Netlogopy: Unlocking Advanced Simulation and Integration for NetLogo Using Python

Nourddine Bouaziz, Belgacem Bettayeb, M'Hammed Sahnoun, Adnan Yassine

Proceedings of the Operational Research Society Simulation Workshop 2025 (SW25)
M. Luis, A. Harper, T. Monks and N. Mustafee, eds.
https://doi.org/10.36819/SW25.019

NETLOGOPY: UNLOCKING ADVANCED SIMULATION AND INTEGRATION FOR NETLOGO USING PYTHON

Nourddine Bouaziz

CESI LINEACT
Saint-tienne-du-Rouvray, France
nbouaziz@cesi.fr

Belgacem Bettayeb

CESI LINEACT
Lille, France
bbettayeb@cesi.fr

M'hammed Sahnoun

CESI LINEACT
Saint-tienne-du-Rouvray, France
msahnoun@cesi.fr

Adnan Yassine

LAMH, University of Le Havre Normandie
Le Havre, France
adnan.yassine@univ-lehavre.fr

ABSTRACT

NetLogo is widely recognized as one of the most popular software tools for agent-based simulation. However, it has notable limitations, particularly the lack of advanced libraries in specialized areas such as optimization, artificial intelligence (AI), and mechanical or electrical modeling. On the other hand, Python is a feature-rich programming language that is increasingly used in various research domains. This study explores the integration of NetLogo and Python to leverage the strengths of both tools. The result of this integration is the Netlogopy library, which allows direct control of NetLogo agents from Python, providing greater flexibility through Pythons ecosystem. Netlogopy is a freely available library that adds an additional layer to existing NetLogo models, enhancing simulation capabilities and making them more accessible to researchers.

Keywords:

Agent-based simulation, NetLogo, Python, Netlogopy, Integration

1 INTRODUCTION

Agent-based simulation (ABS) is a powerful technique for modeling complex systems across various fields, such as biology, economics, and operations management (Wilensky 2015). NetLogo, an agent-based simulation environment, is a popular choice due to its free availability and its ability to model systems with numerous agents (Wilensky 2015). However, while NetLogo excels in handling agents and interactions within complex simulations, it has limitations when it comes to performing more advanced computations, such as optimization, artificial intelligence (AI), or physical process modeling. Python, on the other hand, has emerged as a versatile programming language with a vast range of libraries and capabilities in various scientific fields (Oliphant et al. 2006, McKinney et al. 2010, Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg, Vanderplas, Passos, Cournapeau, Brucher, Perrot, and Duchesnay 2011). This observation has led many researchers to explore ways of combining the user-friendly simulation features of NetLogo with Python's computational power in other domains.

Several attempts to integrate Python and NetLogo have been explored mainly over the last decade. The most common method relies on client-server architectures, where Python controls certain predefined functions in NetLogo by sending commands and retrieving results via scripts (Thiele 2014, Jaxa-Rozen and Kwakkel 2018). While useful, this approach has a significant limitation: it does not allow direct control of agents or manipulation of their behavior from Python. This forces users to switch between the two programming environments, making the integration complex and less flexible.

Previous research to link NetLogo with other programming environments have focused on extending NetLogo's capabilities through external interfaces. For instance, Thiele et al. (Thiele 2014) developed RNetLogo, an interface between NetLogo and R, allowing for advanced statistical analysis of NetLogo models. Similarly, Jaxa-Rozen and Kwakkel (Jaxa-Rozen and Kwakkel 2018) introduced pyNetLogo, a connector that allows NetLogo to be controlled from Python, enabling users to perform simulations and retrieve results within a Python environment.

Py4J provides a bridge between Python and Java, enabling Python programs to dynamically access Java objects. This capability allows developers to directly control NetLogoa Java-based simulation environmentfrom Python. By using Py4J, users can seamlessly interact with NetLogo simulations at the agent level, efficiently managing models without requiring deep expertise in Java. However, despite its efficiency, Py4J may present challenges in more complex interactions and require an in-depth understanding of the underlying Java-Python bridge.

While these integration efforts have expanded the capabilities of NetLogo, significant gaps remain, particularly regarding the direct control of NetLogo agents from Python. Existing interfaces generally allow for running simulations and collecting results but lack the ability to manipulate individual agents or modify their behaviors during runtime directly from Python. This limitation hampers the ability to leverage Python's extensive libraries for machine learning, optimization, and data analysis, which could enhance agent behaviors and decision-making processes within simulations.

In this paper, we introduce **Netlogopy**, an innovative Python library that bridges the gap between NetLogo and Python, allowing for enhanced agent-based simulations by leveraging the strengths of both tools.

The remainder of this paper is structured as follows: The next section introduces the Netlogopy library, detailing its functionality and providing illustrative examples of its application. Section 3 evaluates the performance and feasibility of the library through a comparative analysis. Finally, Section 4 concludes the paper by summarizing the findings and discussing potential future directions for this work.

2 THE NETLOGOPY LIBRARY

In this section, we provide a detailed explanation of the Netlogopy library, including installation instructions and usage examples. The *Netlogopy* library is released as open-source software under the MIT License.[1] This license grants users the freedom to use, modify, and distribute the code under permissive terms. The source code is publicly available on PyPI[2] and GitHub[3]. We encourage the community to contribute to the development of `Netlogopy` by opening issues, submitting pull requests, and sharing feedback. For more information on best practices for open-source simulation code, please see the guidance provided in (Monks, Harper, and Mustafee 2024).

2.1 Installation

To use Netlogopy, it is recommended to create a dedicated Conda virtual environment to avoid conflicts with existing packages and ensure reproducibility. Netlogopy supports Python versions **3.6 and above**, ensuring compatibility with a wide range of Python environments. Follow the steps below:

1. **Install Conda**: If you do not already have Conda installed, download and install Anaconda or Miniconda.
2. **Clone the Repository**: Clone the Netlogopy repository from GitHub to access the `environment.yml` file and other resources.
   ```
   git clone https://github.com/yourusername/netlogopy.git
   cd netlogopy
   ```
3. **Create the Conda Environment**: Use the provided `environment.yml` file to create a new Conda environment:
   ```
   conda env create -f environment.yml
   ```
4. **Activate the Environment**:
   ```
   conda activate netlogopy-env
   ```

[1] `https://opensource.org/licenses/MIT`

[2] `https://pypi.org/project/netlogopy`

[3] `https://github.com/Bouaziz19/netlogopy`

5. **Install Netlogopy**:
   `pip install netlogopy`
6. **Install NetLogo**: Download and install NetLogo from the official site.
7. **Configure Your IDE**: If using Visual Studio Code or another IDE, set the default terminal to the Command Line Interface (CLI) for compatibility.

Note: An example `environment.yml` file is shown below:

```yaml
name: netlogopy-env
channels:
 - defaults
 - conda-forge
dependencies:
 - python=3.9
 - openjdk=11
 - pip
 - pip:
    - netlogopy
```

2.2 Features

Netlogopy provides several key features:

- **Direct Agent Manipulation**: Create and control NetLogo agents (turtles) directly from Python.
- **World Configuration**: Resize the NetLogo world, set backgrounds, and customize patches.
- **Integration with Python Libraries**: Use Python's scientific libraries for computation and analysis within the simulation.
- **Simplified Syntax**: Abstracts the complexity of NetLogos language, allowing users to focus on simulation logic.

2.3 Main Functionalities

Below are examples demonstrating how to use Netlogopy for agent-based simulations through its main functionalities.

2.3.1 Creating and Controlling Agents

```python
import time
from netlogopy.netlogopy import *

if __name__ == "__main__":
  # Specify the NetLogo installation path
  netlogo_home = "path/to/netlogo_home/NetLogo 6.2.2"
  n = run_netlogo(netlogo_home)

  # Resize the NetLogo world
  resize_world(n, 0, 60, 0, 60)

  # Create a turtle agent
  car01 = pyturtle(n, x=20, y=20, shape="car",
           size_shape=4, color=15,
           name="car01", labelcolor=15)

  # Simulation loop
  for i in range(10):
    time.sleep(1)
    print(f"Step {i}")
```

```
  n.close_model()
```

2.3.2 Moving Agents

```
# Move the agent forward by 1 unit each step
for i in range(10):
  time.sleep(1)
  car01.fd(1)
  print(f"Agent moved to position {getxyturtle(n, car01)}")
n.close_model()
```

2.3.3 Interacting Between Agents

```
# Create a second turtle agent
car02 = pyturtle(n, x=5, y=5, shape="car",
          size_shape=4, color=55,
          name="car02", labelcolor=55)

# Create a link (street) between agents
street(n, fromm=car01, too=car02, label="street",
    labelcolor=35, color=35, thickness=0.5)

# Move agent car01 towards car02
for i in range(10):
  time.sleep(1)
  car01.face_to(car02)
  car01.fd(1)
  distance = car01.distanceto(car02)
  print(f"Distance to car02: {distance}")

n.close_model()
```

2.3.4 Setting Background and Customization

```
# Set a background image
background_image = "path/to/image/nelogopy.png"
set_background(n, background_image)
```

2.3.5 Accessing Agent Properties

```
# Get the agent's position
position = getxyturtle(n, car01)
print(f"Agent position: {position}")

# Set the agent's position
car01.setxy(10, 10)
```

2.3.6 Additional Examples

Moving to Another Agent

```
# Move car01 to the position of car02
car01.move_to(car02)
```

Hiding and Showing Agents

```
# Hide the agent car01
car01.hideturtle()
```

Changing Agent Shape

```
# Change the shape of car01 to 'default'
car01.set_shape('default')
```

Calculating Distance Between Agents

```
# Calculate the distance between car01 and car02
distance = distancebetween(n, car01, car02)
print(f"Distance: {distance}")
```

2.3.7 Integrating an Existing NetLogo Model

Netlogopy allows you to layer Python-controlled agents on top of older NetLogo models:

```
import time, os, sys
from netlogopy.netlogopy import *

if __name__ == "__main__":
  netlogo_home = "C:/Program Files/NetLogo 6.2.2"
  path_model = os.path.join(os.path.dirname(__file__),
                "Wolf Sheep Predation.nlogo")
  n = run_netlogo(netlogo_home, path_model)

  # Resize world
  resize_world(n, 0, 70, 0, 55)

  # Initialize the original NetLogo model
  run_command(n, "setup")

  # Add Python-controlled agents
  car01 = pyturtle(n, x=20, y=20, shape="car",
           size_shape=4, color=15,
           name="car01", labelcolor=15)
  car02 = pyturtle(n, x=5, y=5, shape="car",
           size_shape=4, color=55,
           name="car02", labelcolor=55)
  street(n, fromm=car01, too=car02,
      label="street", labelcolor=35,
      color=35, shape="default", thickness=0.5)

  for i in range(100):
    run_command(n, "go")  # Run the NetLogo model step
    time.sleep(0.1)
    print(f"********** {i} ********")
    car01.fd(1)
    if i % 20 == 0:
      car01.setxy(10, 10)
```

2.4 Application Examples

Below are illustrative scenarios demonstrating how Netlogopys integration can enhance simulation capabilities:

Ecological Simulation In ecological modeling, agent-based simulations are used to study interactions between predator-prey species (Wilensky 2015). By integrating Pythons reinforcement learning libraries, predator agents can adapt hunting strategies in real-time (Gunaratne and Garibay 2020).

Industrial Production Systems Agent-based models in operations research help simulate manufacturing processes and supply chains (Bouaziz, Bettayeb, Sahnoun, and Yassine 2024, Siebers, Macal, Garnett, Buxton, and Pidd 2010, Sahnoun, Xu, Belgacem, Imen, David, and Louis 2019). Python-based optimization (e.g., genetic algorithms) can be combined with NetLogos agent structures (Fortin, De Rainville, Gardner, Parizeau, and Gagné 2012).

Robotics Simulation Combining NetLogos agent approach with Pythons ML libraries (e.g., *scikit-learn* (Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg, Vanderplas, Passos, Cournapeau, Brucher, Perrot, and Duchesnay 2011)) allows advanced behaviors for virtual robots (Sahnoun, Xu, Abdelaziz, and Baudry 2019).

3 EVALUATION AND COMPARISON WITH EXISTING LIBRARIES

Our work was inspired by existing libraries such as NL4Py, which parallelizes NetLogo simulations. Netlogopy builds on these concepts by adding richer agent-level commands and a more Python-focused workflow.

To evaluate the feasibility, we implemented a production-line simulation with six machines and six operators in both **Netlogopy** and **pure NetLogo** (same underlying logic). Development in Netlogopy took around 16 hours while building the same logic purely in NetLogo required around 25 hours. Further scenario additions also proved faster in Netlogopy (about one hour) versus NetLogo (around six hours). The more complex the model or the greater the number of inputs, the larger the speed and flexibility gap in favor of Netlogopy.

Table 1: List of Libraries Linking Python and NetLogo

Library	**URL**
pyNetLogo	`https://pynetlogo.readthedocs.io`
Py4J	`https://www.py4j.org`
NL4Py	`https://github.com/chathika/NL4Py`
Netlogopy	`https://pypi.org/project/netlogopy/`

3.1 Comparison of Features

- **pyNetLogo** (Jaxa-Rozen and Kwakkel 2018): Focuses on running NetLogo simulations from Python and retrieving results but lacks direct agent-level manipulation.
- **Py4J**: General-purpose Java-Python bridge; powerful but requires deeper knowledge of NetLogo internals.
- **NL4Py** (Gunaratne 2019): Allows parallelized NetLogo simulations but does not provide fine-grained agent control.
- **Netlogopy**: Direct agent creation/manipulation in Python, plus integrated simulation management.

	Netlogopy	pyNetLogo	Py4J	NL4Py
Language required	Python only	Python, NetLogo code	Python & Java	Python, NetLogo code
Integration w/ Python libs	✓	Limited	Limited	✓
Python-based sim management	✓	-	-	-
Direct agent manipulation	✓	-	-	-

Table 2: Feature Comparison of PythonNetLogo Libraries

4 CONCLUSION

Netlogopy represents a significant advancement in integrating Python and NetLogo for agent-based simulations. By allowing direct manipulation of NetLogo agents and leveraging Pythons extensive libraries, Netlogopy greatly enhances modeling flexibility, reduces development time, and enables more rapid iteration of sophisticated simulators. This seamless interaction between Python and NetLogo opens the door to more powerful and adaptable simulations across various fieldssuch as ecology, social sciences, industrial optimization, and complex systems analysis.

Furthermore, the accessibility of a Python-driven interface broadens the user base by lowering technical barriers. Researchers, educators, and students who may not have deep expertise in Java or NetLogo can now easily engage with advanced agent-based modeling, fostering interdisciplinary collaboration and the adoption of cutting-edge simulation methods.

Ensuring the long-term sustainability and maintainability of Netlogopy is a key priority. We have implemented strategies such as modular code design, comprehensive documentation, open-source collaboration, and continuous integration practices to maintain code quality and reliability. Looking ahead, future developments of Netlogopy could include support for more advanced modeling scenarios (3D simulations, parallel processing, GPU acceleration) and deeper integration with Python-based data analysis and machine learning tools. We also plan to cultivate a vibrant user community to exchange best practices, promote collaboration, and further expand Netlogopy's capabilities.

ACKNOWLEDGEMENT

Acknowledgment is made to the Normandy region and the European Union for supporting this research through the RIN regional research program by funding the AntiHpert project.

REFERENCES

Bouaziz, N., B. Bettayeb, M. Sahnoun, and A. Yassine. 2024. "Incorporating uncertain human behavior in production scheduling for enhanced productivity in Industry 5.0 context". *International Journal of Production Economics*:109311.

Fortin, F.-A., F.-M. De Rainville, M.-A. G. Gardner, M. Parizeau, and C. Gagné. 2012. "DEAP: Evolutionary algorithms made easy". *The Journal of Machine Learning Research* 13 (1): 2171–2175.

Gunaratne, C. 2019. "Evolutionary model discovery: Automating causal inference for generative models of human social behavior".

Gunaratne, C., and I. Garibay. 2020. "Evolutionary model discovery of causal factors behind the socio-agricultural behavior of the ancestral Pueblo". *Plos one* 15 (12): e0239922.

Jaxa-Rozen, M., and J. H. Kwakkel. 2018. "Pynetlogo: Linking netlogo with python". *Journal of Artificial Societies and Social Simulation* 21 (2): 28–31.

McKinney, W. et al. 2010. "Data structures for statistical computing in Python.". In *SciPy*, Volume 445, 51–56.

Monks, T., A. Harper, and N. Mustafee. 2024. "Towards sharing tools and artefacts for reusable simulations in healthcare". *Journal of Simulation*:1–20.

Oliphant, T. E. et al. 2006. *Guide to numpy*, Volume 1. Trelgol Publishing USA.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. "Scikit-learn: Machine learning in Python". *the Journal of machine Learning research* 12:2825–2830.

Sahnoun, M., Y. Xu, F. B. Abdelaziz, and D. Baudry. 2019. "Optimization of transportation collaborative robots fleet size in flexible manufacturing systems". In *2019 8th International Conference on Modeling Simulation and Applied Optimization (ICMSAO)*, 1–5. IEEE.

Sahnoun, M., Y. Xu, B. Belgacem, B. Imen, B. David, and A. Louis. 2019. "Fractal modeling of cyber physical production system using multi-agent systems". In *2019 International Conference on Applied Automation and Industrial Diagnostics (ICAAID)*, Volume 1, 1–6. IEEE.

Siebers, P.-O., C. M. Macal, J. Garnett, D. Buxton, and M. Pidd. 2010. "Discrete-event simulation is dead, long live agent-based simulation!". *Journal of Simulation* 4 (3): 204–210.

Thiele, J. C. 2014. "R marries NetLogo: introduction to the RNetLogo package". *Journal of Statistical Software* 58:1–41.

Wilensky, U. 2015. *An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with Netlogo*. The MIT Press.

AUTHOR BIOGRAPHIES

NOURDDINE BOUAZIZ is a PhD student at CESI LINEACT in Rouen, France. His research interests include human behavior, production scheduling, optimization, and simulation. He can be reached at nbouaziz@cesi.fr

BELGACEM BETTAYEB, Ph.D., MSc-MEng, is an Associate Professor of Industrial Engineering at CESI LINEACT in Lille, France. He obtained his M.Eng. in Industrial Systems Engineering and M.Sc. in Optimization and Systems Safety from the University of Technology of Troyes, France. He later earned his Ph.D. in Industrial Engineering from the University of Grenoble Alps, France. His research interests include Industry 4.0/5.0, Production Planning & Control, Quality Control, and Maintenance Planning. He can be reached at bbettayeb@cesi.fr

M'HAMMED SAHNOUN is a Research Director at CESI LINEACT, specializing in modeling, simulation, and optimization for manufacturing and renewable energy systems. He holds an HDR in Industrial Engineering (Normandy University), a Ph.D. in Automation, and master's degrees in Organization (University of Metz) and Robotics/Intelligent Systems (University of Paris 6), plus an Engineering degree in Automation (National Polytechnic School of Algiers). Formerly a Research Engineer at Grenoble Institute of Technology and Lecturer at the University of Lorraine, he has supervised several PhD theses on industrial data exploitation, complex system management, and dynamic scheduling. He also leads European and national R&D projects applying simulation and optimization across diverse domains. Email: msahnoun@cesi.fr

ADNAN YASSINE is a Full Professor at the Le Havre Normandy University in France and a member of the Laboratory of Applied Mathematics of Le Havre (LMAH). He teaches optimization, Operational Research, and logistics at the Superior Institute of Logistic Studies (ISEL). He is the author and co-author of numerous international publications in numerical and combinatorial optimization, among which some concerning scheduling and logistics problems. He obtained a Ph.D. in Applied Mathematics at the University of Grenoble (France) in 1989 and a Habilitation to Supervise Research (HDR) at the University of Nancy (France) in 1998. Email: adnan.yassine@univ-lehavre.fr