2025 33rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP) | 979-8-3315-2493-7/25/$31.00 ©2025 IEEE | DOI: 10.1109/PDP66500.2025.00049

Exploiting GPU computing for effective Agent-Based simulation: initial experiments

Marzio Pennisi, Giuliana Franceschinis
Computer Science Institute - DiSIT
Università del Piemonte Orientale
Alessandria, Italy
{marzio.pennisi, giuliana.franceschinis}@uniupo.it

Daniele Baccega, Simone Pernice, Irene Terrone
Department of Computer Science
University of Turin
Turin, Italy
{daniele.baccega, simone.pernice, irene.terrone}@unito.it

***Abstract*—Agent-Based Modeling and Simulation (ABMS) has been increasingly applied in various research fields, thanks to the capability of these models to describe fine-grained real-world behavior and to the ease of interpretation by domain experts. However, such models lack a formal definition and well-defined semantics that are common to the different tools supporting ABMS. This may occasionally lead to greater complexity in interpreting the results with respect to other modeling approaches. To address this issue, an ABM semantics that adopts a continuous-time approach and a next-event time advance simulation algorithm has been formally defined and presented. Such an approach may lead to high computation times as it requires recalculations of activity rates for all the agents after each event. In this preliminary study, we exploit the FLAME GPU framework to evaluate the benefits that GPU computing may bring to the performance of our simulation algorithm.**

***Index Terms*—Agent-based modeling, Simulation, GPU computing, FLAME GPU.**

I. Introduction

Mechanistic modeling, along with computer simulation, constitute two essential methods for investigating numerous real-world phenomena. These techniques enable the prediction of a system's temporal evolution, facilitating what-if analyses under scenarios not previously encountered. Possible uses include applications in the fields of biology and immunology, such as immune system simulation for the evaluation of new pharmaceutical compounds in 'in Silico Trials' [1], [2], in the fields of epidemiology and social sciences such as the prediction of the impact of contingency policies during pandemics [3], [4], or in the fields of environment an cyber-security such as the analysis of cyber-attacks to critical systems [5], [6]. This provides a form of extrapolation that is not readily available through other modeling approaches [7].

Real-world phenomena can be modeled by adopting mechanistic modeling in two ways. The first method involves examining the phenomena from a macroscopic perspective, by setting up rules and equations that capture the behavior of the entire system. This is commonly realized through complex systems of differential equations supported by a robust mathematical framework (e.g. [8], [9]). Conversely, the second method involves a microscopic examination, focusing on the behavior of individuals and their interactions. This is facilitated through the simulation of Agent-Based Models (ABMs) [10], a relatively recent computational modeling framework. ABMs represent the system describing the behavior of independent entities known as agents, making it possible to study "emergent phenomena"—complex global system dynamics that arise from the simple actions and interactions of these agents.

Although there is increasing interest in employing ABMs, owing to their adaptability to diverse settings, ability to provide detailed descriptions, and understandability for experts in the application domain, they often lack clearly defined semantics for coupling and scheduling agents and environment behaviors. This may lead to diverse outcomes for the same domain model, not only compared to other computational methods, but also among different ABM programming platforms and simulation tools. There is no universal agreement on defining an agent and its behavior, on the time advance strategy (Fixed Interval or Next Event Time Advance) [11] and the way of managing concurrency and resolving conflicts: in other words, there are several different interpretations of the ABM semantics that drive the system evolution from an initial to a final state.

To address this gap, we developed a modeling pipeline that, starting from the model description based on the graphical high-level formalism—called Extended Stochastic Symmetric Nets (ESSN) [12]—, allows us to develop ABMs with a clear well-defined semantics. In fact, a translation algorithm capable of automatically deriving an ABM model [13] to be used in NetLogo [14] has been integrated into GreatSPN [15].

The simulation algorithm utilizes a Next Event Time Advance (NETA) strategy. Currently, this implementation recalculates the rates for all activities enabling future events, after each state transition (i.e., after an agent acts, changing the state of the system), even if such state transition did not influence the state of the whole system. This clearly causes inefficiencies in various scenarios. To cope with this issue, two possible strategies can be followed. The first strategy we investigated in [16] is based on the adoption of methods exploiting the structural dependencies that can be detected on the model, in order to avoid superfluous recalculations. However, this approach strongly depends on the model structure (i.e., whether the majority of state changes have a local effect) and it introduces an overhead so that not all the models may benefit from it. In this work, we aim to analyze a second strategy that involves a "brute-force" approach exploiting the

979-8-3315-2493-7/25/$31.00 ©2025 IEEE
DOI 10.1109/PDP66500.2025.00049

GPU computing potential to parallelize rate calculations. In this second scenario, all the agents may recalculate their rates in parallel with a consistent speed-up.
Based on our review of the literature, specifically [17], to the best of our knowledge, we found that FLAME GPU [18], [19] (and its updated version, FLAME GPU 2) is currently the only widely available and actively maintained GPU-based simulation framework for agent-based modeling. Other tools mentioned in earlier research either focus on specific use cases, are no longer actively maintained, or are not readily available for practical use. For instance, in [20], [21] no tools have been freely shared.

The paper is organized as follows. In Section II we describe the adopted methodology, including a sketch of the AB-ESSN simulation approach and algorithm (II-A), an introduction to FLAME GPU (II-B), and details about the current implementation of the AB-ESSN algorithm on FLAME GPU (II-C). In Section III we give preliminary results obtained on an epidemiological model of SIRS, in Section IV we discuss the advantages and disadvantages of employing either CPU or GPU computing, and in Section V conclusions and future work are presented.

II. Methodology

A. The AB-ESSN Simulation Approach and Algorithm

The Agent-Based - Extended Stochastic Symmetric Net (AB-ESSN) formalism, introduced in [13], provides a compositional modeling formalism, based on High-Level Stochastic Petri Nets, allowing to graphically define the internal behavior of the different agent types (in the form of Petri Net submodels, including places, corresponding to state variables, and transitions, corresponding to activities that trigger state changes) and their possible interactions through synchronization on labeled transitions.

The complete model, obtained by composing the submodels of all agent types, produces an ESSN whose dynamics can be described by means of a Continuous Time Markov Chain (CTMC). The model is parametric in the number of agents of each type, their initial state (expressed by means of tokens, enriched with structured data, distributed in the places) and the rates of the activities (represented by the ESSN model transitions). The simulation of an AB-ESSN model begins from the initial state and repeatedly evaluates the rates of all enabled *transition instances*, corresponding to all the activities in which each agent can be engaged in that state.

Then, it generates an instance of an exponentially distributed random variable, with a rate equal to the sum of all activity rates in all agents, thus establishing the time to the next event occurrence; the clock is then advanced to that time. Finally, the agent that must act is selected through a random choice weighted according to the rates of the activities of each agent. The event representing the end of one ongoing activity in the selected agent is randomly chosen and actuated. As a result, a new state is reached and the simulation starts all over again until a terminating condition is met. This behavior is obtained by coordinating all agents that act in parallel computing their own rates and performing the appropriate state change when selected.

In [13] an algorithm for the translation of an AB-ESSN model to Netlogo has been defined, which guarantees that the intended semantics is preserved. The simulation procedure becomes very time-consuming as the number of agents increases. Adapting the basic NetLogo code to avoid the update of the agents' rates when not needed could be beneficial. This can be achieved by considering that, in many cases, most of the state changes have a local effect and cause a perturbation on the set (and/or rate) of enabled activities belonging to a small subset of agents. This approach has been presented and discussed in [16]: although it improves the performance, the improved algorithm does not scale to significantly larger agent populations. For this reason we decided to redesign the algorithm following a GPU computing paradigm: currently, the translation from the AB-ESSN model to the FLAME GPU program is not yet automatic, but we plan to work on this line in the near future.

B. FLAME GPU

FLAME GPU (Flexible Large-scale Agent Modeling Environment for Graphics Processing Units) [18], [19] is an advanced GPU-based extension to the FLAME framework designed for high performance. It facilitates the translation of formal agent specifications into optimized CUDA code using C-based scripting. This encompasses essential components of agent-based modeling like diverse agent types, agent communication, and the processes of agent creation and deletion.

FLAME GPU employs advanced techniques to optimize data access and management (such as handling dense and sparse data, using both global and shared memory), optimize kernel functions, manage resource balancing, and improve thread utilization. For instance, each agent is mapped to a GPU thread, which loads batches of tiled messages into the shared memory of each multiprocessor. This approach enables agents to serialize certain message reads with minimal communication overhead, achieving near-optimal GPU hardware performance.

An agent in FLAME GPU is modeled as a communicating X-Machine, an advanced version of the Finite State Machine that incorporates memory. Though the X-Machine has a precise formal definition, X-Machine agents are essentially state machines capable of interacting through messages placed in universally accessible message lists. Agent operations are executed through state transition functions that shift agents from one state to another.

Similar to other CUDA-based applications, FLAME GPU supports the execution of two function types: host functions (which run on the CPU) and device functions (which operate on the GPU). The host is responsible for initiating the CUDA application and overseeing the communication process between the device and the system. Host memory holds and initializes data, also receiving data from the GPU. Moreover, the host oversees the synchronization of the CUDA application, ensuring that it waits for one kernel execution to

finish before initiating another. Conversely, the device executes the actual CUDA kernel, which refers to the GPU-executing code. During kernel execution, device memory retains input data, output data, and local variables. Data transfer between the host and the device occurs through memory allocation and copying when the CUDA application runs. Before kernel execution, data is transferred from host memory to device memory. Post-execution, results are moved from the device's memory to the host's memory. Generally, the device handles kernel operation, while the host manages memory operations and communication between the device and the system. The distinction between host and device function is fundamental for the implementation of our simulation algorithm on top of FLAME GPU.

C. Implementing the AB-ESSN approach on FLAME GPU

The idea behind the implementation of the AB-ESSN algorithm we described earlier is quite straightforward and consists on the implementation of the agents' local rates calculation and action execution on the device, leaving the agent selection, global rate calculation, and time advance on the host. In this way, all the rate recalculations are executed in parallel with minimal effort.

Figure 1 shows the five implementation steps utilized by the algorithm, using the UML Sequence Diagram formalism. Initially, the host executes the initialization functions and shares the global variables with the agents (steps 1 and 2). Then, the agents' code performs the following steps: broadcast messages to share agent information (e.g., position, state, type, etc.) with other agents (step 2.1); count compatible agents for interaction based on message data (step 2.2); and calculate the rates for each agent's potential actions and the cumulative rate (step 2.3).

Then, the host computes the model's overall rate as the sum of the agents' rates using a *reduce* function (step 2.4). Next, the host predicts the timing of the next event and updates the '*CurrentTime*' variable (step 2.4.1), which stores the time progression of the simulation. Furthermore, the host employs a roulette wheel method based on agents' rates to choose and mark the next acting agent (step 3).

Finally, all agents execute steps 4 and 5 in parallel, but only the marked agent will actually determine the next action using the roulette-wheel method on action rates and perform the chosen action. The steps from 2.1 to 5 are repeated in a loop as long as CurrentTime $\leq$ FinalTime.

III. Experiments

The preliminary experiments on the AB-ESSN algorithm implementation have been carried out using a SIRS (Susceptible-Infectious-Recovered-Susceptible) model. The SIRS model is a particular case of the SIR epidemiological model, first used by Kermack and McKendrick in 1927 [22]. In the SIRS model, individuals defined as Susceptible can turn Infectious upon interacting with those already Infectious. Infectious individuals may recover and become Recovered individuals, obtaining immunity. However, after a given time period, Recovered individuals might revert back to Susceptible. The presented implementation also takes into account the spatial distribution of individuals.

Specifically, agents (i.e., individuals) are placed on a 10x10 discrete grid and can move up, down, left, or right at given rates. Furthermore, infection may occur only among agents that are in the same coordinates (x, y). For such a reason, steps 2.1 and 2.2 (message broadcasting and counting, respectively) have been adapted to let Susceptible agents know how many Infectious agents are on the same position, and thus calculate their infection rate accordingly.

Simulations have been carried out starting from an initial number of 100 agents, increasing then this number up to 800 agents. The initial conditions have been set equal for both implementations (NetLogo and FLAME GPU). The NetLogo model has been automatically obtained from an ESSN model drawn with GreatSPN 3 according to what is described in [13], [23] (Figure 2), while the FLAME GPU model has been developed manually. For each initial condition, 100 simulations have been executed, and mean execution times have been recorded. A different random seed has been used for each simulation.

The NetLogo simulations have been executed on an Intel i9-9880H machine with 32GB of RAM. The FLAME GPU simulations have been carried out on a Nvidia© Tesla P100-SXM2-16G made available on the Chameleon Cloud Infrastructure [24]. In both scenarios, runs have been executed sequentially. It is worth noting here that we adopted an old version of FLAME GPU (1.5) for these initial experiments, as FLAME GPU 2 is currently in a pre-release (alpha) state. However, future experiments and implementations will be migrated towards the novel version also to test and exploit newer GPUs.

In Figure 3 we show the mean execution times for both the NetLogo and FLAME GPU implementations as the number of agents increases. Here, it is possible to observe that, the computational time grows exponentially for the NetLogo model, while it remains pretty bounded for the FLAME GPU model. This can be also observed in Figure 4, where the performance speed up (calculated as the ratio between NetLogo and FLAME GPU computational times) is reported. Here we see a growing advantage for the FLAME GPU implementation, which culminates with 24,89x speed improvement for the 800 agents case.

IV. Discussion

Agent-based simulations excel in capturing fine-grained, emergent behaviors that arise from interactions among agents. Key quantitative measures often include computational performance (e.g., runtime efficiency, scalability), agent population size, interaction density, and the ability to handle complex rule sets. Tools like NetLogo provide a simple, intuitive interface that makes them accessible to domain experts without advanced programming skills. However, this simplicity often limits scalability for large-scale or computationally intensive models. In contrast, frameworks like FLAME GPU provide

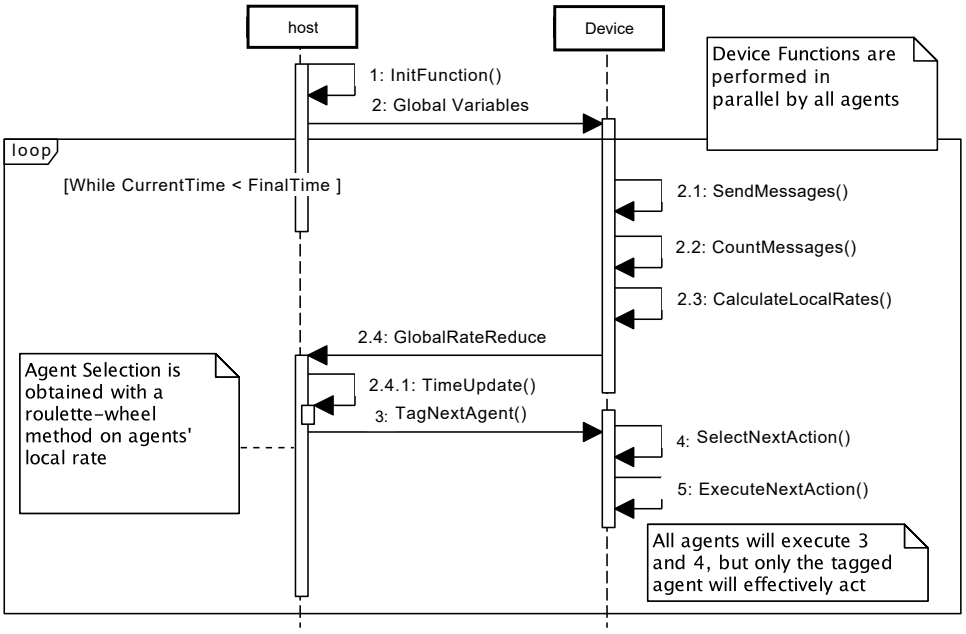

Fig. 1. UML Sequence Diagram for the AB-ESSN Algorithm implementation in FLAME GPU. Numbers from 1 to 5 represent the corresponding steps in the algorithm.

greater computational power and scalability, but require more technical expertise to define agent behaviors and interactions.

Indeed, GPU-based approaches require predefining agent behaviors and interactions in a way that fits the GPU's parallel processing paradigm. This can impose constraints on model design and make it more challenging to implement complex agent interactions compared to purely CPU-based approaches. However, the trade-off is a significant increase in computational efficiency, particularly for large-scale simulations.

It is worth noting that GPU performance is often constrained by memory management, as each type of memory has specific use cases and challenges that need to be addressed to maximize the GPU's potential. For instance, global memory is accessible by all threads across all blocks and typically offers a large capacity, but its access speed is relatively slow. Shared memory, on the other hand, is much faster but limited to threads within a single block. Register memory is extremely fast but has a very small capacity. These limitations can create significant challenges in the structure of agent-based models. Frequent creation and termination of agents require dynamic memory allocation, which in turn demands global synchronization on the device. Additionally, agents often follow diverse behavioral rules, leading to different execution paths within a kernel. This results in thread divergence, reducing parallelism and efficiency. To mitigate such issues, FLAME GPU can be adopted thanks to its capability of efficiently managing GPU resources, such as threads and memory, enabling better performance for agent-based simulations.

For what regards the scalability limitations of GPU-enabled ABM simulations, and in particular of our approach, we note that FLAME GPU has been shown to handle millions of agents effectively. Nevertheless, various factors may influence real-world performance and real scalability. First, the total number of agents that can be managed is heavily influenced by the underlying hardware (e.g., GPU memory and processing power). Additionally, the specific domain model being simulated can have a significant impact on performance. For instance, in many ABM scenarios, agents typically interact with others in their local vicinity. As such, factors like the size of the simulation space and the spatial distribution of agents can greatly influence computational demands. To make things clearer, one should consider a scenario where one million agents are densely packed in the same location and where each agent can, in principle, interact with any other in the same location. This scenario would likely be much more computationally intensive, in particular for our simulation approach, compared to a scenario where the same number of agents is uniformly distributed across a larger simulation grid (e.g., 100×100). Thus, these spatial and interaction patterns directly affect the computational load and the potential scalability due to the way interactions are processed in parallel on GPUs.

V. Conclusions and Future Work

The goal of this preliminary research was to investigate if adopting GPU-computing, by means of the FLAME GPU framework, was beneficial for the implementation and execution of the AB-ESSN simulation algorithm presented in [13], which is particularly demanding from a computational

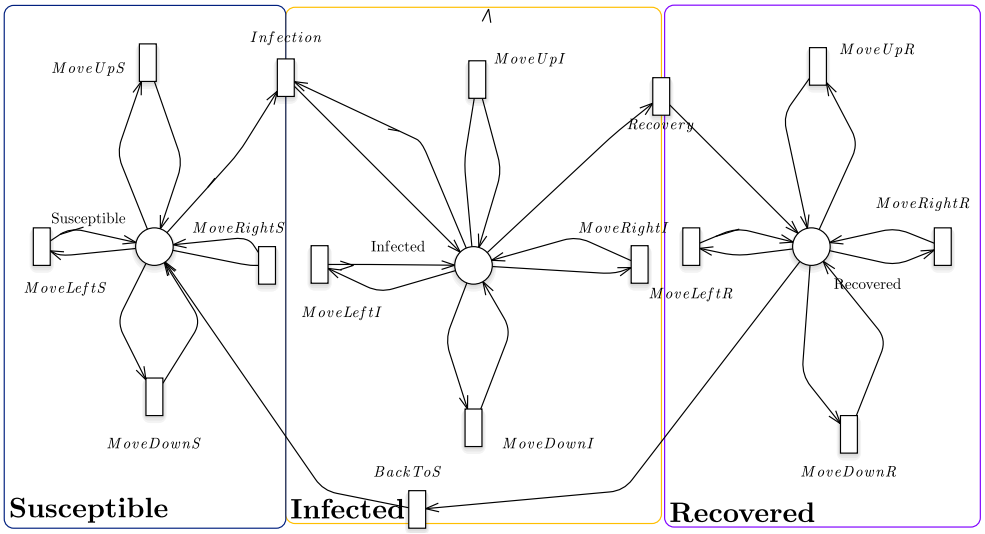

Fig. 2. The ESSN SIR model. Color domains and markings have been removed for readability. Susceptible agents may become Infectious due to the Infection transition; Infectious agents can recover (Recovery transition) and Recovered agents can become Susceptible again (transition BackToS). All agents can move in four directions (Transitions Move*).

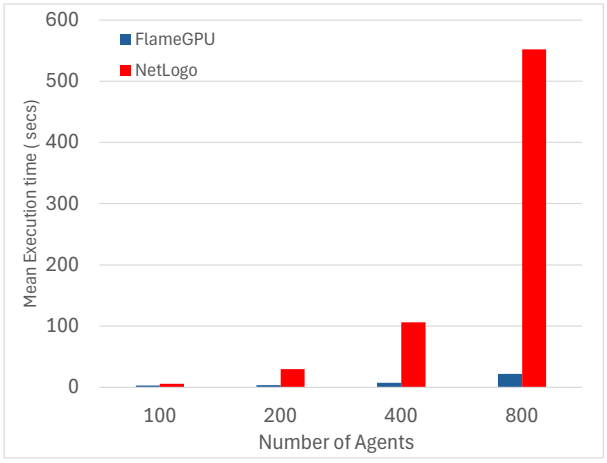

Fig. 3. Mean execution times (calculated over 100 simulations) for the NetLogo and FLAME GPU SIRS AB-ESSN implementations.

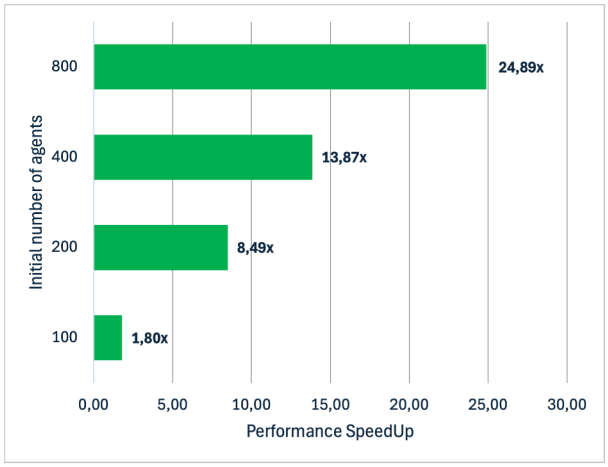

Fig. 4. Performance speed up for the FLAME GPU model with respect to the NetLogo implementation.

perspective, especially in the phase of agents' rates calculation. This aspect is particularly critical, as it represents a limiting factor for the total number of agents to be employed in each simulation (the Achilles' heel for any agent-based model).

While absolute measurements should be taken with a pinch of salt as we are exploiting completely different architectures, it appears clear that GPUs can bring an increasing performance advantage as the number of agents grows, thanks to the parallel execution of the agents' procedures related to rate calculations.

Further experiments, as well as a more refined implementation of the approach, have to be conducted to more accurately estimate the real advantage and applicability of the methodology. To this end, we are currently conducting analyses to better characterize the scalability of our approach, taking into account these factors. The results of these analyses will be presented in future work to provide a more comprehensive understanding of the scalability of GPU-accelerated ABMs modeled using the AB-ESSN approach.

Moreover, we plan to pursue further studies in multiple directions, including the porting towards FLAME GPU 2 [25], [26], a refinement of the current code, the enhancement of the execution efficiency through targeted optimizations, and a comparison with the structural dependency-based approach described in [16]. Also, we plan to evaluate if the integration of

both approaches (the GPU one and the structural dependency-based one) may be useful for improving the simulation times. Finally, we plan to implement the automatic translation, within GreatSPN 3, from an ESSN annotated model directly towards a FLAME GPU model to speed up the modeling and development pipeline.

ACKNOWLEDGMENT

D.B. is a Ph.D. student enrolled in the National Ph.D. in Artificial Intelligence, XXXVII cycle, health and life sciences course organized by Università Campus Bio-Medico di Roma. The authors wish to thank Michael Munì for his initial contribution to this project.

REFERENCES

[1] M. Pennisi, G. Russo, S. Ravalli, and F. Pappalardo, "Combining agent based-models and virtual screening techniques to predict the best citrus-derived vaccine adjuvants against human papilloma virus," *BMC Bioinformatics*, vol. 18, no. S16, Dec. 2017.

[2] M. Pennisi, G. Russo, S. Motta, and F. Pappalardo, "Agent based modeling of the effects of potential treatments over the blood-brain barrier in multiple sclerosis," *J. Immunol. Methods*, vol. 427, pp. 6–12, Dec. 2015.

[3] S. Pernice, P. Castagno, L. Marcotulli, M. M. Maule, L. Richiardi, G. Moirano, M. Sereno, F. Cordero, and M. Beccuti, "Impacts of reopening strategies for COVID-19 epidemic: a modeling study in Piedmont region," *BMC Infectious Diseases*, vol. 20, pp. 1–9, 2020.

[4] D. Baccega, S. Pernice, P. Terna, P. Castagno, G. Moirano, L. Richiardi, M. Sereno, S. Rabellino, M. M. Maule, and M. Beccuti, "An agent-based model to support infection control strategies at school," *Journal of Artificial Societies and Social Simulation*, vol. 25, no. 3, p. 2, 2022.

[5] D. Cerotti, D. Codetta-Raiteri, L. Egidi, G. Franceschinis, L. Portinale, G. Dondossola, and R. Terruggia, "Analysis and detection of cyber attack processes targeting smart grids," in *2019 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe)*. IEEE, Sep. 2019.

[6] D. Cerotti, D. Codetta-Raiteri, G. Dondossola, L. Egidi, G. Franceschinis, L. Portinale, and R. Terruggia, "Evidence-based analysis of cyber attacks to security monitored distributed energy resources," *Appl. Sci. (Basel)*, vol. 10, no. 14, p. 4725, Jul. 2020.

[7] D. Besozzi, "Reaction-based models of biochemical networks," in *Pursuit of the Universal*, A. Beckmann, L. Bienvenu, and N. Jonoska, Eds. Cham: Springer International Publishing, 2016, pp. 24–34.

[8] D. Cerotti, A. Miele, M. Gribaudo, A. Bobbio, and C. Bolchini, "Scalable analytical model for reliability measures in aging VLSI by interacting markovian agents," *Perform. Eval.*, vol. 132, pp. 21–37, Aug. 2019.

[9] C. Bianca and M. Pennisi, "The triplex vaccine effects in mammary carcinoma: A nonlinear model in tune with SimTriplex," *Nonlinear Anal. Real World Appl.*, vol. 13, no. 4, pp. 1913–1940, Aug. 2012.

[10] C. M. Macal and M. J. North, "Agent-based modeling and simulation," in *Proceedings of the 2009 winter simulation conference (WSC)*. IEEE, 2009, pp. 86–98.

[11] A. M. Law, *Simulation Modeling and Analysis, Fifth Edition*. McGraw-Hill, 2015.

[12] S. Pernice and et al., "A computational approach based on the colored Petri net formalism for studying multiple sclerosis." *BMC bioinformatics*, vol. 20, no. 6, pp. 1–17, 2019.

[13] E. G. Amparore, M. Beccuti, P. Castagno, S. Pernice, G. Franceschinis, and M. Pennisi, "From compositional Petri net modeling to macro and micro simulation by means of stochastic simulation and agent-based models," *ACM Trans. Model. Perform. Evaluation Comput. Syst.*, vol. 9, no. 1, pp. 1:1–1:30, 2024. [Online]. Available: https://doi.org/10.1145/3617681

[14] U. . Wilensky, "NetLogo - Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. http://ccl.northwestern.edu/netlogo/ [Accessed: Jan, 27th 2025]," 1999.

[15] E. G. Amparore, G. Balbo, M. Beccuti, S. Donatelli, and G. Franceschinis, "30 years of GreatSPN," in *Principles of Performance and Reliability Modeling and Evaluation*. Springer, 2016, pp. 227–254.

[16] M. Pennisi, E. G. Amparore, and G. Franceschinis, "Exploiting structural dependency relations for efficient agent based model simulation," in *19th European Workshop, EPEW 2023, and 27th International Conference, ASMTA 2023, Florence, Italy, June 20-23, 2023, Proceedings*, ser. LNCS, vol. 14231. Springer, 2023, pp. 353–368. [Online]. Available: https://doi.org/10.1007/978-3-031-43185-2_24

[17] S. Abar, G. K. Theodoropoulos, P. Lemarinier, and G. M. O'Hare, "Agent Based Modelling and Simulation tools: A review of the state-of-art software," *Computer Science Review*, vol. 24, pp. 13–33, 2017.

[18] P. Richmond, D. Walker, S. Coakley, and D. Romano, "High performance cellular level agent-based simulation with FLAME for the GPU," *Briefings in Bioinformatics*, vol. 11, no. 3, pp. 334–347, 02 2010.

[19] M. K. Chimeh and P. Richmond, "Simulating heterogeneous behaviours in complex systems on GPUs," *Simul. Model. Pract. Theory*, vol. 83, pp. 3–17, Apr. 2018.

[20] A. R. Galvão Filho, L. C. Martins de Paula, C. J. Coelho, T. W. de Lima, and A. da Silva Soares, "Cuda parallel programming for simulation of epidemiological models based on individuals," *Mathematical Methods in the Applied Sciences*, vol. 39, no. 3, pp. 405–411, 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/mma.3490

[21] M. Lysenko and R. M. D'Souza, "A framework for megascale agent based model simulations on graphics processing units," *Journal of Artificial Societies and Social Simulation*, vol. 11, no. 4, p. 10, 2008. [Online]. Available: https://www.jasss.org/11/4/10.html

[22] W. Kermack and A. McKendrick, "A contribution to the mathematical theory of epidemics," *Proc. R. Soc. Lond. Ser. Math. Phys. Eng. Sci.*, vol. 115, no. 772, pp. 700–721, 1927.

[23] E. G. Amparore, M. Beccuti, P. Castagno, G. Franceschinis, M. Pennisi, and S. Pernice, "Multiformalism modeling and simulation of immune system mechanisms," *Proceedings - 2021 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2021*, pp. 3259–3266, 2021.

[24] K. Keahey, J. Anderson, Z. Zhen, P. Riteau, P. Ruth, D. Stanzione, M. Cevik, J. Colleran, H. S. Gunawi, C. Hammock, J. Mambretti, A. Barnes, F. Halbach, A. Rocha, and J. Stubbs, "Lessons learned from the chameleon testbed," in *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20)*. USENIX Association, July 2020.

[25] P. Richmond, R. Chisholm, P. Heywood, M. Leach, and M. Kabiri Chimeh, "FLAME GPU," 2021, available: https://www.doi.org/10.5281/ZENODO.5428984.

[26] P. Richmond, "Flame GPU - Flexible-Large Scale Envirnoment for the Graphics Processing unit," 2024, https://flamegpu.com [Accessed: Jan, 27th 2025].