. Manzoni, 513–525. Cham: Springer.

Tisue, S., and U. Wilensky. 2004. "NetLogo: Design and Implementation of a Multi-Agent Modeling Environment". In *Proceedings of the Agent 2004 Conference on Social Dynamics: Interaction, Reflexivity and Emergence*, edited by M. North, D. Sallach, and C. Macal, 161–184. Chicago, IL: Argonne National Laboratory.

Wendel, S., and C. Dibble. 2007. "Dynamic Agent Compression". *Journal of Artificial Societies and Social Simulation* 10(2):9.

AUTHOR BIOGRAPHIES

ROMAIN FRANCESCHINI is a post-doctoral researcher at the University of Antwerp (Belgium) in collaboration with the University of Corsica (France), where he works on modelling methods and techniques to relate multiple levels of abstraction. He received his PhD from the University of Corsica in 2017. His research interests include multi-agent systems, agent-based models and the theory of modeling and simulation. His email address is romain.franceschini@univ-corse.fr.

SIMON VAN MIERLO is a post-doctoral researcher at the University of Antwerp (Belgium). He is a member of the Modeling, Simulation and Design (MSDL) research lab. For his PhD thesis, he developed debugging techniques for modeling and simulation formalisms by explicitly modeling their executors control flow using Statecharts. He is the main developer and maintainer of SCCD, a hybrid formalism that combines Statecharts with class diagrams. His email address is simon.vanmierlo@uantwerpen.be.

HANS VANGHELUWE is a Professor at the University of Antwerp Flanders Make (Belgium), an Adjunct Professor at McGill University (Canada) and an Adjunct Professor at the National University of Defense Technology (NUDT) in Changsha, China. He heads the Modeling, Simulation and Design (MSDL) research lab. In a variety of projects, often with industrial partners, he develops and applies the model-based theory and techniques of Multi-Paradigm Modeling (MPM). His current interests are in domain-specific modeling and simulation, including the development of graphical user interfaces for multiple platforms. To model such reactive systems, he advocates the use of Statecharts to describe their behaviour. His email address is hans.vangheluwe@uantwerpen.be.