WHAT IS IT?

This project was developed during the Santa Fe course Introduction to Agent-Based Modeling 2022. The origin is a Cellular Automata (CA) model to simulate human interactions that happen in the real world. The model presented at referencess uses a market research with real people in two different times: one at time zero and the second at time zero plus 4 months (longitudinal market research).

The paper authors develop an agent-based model whose initial condition was inherited from the results of the first market research response values and evolve it to simulate human interactions with Agent-Based Modeling that led to the values of the second market research, without explicitly imposing rules. Then, compared results of the model with the second market research. The model reached 73.80% accuracy.

In the same way, this project is an Exploratory ABM project that models individuals in a closed society whose behavior depends upon the result of interaction with two neighbors within a radius of interaction, one on the relative "right" and other one on the relative "left". According to the states (colors) of neighbors, a given cellular automata rule is applied, according to the value set in Chooser. Five states were used here and are defined as levels of quality perception, where red (states 0 and 1) means unhappy, state 3 is neutral and green (states 3 and 4) means happy.

There is also a message passing algorithm in the social network, to analyze the flow and spread of information among nodes. Both the cellular automaton and the message passing algorithms were developed using the Python extension.

There are two types of agents (breeds): clients (person shape) and service providers (star shape). Each one of them carries an internal state from 0 to 4, and also the amount of information, a float starting at 0 (no information at all) and greater than that (amount of information carried).

Each agent breed will choose two neighbors within the radius of interaction: an agent with the same breed as itself and an agent of another breed. This set will be used by the cellular automaton algorithm to generate the future state of the agent. Each agent will then move to the XY coordinate between the two neighbors. Note that in the case of lack of two neighbors, the agent can consider its two neighbors as a single other agent.

Information starts at level 1.00 for the individual with the biggest degree (connections in the social network), and 0.00 for all the others. It's possible to note in the plot that the information flows through the network, increasing or decreasing its value over time.

Besides the interaction and the formation of social networks and information spread, the system is also subject to levels of temperature of the environment, measured with a sensor attached to Arduino, following the sketch presented at the INSTALLATION section. Patches allow free movement of agents. However, if you have the opportunity to connect the Arduino device, as presented in the Installation chapter, you will notice how higher and lower temperatures influence the amount of movement steps taken by agents. The lattice is not a toroid, meaning it is not wrapped at borders.

As Inputs the model offers the possibilities described at the Chapter HOW IT WORKS - INITIALIZATION. Outputs can be accessed via the Behavior Space setup saved as experiment-rule232, with 10 runs to generate more robustness of the model, and also at the HOW TO USE IT section.

As final considerations about Setup and Run, the user must enter that local path for saving the model (if desired). A CSV file with the connections FROM and TO of the social network must be loaded, as stated in Chapter INSTALLATION. For both tasks, be sure you have the appropriate permissions to save/read the file in the location chosen.

As a final advice, extensions [py nw arduino csv matrix] are used. Be sure to activate them in Tools / Extensions.

INSTALLATION

To install **Python** in Windows, follow this tutorial:
https://docs.anaconda.com/anaconda/install/windows/

or download directly from:

https://repo.anaconda.com/archive/Anaconda3-2019.07-Windows-x86_64.exe

To install Python and libraries in Linux run:
wget https://repo.anaconda.com/archive/Anaconda3-2019.07-Linux-x86_64.sh
sudo bash Anaconda3-2019.07-Linux-x86_64.sh

For Linux, set the Anaconda folder as /home/anaconda3 so that you don't have permission issues.

Anaconda is suggested in order to have the libraries required by this notebook: numpy and scipy.

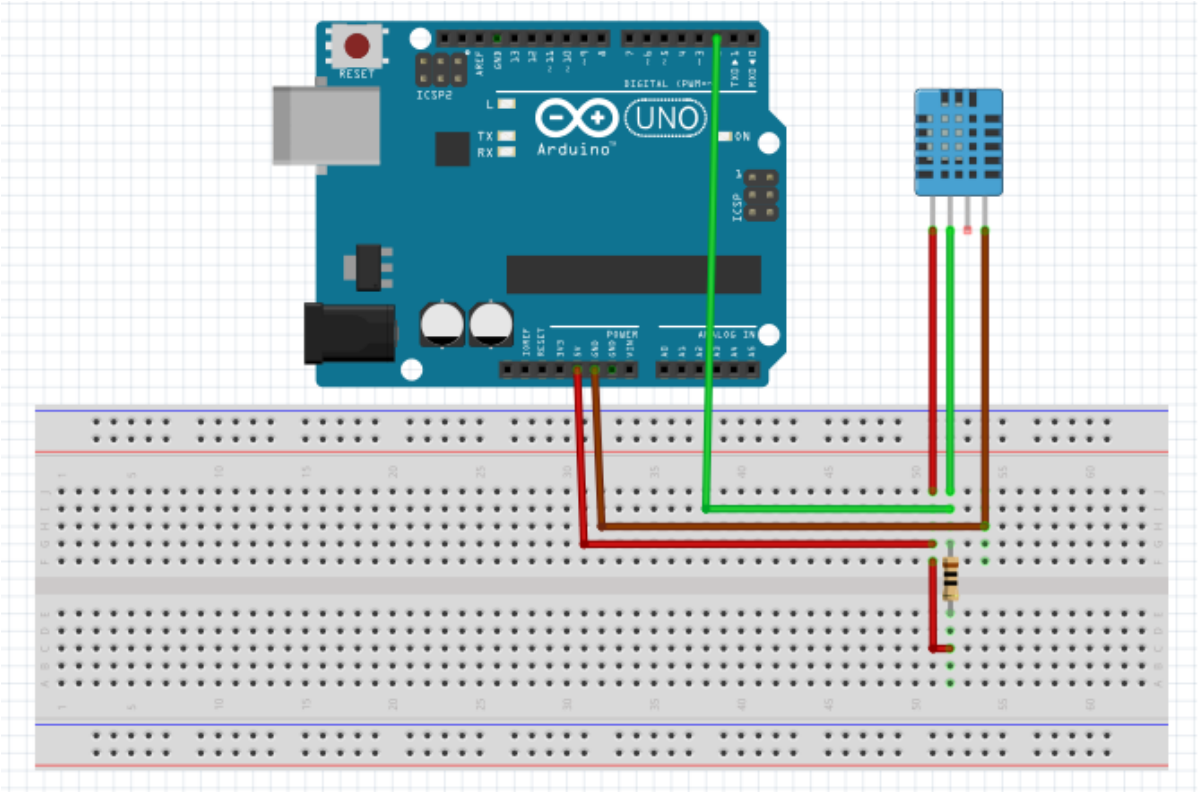
The model works without Arduino installation. However, if you opt for using **Arduino**, first download the Arduino software at:

https://www.arduino.cc/en/software

Install it and plug Arduino. Then, get the DHT library from Arduino and download the project .ino file from:

>https://raw.githubusercontent.com/RubensZimbres/Repo-2022/main/SantaFe/NetLogo/Arduino/arduino-example-sketch.ino

After that, connect the Temperature/Humidity sensor DHT11 to your Arduino, as the schema in the picture below:



In Arduino software, compile and upload to the board. If you have issues with Linux, run:

sudo chmod a+rw /dev/ttyACM0

Open Tools / Serial Monitor in Arduino Software and check if the sensor is indeed reading the humidity and temperature. Close the Monitor so that the port will be available for NetLogo.

Then, activate the Python extension in NetLogo and configure the Python path, for instance: /home/anaconda3/bin/python

You will also need the **CSV** file to connect nodes, available at:

https://raw.githubusercontent.com/RubensZimbres/Repo-2022/main/SantaFe/NetLogo_final/Best_v3/social_80.csv

Be sure to edit the CSV path in the code at section Setup.

HOW IT WORKS

CELLULAR AUTOMATON

With 2 cellular automaton (CA) states, you have 256 possible rules. With 5 states, also used here, you have 2350988701644575015937473074444491355637331113544175043017503412556834518909454345703125 possible rules. The biggest issue here is to know which will be the

outcome for a chosen rule. The rule of referred article and used here is a CA 5-state radius-one rule number 21590625125649876448194552191168939458959585281520212287057525638079592376559119505491 24.

Initially, each individual has its own state. As an amorphous cellular automaton, the neighborhood is not limited to immediate neighbors in the lattice. The agent will interact with any of the other 2 individuals within the defined radius of interaction. This model have two breeds, one is clients and the other service providers.

The behavior of an agent is defined by the type of neighbors he/she has at time t. So, at time t+1 the states of all agents that interacted with two neighbors are updated according to the rule.

Cellular Automata principle: for each cell in the grid with its position c(i,j) where i and j are the row and the column respectively, a function Sc(t)=S(t;i,j) is associated with the lattice to describe the cell c state in time t. So, in a time t+1, state S(t+1,i,j) is given by:

$S(t+1;i, j) = [S(t;i,j)+\delta] \bmod k$

where $-k \leq \delta \leq k$ and k is the number of cell c states. The formula for δ is:

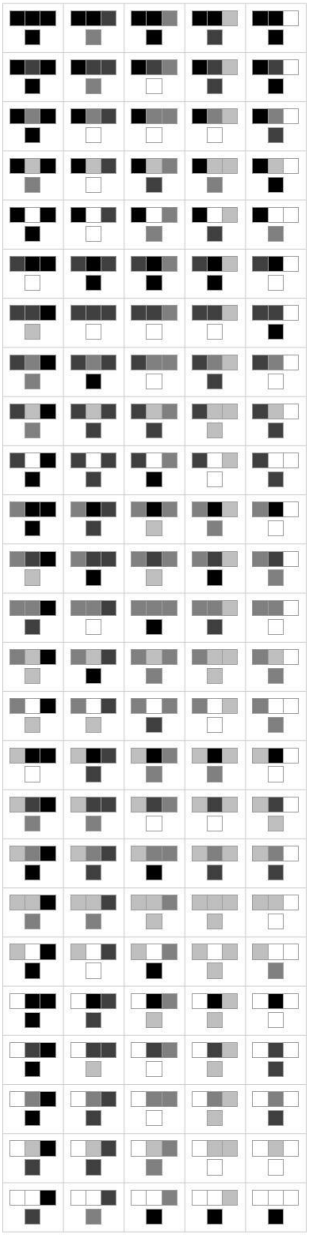
$\delta = \mu$ if condition (a) is true

$\delta = -S(t;i,j)$ if condition (b) is true

$\delta = 0$ otherwise

where a and b change according to the rule.

The chosen cellular automaton rule has the following transition table to guide the state update, according to the states of its 2 neighbors.



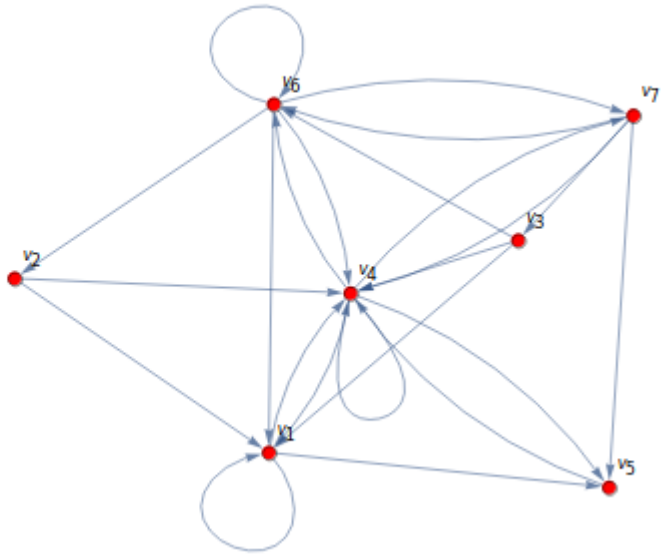
In a two-dimensional lattice, the chosen rule has the following behavior:

The cellular automaton algorithm was developed using the Python extension for NetLogo and is able to support rules with 2 to 5 states.

MESSAGE PASSING ALGORITHM

Regarding the message passing algorithm, it was also developed in Python from the adjacency matrix (A) collected from the NetLogo run.

```
1  0  0  1  1  0  0
1  0  0  1  0  0  0
1  0  0  1  0  1  0
1  0  0  1  1  1  1,
0  0  0  1  0  0  0
1  1  0  1  0  1  1
0  0  1  1  1  1  0
```



The calculation of information flow is given by the following algorithm:

- Calculate the Normalized Adjacency Matrix, by adding the Identity Matrix

$$\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$$

$$\tilde{A} = A + I$$

- Then calculate the Diagonal Degree Matrix

$$(D)_{ij} = \delta_{i,j} \sum_{k} A_{i,k}$$

Initially, only the agent with more links has information equal to 1, then, this information spreads over the network, by successively multiplying this initial vector of information [1,0,0,0,0....] by the Diagonal Degree Matrix. Then, information flows over the network [0.6,0.22,0.27,0.1,0,0,0...].

ARDUINO AND MOVEMENT STEPS

The temperature sensor will generate N MOVEMENT-STEPS, according to the following equations:

First Law Thermodynamics:

$$\Delta\varepsilon = \Delta heat - work$$

Other useful formula:

$$work = force \times displacement$$

$$Force = mass \times acceleration$$

$$work = mass \times acceleration \times displacement$$

$$\Delta heat - \Delta\epsilon = mass \times \frac{distance}{time^{2}} \times distance$$

Given mass constant, we have:

$$\Delta heat - \Delta\epsilon = k\frac{distance^{2}}{time^{2}}$$

$$distance = \frac{\sqrt{time^{2} \times |\Delta heat - \Delta\varepsilon|}}{k}$$

Given synchronous update and movement-steps variable, we have:

$$movement\_steps = \frac{\sqrt{|\Delta heat - \Delta\varepsilon|}}{k}$$

$$\Delta\varepsilon = change\ in\ cellular\ automata\ state$$

$$\Delta heat = difference\ in\ temperature\ sensor$$

INITIALIZATION

The system is initialized choosing:

ARDUINO-ON: choose if Arduino is set to ON or OFF. If Arduino is ON and properly set up, it is supposed that humidity and temperature values show up. If not, check your installation and if all wires and cables are connected.
LINKS-TO-USE: if they are directed or undirected
FRAC-PROVIDERS: fraction of the agents that are service providers. The other fraction is composed of clients.
PERCENT-UNHAPPY: amount of agents with state < 3
MOVEMENT-STEPS variable defines how much an agent should move in case there are no neighbors available inside the radius defined in Setup, or even if it is unhappy.
The RADIUS-OF-INTERACTION variable helps the agent to decide the radius in which neighbors will be chosen.

The MUTATED variable. Using a genetic metaphor, the idea here is to add diversity to allow the evolution of the social network.
CA-BASE: number of states of the cellular automaton
CA_RULE: cellular automaton rule. Note that 2 states support rules up to 255.
LOAD-GRAPHML: keep it OFF unless you are loading a previous saved configuration
LAYOUT: option as radial, spring and circular
EPOCHS: how many cycles will last the model.

The temperature is collected by a DHT11 sensor attached to Arduino and influences the degree of movimentation of agents. In colder temperatures, agents move less, and in high temperatures, agents move more steps. This is similar to physical properties of the states of matter.

It's also possible to enter the path where you want to save the model as a GraphML file.

ITERATIVE

The agents will randomly choose their left and right neighbors. Clients and service providers will choose a client and a service provider as neighbors.

In case there are not two neighbors within the radius defined in RADIUS-OF-INTERACTION, they will move MOVEMENT-STEPs until they find two available neighbors. In this movement, relative radius is updated. As soon as they find two neighbors, they update their X,Y coordinates as the mean of neighbors position and their state according to the colors of each one of the neighbors, following the transition table of the chosen CA rule.

As interactions happen, the number of cliques, the closeness, betweenness and communities are recorded, as well as the degree distribution and the states in the social network. For the sake of simplicity and ease of visualization, states less than 3, unhappy, are represented by red, state 3, neutral is represented by yellow and states 3 and 4 are green.

The button IDs, when pressed, will show the labels of agents when the mouse is over. The size of agents shows how much information he/she carries.

HOW TO USE IT

Click the SETUP button to set up the agents. You can choose equal numbers of red and green agents, but you can change it using PERCENTAGE-UNHAPPY. The agents are set up so no patch has more than one agent. Click GO to start the simulation.

If you choose ARDUINO-ON, wait for temperature and humidity to show up. If they don't show up when you click SETUP, go to Arduino software Monitor and check if the sensor is really reading temperature and humidity. Click Setup again and click Go.

When you stop the iterations, you can check the number of turtles unhappy, neutral and the number of turtles happy. If Arduino is connected, HUMIDITY and TEMPERATURE of the environment are shown.

Below these monitors MEAN CLUSTERING COEFFICIENT of turtles is presented and also the minimum and maximum DEGREES (number of connections), as well as total LINKS of the social network.

STRUCTURAL HOLES are individuals in the social network that have connections that generate holes in the density or even isolated individuals. The STRENGTH OF TIES can be seen by the thickness of the connections (links) among agents.

After you stop the model run, when you click BIGGEST CLIQUES, you will find the individual with more cliques in the social network. A clique is a subset of a network in which the actors are more closely and intensely tied to one another than they are to other members of the network. Think of it as a group of people connected by strong social ties.

DETECT COMMUNITIES

You can also detect communities. Netlogo interface will show each community with its members and correspondent colors. This option detects community structures present in the network. It does this by maximizing modularity using the Louvain method. The Louvain method is a greedy optimization of modularity, a value between −0.5 (non-modular clustering) and 1 (fully modular clustering) that measures the relative density of edges inside communities with respect to edges outside the community tested. In a detected community, the modularity will increase. Then, community nodes are grouped to restart the algorithm.

CLOSENESS

The Closeness button will show you the closeness of each agent to the rest of the network. Closeness centrality indicates how close a node is to all other nodes in this network. It is calculated as the average of the shortest path length from the node to every other node in the network. A smaller value means that the given agent is closer to other nodes of the network.

BETWEENNESS

To calculate the betweenness centrality of an agent, you take every other possible pairs of agents and, for each pair, you calculate the proportion of shortest paths between members of the pair that passes through the current agent. The betweenness centrality for each node is the sum of the numbers of these shortest paths that pass through the node.

The three plots in the interface show the distribution of states, the degree distribution as an histogram and the mood evolution. Note that the MOOD output may be very similar to the oscillating behavior found by Brian Arthur (1994).

If you decrease the MOVEMENT-STEPS, RADIUS OF INTERACTION and the SPEED, you can see how communities are being formed. Try MOVEMENT-STEPS = 2 and RADIUS-OF-INTERACTION = 2.

PLOTS:

The first plot on the left DISTRIBUTION OF STATES shows the total number of agents with the states happy, unhappy and neutral.

The plot MOOD EVOLUTION shows how the mood (sum of states) of clients and service providers evolve over time.

The third plot shows the TOTAL AMOUNT OF INFORMATION OVER TIME considering the whole artificial society.

The fourth plot shows INFORMATION REACH, meaning how far the information flows in the social network. A bigger value means more nodes have access to information.

THINGS TO NOTICE

When you execute SETUP, the red and green agents are randomly distributed throughout the neighborhood. Clients are turtles with shape "person" and service providers are turtles with shape "star". Each agent moves towards its neighbors. When in the new locations, they may alter the equilibrium of the local population, prompting other agents to get close or to leave.

Notice the emergence of clusters of deep connected individuals. Note that according to the RADIUS-OF-INTERACTION, CA-RULE and MOVEMENT-STEPS a polarization or collapse may happen. Note that polarization happens even if agents have the same state (opinion).

Notice the weight of edges following multiple interactions with neighbors. You can find the most influential individuals in the network. Notice also that the size of nodes increases as the agents have more connections.

Notice if information quality fades along the interactions. According to Gillani et al., (2018) homophily creates “echo chambers” that degrade the quality, safety, and diversity of discourse.

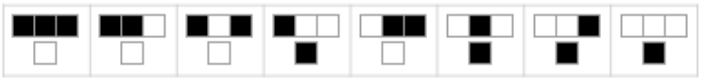
Note what happens in the network dynamics if the temperature of the environment goes up or down. Note also how different setups change the flow of information and the number of agents reached.

THINGS TO TRY

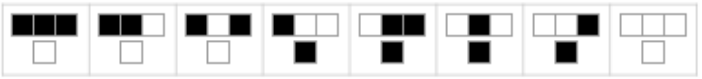
Try to alter the amount of movement the agent does when he has few neighbors. You can also change the radius an agent will consider when interacting with others. Does a small radius isolate him/her in the social dynamics ?

Make experiments with these 2-state CA interesting rules, altering the rule along with the other Inputs. You will notice that according to Setup, a cellular automaton rule behaves differently. These are the behaviors in a two dimensional lattice.

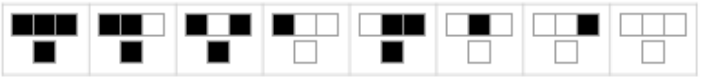
23 - minority rule



30 - random rule



232 - majority rule



You can also try to choose left and right neighbors based on state similarity.

You can make experiments with variations of temperature and how this affects the dynamic of the system.

Try different setups. Maybe you will reach a state where the system may reach an equilibrium, stop moving and stop creating new links. In this case, increase the temperature of the system in the Arduino DHT11 sensor and see what happens.

Try also the opposite, when the social ties are forming, decrease environment temperature and see if the system reaches an equilibrium.

ANALYSIS

The Behavior Space is set up to handle different runs (10) for each one of the changing variables values, in order to deliver more robust results.

EXTENDING THE MODEL

Incorporate social networks concepts into this model. For instance, have unhappy agents decide on a new location based on information about the other communities, like betweenness centrality, closeness of agents, etc.

Find the perfect setup for the rapid emergence of happiness and a good quality perception.

REFERENCES

Albaum, G., (1967). Information flow and decentralized decision making in marketing. California Management Review, 9(4), 59-70.

Arthur, W.B. (1994) Inductive Reasoning and Bounded Rationality. The American Economic Review, Vol. 84, No. 2, Papers and Proceedings of the Hundred and Sixth Annual Meeting of the American Economic Association, pp. 406-411.

Axelrod, R. (1997). Advancing the Art of Simulation in the Social Sciences. Handbook of Research on Nature Inspired Computing for Economy and Management, Jean-Philippe Rennard (Ed.).Hersey, PA: Idea Group.

Axelrod, R. (1997). Complexity of cooperation. New Jersey: Princeton University Press, 1997.

Bertalanffy, L. (1950). An Outline of General System Theory. British Journal of the Philosophy of Science, 1950

Cederman, L.E., (2003). Computational models of social forms: Advancing generative macro theory. Paper prepared for presentation at the 8 th Annual Methodology Meeting of the American Sociology Association, University of Washington, Seattle.

Epstein, J.M. & Axtell, R., (1996). Growing artificial societies: Social science from the bottom up. MIT Press, Cambridge.

Flache, A. Hegselmann, R. (2001). Do Irregular Grids make a Difference? Relaxing the Spatial Regularity Assumption in Cellular Models of Social Dynamics. Journal of Artificial Societies and Social Simulation, v. 4, n. 4.

Ganguly, N.; Sikdar, B.K.; Deutch, A.; Canright, G.; Chaudhuri, P.P. A survey on cellular automata.

Granovetter M.S. (1973) The Strength of Weak Ties. American Journal of Sociology. Vol. 78, No. 6, pp. 1360-1380

Macy, M.W.; Willer, R. (2002). From factors to actors: Computational sociology and agent-based modeling. Annual Review of Sociology, v. 28.

Pew Research Center. (2014). Political Polarization in the American Public.

Sawyer, R.K. (2003). Artificial societies: Multiagent systems and the micro-macro link in sociological theory. Sociological Methods and Research, v. 31, n. 3, Feb 2003.

Sawyer, R.K. (2004). Social explanation and computational simulation. Philosophical explorations, v. 7, n. 3.

Schelling, T. (1978). Micromotives and Macrobehavior. New York: Norton.

Tesfatsion, L., (2005). Agent-based computational economics: A constructive approach to economic theory. Forthcoming in Judd, K.L. Tesfatsion, L. Handbook of Computational Economics. North-Holland.

Vicsek, T. (2002) Complexity: The Bigger Picture. Nature, v. 418.

Wolfram, S. (2002). A new kind of science. Canada: Wolfram Media Inc.

Zimbres, R.A. (2006) Modelagem Baseada em Agentes: uma Terceira Maneira de se Fazer Ciência? ANPAD, Presented at Encontro da Associação Nacional de Pós-Graduação e Pesquisa em Administração.

Zimbres, R.A.; Brito, E.P.Z.; Oliveira, P.P.B. (2008) Cellular automata based modeling of the formation and evolution of social networks: A case in Dentistry. In: J. Cordeiro and J. Filipe, eds. Proc. of the 10th Int. Conf. on Enterprise Information Systems, INSTICC Press: Setùbal-Portugal, Vol. III: Artificial Intelligence and Decision Support Systems, pp. 333-339.

Zimbres, R.A., Oliveira,P.P.B. (2009) Dynamics of Quality Perception in a Social Network: A Cellular Automaton Based Model in Aesthetics Services. Electronic Notes in Theoretical Computer Science, Elsevier, 252 pp 157–180.

ACKNOWLEDGEMENTS

The author thanks Google Developers Experts for supporting his research activities. Special thanks also to Professor William Rand and NetLogo Stackoverflow community, who made this model possible.

HOW TO CITE

If you mention this model or the NetLogo software in a publication, we ask that you include the citations below.

For the cellular automata model:

* Zimbres, R.A., Oliveira, P.P.B. (2009). Dynamics of Quality Perception in a Social Network: A Cellular Automaton Based Model in Aesthetics Services. Electronic Notes in Theoretical Computer Science, Elsevier, Volume 252, 1, Pages 157–180. https://www.sciencedirect.com/science/article/pii/S1571066109003740

Please cite the NetLogo software as:

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

COPYRIGHT AND LICENSE