Spatial Dynamics with Cellular Automata Modelling

Eike Blomeier[1]

[1]Z_GIS, University of Salzburg, Austria: eike.blomeier@stud.sbg.ac.at

Abstract

Agent-based modeling becomes more and more popular across all different kinds of scientific fields. It is used to model real-world phenomena like the spread of viruses or the urban sprawl as well as highly theoretical models like the Game of Life. While some modelling software requires advanced programming skills, others are really comfortable to use and learn. The focus is on this modelling software. Therefore, NetLogo and GAMA are introduced and analyzed for their strengths weaknesses especially in the GIS domain. Eventually it will be concluded that GAMA has more strengths in this field while NetLogo is richer in available plugins.

Content

Figures

Introduction

Nowadays, an increasing world population leading to growing settlement areas, climate change and corresponding disaster management, the spread of viruses like SARS-CoV-2, or the increasing human expanding in the world's last unpopulated areas like the rainforests in Borneo or the Amazonian rainforest makes it more important than ever to discuss impacts proceedings before an irreversible damage is done. This raises the interest in answering key policy questions of the underlying processes and understanding for human behaviour (Wu & Hobbs, 2002) and the corresponding feedbacks between humans and natural systems ((Pickett et al., 2001), (Levin, 2006), (Liu et al., 2007), (Turner et al., 2007), (Grimm et al., 2008)).

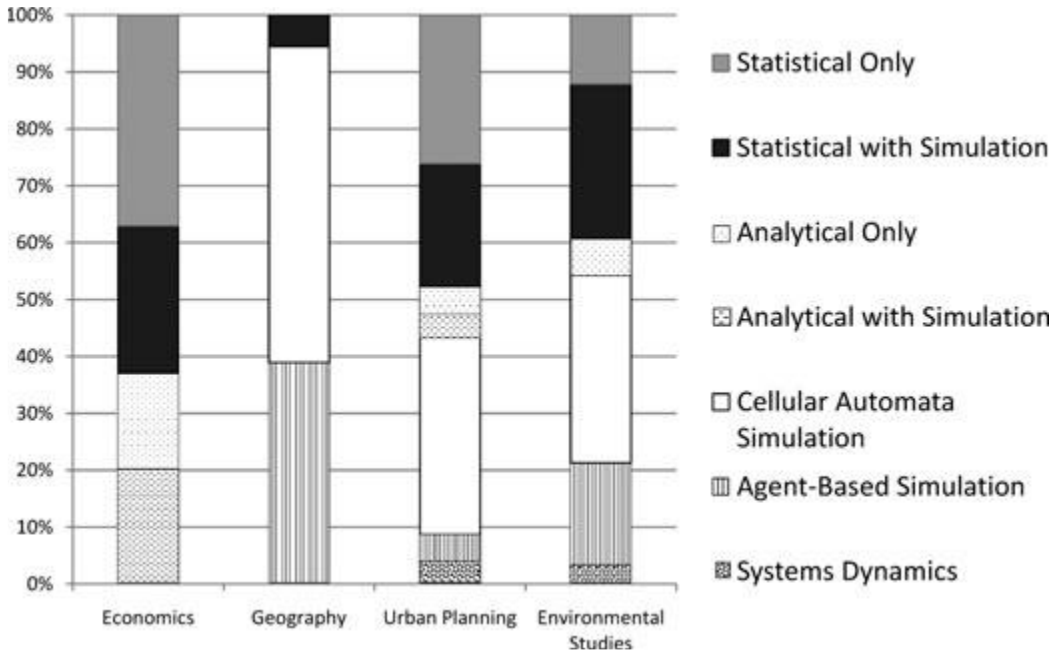

Figure 1 Relative proportion of modeling methods in different scientific disciplines (Irwin, 2010)

But the interest in systems to answer those questions is not a new one. Already in 1988, Lucas pointed out that in many contexts' agglomeration forces are the source of the increasing returns that lead to growth. And in 1991, Krugman identified the most challenging problem: The difficulty of developing a common framework that incorporates both, the spatial and the temporal dimensions. What he was basically asking for was the need of a dynamic spatial theory (Desmet & Rossi-Hansberg, 2010). Spatially explicit models (SEMs) are tackling this need. Tong defined SEMs as *"location-based computational approaches that can reproduce the dynamics of geographical phenomena"*. To get a better overview, SEMs can be categorized into two categories. On the one side,

platforms and toolboxes like Repast, SWARM, JADE or CORMAS are allowing to define models using a high-level generic programming language (C++, Java, Python…). These platforms are taking the advantage of the many available external libraries and the integration of numerous data and rich agent behaviours (Taillandier, Gaudou, et al., 2019). But strong computer science skills are required, and field scientists/experts must rely on computer scientists to develop models ((Drogoul et al., 2013b), (Taillandier, Gaudou, et al., 2019)). On the other side, platforms have been developed using a specific modelling language to tackle that problem. NetLogo and GAMA are belonging to this category. Even though, they are simpler to use than platforms from the first category, they still require some basic skills in algorithmic (Taillandier, Gaudou, et al., 2019).

In this paper, it tried to be answered how useful SEMs are to answer questions in the field of spatial dynamics. As a sub question it will be answered which modelling framework tackles the needs of location-based simulations better. In order two answer those questions, the paper is threefold. First, spatial dynamics will be briefly explained. Second a brief introduction on Cellular Automata will be given. Third the NetLogo and GAMA will be introduced and analysed regarding their GIS and large dataset capabilities.

Spatial Dynamics

Modeling dynamic processes, including many different stakeholders and interrelations between these stakeholders can already be very difficult. Incorporating space into dynamic frameworks makes this not easier. Actually, by integrating space into such a framework, the dimensionality of the problem increases (Desmet & Rossi-Hansberg, 2010). But for some processes, the spatial dependence of the process may be a not neglectable factor. Therefore *"Spatial dynamics pertains to a dynamic process that is spatially dependent"* (Irwin, 2010). In other words, spatial dynamics are describing a spatially dynamic process. In such a process, the change over time at one location is dependent and/or influences the state or changes in the state at other location (Irwin, 2010). This spatial interdependence may emerge, e.g., from local interaction among spatially distributed agents or feedbacks, caused by the decisions of many individual agents over space and time. However, the matter that a process is dynamic and spatially heterogeneous need not lead to be automatically a

spatial dynamic process (Pickett et al., 2001). For example, land development as a function of spatially heterogeneous and exogeneous soils is a process that generated a spatial pattern, but shortfalls spatial dynamics. This process in independent of the land use at other locations. But if the land development is a response to local land use spillovers, e.g. increasing population, it is a spatial dynamic process arising from the internal interactions of neighboring agents (Irwin, 2010).

Cellular Automata

Cellular automata (CA) are mostly used today as simulation frameworks to model how spatially extended phenomena, emerging from local processes and depending on the spatial arrangement of the environment, evolve over an assured period of time (Wallentin, 2019). With CA, Tobler's First Law of geography – *"Everything is related to everything else, but near things are more related than distant things"* (Tobler, 1970), is implemented into simulation modelling. Even though, CA are used today for simulation modelling, the idea behind CA was already proposed by Johann von Neumann in his paper "*The general and logical theory of automata"* (Neumann, 1963) and implemented by John Conway as the *Game of Life* in 1970 (Gardner, 1970).

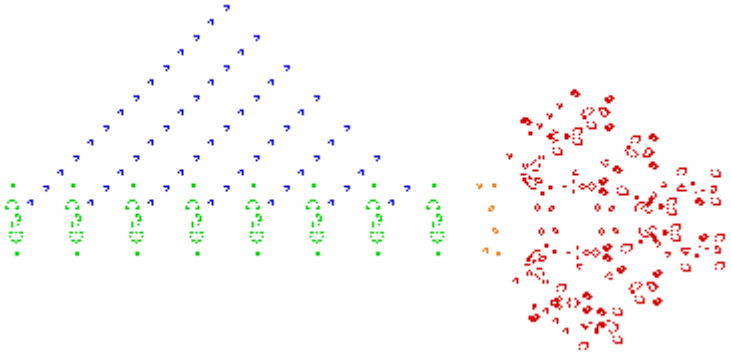

Figure 2 Screenshot of a puffer type breeder (red), that leaves glider guns (green) in its wake, which in turn create gliders (blue)[1]

With the increasing computational power in the past decades (Ravì et al., 2017), CA became more and more popular to solve theoretical and applied problems of

1 https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life#/media/File:Conways_game_of_life_breeder.png

spatial processes (Wallentin, 2019). At the same time, CA are still based on the same concepts invented by Johann von Neumann:

1. Cells
2. Neighbourhood
3. Transition rules

At every time step of the CA simulation, the neighbourhood and transition rules are applied to each cell simultaneously to compute the cells state in the next time step.

Cells

Like in nature, where cells can be of every size and shape (reference), cells in the grid of a CA can be of various shapes and sizes as well. While regular grids are the most typical (Wallentin, 2019),

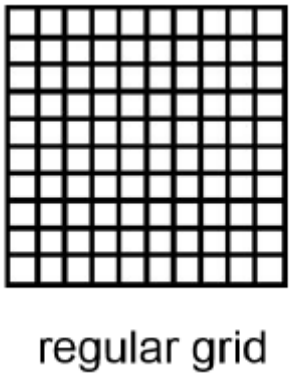

Figure 3 Regular Grid[2]

all different kind of cell shapes in the CAs grid are possible. From regular hexagonal grids to irregular Voronoi polygons or three-dimensional grids are possible, to be a best match for the applied use-case.

[2] UNIGIS

Figure 4 Hexagonal, Voronoi and 3D grids[3]

Continuatively, the type of the cell state must be defined (Figure 5). While for some scenarios, e.g. a simple tree cover model a binary cell type might be enough (tree or no tree in the cell), a wildfire model already needs more categories and therefore a categorical cell type. In addition, models that are more complex already need graduated cells or even a mixture of graduated and categorical cell types.

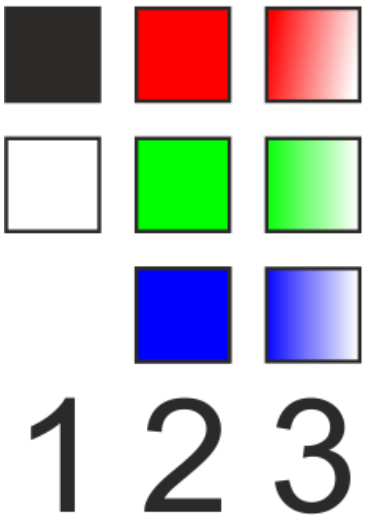

Figure 5 Cell types – (1) binary, (2) categorical, (3) graduated

Neighbourhood

Like in chess, where the rules of the game are defining the ability to move for every chess piece and therefore states the fields a chess piece can reach from its current position. Similar to the chess game, the neighbourhood in a CA can be of any kind but must be composed of type, distance and direction. While the major neighbourhood types are the *von Neumann neighbourhood* and the *Moore neighbourhood* (Figure 6).

[3] UNIGIS

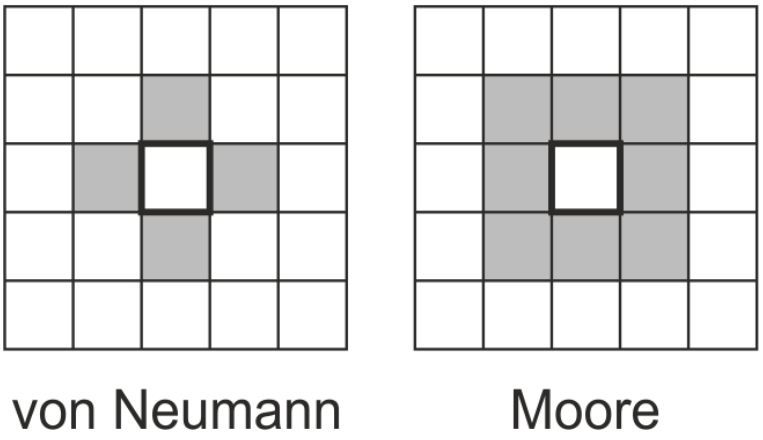

Figure 6 von Neumann neighbourhood (left) and Moore neighbourhood (right)

The distance of the neighbourhood (Figure 7) is only limited by the size of the grid of the CA itself.

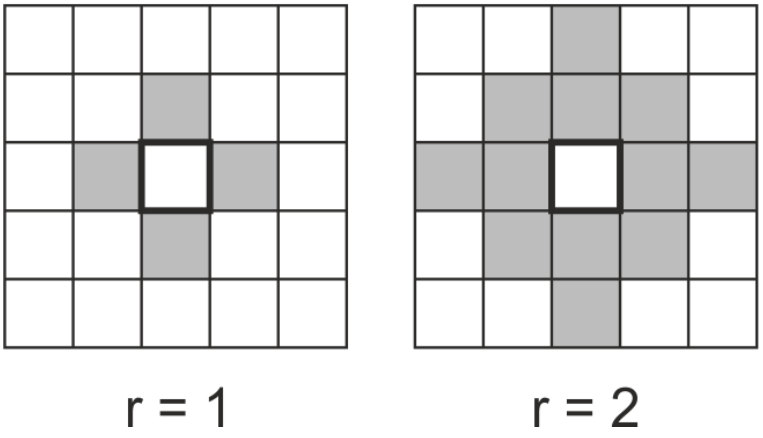

Figure 7 Neighbourhood distance 1 cell (left) and 2 cells (right)

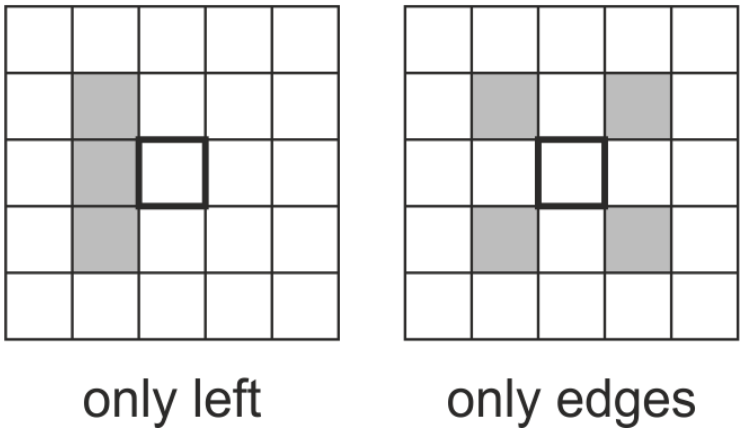

Figure 8 Neighbourhood direction

Additionally, to overcome artefacts rooted from the geometric composition of the CAs underlying grid, the neighbourhood can be of a stochastic nature.

Transition rules

The transition rules of a CA can also be referred as the *process engines* of the CA. Transition rules applied to all cells in the grid simultaneously and specify the way cells are changing from one time step to another.

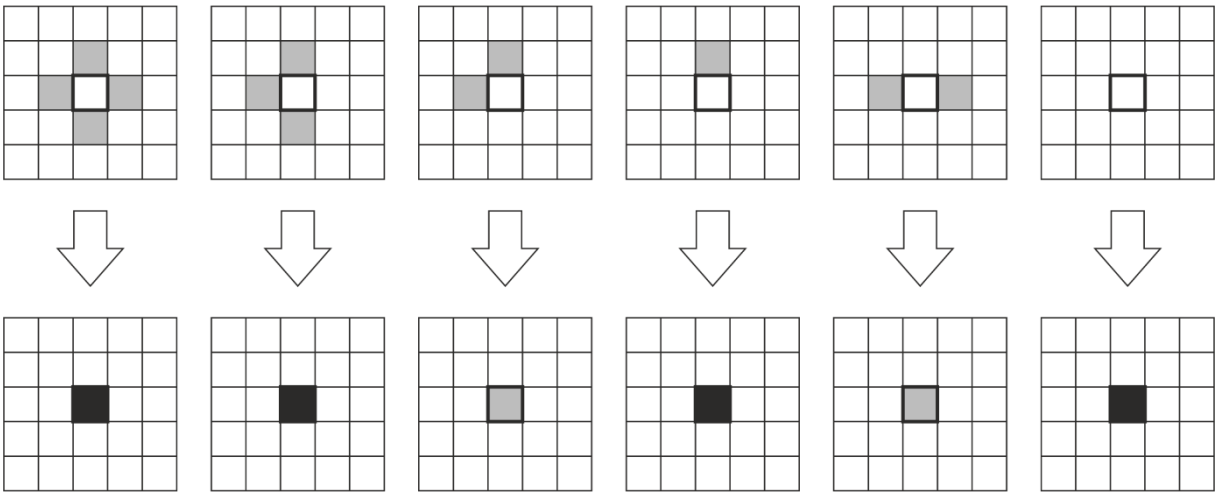

Figure 9 If-else transition rules based on the Game of Life

If-else statements are the easiest transition rule and mostly used for abstract and theoretical models (Figure 9). Applying deterministic transition rules to the same input pattern will always result in the same output pattern. By adding randomness to the transition rules, the outcome will be slightly different for each run. This makes a CA more life-like and realistic to how nature behaves (Figure 10).

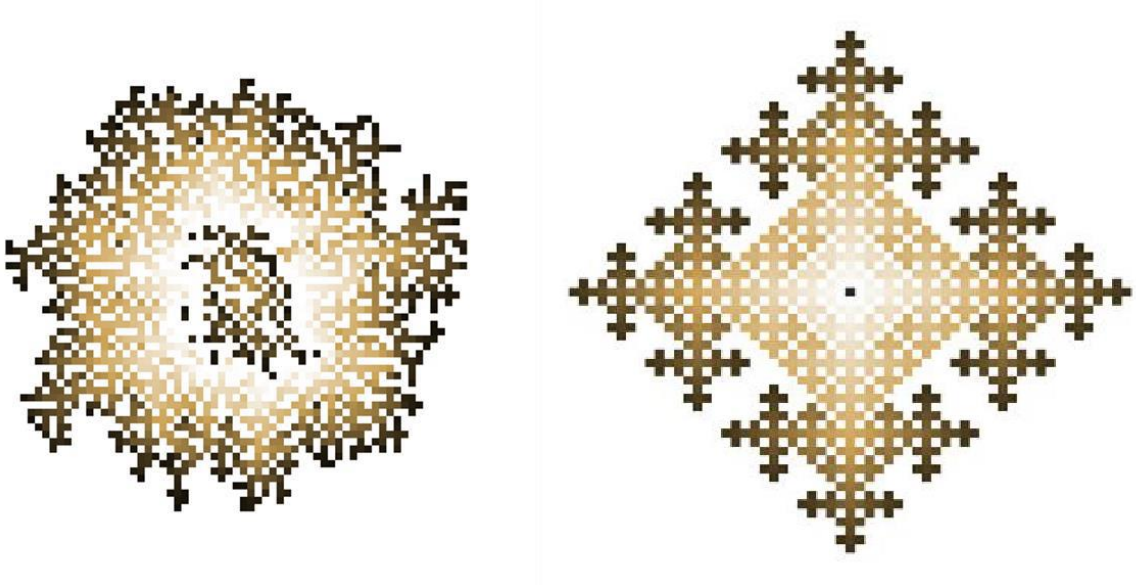

Figure 10 Outcome of a stochastic CA (left) and deterministic CA (right)[4]

Those if-else rule-based CA are called constrained CA. The transition rules are only applied if a local condition is true and therefore, constrained CAs are resulting in very realistic results (Wallentin, 2019). Moreover, a second type of CA has been developed. The so-called probabilistic CA are not using if-else rules but encoding correlative relations between one or multiple explaining variables and the dependent variable (Figure 11). Typically, probabilistic CA are relying on existing datasets to infer statistical relationships of Markov chain matrices (Wallentin, 2019).

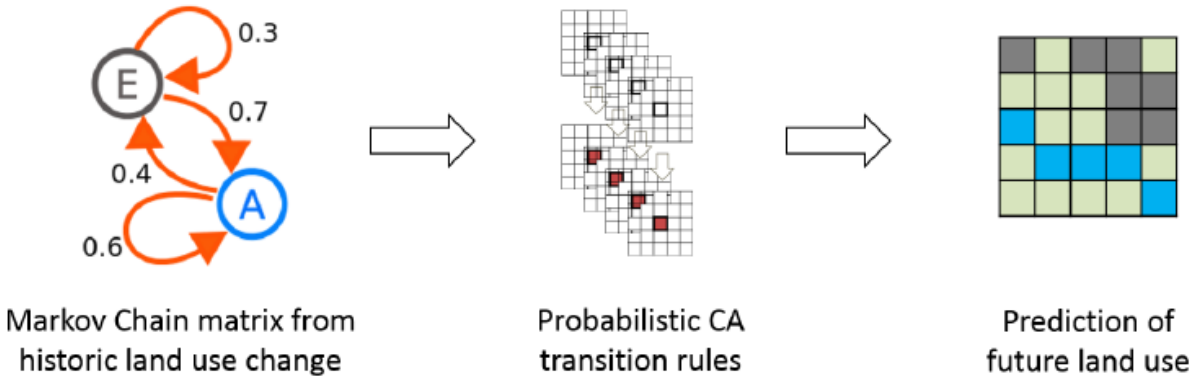

Figure 11 Probabilistic CA using a Markov Chain matrix[5]

[4] UNIGIS (modified)
[5] UNIGIS

Assessment methods for Cellular Automata

To make CA a valuable resource for research, model assessment methods are needed to proof the validity of the CA output. For most models, especially in the field of land use change and urban growth, multiple methods are used simultaneously to proof the models validity. For CA model assessment, cell-by-cell comparison and landscape analysis are the most frequently used ones. In the terms of all validation metrics, overall accuracy is ranked first and standard Kappa coefficient is ranked second (Tong). To achieve the most accurate results, the parameters must be adjusted and precise model assessment must be conducted. Due to the many factors in a CA like cell size, transition rules, agents, the calibration process is very challenging and takes up a lot of time in the development of CA. Additionally, it must not be disregarded that there will be a difference between the simulated result and the actual *ground truth*. For a reliable CA this difference should be within acceptable limits (Tong).

Pursuant to (Tong), the three major types of model assessment are:

1. Dataset assessment
2. Procedure assessment
3. Result assessment

To perform the model assessment and related metrics, CA software and/or other software packages can be used.

Dataset assessment

Dataset assessment allows to evaluate the reliability of the input data. On the one hand, major errors in the input data may be an introductory element of uncertainties in the CA output and therefore must not be ignored because they can be propagated through the model during the simulation. On the other hand, if the accuracy of the input data is high, errors in the output of the CA model are probably more cause by the modelling process rather than the raw input dataset. In summary, dataset assessment is used to evaluate the input data reliability to provide useful insights for controlling modelling errors (Tong & Feng, 2020).

Procedure assessment

Since each process plays an essential role in the result quality of a CA, and especially the transition rules are the core of CA and subsequently well-fitted

transition rules are resulting in results that are more accurate. Procedure assessment especially provides insights into the statistics about how fittingly transition rules and input data are working together, as well as, how efficient the model runs. A first indicator of the efficiency is the run-time of the model, as the run-time is a clear indicator for computing performance. Given that this paper isn't about to explain and compare the different assessment methods in depth, an informative overview about procedure assessment is given in the paper *"A review of assessment methods for cellular automata models of land-use change and urban growth"* (Tong & Feng, 2020).

Result assessment

Result assessment compares the end-state and change of the simulation with the actual result by, for example, visual inspection. It can be realized by several methods to delineate the three aspects of the simulation results: end-state, change, and the explanatory ability of driving factors (Tong & Feng, 2020). The easiest, but highly subjective method is a visual inspection to examine similarities and differences between the simulated result and an actual result. Cell-by-cell comparison methods are detecting whether the state of each cell matches between the simulation results and the actual results. This method generates a cross-tabulation matrix, including accuracy and error statistics. The accuracy and error statistics are discussed in the paper *"A review of assessment methods for cellular automata models of land-use change and urban growth"* (Tong & Feng, 2020).

In a nutshell, result assessment is providing a comprehensive evaluation of the overall model performance.

Software

As stated in the introduction there are plenty of different modelling tools and software packages on the market. While modelling tools like Repast or CORMAS require advanced algorithmic skills, platforms like NetLogo or GAMA are using high-level agent-based modelling languages allowing less experienced users to develop and simulate models on their own. But it comes with a trade-off. Compared to the other group of modelling frameworks, they are not as good in handling large datasets and computational power. Nevertheless, the advantages

are that anyone can develop and execute models for own purposes, makes them very attractive to scientists from all different kinds of scientific fields.

NetLogo

The NetLogo software package is developed by the Center for Connected Learning and Computer-Based Modeling at Northwestern University, Illinois. It is written in Java to aid portability and compatibility, was first released in 1999, and runs on the most common personal computing operating systems (Microsoft Windows, Mac OS, and Linux) as well as in a web environment. NetLogo is freely downloadable and licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License. When downloading and installing NetLogo, it comes with a documented library supporting over 150 sample models, allowing the user to learn and adapt NetLogo quite fast. A key feature is its main purpose as a multi-agent programming language and modeling environment, and supports as a Lisp family member, multiple agents, and concurrency. Those agents are programmable, can interact with each other, and perform multiple tasks concurrently, which allows users to explore connections between micro-level behaviors of individual agents and macro-level patterns coming up from the agents' interactions (Tisue & Wilensky, 2004). They are called "turtles", when they can move, and "patches" otherwise. This makes NetLogos main focus on simulation natural and social phenomena (Tisue & Wilensky, 2004). While the "low-threshold, high-ceiling" design philosophy is inherited from Logo (Blikstein et al., 2005). Therefore, NetLogo is a satisfactory language to model time evolving and complex systems (Tisue & Wilensky, 2004) and has been widely used by a broad audience. An audience, starting at the very beginning with pupils from elementary school to scientists from various disciplines (Sklar, 2007). Especially the possibility of NetLogo to add available extensions, for example to run automated experiments, 3D visualizations, connecting to external physical devices and important for the GIS to import GIS and GIS raster data makes it a sophisticated software for researchers. Additionally, NetLogo can be extended with own Java extensions (Sklar, 2007) and be connected to other programming languages like python and R. Another mentionable feature of NetLogo is, that NetLogo allows to save a created models as Java applets to grant a smooth integration of the model into a web page (Sklar, 2007). On the downside, the

advantages of OS independency and good extendibility by developing NetLogo in Java, it is too slow for intense computations over many iterations (Sklar, 2007).

Despite some disadvantages, for example NetLogos old fashioned design and handling or lacking performance on big simulations, the advantages are prevailing. NetLogo demonstrated to be a practical tool to prototype simulations. Developed to investigate research questions in the field of modeling real-world phenomena using multi-agent systems (Sklar & Davies, 2005) and studying interaction mechanism in multi-agent systems ((Sklar et al., 2006), (Reich & Sklar, 2006)).

GAMA

GAMA is an acronym and stands for *GIS & Agent-based Modeling Architecture*. It is based on the Eclipse IDE and therefore executable on the three major OS (Microsoft Windows, Mac OS, and Linux). As a collaborative open-source software, GAMA can be freely downloaded and used under the GPL license. When being downloaded, the software already comes with a library including over 300 models, to show the users different features, classic models like the predator-prey model and tutorials to ease onboarding with the software. Using the Eclipse IDE allows GAMA, to take the advantages of (a) users are already feeling comfortable with the interface before using GAMA and (b) offers the modeler all the benefits from a modern IDE (Taillandier, Gaudou, et al., 2019). For example, auto-coloring, auto-compilation, auto-completion, auto-formatting, and a like large scale of tools to manage models and simulations. But the main reason for the development of GAMA was not to put a modeling software into a common IDE. It was the need to overcome the weak integration of GIS data in simulations and addressing further shortcoming on other simulation platforms (Taillandier, Gaudou, et al., 2019). Accordingly, GAMA allows naturally to import and integrate GIS vector and raster data, without needing additional plugins, into the simulation environment. Because it was developed with the focus on addressing GIS data, GAMA comes with features known from professional GIS software. The modeler can create agents directly from real data by importing shapefiles (and other file formats) and GAMA will take care of the spatial projection and reading of the data ((Drogoul et al., 2013a), (Taillandier, Gaudou, et al., 2019)). To make spatial simulations more efficient, GAMA makes usage of the dynamic Quadtree structure and updated this structure according to the movement of the

agents (Taillandier, Gaudou, et al., 2019). Furthermore, GAMA proposes different high-level geometry transformations, for example buffer, convex hull, etc. (Drogoul et al., 2013a), and algorithms to compute the shortest path on graphs as well as distances and shortest path on grids (Taillandier, Gaudou, et al., 2019). This is provided by GAML, GAMAs high-level programming language.

GAML (GAMA Modeling Language), the GAMAs agent-oriented programming language. Taking its roots in object-oriented languages like Java or Smalltalk, GAML extends the object-oriented programming approach with compelling agent concepts for a better expressiveness in models. Furthermore, by following OOP design patterns, GAML enriches "older" agent-based modeling languages with modern computing notions like inheritance, type safety or multi-level agency (Taillandier, Gaudou, et al., 2019). In GAML, agents are called species and provided with a set of attributes, actions, and reflexes. A special form of species are *grids*, which are defined by the number of rows and columns and the size of the cells or by using a raster (geographical) file. Moreover, two very specialized species are extant. Firstly, a species called experiment, containing the output of the aforementioned species, and secondly, the model species containing everything needed for the model.

Further interesting features implemented in GAMA are event layers, allowing an action when an event occurs (Taillandier, Grignard, et al., 2019). Advanced visualization features to provide 3D visualization with textures, complex lights, and custom dynamic cameras (Taillandier, Grignard, et al., 2019). Communication with other programs and devices through different protocols (MQTT, UDP, TCP) (Taillandier, Grignard, et al., 2019). Adaption of the modular modeling approach to develop and re-use more complex and specified models in a modular way (Taillandier, Gaudou, et al., 2019). Additionally, own Java-plugins can be added to GAMA and packages like GAMAR or OpenMole providing useful features. GAMAR has been developed to create experiments, run simulations and analyze results with the R scripting language and OpenMole allows GAMA to run on supercomputers using the OpenMole framework.

Despite the feature richness and possibilities to create and run simulations, GAMA faces the problem of high-level programming languages. Due to abstractions made for usability, the performance is lacking in bigger experiments.

Nevertheless, GAMA provides field experts, modeler, and computer scientists with a thorough modeling and simulation environment for building rich spatially explicit agent-based simulations (Drogoul et al., 2013b). It has been used in wide scale GIS concerning application, for example the simulation of organization evacuation in case of a tsunami at the city of Nha Trang in Vietnam, the identification of city management strategies of Dijon and Grenoble in France as well as natural resource management and epidemiology (Drogoul et al., 2013b).

Conclusion

In this paper a brief introduction to spatial dynamics and cellular automata was given to make the reader more conscious about what is important for spatial simulations in the GIS domain. Subsequent, the NetLogo and GAMA agent-based modelling platforms were introduced with their strengths and weaknesses. While the strengths of NetLogo clearly are the long development period and a numerous number of available plug-ins, GAMA scores with its development focus on GIS data and to run it on supercomputer. To sum it up, GAMA is the recommended platform when it comes to the terms of GIS data-based simulations.

References

Blikstein, P., Abrahamson, D., & Wilensky, U. (2005). Netlogo: Where we are, where we're going. *Proceedings of the Annual Meeting of Interaction Design and Children, Press*, *23*.

Desmet, K., & Rossi-Hansberg, E. (2010). ON SPATIAL DYNAMICS*. *Journal of Regional Science*, *50*(1), 43–63. https://doi.org/https://doi.org/10.1111/j.1467-9787.2009.00648.x

Drogoul, A., Amouroux, E., Caillou, P., Gaudou, B., Grignard, A., Marilleau, N., Taillandier, P., Vavasseur, M., Vo, D.-A., & Zucker, J.-D. (2013a). Gama: A spatially explicit, multi-level, agent-based modeling and simulation platform. *International Conference on Practical Applications of Agents and Multi-Agent Systems*, 271–274.

Drogoul, A., Amouroux, E., Caillou, P., Gaudou, B., Grignard, A., Marilleau, N., Taillandier, P., Vavasseur, M., Vo, D.-A., & Zucker, J.-D. (2013b). Gama: multi-level and complex environment for agent-based models and simulations. *12th International Conference on Autonomous Agents and Multi-Agent Systems*, 2--p.

Gardner, M. (1970). MATHEMATICAL GAMES The fantastic combinations of John Conway's new solitaire game "life." *Scientific American*, *223*, 120–123.

Grimm, N. B., Faeth, S. H., Golubiewski, N. E., Redman, C. L., Wu, J., Bai, X., & Briggs, J. M. (2008). Global change and the ecology of cities. *Science*, *319*(5864), 756–760.

Irwin, E. G. (2010). NEW DIRECTIONS FOR URBAN ECONOMIC MODELS OF LAND USE CHANGE: INCORPORATING SPATIAL DYNAMICS AND HETEROGENEITY*. *Journal of Regional Science*, *50*(1), 65–91. https://doi.org/https://doi.org/10.1111/j.1467-9787.2009.00655.x

Krugman, P. (1991). Increasing returns and economic geography. *Journal of Political Economy*, *99*(3), 483–499.

Levin, S. A. (2006). Learning to live in a global commons: socioeconomic challenges for a sustainable environment. *Ecological Research*, *21*(3), 328–333.

Liu, J., Dietz, T., Carpenter, S. R., Alberti, M., Folke, C., Moran, E., Pell, A. N., Deadman, P., Kratz, T., Lubchenco, J., & others. (2007). Complexity of

coupled human and natural systems. *Science*, *317*(5844), 1513–1516.

Lucas Jr, R. E. (1988). On the mechanics of economic development. *Journal of Monetary Economics*, *22*(1), 3–42.

Neumann, J. (1963). *The General and Logical Theory of Au-tomata*.

Pickett, S. T. A., Cadenasso, M. L., Grove, J. M., Nilon, C. H., Pouyat, R. V, Zipperer, W. C., & Costanza, R. (2001). Urban Ecological Systems: Linking Terrestrial Ecological, Physical, and Socioeconomic Components of Metropolitan Areas. *Annual Review of Ecology and Systematics*, *32*(1), 127–157. https://doi.org/10.1146/annurev.ecolsys.32.081501.114012

Ravì, D., Wong, C., Deligianni, F., Berthelot, M., Andreu-Perez, J., Lo, B., & Yang, G.-Z. (2017). Deep Learning for Health Informatics. *IEEE Journal of Biomedical and Health Informatics*, *21*(1), 4–21. https://doi.org/10.1109/JBHI.2016.2636665

Reich, J., & Sklar, E. (2006). Robot-sensor networks for search and rescue. *IEEE International Workshop on Safety, Security and Rescue Robotics*, *22*.

Sklar, E. (2007). NetLogo, a multi-agent simulation environment. *Artificial Life*, *13*(3), 303–311.

Sklar, E., & Davies, M. (2005). Multiagent simulation of learning environments. *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, 953–959.

Sklar, E., Shut, M., Diwold, K., & Parsons, S. (2006). Exploring Coordination Properties within Populations of Distributed Agents. *AAAI Spring Symposium: Distributed Plan and Schedule Management*, 121–127.

Taillandier, P., Gaudou, B., Grignard, A., Huynh, Q.-N., Marilleau, N., Caillou, P., Philippon, D., & Drogoul, A. (2019). Building, composing and experimenting complex spatial models with the GAMA platform. *GeoInformatica*, *23*(2), 299–322.

Taillandier, P., Grignard, A., Marilleau, N., Philippon, D., Huynh, Q.-N., Gaudou, B., & Drogoul, A. (2019). Participatory modeling and simulation with the gama platform. *Journal of Artificial Societies and Social Simulation*, *22*(2).

Tisue, S., & Wilensky, U. (2004). Netlogo: A simple environment for modeling complexity. *International Conference on Complex Systems*, *21*, 16–21.

Tobler, W. R. (1970). A Computer Movie Simulating Urban Growth in the Detroit Region. *Economic Geography*, *46*(sup1), 234–240.

https://doi.org/10.2307/143141

Tong, X., & Feng, Y. (2020). A review of assessment methods for cellular automata models of land-use change and urban growth. *International Journal of Geographical Information Science*, *34*(5), 866–898. https://doi.org/10.1080/13658816.2019.1684499

Turner, B. L., Lambin, E. F., & Reenberg, A. (2007). The emergence of land change science for global environmental change and sustainability. *Proceedings of the National Academy of Sciences*, *104*(52), 20666–20671.

Wallentin, G. (2019). *Spatial Simulation*.

Wu, J., & Hobbs, R. (2002). Key issues and research priorities in landscape ecology: an idiosyncratic synthesis. *Landscape Ecology*, *17*(4), 355–365.