# Distribution Model: Separation of Concerns to Facilitate the Distribution of Agent-Based Models

Lucas Grosjean(1) Alexis Drogoul(1)
Bénédicte Herrmann(2) Nghi Quang Huynh(3)
Christophe Lang(2) Nicolas Marilleau(1)
Laurent Philippe(2)

1. UMI 209, UMMISCO, IRD, Sorbonne Université, Bondy, France
2. Université Marie et Louis Pasteur, CNRS, institut FEMTO-ST, F-25000 Besançon, France
3. Can Tho University, Can Tho, Vietnam

**Abstract**:

Agent-based modeling is a powerful tool for simulating complex systems. Some models require large amounts of agents and data. Most agent simulation platforms run models sequentially and cannot run large models in a reasonable time, or at all. To solve these problems, Agent-Based Models can use distributed computing to spread the load and/or the data over multiple computing processors. The distribution of ABM execution, however, raises complex issues that require advanced skills to be addressed. This paper presents the concept of *Distribution Model* that aims at simplifying the distribution of ABMs. Following the separation-of-concerns approach, we propose a flexible framework enabling a clear division between the model thematic and its distributional aspects, fostering greater flexibility in design and implementation. We present a model-based agent distribution system that uses agents to address distribution challenges. We demonstrate its capabilities using the GAMA platform, highlighting how it simplifies model distribution for researchers.

**Kkeywords** Distribution High Performance Computing Agent-Based Model

## 1 Introduction

Agent-Based Models (or ABMs) are used to simulate and understand increasingly complex real-world systems. As the scale of these models and the accuracy of the modeled behaviors increase, performance issues arise when they are run sequentially in a single-processor simulation, limited by the available computing resources, sometimes to the point that it can no longer be executed in a reasonable time, or even at all. In this case, the use of high-performance computing (HPC) techniques, by distributing the simulation on multiple processors opens up interesting prospects for overcoming resource constraints and hardware limitations. Despite the intrinsically "distributed" nature of multi-agent systems, these systems describe a single world with synchronized interactions and shared data, making their distribution challenging. Distributed agent-based simulations, for instance, require frequent synchronization operations, such as updating the state of the environment and the agent perceptions. The distribution of an agent-based model is therefore neither a trivial task, nor easily generalizable.

Some platforms offer features to run ABMs on distributed architectures, including HPC architectures. But the distribution remains tricky for modelers, since mastering HPC platforms requires specialized expertise in parallel programming and distributed systems. On the other hand, thanks to their rich, comprehensive environment, traditional agent-based modeling platforms (like GAMA or NetLogo) are easy to use, accessible to modelers of all experience levels and various disciplines. A new approach is needed, to bridge the gap between modeler-oriented ABM platforms and the advanced functionalities of distribution platforms (like RepastHPC or D-Mason).

This paper introduces a novel distribution paradigm for Agent-Based Models relying on the separation of concerns: **Distribution Model**. By employing modeling techniques, Distribution Model effectively **decouples** the ABM definition from its distribution process in two coupled models. The Distribution Model relies on the use of a modeling language that **empowers regular modelers** to independently distribute their ABMs, without requiring specialized technical skills.

In Section 2, we analyze gaps and weaknesses of existing ABM distribution platforms from the modeler perspective. In Section 3, we detail the concept of *Distribution Model*, the deployment of the model on a distributed architecture and how it improves the development cycle. In Section 4, we present an implementation leveraging modeling techniques using agents to solve specific distribution issues: *Distribution Agent*. Section 5 presents our implementation of the concepts of Distribution Model and distribution agents in the GAMA platform. Section 6, concludes the article.

## 2 Understanding Modeler Challenges in ABM Distribution

In the vast field of agent-based modeling and simulation, many platforms with various goals and features exist but few allow for the distribution of models across multiple processors. Platforms such as NetLogo [33], GAMA [31], Evoplex, Jadex [5], or Agentscript, allow the execution of agent-based models in a centralized computing environment, without any particular distribution capability. In contrast, some platforms are specifically designed to support distribution, such as FPMAS [6], PDES-MAS [29], Pandora [25], MASS [9], FAME-core [26], or Care HPS [2]. Other centralized platforms, such as DEVS-Suite [21], AnyLogic [4], Jade [1], offer the ability to execute distributed models in specific scenarios. Finally, although initially centralized, some platforms have since proposed developments allowing them to distribute a simulation, such as Flame with FLAME GPU [22], Repast [20] with RepastHPC [11], and Mason [18] with D-MASON [12]. The goal of this section is to examine modeler experiences with existing distribution platforms, in order to propose solutions for enhancing accessibility and for lowering the required skill level for platform operation. Our analysis starts with an exploration of the distribution platform strategies for mitigating the issues arising from distributed ABM. The second phase of our analysis centers on the modeler experience when distributing his own model with current distribution platform. This analysis concludes by examining the current state of ABM distribution by trying to argue in favor of separating the model development from the model distribution.

## 2.1 Platform Approaches to Address Distribution Issues

Platforms approach to distribute an ABM is to divide a simulation across several simulation processes that are, in turn distributed on distributed platform nodes. This distribution introduces several issues, each distribution platform offers unique approaches to solving one or more of these issues. Some studies, [8], [24] present and compare these different approaches. Based on these studies, we have identified four main issues and we outline the strategies used by the different distribution platforms.

- **Communication** protocols ensure data exchange between processes in the distributed system. Most of platforms rely on Message Passing Interface (MPI) [32] to exchange data and messages between the simulation processes. Flame's communication mechanism is based on a synchronized message board that operates on a publish-subscribe paradigm [10].
- **Partitioning** is the strategy used to divide the environment and agents across the simulation processes. The majority of platforms use a grid approach, assigning each grid cell and its agents to a process. CareHPS proposes both a grid-based approach and a voronoid-based partitioning [27]. Flame [19] and FPMAS uses a graph system to assign agents to processes.
- **Load balancing** dynamically redistributes, the workload evenly among processes, to optimize performance. Grid-based approaches use load balancing to adjust cell sizes. For graph approach, FPMAS proposes different solutions using Zoltan library. Flame's strategy utilizes round-robin algorithm, storing agents in a queue and cyclically assigns them to processes. It achieves low-cost execution time but requires costly global MPI communications.
- **Data synchronization** mechanism provides agents with the most up-to-date information. FPMAS introduces synchronization modes [7], interactions are classified according to read/write operations and location (local or remote). Other platforms strategy for data synchronization can be described using FPMAS definition: GhostMode for RepastHPC, GlobalGhostMode for D-Mason. Data synchronization in Flame relies on synchronized message boards. Pandora divides each cell in four equal sub-portions, each process cyclically runs the portions in the same order to avoid conflicts.

## 2.2 ABM Distribution: a Complex and Time-Consuming Task

A modeler, with limited distribution experience and working on an ABM facing performance issues, may encounter difficulties with the previously listed issues. Using one of the distribution platforms should be the solution in this case. Current distribution platforms are however constrained by their approaches to addressing distribution, each offering a single solution for each issue, limiting their flexibility and adaptability. This forces modeler to carefully choose the distribution platform as the platform might not suit the ABM needs and requires in-depth knowledge and experience in application distribution. Finally this may discourage modelers from using them to distribute their models. [23] analyze the characteristics of existing distribution platforms and, among them, in the programming languages. Interestingly, the programming language used to develop the platform is identical to the language used by the modeler to distribute its model: RepastHPC uses C++/ReLogo, D-Mason and JADE use Java, PDES-MAS and FPMAS use C/C++ and Flame uses C and XML-Schema. While widely used, general programming languages may not suit modelers accustomed to user-friendly modeling languages like NetLogo or GAMA, which offer high-level abstractions. Furthermore, even with distribution-focused platforms (more like libraries than IDEs), modelers still face a significant distribution workload.

Note that a large number of projects implements tailored approaches: e.g. strip-based [3], voronoid-based [27], graph-based [15] [16]. Other works implement tailored load-balancing algorithm: strip [13] or proximity criteria [28].The existence of custom solutions indicates that distribution platforms may not be readily adaptable to diverse needs, forcing modelers to develop their solutions. For easier ABM distribution, modelers need user-friendly platforms with integrated distribution features and multiple solutions tailored to model characteristics, minimizing the technical workload.

### 2.3 Modeling and Distributing: an Amalgam?

In the current distribution platforms, the separation of concerns principle stated by Dijkstra [14] is not respected: platforms require modelers to change the code of their models to incorporate the necessary code for their distribution. However, the link between model and distribution code can harm maintainability and adaptability. Incorrect distribution can lead to result changes, inconsistencies, or errors. Assessing this combined ABM/distribution is complex and requires distribution and modeling expertise.

ABMs are designed for adaptability through modifiable agent properties and behaviors for new features. However, embedding distribution functions complicates system changes, as even minor modifications might necessitate altering distribution functions, risking errors. Prior points highlight the necessity of a new paradigm to facilitate simulation distribution with minimal model changes. We argue for a clear boundary between the modeling and distribution to eliminate possible confusions and facilitate the distribution process. To our knowledge, only [30] separates the modeling from its distribution. It uses agent-based models to manage distribution aspects, such as model deployment, communications, consistency (data projection), flexibility and load balancing.

## 3 Decoupling Thematic Modeling from Distribution

We facilitate model distribution by decoupling the Thematic Model (the phenomenon being modeled) from the Distribution Model, a separate agent-based system for distributing it. To distribute the Thematic Model, we deploy an instance of the Thematic Model and the Distribution Model on each available core.

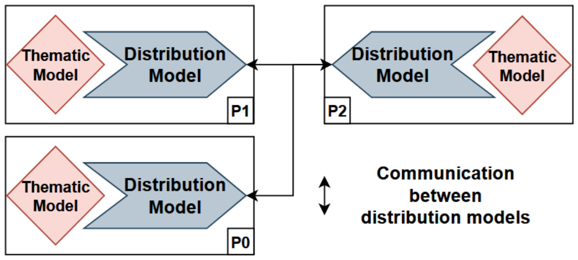


Figure 1: Thematic and Distribution Models: an instance of each model is run on each process. Distribution models communicate and exchange data/agents.

Figure 1 presents the positioning of the instances of the Distribution Model and Thematic Model on a distributed environment. The Distribution Model directly accesses the data from the Thematic Model through a weak coupling relationship [17]. The Distribution Model orchestrates distribution, retrieving Thematic Model data via its getters and setters. The primary innovation of the Distribution Model lies in the complete outsourcing of distribution functions and of the algorithms addressing distribution issues, which until now were implemented directly within the Thematic Model.

## 4 Modeling Technique for the Distribution

In this section, we present how we use agent modeling techniques in the Distribution Model to solve distribution issues for the Thematic Model. The use of a modeling language for Thematic and Distribution Models provides an elegant and efficient solution to simplify the distribution process for modelers. Note that this approach could also give modelers an opportunity access to the distribution level, improve their distribution skills and use their model knowledge to efficiently distribute their model.

Since the Distribution Model uses a modeling language and our context is agent-based, it's logical to base it on agents. Thus, we introduce **Distribution Agents** for distribution challenges. Each Distribution Agent handles a specific distribution issue. We define: **Communication Agent** (inter-instance communication), **Partitioning Agent** (initial partitioning and its issues), **Load Balancing Agent** (dynamic workload distribution), and **Data synchronization Agent** (data consistency during interactions)

Figure 2 illustrates a Distribution Model instance that ensures the decoupling between the modeling and the distribution. The figure shows the handling of distribution issues outside of the Thematic Model. Distribution Agents directly access Thematic Model data via their co-model relationship. The Distribution Model depends on Distribution Agents to gather data from the Thematic Model and to deploy distributed algorithms to effectively distribute the simulation.

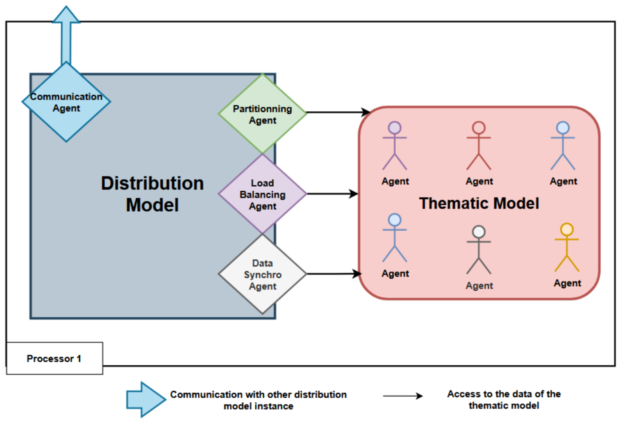


Figure 2: Internal functioning of a Distribution Model instance: Distribution Agents accessing the data of the Thematic Model instance.

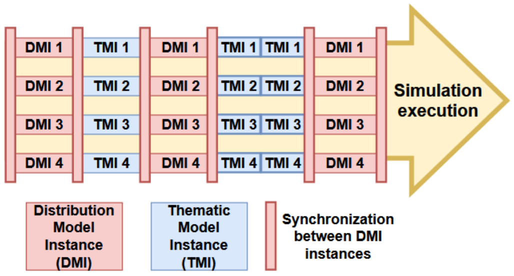

DMI 1
DMI 2
DMI 3
DMI 4
TMI 1
TMI 2
TMI 3
TMI 4
Simulation execution
Distribution Model Instance (DMI)
Thematic Model Instance (TMI)
Synchronization between DMI instances

Figure 3: Synchronized Execution of Thematic and Distribution Models.

## 4.1 Models Execution and Synchronization

The composition of the Distribution Model is closely linked to the Thematic Model, as the issues to be managed depend on the Thematic Model characteristics. Thanks to the modular design of the Distribution Model, the modeler can select the relevant Distribution Agents needed to distribute its Thematic Model.

When instantiating the Distribution Model (DM) and Thematic Model (TM) processes, the initial step is to create the DM environment and the Distribution Agents. The DM environment is identical to the TM environment. The Thematic Model's specific challenge dictates the Distribution Model's initial setup. For instance, complex, dynamic environments might require fine-grained partitioning, while many independent agents might necessitate even distribution for parallel execution.

DM are controlling the execution of the TM and can trigger the execution of the next cycle of the TM. After running this cycle, the DM instance can synchronizes with the other DM instances. Note that, depending on the model needs, a DM instance can run several cycles of its TM. Figure 3 shows the model pair's cycle progression, illustrating that TM instances can execute multiple times between DM cycles (e.g., once between the first two, twice before the third). Within the red section (DMI) in Figure 3, Distribution Agents have the capability to synchronize, communicate, and execute their behavior, thereby enabling Distribution Agent behavior to be triggered. For example, the Load Balancing Agent behavior can detect an unbalanced workload between processes and trigger a new partitioning.

## 4.2 Distributed Algorithms with Distribution Agents

The Distribution Model and the agent framework are used to implement distributed algorithms, synchronized at each execution step, in distributed agent instances. Diverse solution outlined earlier in Section 2 can be used.

For illustrative purposes, we detail a Partitioning Agent in the context of environmental partitioning in Alg 1. The K-MEAN algorithm divides a dataset into K clusters, based on data point proximity. It can be used to group spatially close agents onto the same process. When the Load Balancing Agent detects an uneven workload distribution among processes, it triggers the instance of the Partitioning Agent on each process to define a new partition. Each instance of the Partitioning Agent manages a centroid that defines the boundaries of its process.

The behavior of each instance of the Partitioning Agent can be defined as follows:

- *sendCentroid():* Partitioning Agent broadcasts the coordinates of its centroid to all other Partitioning Agents located on different processes
- *getClosestCentroids():* Partitioning Agent computes the distance between each data point, on the current process, and all the centroids. It then assigns point agents to the nearest cluster centroid to be migrated later.
- *exchangeDataPoint():* The Partitioning Agent asks the Communication Agent to migrate agents that are closer to another centroid located on a different process.
- *updateCentroidLocation():* Partitioning Agent updates the centroid coordinate based on newly assigned data point agents. The new location is the average location of the data point agents on the current process.

**Algorithm 1** Behavior of the Partitioning Agent using a distributed K-MEAN for partitioning

**Output**: An updated partition for each process

```
1: sendCentroid()
2: getClosestCentroids()
3: exchangeDataPoint()
4: updateCentroidLocation()
```

This distributed algorithm is one example of the potential applications of the distributed model approach to solve issues related to ABM distribution. By using Distribution Agents, we can effectively describe and implement various partitioning techniques, including partitioning used by previously presented distribution platforms.

## 4.3 Bridging the Gap Between Modelers and Distribution

The modularity of the architecture and its distribution-based decomposition allow for the modelers to adopt various approaches to use and to customize Distribution Agents, depending on their individual skills and needs, and to adapt distributed algorithms to fit specific Thematic Models needs. In addition, whenever new Distribution Models are developed, they can be added to a base of solutions to be reused at the following levels:

- **Agent Library:** Modelers have access to a library of ready-to-use Distribution Agents with pre-defined behaviors, giving them access to ready to use DM library. These agents can be directly used in a DM without requiring any additional work for the modeler.
- **Agent Personalization:** Modelers can modify the behavior, skills, actions and features of existing Distribution Agent from the Agent Library to suit their specific needs.
- **Advanced Agent Development:** Modelers can customize the distribution of their ABM through modeling or design customized solutions with new Distribution Agents.

## 5 Distribution Model Implementation With the GAMA Platform

Having explored the fundamental concepts of the Distribution Model, we now focus on the concrete implementation of Distribution Models. In this regard, the GAMA platform proves to be a suitable tool, since it is an easy-to-use, open-source modeling and simulation environment for creating spatially explicit agent-based simulations. Modelers can specify agent behavior in GAMA using GAML, a specialized modeling language.

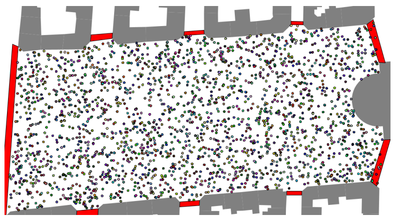


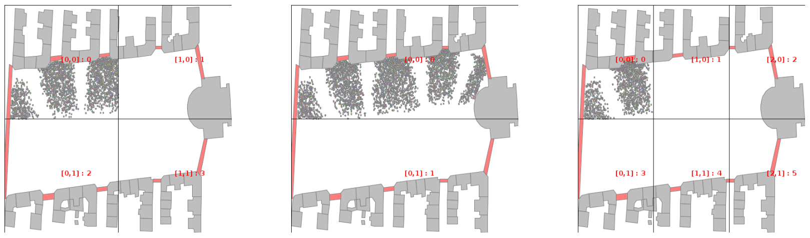
Figure 4: GAMA Thematic Model for crowd evacuation. Dots represent People agent, red shapes are exits.

We present an implementation of the Distribution Model and Distribution Agents concepts in the GAMA platform. We select a typical distribution scenario: a Thematic Model which needs to be distributed.We then implemented various Distribution Models with different approaches to distribute this Thematic Model [1]. For Thematic Model, we chose a model describing a crowd evacuation where obstacles are the buildings, from the GAMA model library. People agents are placed in the environment and have to escape through an exit. A snapshot of this GAMA Thematic Model is shown in Figure 4. People Agents (colored dots) move towards one of the exits (red shapes). Multiple Distribution Models with different partitioning and load balancing methods to illustrate the flexibility of our approach are presented in the following.

### 5.1 Static Partitioning : GAMA Grid Species

We present here a static partitioning solution based on a grid approach. The model uses a Grid Agent to specify the grid dimensions (rows and columns) and neighbor cell positions. Grids in the Distribution Model are easily defined with the GAMA platform.

[1]The implementations of these Distribution Models are available on our GitHub repository `https://anonymous.4open.science/r/Gama_distribution-9906`



(a) 2x2 GAML grid (b) 1x2 GAML grid (c) 3x2 GAML grid

Figure 5: Different configurations of the Grid Agent used for the partitioning of the Thematic Model. Different configurations can be defined from the same Distribution Model.

Using the same Grid Agent, we can create various grid patterns. Figure 5 illustrates the partitioning possibilities enabled by the grid approach.

Each process mirrors the environment and manages one grid cell. At the end of each simulation step, a Partitioning Agent relocates any People agents to the process responsible for their current cell, ensuring spatial continuity. The static grid partitioning, while simple and generic, cannot adapt to agent movement, causing workload imbalances and inefficient process usage, and fails to leverage the Thematic Model. The next section will present a more elaborate solution using dynamic partitioning.

## 5.2 Load Balancing with Distributed K-MEAN

To dynamically load balance and partition the workload between processes of the GAMA crowd evacuation Thematic Model, we developed a Distribution Model using the previously introduced K-MEAN distributed algorithm. In this DM, the Partitioning Agent and Load Balancing Agent implement the KMEAN distributed algorithm in their agent behavior to adapt the dynamic partitioning to the people agent movement. Figure 6 shows 4 snapshots of the Thematic Model with 5 clusters that dynamically change.

Despite distinct agent assignment methods (Partitioning in Grid, Load Balancing in KMEAN), both models utilize the same Communication Agent and snapshot/Thematic Model cycle execution, illustrating the potential for reuse provided by our approach.

## 5.3 Evacuation Model Distribution Performance Analysis

The comparison between KMEAN and GRID Distribution Models reveals good performance[2] across different scales of simulation (see Figure 7). In examining the centralized versus distributed execution, both models demonstrate substantial improvements in performance compared to the centralized approach. This is evident when simulating 20,000 agents, where the centralized execution requires 305 seconds, while both distribution methods achieve dramatic reductions: KMEAN reduces the time to 67.1 seconds

[2]Computations have been performed on the supercomputer facilities of the Mésocentre de calcul de Franche-Comté.

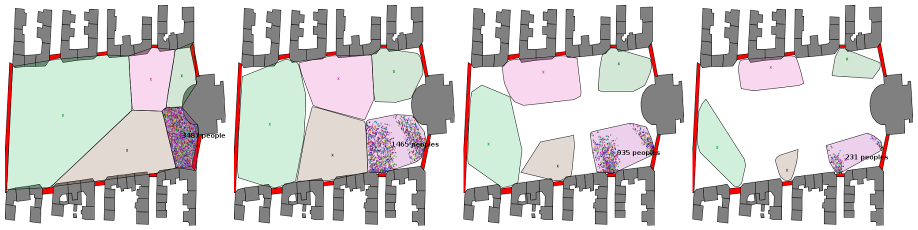


Figure 6: Distributed simulation of the Thematic Model using the distributed KMEAN. Five instances of the DM are used to distribute the Thematic Model. Each shape is managed by a different process and formed by the K-MEAN distributed algorithm. The pictures present the distributed simulation's evolution over time.

and GRID achieves 64.8 seconds with 8 processors, representing approximately 78% and 79% improvements respectively. The even distribution of exits in this scenario enables good performance for both models. A more spatially focused exit configuration would likely disadvantage GRID compared to KMEAN which is more flexible with the integrated load balancing. Nonetheless, this demonstrates the effectiveness of the Distribution Model approach to gain performance, even with a simple Distribution Model like GRID.

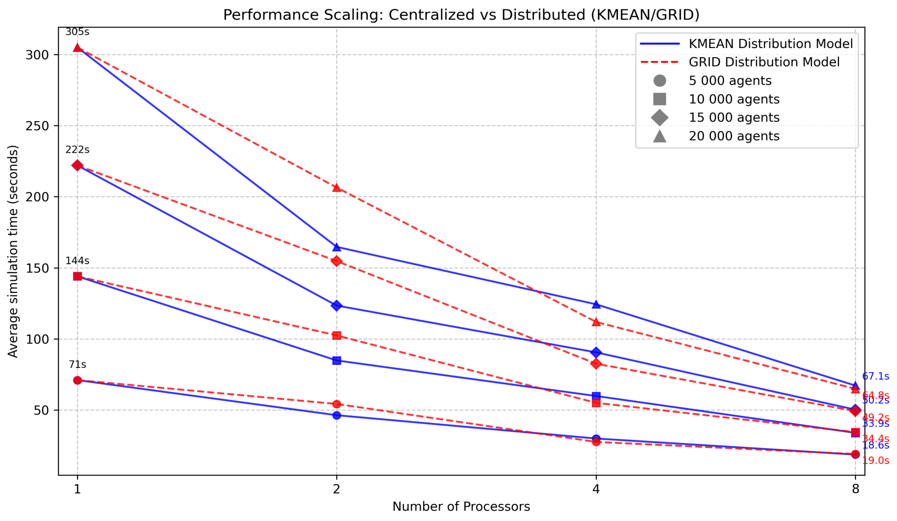


Figure 7: Performance of KMEAN and GRID Distribution Models for evacuation simulations with varying numbers of agents (5,000 to 20,000). The figure shows the average simulation time as a function of processor count used (1 to 8).

Results demonstrate that both Distribution Models successfully address the performance challenges of large-scale agent-based evacuation simulations, with each offering distinct advantages depending on the specific use case. Results strongly indicate the validity and effectiveness of our Distribution Model concept, particularly in handling the complexities of large-scale agent-based simulations. The observed performance demonstrates its capability to manage and distribute the computational workload efficiently as the number of agents increases significantly.

## 6 Conclusion and Future Work

This paper introduces a novel approach to ABM distribution by separating thematic concerns (Thematic Model) from distribution concerns (Distribution Model). The benchmark results highlight how this decoupling enables a flexible and scalable approach, allowing researchers to focus on model design and choose distribution support independently of implementation. Our approach uses modeling languages to enable regular modelers to distribute their ABMs independently, without needing distribution expertise, through a flexible modular Distribution Model based on Distribution Agents. The reusability and extensibility of Distribution Models and agents, along with a predefined library, enable broad applicability. Our experiments demonstrated successful GAMA model distribution without modifying GAMA or the Thematic Model. Future work will expand our approach with more diverse Distribution Models and agents to improve scalability and efficiency for large ABM simulations. Further research will aim to establish a formal approach for distribution model development.

## References

[1] Bellifemine, F., Poggi, A., Rimassa, G.: Developing multi-agent systems with jade. In: Intelligent Agent Theories Architectures and Languages. Springer Berlin Heidelberg (2001)

[2] Borges, F., Gutierrez-Milla, A., Luque, E., Suppi, R.: Care HPS: A high performance simulation tool for parallel and distributed agent-based modeling. Future Generation Computer Systems **68**, 59–73 (2017)

[3] Borges, F., Gutierrez-Milla, A., Suppi, R., Luque, E.: Strip partitioning for ant colony parallel and distributed discrete-event simulation **51**, 483–492 (2015)

[4] Borshchev, A.: The Big Book of Simulation Modeling: Multimethod Modeling with AnyLogic 6. AnyLogic North America (2013)

[5] Braubach, L., Pokahr, A.: The Jadex Project: Simulation. In: Multiagent Systems and Applications, vol. 45, pp. 107–128 (2013), intelligent Systems Reference Library

[6] Breugnot, P., Herrmann, B., Lang, C., Philippe, L.: A Synchronized and Dynamic Distributed Graph structure to allow the native distribution of Multi-Agent System simulations. In: International Conference on Parallel, Distributed and Network-Based Processing (2021)

[7] Breugnot, P., Herrmann, B., Lang, C., Philippe, L.: Data Synchronization in Distributed Simulation of Multi-Agent Systems. In: Advances in Practical Applications of Agents, Multi-Agent Systems, and Complex Systems Simulation. The PAAMS Collection. Springer (2022)

[8] Breugnot, P.: Distribution et synchronisation des simulations de Systèmes Multi-Agents. Phd thesis, Université Bourgogne Franche-Comté (2023)

[9] Chuang, T., Fukuda, M.: A Parallel Multi-agent Spatial Simulation Environment for Cluster Systems. In: 2013 IEEE 16th International Conference on Computational Science and Engineering. pp. 143–150. IEEE, Sydney, Australia

[10] Coakley, S., Gheorghe, M., Holcombe, M., Chin, S., Worth, D., Greenough, C.: Exploitation of high performance computing in the FLAME agent-based simulation framework. In: IEEE 14th International Conference on High Performance Computing and Communication (2012)

[11] Collier, N., North, M.: Parallel agent-based simulation with Repast for High Performance Computing. SIMULATION **89**(10), 1215–1235 (2013)

[12] Cordasco, G., De Chiara, R., Raia, F., Scarano, V., Spagnuolo, C., Vicidomini, L.: Designing Computational Steering Facilities for Distributed Agent Based Simulations (2013)

[13] Cosenza, B., Cordasco, G., De Chiara, R., Scarano, V.: Distributed load balancing for parallel agent-based simulations. In: 2011 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing. pp. 62–69. IEEE (2011)

[14] Dijkstra, E.W.: On the Role of Scientific Thought. In: Selected Writings on Computing: A personal Perspective, pp. 60–66. Springer New York, New York, NY (1982)

[15] Glake, D., Ocker, F., Lenfers, U., Clemen, T.: Adaptive partitioning for distributed multi-agent simulations. In: Proceedings of the 2022 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation. pp. 13–24. ACM (2022)

[16] Groen, D., Papadopoulou, N., Anastasiadis, P., Lawenda, M., Szustak, L., Gogolenko, S., Arabnejad, H.: Large-scale parallelization of human migration simulation (2024)

[17] Huynh, Q.N.: CoModels, engineering dynamic compositions of coupled models to support the simulation of complex systems (2016)

[18] Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., Balan, G.: MASON: A Multiagent Simulation Environment. SIMULATION **81**(7), 517–527 (2005)

[19] Márquez, C., César, E., Sorribes, J.: Graph-based automatic dynamic load balancing for HPC agent-based simulations. In: Euro-Par 2015: Parallel Processing Workshops, vol. 9523, pp. 405–416. Springer International Publishing (2015)

[20] North, M., Collier, N., Ozik, J., Tatara, E., Macal, C., Bragen, M., Sydelko, P.: Complex adaptive systems modeling with Repast Simphony. Complex Adaptive Systems Modeling (2013)

[21] Orooji, F., Sarjoughian, H.S., Taghiyareh, F.: Modeling & simulation of educational multi-agent systems in DEVS-suite. pp. 956–961 (2011)

[22] Richmond, P., Walker, D., Coakley, S., Romano, D.: High performance cellular level agent-based simulation with FLAME for the GPU. Briefings in Bioinformatics **11** (2010)

[23] Rousset, A., Herrmann, B., Lang, C., Philippe, L.: A survey on parallel and distributed multi-agent systems for high performance computing simulations. Computer Science Review **22**, 27–46 (2016)

[24] Rousset, A.: Contribution à la distribution et à la synchronisation des Systèmes Multi-Agents sur les super-calculateurs. Phd thesis, Université de Franche-Comté (2016)

[25] Rubio-Campillo, X.: Pandora: A Versatile Agent-Based Modelling Platform for Social Simulation (2014)

[26] Schimeczek, C., Deissenroth-Uhrig, M., Frey, U., Fuchs, B., Ghazi, A., Wetzel, M., Nienhaus, K.: FAME-Core: An open Framework for distributed Agent-based Modelling of Energy systems. Journal of Open Source Software (2023)

[27] Solar, R., Suppi, R., Luque, E.: High performance distributed cluster-based individual-oriented fish school simulation **4**, 76–85 (2011)

[28] Solar, R., Suppi, R., Luque, E.: Proximity load balancing for distributed cluster-based individual-oriented fish school simulations **9**, 328–337 (2012)

[29] Suryanarayanan, V.and Theodoropoulos, G., Lees, M.: PDES-MAS: Distributed Simulation of Multi-agent Systems. Procedia Computer Science **18**, 671–681 (2013)

[30] Sébastien, N.: Distribution et Parallélisation de Simulations Orientées Agents. Ph.D. thesis, Université de la Réunion (2010)

[31] Taillandier, P., Gaudou, B., Grignard, A., Huynh, Q.N., Marilleau, N., Caillou, P., Philippon, D.and Drogoul, A.: Building, composing and experimenting complex spatial models with the GAMA platform. GeoInformatica **23**, 299–322 (2019)

[32] Walker, D.W., Dongarra, J.J.: Mpi: a standard message passing interface. Supercomputer **12**, 56–68 (1996)

[33] Wilensky: Wilensky (1999). NetLogo http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.