MultiRepast4py: A Framework for Agent-Based Simulations on Multilayer Networks

Keng-Lien Lin and Parinaz Naghizadeh

University of California San Diego, San Diego CA 92092, USA
{kel030, parinaz}@ucsd.edu

Abstract. Agent-Based Simulations (ABS) offer a powerful approach for analyzing how individual agents' decisions and interactions within networked systems lead to system outcomes. ABS have been widely used across various fields, including in the study of the spread of diseases and (mis)information. However, traditional ABS platforms, such as NetLogo and Repast, simplify models by assuming a single network of interactions between agents. In reality, agents' interactions are typically multi-layered (i.e., involve multiple interconnected networks that influence agents' decisions). To address this limitation, we developed `MultiRepast4py`, a multilayer simulation tool extending the simulation capabilities of Repast4py. Our framework enables simulations on multilayer networked systems by efficiently reconstructing network data and utilizing agent attributes, allowing agents to dynamically access multilayer connections during simulation. By maintaining Repast4py's scalability and minimizing memory overhead, `MultiRepast4py` ensures high performance for large-scale simulations. Through simulation examples on the spread of information in social networks, we showcase how `MultiRepast4py` can enable more comprehensive agent-based simulations, guiding improved predictions and interventions.

Keywords: Agent Based Simulations · Multilayer Networks · Repast.

1 Introduction

Agent-based models and simulations have become a widely used computational approach for analyzing how the behavior and interactions of autonomous agents give rise to complex, system-level outcomes [20]. In an agent-based simulation (ABS), agents are first endowed with a set of attributes and decision-making heuristics. The simulation then allows these agents to interact repeatedly over time, with their interactions governed by an *interaction topology* (referred to here as a *network*). By running large-scale experiments in a cost-effective way, ABS enable researchers to analyze how various "micro"-scale assumptions on agents (e.g., their movement patterns, their propensity to catch a disease, or adopt an opinion), as well as different interventions to change agents' behavior or their network (e.g., limiting agents' exposure to content on social networks, or isolation/vaccination strategies), can (collectively) alter the "macro" outcomes (e.g., spread of a disease or misinformation).

There currently exist several commonly used simulations platforms for ABS, built on different programming languages and offering different levels of customization; these include NetLogo [28], Repast [9], MASON [19], FLAME [16], and Swarm [21]; we refer the interested reader to [18] for a comparison of (some of) these platforms. The versatility of ABS and the accessibility of these platforms has led to the wide use of ABS by researchers in many fields. For instance, ABS have been leveraged to gain insights into COVID-19 dynamics and interventions [15], healthcare resource management [7], spatio-temporal dynamics of crime [24], supply chain management [14], and policy design in labor markets [10]. The capabilities of these platforms have also been extended in several directions, including by integrating micro-findings from human subject experiments into agents' decision models [27], advocating for data-driven ABS [23], and developing strategies for scaling-up to accommodate data-intensive simulations [3].

The gap. Despite their widespread use and recent advances, existing ABS platforms assume that agents' interactions occur within a *single* interaction topology/network (e.g., they account for the spread of misinformation over one social network, or the spread of a disease over one mode of interaction). However, there exist many settings, including in the study of disease spread and information dynamics, where this approach overlooks the complexities of intersectionality of humans - how an individual's experiences and outcomes can be shaped by their multiple social identities and networks. For instance, individuals with accounts on two social media platforms (e.g., TikTok and Instagram) follow, and are followed by, different accounts on each platform, can cross-post the same content on both platforms, and may further use these platforms at differing frequencies and for different purposes; studying the spread of (mis)information on any one of these platforms in isolation would fail to capture these nuances. Similarly, a family network and a workplace network may differ in size, interaction frequency, and types of interactions, yet both can significantly influence an individual's exposure to a disease; isolation or vaccination policies that do not explicitly account for these differences may therefore be suboptimal.

Multilayer networks. For the situations described above, and other similar contexts, *multilayer networks* have been proposed as a model to simultaneously account for the multiple modalities of interactions between agents; see [4, 17, 1, 25, 5] for surveys of this field. The study of multilayer networks, as opposed to the study of their constituent *single-layer* networks in isolation, can offer a nuanced understanding of how the "micro" differences between the various modalities of interactions (ranging from the agents' attributes on each network, to the differences in each network's topology, to the frequency with which agents are actively interacting with others in each network) impact macro outcomes. Existing works have used the formalism of multilayer networks to study game theoretical decision making over multiple interaction/information modalities (e.g., [26, 13, 2, 11]), to evaluate the resilience of networks of networks against failures or attacks, often by studying percolation (e.g., [6, 12, 30, 22, 31, 32, 4]), and to analyze dynamical processes on interacting networks (such as diffusion and spreading processes) to identify the critical thresholds for an outbreak (in epidemic modeling) or consensus (in the study of opinion dynamics) on these networks (e.g., [29, 33, 4]). Despite the broad applicability of multilayer network models and the increasing body of research dedicated to them, to the best of our knowledge, no existing agent-based simulation platform supports multilayer networks. This limitation confines both current and potential future research to smaller-scale, custom-built simulation environments. While experiments involving multilayer systems have been conducted, they all require researchers to develop their own frameworks, as no platform currently offers native support for such models.

Our contributions. To address these limitations, we have developed `MultiRepast4py`, a multilayer agent-based simulation framework. This framework builds on an existing ABS simulation platform (specifically, Repast4py [9]), enabling researchers to model complex, multilayered interactions without needing extensive programming skills, making it accessible to researchers across various disciplines. The framework's design maintains Repast4py platform's flexibility and scalability, which can enable leveraging High-Performance Computing (HPC) resources for large-scale (multilayer) simulations. It is also prepared for integration with data-driven approaches, allowing researchers to easily incorporate real-world data into their models.

In more detail, our framework makes the following key contributions:

1. **Multilayer Agent-Based Simulation Capability:** `MultiRepast4py` introduces the ability to run *multilayer* agent-based simulations by building on the established Repast4py platform. This bridges a crucial gap in ABS technology, allowing for more nuanced and realistic modeling of complex systems where agents interact across multiple interconnected networks.
2. **Scalability and Flexibility:** `MultiRepast4py` retains all the features that make the Repast suite powerful, including its renowned scalability and flexibility. This ensures that researchers can tackle large-scale, complex simulations without sacrificing computational efficiency.
3. **Seamless Integration:** `MultiRepast4py` seamlessly integrates with Repast4py, requiring minimal modifications to model logic. By reconstructing network files, users need only add one attribute to their agents to access multilayer connections during simulation. This design ensures accessibility for users familiar with the Repast family.
4. **Opportunities for Customization:** `MultiRepast4py` is developed in Python, and remains open to additional features and enhancements. This includes the potential for incorporating data-driven models, further increasing the framework's relevance and applicability in an era of big data.

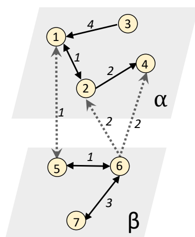
Illustration through simulation studies. To demonstrate the practical utility of `MultiRepast4py`, we present a case study analyzing rumor propagation across interconnected social networks. This is done through two complementary simulations: a reduced-scale proof-of-concept and a full-scale validation. The study builds upon Repast4py's single-layer rumor model, explicitly demonstrating how to enhance conventional agent-based simulations with multilayer capabilities. The layers represent distinct social media platforms with unique network topologies (Erdős-Rényi random graphs) and interaction patterns. Through parameterized layer configurations, we model critical real-world phenomena like cross-platform connectivity and varying interaction frequencies – features impossible to capture in single-layer ABS. Our small-scale experiment reveals how nodes with balanced cross-layer connectivity (combined degree centrality = 7) outperform those with superior single-layer positions, while the large-scale extension demonstrates persistent multilayer effects in more realistic networks (50,000 nodes per layer). By comparing single-layer versus multilayer seeding strategies, we empirically validate that ignoring platform interdependence leads to suboptimal diffusion predictions – a critical limitation for social media analytics. The seamless scaling from 25 to 50,000 agents per layer further demonstrates our framework's computational feasibility, executing efficiently on consumer-grade hardware. This case study exemplifies how `MultiRepast4py` enables researchers to 1) convert existing single-layer models into multilayer systems and 2) identify emergent phenomena arising from cross-layer interactions.

2 Multilayer Networks

We model *multilayer networks* as structures consisting of multiple *single-layer networks* connected together, with each layer corresponding to a particular type of social relation, mode of interaction, or information channel, between agents.

Formally, each layer α of a multilayer network is a network represented by a graph $\mathcal{G}^\alpha =< \mathcal{N}^\alpha, \mathcal{A}^\alpha >$, where $\mathcal{N}^\alpha$ denotes the set of agents in layer α and $\mathcal{A}^\alpha$ denotes the *intra-network* adjacency matrix. An agent $m \in \mathcal{N}^\alpha$ could be, e.g., an individual in a social network. An edge $a^\alpha_{mn} \in \mathcal{A}^\alpha$ represents the dependency between agents m and n in $\mathcal{G}^\alpha$, and can capture, e.g., the exchange of information (in-person or virtual). We assume that interactions are undirected and weighted (reflecting mutual dependencies, but with potentially different strengths).

In addition, as these layers do not operate in isolation, there exist connections between nodes in different layers, captured using an *inter-network* adjacency matrix $\mathcal{B}^{\alpha,\beta} \in \mathbb{R}^{\mathcal{N}^\alpha \times \mathcal{N}^\beta}$. An edge $b^{\alpha,\beta}_{mn} \in \mathcal{B}^{\alpha,\beta}$ indicates that the decisions made by agent m in $\mathcal{G}^\alpha$ are linked to those of agent n in $\mathcal{G}^\beta$. In this paper, we focus on the case of an *identity* inter-network adjacency matrix. These matrices capture a special case of multilayer networks in which the different layers consist of the *same* set of nodes/agents, but where the nature of the relation between the nodes being different in each layer; these are also referred to as *multiplex networks* in the literature [4]. Multiplex networks are primarily used when agents have access to different communication or interaction modalities. Examples include the spread of social influence campaigns between social networks (e.g. Twitter in layer α and Facebook in layer β), or the spread of diseases as individuals interact with others in both their family (layer α) and work (layer β) networks.



$$A_\alpha = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 2 \\ 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad A_\beta = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 3 \\ 0 & 3 & 0 \end{bmatrix}$$

$$\bar{A} = \begin{array}{c} \mathcal{A}^{\alpha} \qquad\qquad\quad \mathcal{B}^{\alpha,\beta} \\ \left[\begin{array}{cccc|ccc} 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 2 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 2 & 0 & 2 & 1 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 \end{array}\right] \\ \mathcal{B}^{\beta,\alpha} \qquad\qquad\quad \mathcal{A}^{\beta} \end{array}$$

Fig. 1. An example of a multilayer network, and its intra-layer and inter-layer adjacency matrices.

An illustration of a multilayer network is shown in Figure 1. As illustrated in the figure, the inter-network and intra-network adjacency matrices can be collected into a single "supra-adjacency" matrix $\bar{A}$. One might then propose that the interactions of agents can be viewed as happening over a single-layer network with adjacency matrix $\bar{A}$. We note, however, that the multilayer network is different from this single-layer network with adjacency matrix $\bar{A}$. First, a multilayer model can impose different structural properties on the adjacency matrices in each layer, and enables us to investigate their impact on emergent phenomena accordingly. For example, real-world data can be used to learn the different structural properties of two social networks separately, and allow for each to be reflected independently in the ABS environment. Further, disturbances or information from a node may spread within each layer following a different process, and at a different time scale. For instance, individuals may interact with their co-workers during the week, and with their extended family and friends over the weekend; they may also engage in different activities with each group, therefore impacting the likelihood of the spread of a disease between them in each context differently. Lastly, multilayer network simulations allow us to distinguish how regulators can propose and enact interventions in each layer.

3 `MultiRepast4py`: A Multilayer ABS Framework

Given the limitations of traditional ABS platforms and the importance of incorporating multilayer functionalities, our objective is to create ABS tools that enable agent-based simulations over multilayer networks. In this section, we first outline our evaluation of existing simulation platforms to identify those best suited for developing and implementing a multilayer ABS framework (Section 3.1). We then provide details on our implementation of `MultiRepast4py`, and outline the main challenges that were addressed (Section 3.2).

3.1 Platform Selection and Rationale

As noted in the introduction, there exist several widely used platforms for ABS, built on different programming languages; these include NetLogo [28], Repast

[9], MASON [19], FLAME [16], and Swarm [21], and their extensions. Among these, NetLogo is the highest-level platform, with a relatively simple programming language and a well-developed graphical interface; however, while NetLogo offers robust features for certain applications (e.g., grid-based ABMs), it presents limitations for highly specialized or complex simulations. The remaining platforms (e.g., Repast, MASON, Swarm) provide a *framework* (a set of concepts for describing an agent-based model and the required simulation components) along with a *software library* (implementations of the framework using customized simulation tools). While these require additional programming skills, they facilitate customization, as well as integration with a wide range of data analysis and machine learning libraries to enable data-driven ABS. Our focus in this work is similarly on developing a framework and library for multilayer ABS, while maintaining the flexibility for customization and data-driven implementations.

Accordingly, Repast4py [8], a Python-based member of the Repast Suite, was chosen as the development platform for the following reasons:

1. *Scalability*: Repast4py simplifies the construction of large-scale ABMs that can be distributed across multiple processing cores using MPI, enabling efficient execution of complex simulations.
2. *Flexibility*: The platform's dynamic simulation step capabilities, facilitated by the scheduling feature, enables a high degree of customization, ensuring the seamless implementation of our multilayer approach.
3. *Extensibility*: Built on Python, Repast4py inherently supports integration with a wide range of data analysis and machine learning libraries, facilitating data-driven modeling and future expansions.

3.2 Implementation Details

The multilayer functionality in `MultiRepast4py` is implemented through a structured process that embeds each agent's multilayer edges into a unified data structure. Specifically, connections from the network files are mapped using the node identifiers (`node_id`) defined in the first file, ensuring consistent agent identification across all layers.

Each agent is associated with its own `shadow_data`, a list of dictionaries where each dictionary corresponds to a specific layer and stores that agent's outgoing edges for that layer. This design enables efficient differentiation of interlayer connections while maintaining a cohesive representation for each agent.

The implementation is organized into four sequential stages, each addressing a critical aspect of our multilayer simulation framework. First, multilayer network files are parsed and merged into a unified structure, creating a cohesive representation of all layers. Second, edge data is transformed—compressed and encoded—to satisfy the Repast4py's network file format requirements while enhancing memory efficiency. Third, agents are initialized by reconstructing their multilayer connections from the processed data, enabling a decentralized setup that supports parallel execution. Finally, the simulation step function is adapted to allow layer-specific interactions, ensuring that each simulation cycle accurately reflects the dynamics of the corresponding layer. Together, these stages facilitate efficient memory usage and scalable parallel processing while preserving the essential semantics of multilayer networks. The following sections provide detailed explanations of each component.

Network File Parsing. In this stage, multiple network files are provided as input, each representing a layer. The file contains agent information in the form of unique identifiers (UIDs): (`node_id`, `agent_type`, and `rank`) and the edge connections between nodes. These network files can be generated using Repast4py's `write_network` function. The parsing process constructs a data structure where each agent maintains a list of dictionaries representing its outgoing edges, indexed by layer:

$$[\{(\text{uid}) : \text{edge\_weight}, \dots\}, \dots]$$

By storing outgoing edges as agent attributes, the model achieves greater decentralization, thereby reducing the need for message passing between ranks. A list of dictionaries is employed instead of a nested dictionary because, in most cases, the number of layers remains fixed, making a list a more memory-efficient choice. However, at the layer level, edges may be dynamically added or modified, justifying the use of a dictionary structure for efficient access and updates.

Data Compression & Encoding. Outgoing edge lists for agents cannot be directly stored in Repast4py network files, as the framework's network file format prohibits nested data structures and requires string-based keys (e.g., connection tuples must be formatted as strings). To ensure compatibility with Repast4py's network serialization requirements, edge data undergoes a four-step transformation:

1. **Stringification**: Tuple keys are converted to strings for JSON compatibility.
2. **UTF-8 Encoding**: Encode JSON-serialized structure with UTF-8.
3. **zlib Compression**: Reduce storage overhead using zlib compression
4. **Base64 Encoding**: Convert compressed bytes to an ASCII-safe string via Base64 to avoid file encoding conflicts.

The final encoded string is stored in the following format

{"data" : "[Base64 string]"}

Agent Initialization. During the agent initialization phase, the `read_network` function is invoked to load the modified network file. A custom agent creation function is then employed to decode the JSON-encoded attribute, restoring it to its original list-of-dictionaries format. This approach allows the initialization process to be executed in parallel across multiple ranks, as each agent independently retrieves and reconstructs its respective edge data. Consequently, each agent gains access to its outgoing edges across all layers while maintaining a scalable structure.

Algorithm 1 Multi-Layer Network Reconstruction

Require: List of network file paths *file_paths*

Ensure: Unified network file with compressed multi-layer edges

```
Parsing Phase
procedure ReconstructNetworkFiles(file_paths)
    num_layers ← len(file_paths)
    node_info ← {}, all_nodes ← ∅
    agents ← defaultdict(λ : [{} for _ in range(num_layers)])
    for layer_index, file_path in file_paths do
        lines ← ReadContents(file_path)
        directed ← ParseHeader(lines[0])
        found_edges ← False
        for line in lines[1 :] do
            if not found_edges then
                if line.strip() = "EDGES" then
                    found_edges ← True
                    Continue
                end if
                if layer_index = 0 then
                    Parse node_id, agent_type, rank ← line
                    node_info[node_id] ← (node_id, agent_type, rank)
                    all_nodes.add(node_id)
                end if
            else
                Parse edge (u, v, weight) ← line
                agents[u][layer_index][v] ← weight
                if not directed then
                    agents[v][layer_index][u] ← weight
                end if
            end if
        end for
    end for
Compression Phase
    compressed_edges ← []
    for node_id in sorted(all_nodes) do
        uid ← node_info[node_id]
        edge_dict ← agents[node_id]
        compressed ← CompressEdges(edge_dict)
        compressed_edges.append(compressed)
    end for
File Writing Phase
    Create copy of base file (First file in list)
    Inject compressed_edges {'data' : compressed_edges} as node attributes
    Write modified content to new file
end procedure
```

Modify Step function. The multilayer simulation is facilitated through Repast4py's scheduling mechanism. To enable multilayer simulations, a modified step function is defined in `MultiRepast4py`, which accepts a layer parameter to specify the active layer for a given simulation step. The function determines whether an agent participates in a particular layer by checking the presence of outgoing edges within that layer. The spreading process is then executed by iterating over the agent's outgoing edges in the specified layer, allowing users to define custom propagation mechanisms. Specifically, an agent's list of neighbors can be retrieved using:

```
Agent.shadow_data[layer].keys()
```

If an agent has no outgoing edges in a given layer, this function returns an empty list, otherwise, it returns a list of tuple(uids).

4 Experimental Demonstration

To demonstrate the capabilities of our multilayer simulation framework, we present a case study focused on information dynamics – specifically, rumor propagation. This topic is commonly explored in agent-based simulations and serves as an excellent example to highlight the advantages of our multilayer approach.

This case study is based on modifications of existing demo, Tutorial 2 - The Rumor Network Model from Repast4py. We selected this example to better illustrate the process of transforming a single-layer ABS into a multilayer ABS. The simulation was executed on an Apple M2 chip with 8GB RAM, showing that our framework is accessible on standard computing platforms.

4.1 Multilayer Analysis of Rumor Propagation Through Cross-Platform Interaction

This case study demonstrates how multilayer network simulations reveal propagation dynamics that conventional single-layer analyses cannot capture, particularly through the mechanism of cross-platform information diffusion.

Model Configuration. We implement a two-layer multiplex network model where each layer contains 25 nodes representing individuals active on distinct social platforms. The experimental configuration employs a reduced-scale network to enable clear demonstration of multilayer interaction effects, with larger-scale validation presented in Section 4.2.

- **Network Topology:** Both layers' connections are generated via Erdős-Rényi random graphs($G(n, p)$ model, $n = 25, p = 0.1$), producing sparse networks with average degree $k = 2.5$. This model was selcted for its simplicity, controllability, and reproducibility.
- **Agent State Model:** Each agent has the following attribute
  - `received_rumor`: Binary state (0=uninformed, 1=informed)

- `shadow_data`: Multilayer adjacency list storing intra-layer connections

Propagation Dynamics. The rumor dissemination process operates as:

1. Both layer activates at each time step
2. Within the activate layer, Informed nodes attempt transmission to adjacent uniformed neighbors
3. Per-contact infection probability $\beta = 0.005$
4. New informed nodes participate in next spreading cycle

Simulation Protocol. We execute 100 Monte Carlo replications for each of 25 network seeds, yielding 2,500 independent simulations (100 time steps each). All results present ensemble averages with stochastic effects mitigated through this extensive sampling.

Combined Degree Centrality. For a multilayer network with L layers, the **combined degree centrality** C_D^{combined} of an agent m is defined as:

$$C_D^{\text{combined}}(m) = \sum_{\gamma=0}^{L-1} \frac{1}{T_\gamma} \cdot C_D^\gamma(m), \tag{1}$$

- $C_D^\gamma(m)$ denotes the degree centrality of agent m in layer γ,
- T_γ is the execution interval of layer γ, specifying how frequently the layer is active; it must be strictly positive ($T_\gamma > 0$).

Empirical Findings. Figure 2 demonstrates the propagation patterns for three strategic seed nodes, quantified through cumulative adoption curves. Table 1 provides structural context through degree centrality measures.

Table 1. Node Centrality Measures Across Network Perspectives. Combined degree centrality is calculated as the sum of intra-layer degree centralities, weighted by the reciprocal of each layer's activation interval ($T_0, T_1 = 1$).

	Degree Centrality		
Seed Node	Layer 1	Layer 2	Combined
Agent 2	3	4	**7**
Agent 23	**4**	1	5
Agent 24	1	**5**	6

The propagation dynamics reveal several critical insights. Despite Agents 23 and 24 exhibiting maximal intra-layer centrality in their respective platforms ($Layer1 : C_D = 4, Layer2 : C_D = 5$), Agent 2 demonstrates superior global propagation efficiency (AUC = 365.65 vs 274.62 and 312.30) due to its cross-platform connectivity (Combined $C_D = 7$). This emergent property remains invisible to single-layer analyses - traditional centrality metrics would prioritize Agents 23 and 24 as optimal seeds within their native platforms, overlooking the critical role of inter-layer connectivity in system-scale diffusion. This

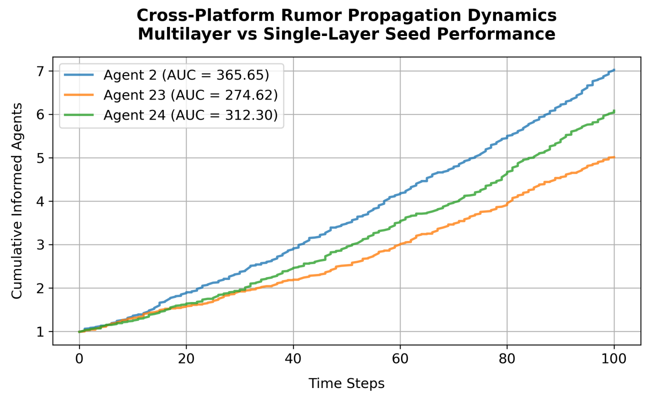


Fig. 2. Rumor propagation across multilayer networks. Curves represent mean adoption rates (N=100 simulations) for three seed nodes. Area Under Curve (AUC) values quantify total propagation efficiency.

highlights the importance of multilayer network analysis, made possible by the `MultiRepast4py`, in accounting for the impacts of different interacting networks on global outcomes.

4.2 Large-Scale Validation of Multilayer Propagation Dynamics

To evaluate the scalability and robustness of our framework, we extend the analysis presented in Section 4.1 by conducting a full-scale simulation of cross-platform information diffusion. This experiment quantifies the persistence of multilayer interaction effects in realistic network configurations and demonstrates the computational feasibility of large-scale multilayer agent-based simulations.

Model Configuration. We implement a two-layer multiplex network with 50,000 agents per layer, preserving the structural consistency of the small-scale demonstration while scaling the network parameters to reflect real-world social platforms.

- **Network Topology:** Each layer is instantiated as an Erdős-Rényi random graph$(G(n,p)model, n = 50000, p = 0.0005)$, yielding networks with an average degree $k = 25$.
- **Agent State Model:** As described in Section 4.1.

Propagation Dynamics: As detailed in Section 4.1, we set $\beta = 0.004$ to decelerate the spread, enhancing clarity and interpretability in the plot.

Simulation Protocol: We perform 100 Monte Carlo replications for each of three network seeding strategies, resulting in 300 independent simulations (each spanning 100 time steps). Seed nodes are selected according to three criteria:

- Top 10 degree centrality nodes in Layer 1 (single-layer perspective)
- Top 10 degree centrality nodes in Layer 2 (single-layer perspective)
- Top 10 combined degree centrality nodes (multilayer perspective)

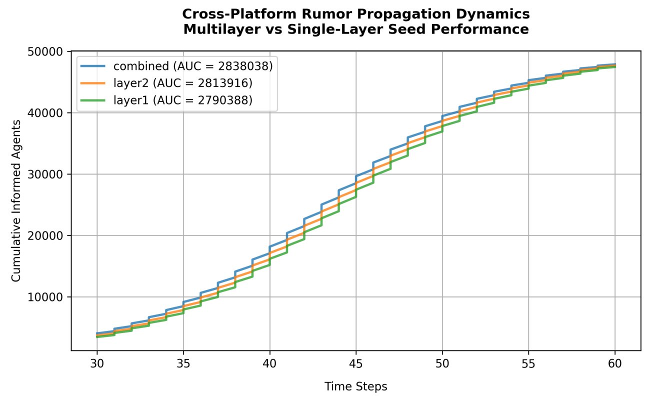


Fig. 3. Large-scale rumor propagation across Multilayer Networks. Curves depict the mean adoption rates (N=100 simulations) across three seeding strategies.

Empirical Findings. Figure 3 illustrates the propagation trajectories for each seeding strategy, with the Area Under the Curve (AUC) serving as a quantitative measure of overall dissemination efficiency.

Although the absolute differences in AUC values are modest relative to those observed in the small-scale demonstration, statistical analyses (t-test, Combined vs. Layer 1, p-value: 0.0007) confirm that the disparities are significant. Notably, the combined centrality seeds achieve approximately 10% greater population penetration at key diffusion milestones compared to the single-layer strategies. These findings underscore the importance of cross-platform connectivity in enhancing global diffusion efficiency in large networks.

5 Conclusion

We have developed `MultiRepast4py`, a multilayer agent-based simulation framework. This framework builds on an existing ABS simulation platform (specifically, Repast4py [9]) enabling researchers to model complex, large-scale, multilayered interactions without needing extensive programming skills, making it accessible to researchers across various disciplines. We used simulations studies on rumor spreading to highlight the advantages of our multilayer approach. Our use cases were selected from existing ABS on single-layer networks, to further illustrate the process of transitioning from single-layer to multilayer ABS using our proposed platform. As noted in the introduction, our framework has the potential to be integrated with data analysis techniques to enable data-driven (multilayer) ABS; we view this as an important directions of future work.

Acknowledgments. This work is supported by the University of California San Diego Summer Research Internship program, and by the NSF under award CCF-2416311.

Disclosure of Interests. The authors declare no competing interests.

References

1. Aleta, A., Moreno, Y.: Multilayer networks in a nutshell. Annual Review of Condensed Matter Physics **10**, 45–62 (2019)
2. Battiston, F., Perc, M., Latora, V.: Determinants of public cooperation in multiplex networks. New Journal of Physics **19**(7), 073017 (jul 2017). https://doi.org/10.1088/1367-2630/aa6ea1, https://dx.doi.org/10.1088/1367-2630/aa6ea1
3. Bhattacharya, P., Ekanayake, S., Kuhlman, C.J., Lebiere, C., Morrison, D., Swarup, S., Wilson, M.L., Orr, M.G.: The matrix: An agent-based modeling framework for data intensive simulations. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems. pp. 1635–1643 (2019)
4. Boccaletti, S., Bianconi, G., Criado, R., Del Genio, C.I., Gómez-Gardenes, J., Romance, M., Sendina-Nadal, I., Wang, Z., Zanin, M.: The structure and dynamics of multilayer networks. Physics reports **544**(1), 1–122 (2014)
5. Bródka, P., Musial, K., Jankowski, J.: Interacting spreading processes in multilayer networks: A systematic review. IEEE Access **8**, 10316–10341 (2020). https://doi.org/10.1109/ACCESS.2020.2965547
6. Buldyrev, S.V., Parshani, R., Paul, G., Stanley, H.E., Havlin, S.: Catastrophic cascade of failures in interdependent networks. Nature **464**(7291), 1025–1028 (2010)
7. Cabrera, E., Taboada, M., Iglesias, M.L., Epelde, F., Luque, E.: Optimization of healthcare emergency departments by agent-based simulation. Procedia computer science **4**, 1880–1889 (2011)
8. Collier, N., Ozik, J.: Distributed agent-based simulation with repast4py. In: 2022 Winter Simulation Conference (WSC). pp. 192–206. IEEE (2022)
9. Collier, N.: Repast: An extensible framework for agent simulation. The University of Chicago's social science research **36**, 2003 (2003)
10. De Bufala, N., Kant, J.D.: An evolutionary approach to find optimal policies with an agent-based simulation. In: 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), (2019)
11. Ebrahimi, R., Naghizadeh, P.: United we fall: On the nash equilibria of multiplex network games. In: 2023 59th Annual Allerton Conference on Communication, Control, and Computing (Allerton). pp. 1–8. IEEE (2023)
12. Gao, J., Buldyrev, S.V., Stanley, H.E., Havlin, S.: Networks formed from interdependent networks. Nature physics **8**(1), 40–48 (2012)
13. Gómez-Gardeñes, J., Reinares, I., Arenas, A., Floría, L.M.: Evolution of cooperation in multiplex networks. Scientific Reports **2**(1), 620 (Aug 2012). https://doi.org/10.1038/srep00620, https://doi.org/10.1038/srep00620
14. Julka, N., Srinivasan, R., Karimi, I.: Agent-based supply chain management—1: framework. Computers & Chemical Engineering **26**(12), 1755–1769 (2002)
15. Kerr, C.C., Stuart, R.M., Mistry, D., Abeysuriya, R.G., Rosenfeld, K., Hart, G.R., Núñez, R.C., Cohen, J.A., Selvaraj, P., Hagedorn, B., et al.: Covasim: an agent-based model of covid-19 dynamics and interventions. PLOS Computational Biology **17**(7), e1009149 (2021)
16. Kiran, M., Richmond, P., Holcombe, M., Chin, L.S., Worth, D., Greenough, C.: Flame: simulating large populations of agents on parallel hardware architectures. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1. pp. 1633–1636 (2010)
17. Kivelä, M., Arenas, A., Barthelemy, M., Gleeson, J.P., Moreno, Y., Porter, M.A.: Multilayer networks. Journal of complex networks **2**(3), 203–271 (2014)

18. Kravari, K., Bassiliades, N.: A survey of agent platforms. Journal of Artificial Societies and Social Simulation **18**(1), 11 (2015). https://doi.org/10.18564/jasss.2661, http://jasss.soc.surrey.ac.uk/18/1/11.html
19. Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., Balan, G.: Mason: A multiagent simulation environment. Simulation **81**(7), 517–527 (2005)
20. Macal, C.M., North, M.J.: Agent-based modeling and simulation. In: Proceedings of the 2009 winter simulation conference (WSC). pp. 86–98. IEEE (2009)
21. Minar, N., Burkhart, R., Langton, C., Askenazi, M., et al.: The swarm simulation system: A toolkit for building multi-agent simulations (1996)
22. Parandehgheibi, M., Modiano, E.: Robustness of interdependent networks: The case of communication networks and the power grid. In: 2013 IEEE Global Communications Conference (GLOBECOM). pp. 2164–2169. IEEE (2013)
23. Rand, W.: Theory-interpretable, data-driven agent-based modeling. Social-behavioral modeling for complex systems pp. 337–357 (2019)
24. Rosés, R., Kadar, C., Gerritsen, C., Rouly, C.: Agent-based simulation of offender mobility: Integrating activity nodes from location-based social networks. In: Aamas. pp. 804–812 (2018)
25. Salehi, M., Sharma, R., Marzolla, M., Magnani, M., Siyari, P., Montesi, D.: Spreading processes in multilayer networks. IEEE Transactions on Network Science and Engineering **2**(2), 65–83 (2015)
26. Shahrivar, E.M., Sundaram, S.: The game-theoretic formation of interconnections between networks. IEEE Journal on Selected Areas in Communications **35**(2), 341–352 (2017)
27. Smith, E.B., Rand, W.: Simulating macro-level effects from micro-level observations. Management Science **64**(11), 5405–5421 (2018)
28. Tisue, S., Wilensky, U.: Netlogo: A simple environment for modeling complexity. In: International conference on complex systems. vol. 21, pp. 16–21. Citeseer (2004)
29. Yağan, O., Gligor, V.: Analysis of complex contagions in random multiplex networks. Physical Review E **86**(3), 036103 (2012)
30. Yağan, O., Qian, D., Zhang, J., Cochran, D.: Optimal allocation of interconnecting links in cyber-physical systems: Interdependence, cascading failures, and robustness. IEEE Transactions on Parallel and Distributed Systems **23**(9), 1708–1720 (2012)
31. Zhang, Y., Arenas, A., Yağan, O.: Cascading failures in interdependent systems under a flow redistribution model. Physical Review E **97**(2), 022307 (2018)
32. Zhang, Y., Yağan, O.: Robustness of interdependent cyber-physical systems against cascading failures. IEEE Transactions on Automatic Control **65**(2), 711–726 (2019)
33. Zhuang, Y., Yağan, O.: Multistage complex contagions in random multiplex networks. IEEE Transactions on Control of Network Systems **7**(1), 410–421 (2019)