Soil: An Agent-Based Social Simulator in Python for Modelling and Simulation of Social Networks

Jesús M. Sánchez, Carlos A. Iglesias(✉), and J. Fernando Sánchez-Rada

Intelligent Systems Group, DIT, E.T.S. de Ingenieros de Telecomunicación, Universidad Politécnica de Madrid, 28040 Madrid, Spain
jesusmanuel.sanchez.martinez@alumnos.upm.es,
{cif,jfernando}@dit.upm.es
http://www.gsi.dit.upm.es

Abstract. Social networks have a great impact in our lives. While they started to improve and aid communication, nowadays they are used both in professional and personal spheres, and their popularity has made them attractive for developing a number of business models. Agent-based Social Simulation (ABSS) is one of the techniques that has been used for analysing and simulating social networks with the aim of understanding and even forecasting their dynamics. Nevertheless, most available ABSS platforms do not provide specific facilities for modelling, simulating and visualising social networks. This article aims at bridging this gap by introducing an ABSS platform specifically designed for modelling social networks. The main contributions of this paper are: (1) a review and characterisation of existing ABSS platforms; (2) the design of an ABSS platform for social network modelling and simulation; and (3) the development of a number of behaviour models for evaluating the platform for information, rumours and emotion propagation. Finally, the article is complemented by a free and open source simulator.

1 Introduction

Social Networks (SNs) have a great impact in our lives. While they started to improve and aid communication, nowadays they are used both in professional and personal spheres, affecting different aspects ranging economic [11] to health outcomes [22].

The emergence of social computing [45] has raised the interest in the design, analysis and forecasting of social systems. To this end, Social Computing is a cross-disciplinary field with theoretical underpinnings including both computational and social sciences, as well as research from areas such as social psychology, human computer interaction, Social Network Analysis (SNA), anthropology, sociology, organization theory, and computing theory.

One of the fields where ABSS has been applied is the analysis and simulation of social networks, in applications such as viral marketing [40], innovation diffusion [20], rumour propagation [23]. In fact, some authors [33] propose that the use of social media in agent based simulations can leverage the input data

Y. Demazeau et al. (Eds.): PAAMS 2017, LNAI 10349, pp. 234–245, 2017.
DOI: 10.1007/978-3-319-59930-4_19

problem in ABSS, since capturing data from individuals is an expensive and difficult task in longitudinal studies.

Nevertheless, there is a lack of ABSS platforms that provide support for social network modelling. Thus, we aim at bridging this gap by designing and developing an ABSS in Python specifically designed for social networks which benefits from the wide number of available Python libraries for network analysis and machine learning.

The remainder of the article is organised as follows. First, we review existing ABSS platforms to justify why they are not suitable for our problem in Sect. 2, as well as applications of ABSS to social network analysis. Based on this, we present a set of requirements for the desired platform in Sect. 3. Then, the proposed model, architecture and simulation workflow are presented in Sect. 4. The platform has been evaluated through the development of a library of models which is described in Sect. 5. We conclude with Sect. 6 and provide an outlook of future work.

2 Review of ABSS Platforms for Modelling SNs

In recent years numerous ABSS have been developed, as shown by Railsback et al. [34] and Nikolay et al. [31]. Based on this latter work that reviews 55 ABSS platforms, we have reviewed ABSS platforms to evaluate their suitability for modelling social networks, attending to the following aspects: (i) type of platform (general purpose or domain specific), (ii) programming language, (iii) expertise in its application to SNs, (iv) whether the framework provides SNA facilities and (v) whether the license is Open Source (OS). Table 1 summarizes the platforms and the reviewed aspects.

From the initial list provided in [31] we have filtered out platforms that are under a commercial license (e.g. cougaar), not actively developed (e.g. ABLE), focused on training (e.g. AgentSheets), or otherwise not directly focused on simulation (e.g. ECJ or Jade). The resulting set of platforms is Common-Pool Resources and Multi-Agent Systems (Cormas) [7], Madkit [14], Mason [24], NetLogo [35], Repast [32], SeSam [16] and Swarm [27]. Based on our literature research, we have added some additional platforms: UbikSim [9], EscapeSim [41], HashKat [38], Mesa [28], Krowdix [6] and Multi-Agent Scalable Runtime platform for Simulation (MASeRaTi) [2].

Cormas [7] is a general ABSS platform dedicated to natural and common resource management. There is a work [36] that models a social network of innovation diffusion in the medical domain. Madkit [14] is a general multiagent platform which relies on organization concepts and includes simulation facilities. Kodia et al. [21] describe a model where investors are tied by relationships such as friendship, trust and privacy. Mason [24] is a popular multiagent simulation. There is an extension *socialnets* that provides simple network statistics and a bridge to the Java Graph library Jung.[1] Some authors [40] have used Mason for

[1] http://jung.sourceforge.net/.

Table 1. Review of ABSS platforms

Name	Domain	Language	SNs	SNA	OS
Cormas	Generic	VisualWorks	✓	✗	✓
NetLogo	Generic	NetLogo, Scala & Java	✓	✗	✓
Swarm	Generic	Objective-C, Java	✓	✗	✓
MadKit	Generic	Java	✓	✗	✓
MASON	Generic	Java	✓	✗	✓
Repast	Generic	Java	✓	✓	✓
SeSam	Generic	Java	✗	✗	✓
MASeRaTi	Generic	Java	✗	✗	✓
Mesa	Generic	Python	✗	✗	✓
UbikSim	AmI	Java	✗	✗	✓
EscapeSim	Evacuation	Java	✗	✗	✓
HashKat	Social networks	C++	✓	✓	✓
Krowdix	Social networks	Java	✓	✓	✗
Soil	Social networks	Python	✓	✓	✓

modelling viral marketing in Twitter. To this end, authors usually complement Mason with other libraries and tools e.g. GraphStream[2] for synthetic network generation and dynamic network visualisation, iGraph[3] for centrality and network measures, and Gephi[4] for detailed analysis of the network. NetLogo [35] is a multiagent programming and simulation environment. It includes facilities for network representations although not for network analysis. An outdated extension to NetLogo is described in [5], where the network analysis and visualisation tool Pajek[5] is integrated. In addition, there are some available models of social networks (e.g. Social circles [15]), but they do not provide facilities for analysing or building new models. Repast [32] is an agent based simulation platform that provides a large library of simulation models. Repast has been extended for SNA [19]. The library Repast Social Network Analysis (ReSoNetA) adds network functionality to RepastJ. It provides a number of network metrics (centrality, prestige and authority) based on the graph Java library Jung as well as visualisation facilities. This library exploits Repast's built-in facilities for network modelling. In addition, other works such as van Maanen [25] have used Repast for modelling social influence in Twitter. SeSam [16] provides a generic environment for agent based simulations but it has not been applied for social network modelling. Swarm [27] is a well known agent-based simulator that has been applied to social network problems such as open source project dynamics [27].

[2] http://graphstream-project.org/.
[3] http://igraph.org/.
[4] https://gephi.org/.
[5] http://mrvar.fdv.uni-lj.si/pajek/.

While the previous ABSS platforms were designed for its application to a wide variety of domains, other platforms, such as UbikSim [9] and EscapeSim [41] has been specifically designed for a particular domain, such as Ambient Intelligence (AmI) and evacuation.

HashKat [38] is a C++ ABSS platform specifically designed for the study and simulation of social networks. It includes facilities for network growth and information diffusion, based on a kinetic Monte Carlo model. It exports information to be processed by machine learning libraries such as NetworkX[6] or R's iGraph and network visualisation with Gephi.

Mesa [28] is an ABSS platform that aims at providing a Python alternative to traditional Netlogo, MASON or Repast. It enables in-browser visualisation and takes advantage of Python ecosystem. Krowdix [6] is a Java ABSS for social networks but it is not open source. It uses JUNG for network functions and JFreeChart[7] for visualisation. The simulation model considers users, their relationships, user groups and interchanged contents. It has been applied to Twitter and Facebook. MASeRaTi [2] is a distributed and scalable ABSS that uses the Belief-Desire-Intention (BDI) framework lightjason [3], that extends the agent-oriented programming language AgentSpeak.

To summarise, except for HashKat and Krodix, ABSS platforms do not provide support for the analysis of social networks, although some platforms have already been used for this purpose. Moreover, most ABSS platforms are programmed in Java. MASeRaTi follows a different approach where agents can be programmed based on a BDI model. Main challenges for applying existing platforms to social networks come from their underlying models, frequently tied to spatial models.

3 ABSS Requirements for Social Networks

Based on the previously presented review of ABSS platforms and their application to SN analysis and simulation, we have identified the requirements listed below, which are structured in network and agent model.

Network model. The network level groups all the functionalities related to the structural aspects of the social network. The following requirements have been identified:

– *Generation of synthetic graphs.* Even though accessing real social network graphs is critical, real datasets have a number of disadvantages [39]. First, sharing large social graphs is challenging, since they should be anonymised and there are limitations in the way they can be shared (for example, only tweet ids can be shared in Twitter, which requires collecting the dataset with API restrictions and difficulties in reproducing the original dataset since some tweets could be no longer available). Second, the availability of a small number of social graphs can limit the statistical confidence in the experimentation

[6] https://networkx.github.io/.
[7] http://www.jfree.org/jfreechart/.

results. Finally, obtaining real datasets suitable for the desired experimentation can be difficult and require a great effort. Thus, synthetic graph generated by measurement-calibrated graph models [39] so that graph models are fitted to a real social graph, and the simulation are realistic. The platform should provide implementation of classical social graph models [39] (e.g. Barabasi-Albert model [4], Random Walk [44], etc.) and should be extensible to innovative models.
- *Graph traversing and visualisation.* The platform should provide functionalities for traversing social graphs and visualising social structure, in order to be applied to diffusion models [13].
- *SNA functionalities.* Several functionalities should be available for the analysis of the social graph, such as calculation of social metrics (e.g. centrality, betweenness, etc.) as well as algorithms for community detection.
- *Export and import of network model.* There should be facilities for importing and exporting social graphs, based on popular formats such as Graph Modelling Language (GML) [18], GraphML [8] and Graph Exchange XML Format (GEXF) [12].

Agent model. The agent level models the agent characteristics, their state, how agent state evolves in every simulation step. Following the modelling steps proposed by Macal and North [26], we outline the requirements for social network modelling. Platform should allow users to: (i) define agent type definition and attributes (e.g. sentiment, frequency of tweeting, number of followers, etc.); (ii) define interactions with the environment, that represent external factors to agent decision, such as news or market evolution; and (iii) specify methods to update agent state based on their interaction with other agents and the environment. This include the capability to update the agent social network (i.e. creation or modification of social links).

Non functional requirements. Regarding non functional requirements, several aspects have been considered. First of all, the *programming language* is an important decision. In order to provide a homogeneous programming environment, network and machine learning libraries should be available. Both Java and Python fulfill these requirements, as we have introduced previously. As previously outlined, it is very important that ABSS provide *interactive experimentation facilities* that enable researchers to run and define their experiments. In this regard, most platforms ABSS platforms such as Mason or Repast provide configurable and extensible *configuration facilities* [43]. *Scalability* has been recently addressed by a number of researchers [1,2]. The ability to distribute agents across machines or big data processing infrastructures can be required for the simulation of large scale social networks. Finally, *extensibility and reusability* of simulation models should be encouraged [37], so that researchers can benefit from a library of tested simulation models that can be used, extended and adapted to model new behaviours.

4 Soil Platform

4.1 Design Decisions

The first design decision is the selection of Python [30], given its increased popularity, its very gradual learning curve, readability, clear syntax and availability of libraries for network processing and machine learning. In addition, we consider the interactive analysis of the IPython interface[8] very beneficial for simulation. From the reviewed platforms, only one platform is available in Python, Mesa, but it does not provide network facilities yet and is still in constant evolution. Hence, we evaluated different options to extend Mesa for this scenario. Another alternative was to extend nxsim[9], a Python library that provides a basic ABSS framework, based on Simpy [29]. We eventually chose nxsim due to its simplicity and robustness.

Regarding the network model, we have opted for NetworkX, which is the de-facto standard library for SNA analysis of small to medium networks. For massive networks, the transition to NetworkKit [42] is straight forward. NetworkX provides functionalities for manipulating and representing graphs, generators of classical and popular graph models, and graph algorithms for analysing graph properties. In addition, NetworkX is interoperable with a great number of graph formats, including GML, GraphML JSON and GEXF.

For network visualization, we have selected Gephi, an open-sourced software for network and graph interactive analysis. Gephi is able to render in 3D and real-time large and complex networks. In addition, both NetworkX and Gephi support the format GraphML, so a graph generated with NetworkX can be explored with Gephi in every simulation step. Finally, configurability will be achieved with configuration files.

4.2 Simulation Model for Social Networks

We propose a simulation model of SNs consisting of users represented by agents and a network that represents the social links between users. Agent are characterised by their state (e.g. infected) and the behaviours they can carry out in every simulation step, usually depending on the user state. Each behaviour defines the actions carried out (e.g. tweeting, following a user, etc.) and how the agent state evolves, depending on external factors (e.g. news about a topic) or social factors (e.g. opinion of their friends). Probabilities defined in the configuration control the frequency of actions in every behaviour.

This simulation model has been implemented in the architecture shown in Fig. 1 and consists of four main components.

The *NetworkSimulation* class is in charge of the network simulator engine. It provides forward-time simulation of events in a network based on nxsim and Simpy. Based on configuration parameters, a graph is generated with NetworkX

[8] https://ipython.org/.
[9] https://pypi.python.org/pypi/nxsim.

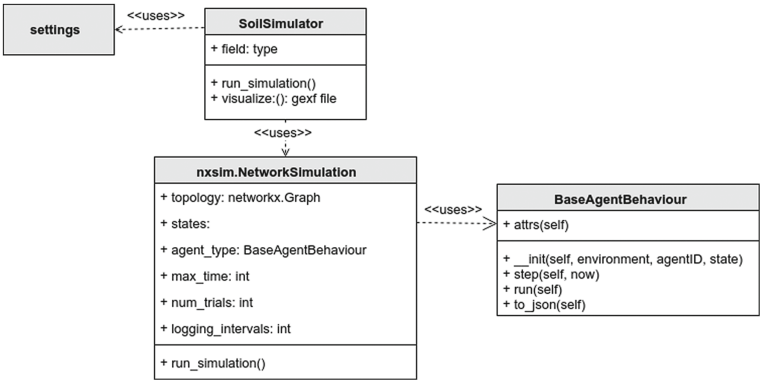

Fig. 1. Simulation components

and an agent class is populated to each network node. The main parameters are the network type, number of nodes, maximum simulation time, number of simulations and timeout between each simulation step.

The *BaseAgentBehaviour* class is the basic agent behaviour that should be extended for each social network simulation model. It provides a basic functionality for generation of a JSON file with the status of the agents for its analysis with machine libraries such as Scikit-Learn.

The *SoilSimulator* class is in charge of running the simulation pipeline defined in Sect. 4.3, which consists in running the simulation and generating a visualisation file in GEXF which can be visualised with Gephi. In addition, interactive analysis can be done through IPython notebooks.

Settings groups the general settings for simulations and the settings of the different models available in Soil's simulation model library.

4.3 Simulation Workflow

An overview of the system's flow is shown in Fig. 2. The simulation workflow consists of three steps: configuration, simulation and visualization.

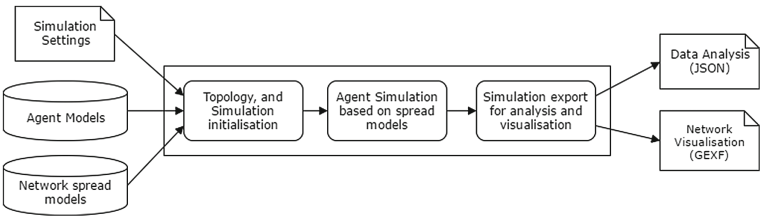

Fig. 2. Social simulator's workflow

In the first step, the main parameters of the simulation are configured in the *settings.py* file. The main parameters are: network graph type, number of agents, agent type, maximum time of simulation and time step length. In addition, the parameters of the behaviour model should be configured (e.g. initial states or probability of an agent action). Agent behaviours should be selected from the provided library or developed extending the *BaseAgentBehaviour* class.

Once the simulation is configured, the next step is the simulation, that can be done step by step or a number of steps. The class *BaseAgentBehaviour* stores the status of every agent in every simulation step into a JSON file to be exported once the simulation is finished. This allows us to automatise the process of generating the .gexf file.

Finally, users can carry out further analysis with the JSON file as well as visualize the evolution the simulation with the generated .gexf file with the tool Gephi, as shown in Fig. 5.

5 Test Cases

We have evaluated Soil in the development of a number of simulation models. In these experiments, we have used the Barabasi-Albert network generation model [4].

The models included in the library deal with viral marketing in Twitter [40], infection (SISa [17]), sentiment correlation the social network Weibo [10], Bass model [35] and Independent Correlation Model [35] of information diffusion in social networks.

In order to illustrate the functionalities of Soil, we review the Viral Marketing model [40], which is based on rumour propagation models. In it, agents have four potential states: neutral, infected, vaccinated and cured. This model includes the fact that infected users who made a mistake believing in the rumour will not be

```
class ControlModelM2(BaseBehaviour):
# init states
def step(self, now):
if self.state['id'] == 0:  #Neutral
self.neutral_behaviour()
elif self.state['id'] == 1:  #Infected
self.infected_behaviour()
...
def infected_behaviour(self):
# Infected
neutral_neighbors = self.get_neighboring_agents(state_id=0)
for neighbor in neutral_neighbors:
if random.random() < self.prob_infect:
neighbor.state['id'] = 1  # Infected
```

Fig. 3. Code snippet of an infected behaviour

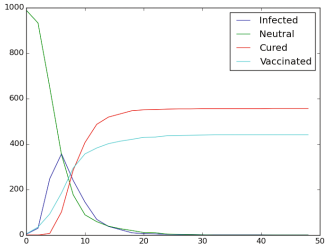

Fig. 4. Agent evolution

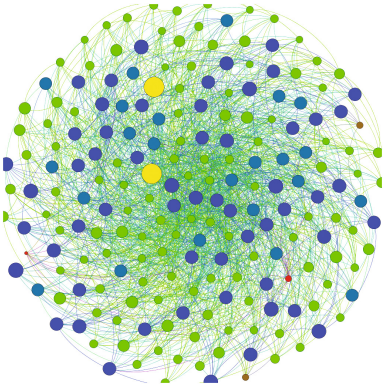

Fig. 5. Network visualization

in favour of spreading their mistakes through the network. A example of how behaviours are programmed is shown in Fig. 3. This behaviour shows that an infected agent first selects its neutral neighbours and infects them with a given probability. Figures 4 and 5 show the evolution of agent states and network visualisation, respectively.

6 Conclusions and Outlook

While generic ABSS provide a suitable framework, we think that further research on ABSS platforms for specific domains is needed. In this paper we have reviewed the existing frameworks and the requirements for modelling and simulation of social networks.

Soil is a modern ABSS for social networks developed in Python that benefits from the Python ecosystem. It has been applied to a number of social network simulation models, ranging from rumour propagation to emotion propagation and information diffusion. Additionally, it is fully open source, cross-platform and produces outputs compatible with SNA packages and network visualisation tools. The platform has been designed for research purposes, and has focused on simplicity of developing new simulation models. Soil allows the generation of dynamic networks and its animation thanks to the use of Gephi. In spite of the growing development of the Python ecosystem, there are still some functionalities, such as Exponential Random Graph Model (ERGMs) which are better supported in other environments such as R with the statnet[10] package, which provides a wide range of functionality for the statistical analysis of social networks. In particular, these models are very interesting for fitting models given a network data set. As future work, we aim at evaluating and integrating implementations such as ergm[11].

[10] http://statnetproject.org.
[11] https://github.com/jcatw/ergm.

Lastly, Soil is work in progress. We aim at improving the experimentation and visualisation facilities provided by the platform, and improve the platform through its application in more use cases and through the collaboration with other research groups.

Acknowledgements. This work is supported by the Spanish Ministry of Economy and Competitiveness under the R&D projects SEMOLA (TEC2015-68284-R) and EmoSpaces (RTC-2016-5053-7), by the Regional Government of Madrid through the project MOSI-AGIL-CM (grant P2013/ICE-3019, co-funded by EU Structural Funds FSE and FEDER), and by the European Union through the project MixedEmotions (Grant Agreement no: 141111).

References

1. Aaby, B.G., et al.: Efficient simulation of agent-based models on multi-GPU and multi-core clusters. In: Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, SIMUTools 2010, Torremolinos, Malaga, Spain. ICST (2010)
2. Ahlbrecht, T., Dix, J., Köster, M., Kraus, P., Müller, J.P.: A scalable runtime platform for multiagent-based simulation. In: Dalpiaz, F., Dix, J., Riemsdijk, M.B. (eds.) EMAS 2014. LNCS, vol. 8758, pp. 81–102. Springer, Cham (2014). doi:10.1007/978-3-319-14484-9_5
3. Aschermann, M., et al.: LightJason: a BDI framework inspired by Jason. Technical report IfI-16-04, Depart. Department of Computer Science, TU Clausthal, Germany (2014)
4. Barabáasi, A.-L., Albert, R.: Emergence of scaling in random networks. Science **286**(5439), 509–512 (1999)
5. Berryman, M.J., Angus, S.D.: Tutorials on agent-based modelling with NetLogo and network analysis with Pajek. In: Complex Physical, Biophysical and Econophysical Systems, vol. 1. World Scientific, Hackensack (2010)
6. Blanco-Moreno, D., Cárdenas, M., Fuentes-Fernández, R., Pavón, J.: Krowdix: agent-based simulation of online social networks. In: Bazzan, A.L.C., Pichara, K. (eds.) IBERAMIA 2014. LNCS, vol. 8864, pp. 587–598. Springer, Cham (2014). doi:10.1007/978-3-319-12027-0_47
7. Bommel, P., et al.: Cormas, an agent-based simulation platform for coupling human decisions with computerized dynamics. In: ISAGA 2015: Hybrid Simulation and Gaming in the Network Society (2015)
8. Brandes, U., et al.: Graph markup language (GraphML). In: Handbook of Graph Drawing and Visualization 20007 (2013)
9. Campuzano, F., Garcia-Valverde, T., Garcia-Sola, A., Botia, J.A.: Flexible simulation of ubiquitous computing environments. In: Novais, P., Preuveneers, D., Corchado, J.M. (eds.) Ambient Intelligence - Software and Applications. AINSC, vol. 92, pp. 189–196. Springer, Heidelberg (2011)
10. Fan, R., et al.: Anger is more influential than joy: sentiment correlation in Weibo. In: CoRR abs/1309.2402 (2013)
11. Granovetter, M.: The impact of social structure on economic outcomes. J. Econ. Perspect. **19**(1), 33–50 (2005)
12. Group, G.W.: GEXF file format. GEXF Working Group (2009)

13. Guille, A., et al.: Information diffusion in online social networks: a survey. ACM SIGMOD Rec. **42**(2), 17–28 (2013)
14. Gutknecht, O., Ferber, J.: The MadKit agent platform architecture. In: Wagner, T., Rana, O.F. (eds.) AGENTS 2000. LNCS, vol. 1887, pp. 48–55. Springer, Heidelberg (2001). doi:10.1007/3-540-47772-1_5
15. Hamill, L., Gilbert, N.: Social circles: a simple structure for agent-based social network models. J. Artif. Soc. Soc. Simul. **12**(2), 3 (2009)
16. Herrler, R., Fehler, M.: SeSAm: implementation of agent based simulation using visual programming. In: Components (2006)
17. Hill, A.L., et al.: Emotions as infectious diseases in a large social network: the SISa model. Proc. Roy. Soc. Lond. B Biol. Sci. **277**(1701), 3827–3835 (2010)
18. Himsolt, M.: GML: a portable graph file format. Technical report. Universität Passau (1997)
19. Holzhauer, S.: Developing a Social Network Analysis and Visualization Module for Repast Models, vol. 4. Kassel University Press GmbH, Kassel (2010)
20. Kiesling, E., et al.: Agent-based simulation of innovation diffusion: a review. CEJOR **20**(2), 183–230 (2012)
21. Kodia, Z., Said, L.B., Ghedira, K.: Stylized facts study through a multi-agent based simulation of an artificial stock market. In: Li Calzi, M., Milone, L., Pellizzari, P. (eds.) Progress in Artificial Economics. Lecture Notes in Economics and Mathematical Systems, vol. 645, pp. 27–38. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13947-5_3
22. Korda, H., Itani, Z.: Harnessing social media for health promotion and behavior change. Health Promot. Pract. **14**(1), 15–23 (2013)
23. Liu, D., Chen, X.: Rumor propagation in online social networks like twitter - a simulation study. In: 2011 Third International Conference on Multimedia Information Networking and Security, November 2011
24. Luke, S.: MASON: a multiagent simulation environment. Simulation **81**, 517–527 (2005)
25. van Maanen, P.-P., van der Vecht, B.: An agent-based approach to modeling online social influence. In: Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining. ACM (2013)
26. Macal, C.M., North, M.J.: Tutorial on agent-based modeling and simulation. In: Simulation Conference, 2005 Proceedings of the Winter. IEEE (2005)
27. Madey, G., et al.: Agent-based modeling of open source using Swarm. In: AMCIS 2002 Proceedings (2002)
28. Masad, D., Kazil, J.: MESA: an agent-based modeling framework. In: Proceedings of the 14th Python in Science Conference (SCIPY 2015) (2015)
29. Matloff, N.: Introduction to discrete-event simulation and the Simpy language. In: Davis, CA. Dept of Computer Science. University of California at Davis. Retrieved 2 Aug 2008
30. McKinney, W.: Python for Data Analysis. O'Reilly, Sebastopol (2012)
31. Nikolai, C., Madey, G.: Tools of the trade: a survey of various agend based modeling platforms. J. Artif. Soc. Soc. Simul. **12**(2), 2 (2009)
32. Ozik, J., Collier, N., Combs, T., Macal, C.M., North, M.: Repast simphony statecharts. J. Artif. Soc. Soc. Simul. **18**(3), 11 (2015). http://jasss.soc.surrey.ac.uk/18/3/11.html
33. Padilla, J.J., et al.: Leveraging social media data in agentbased simulations. In: Proceedings of the 2014 Annual Simulation Symposium. Society for Computer Simulation International (2014)

34. Railsback, S.F., et al.: Agent-based simulation platforms: review and development recommendations. Simulation **82**(9), 609–623 (2006)
35. Rand, W., Wilensky, U.: An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with NetLogo. MIT Press, Cambridge (2015)
36. Ratna, N.N., et al.: Diffusion and social networks: revisiting medical innovation with agents. In: Qudrat-Ullah, H., Spector, J.M., Davidsen, P.I. (eds.) Complex Decision Making, pp. 247–265. Springer, Heidelberg (2008)
37. Robinson, S., et al.: Simulation model reuse: definitions, benefits and obstacles. Simul. Model. Pract. Theory **12**(7), 479–494 (2004)
38. Ryczko, K., et al.: Hashkat: large-scale simulations of online social networks. In: arXiv preprint arXiv (2016)
39. Sala, A., et al.: Measurement-calibrated graph models for social network experiments. In: Proceedings of the 19th International Conference on World Wide Web. ACM (2010)
40. Serrano, E., Iglesias, C.A.: Validating viral marketing strategies in Twitter via agent-based social simulation. Expert Syst. Appl. **50**(1), 140–150 (2016)
41. Serrano, E., et al.: Towards a holistic framework for the evaluation of emergency plans in indoor environments. Sensors **14**(3), 4513–4535 (2014)
42. Staudt, C., et al.: NetworKit: an interactive tool suite for high-performance network analysis. In: CoRR abs/1403.3005 (2014)
43. Szufel, P., et al.: Controlling simulation experiment design for agent-based models using tree representation of parameter space. Found. Comput. Decis. Sci. **38**(4), 277–298 (2013)
44. Vázquez, A.: Growing network with local rules: preferential attachment, clustering hierarchy, and degree correlations. Phys. Rev. E **67**(5), 056104 (2003)
45. Wang, F.Y., et al.: Social computing: from social informatics to social intelligence, March 2007