Available online at www.sciencedirect.com
ScienceDirect
Applied Mathematics and Computation 201 (2008) 371–377
APPLIED MATHEMATICS AND COMPUTATION
www.elsevier.com/locate/amc

An agent-based computational study of wealth distribution in function of resource growth interval using NetLogo

Romulus-Catalin Damaceanu

Petre Andrei University of Iasi, Faculty of Economics, Research Economic Department, Gica Voda 13, Iasi 700400, Romania

Abstract

We describe an agent-based computational model that simulates the distribution of wealth in three classes: upper, middle and lower. The experimental data show us that: (1) the wealth of economy based on renewable resources is increasing if the resource growth interval is decreasing with the condition that the other factors remained unchanged; (2) the wealth of an economy based on renewable resources is higher in comparison with the wealth of an economy based on nonrenewable resources. This conclusion stresses the fact that the global economy must focus on using renewable resources because this approach may increase the global wealth.

Keywords: Simulation software; Agent-based computational model; Computational experiments

1. Introduction

The development of theory and applications of multi-agent systems determined in the last years a real revolution regarding the modeling of complex economic systems [1–8]. Most of the systems analyzed by Social Sciences (including Economics) are complex dynamic systems. The modeling style frequently used before the multi-agent theory was based on equations. This way of modeling was firstly used in Physics in the 18th century and was gradually extended in others disciplines, including Economics. The computable models based on equations have as starting point the Walrasian equilibrium model devised by the 19th century French economist Leon Walras (1834–1910) [9]. This type of computable models are limited by hypothesis and conditions formulated in order to solve the equations [10–12].

Agent-based modeling is an alternative approach of complex systems not opposed to equation-based modeling. These two approaches can be combined for modeling economic complex systems. Researchers can now model a large variety of complex phenomena associated with market economies. A part of these tools are given by agent-based modeling that uses computational methods to study economies in the frame of some controlled experimental conditions [13].

In the frame of this paper, we are going to use an agent-based model of wealth distribution implemented in NetLogo that is a modeling environment based on agents destined for simulations of natural and social

E-mail address: romulus_catalin_damaceanu@yahoo.com

0096-3003/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2007.12.042

phenomena. Uri Wilensky designed it in the year 1999 and it is in a process of development and modernization in the frame of Center for Connected Learning and Computer-Based Modeling – Northwestern University, Illinois, USA. NetLogo is written in Java language and can be run on all major platforms (Mac, Windows, Linux, etc.). In addition, individual models can be run as Java applets inside web pages. NetLogo is freeware and can be downloaded from the next web address: http://ccl.northwestern.edu/netlogo/.

Netlogo uses three types of agents: turtles, patches and observer [14]. Turtles are agents that are moving inside the world. The world is a bi-dimensional lattice composed by patches. The observer does not have a specific location – we can imagine it like an entity that observes the world composed by turtles and patches.

When NetLogo is run first time, there are no turtles. The observer can create new turtles. In addition, the patches can do the same thing. Turtles and patches have coordinates determined by the variables *xcor* and *ycor* for turtles and, respectively, *pxcor* and *pycor* for patches. The patch with the coordinate $(0,0)$ is called origin. *pxcor* is growing ($pxcor > 0$) if we are moving to the right and is dropping ($pxcor < 0$) if we are moving to the left in comparison with the origin. In the same time, *pycor* is growing ($pycor > 0$) if we are moving up and is dropping ($pycor < 0$) if we are moving down in comparison with the origin. The total number of patches is determined by the parameters *min-pxcor*, *max-pxcor*, *min-pycor* and *max-pycor*.

In NetLogo, commands and reporters tell agents what to do. A command is an action that an agent must execute. A reporter calculates a result and reports it. NetLogo uses three types of variables: global variables, turtles variables, patches variables and system variables. The first type of variables may be accessed by any type of agent. The next two types can be accessed only by the agent inside whom the variables were created. The last type of variables is predefined by NetLogo. Examples of system variable are the next: *color* (sets the color of turtle), *pcolor* (sets the color of patch), *xcor*, *ycor*, *heading* (sets the orientation in space of turtles), *pxcor*, *pycor*, etc.

In NetLogo, you have the choice of viewing models found in the Models Library, adding other models to existing ones, or creating your own models. The NetLogo interface was designed to meet all these needs. The interface is divided into two main parts: NetLogo menus and the main NetLogo window. The main window is divided into the next tabs: "Interface", "Procedures" and "Information". Only one tab at a time can be visible, but you can switch between them by clicking on the tabs at the top of the window. Right below the row of tabs is a toolbar containing a row of buttons. The available buttons vary from tab to tab. The "Interface" tab is where you watch the running of model. This tab also has tools that you can use to inspect and alter what's going on inside the model. When you first open NetLogo, the "Interface" tab is empty except for the View, where the turtles and patches appear, and the Command Center, which allows you to enter NetLogo commands.

The "Procedures" tab is the workspace where the code for the model is stored. The commands that you only want to use immediately go in the Command Center; the commands that you want to save and use later, over and over again, are found in the "Procedures" tab.

The "Information" tab provides an introduction to the model and an explanation of how to use it, things to explore, possible extensions, and NetLogo features.

The rest of the paper is organized as follows: Section 2 describes the model of wealth distribution, Section 3 presents the computational experiments done with this model and Section 4 presents the conclusions.

2. Description of computational model

The model described in this section simulates the distribution of wealth and it is based on the model of Wilensky [15] and gives to this the next improvements:

(i) it uses resources instead of grain;
(ii) the turtles can inherit the wealth of their parents;
(iii) the model has an additional switch *renewable-resources* $= \{true, false\}$; in the case when *renewable-resources* $= true$ then the resources are renewable and these can be regenerated in function of resource growth interval, otherwise these are nonrenewable and cannot be regenerated.

Each patch has an amount of resources and a certain resources capacity (the amount of resources it can grow). The turtles collect resources from the patches and process the resources in order to survive. How much resources each turtle accumulates is his or her wealth.

The model has two control parameter called *renewable-resources* and *resource-growth-interval* that describes how quickly the resources are growing. This last parameter has values between 1 and 10.

In Fig. 1, there is described the structure of computational model implemented in NetLogo. The step (1) setup up the next parameters of the computational model:

– *max-resource* = 50 – maximum amount of resource any patch can hold;
– *num-people* = 500 – the number of turtles;
– *max-vision* = 15 – maximum possible vision;
– *metabolism-max* = 8 – maximum possible metabolism of turtles;
– *life-expectancy-min* = 1 – minimum life expectancy of turtles;
– *life-expectancy-max* = 75 – maximum life expectancy of turtles;
– *percent-best-land* = 25 – the percentage of best land endowed with resources;
– *num-resource-grown* = 10 – the amount of resource that grows per one patch of land.

In addition, the step (1) set up the next initial variables used by computer simulation:

– *clock* = 0 – keeps the number of simulation steps;
– *random-seed* = 47823 – the seed of the pseudo-random number generator. The seed may be any integer in the range supported by NetLogo (−2147483648 to 2147483647). The random numbers used by NetLogo are what is called "pseudo-random" that means they appear random, but are in fact generated by

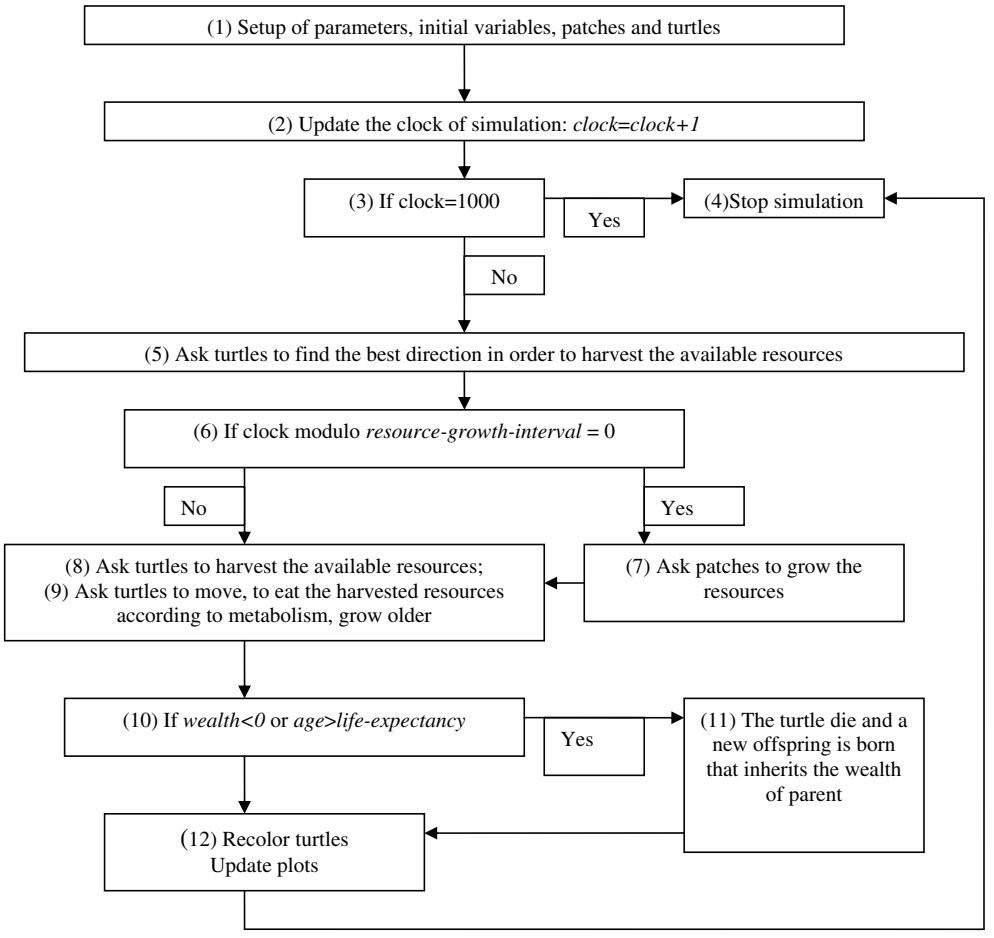

Fig. 1. The structure of computational model.

a deterministic process. "Deterministic" means that you get the same results every time, if you start with the same random "seed". If you do not set the random seed yourself, NetLogo sets it to a value based on the current date and time. If you want your model run to be reproducible, you must set the random seed before the simulation starts.

Step (2) updates the variable clock that counts the clock of simulation using the relation $clock = clock + 1$. The next step (3) checks if clock is equal with 1000. If this condition is fulfilled then the simulation is stopped using step (4). Otherwise, the step (5) asks turtles to find the best direction in order to harvest the available resources using the next algorithm – see Fig. 2:

1. turn your head in the *position* 0;
2. set *best-direction* = 0;
3. set *best-amount* = *resource-ahead*;
4. turn your head to the right in the *position* 90;
5. if (*resource-ahead* > *best-amount*) then (set *best-direction* = 90 and set *best-amount* = *resource-ahead*);
6. turn your head to the right in the *position* 180;
7. if (*resource-ahead* > *best-amount*) then (set *best-direction* = 180 and set *best-amount* = *resource-ahead*);
8. turn your head to the right in the *position* 270;
9. if (*resource-ahead* > *best-amount*) then (set *best-direction* = 270 and set *best-amount* = *resource-ahead*).

Resource-ahead is a reporter called by every turtle that computes the resources available on patches. This reporter has a variable *vision* computed using the relation: $vision = 1 + random(max\text{-}vision)$, where *random (max-vision)* is a reporter that computes a random integer number = [0, *max-vision*−1]. The algorithm used by the reporter *resource-ahead* is the next:

1. total = 0
2. how-far = 1
3. for $i = 1$ to vision
   [total = total + resource-here-of-patch-ahead how-far how-far = how-far + 1]
4. report total

The expression resource-here-of-patch-ahead how-far computes the resource found on the patch that is how-far "ahead" of the calling turtle, that is, along the turtle's current heading.

Step (6) checks if clock modulo *resource-growth-interval* = 0. If this condition is fulfilled then the patches are asked to grow the resources (step (7)) and the simulation continues with step (8). Otherwise, the simulation goes directly to the step (8) that asks turtles to harvest the available resources. Step (9) asks turtles to move, to eat the harvested resources according to their metabolism and to grow older.

Step (10) checks for every turtle if *wealth* < 0 or *age* > *life-expectancy* and if this condition is fulfilled then the simulation goes to step (11) where the turtle dies and a new offspring is born that inherits the wealth of parent. Otherwise, the simulation goes to step (12) that recolor the turtles and update the plots of the simulation. From step (12), the simulation goes to step (2).

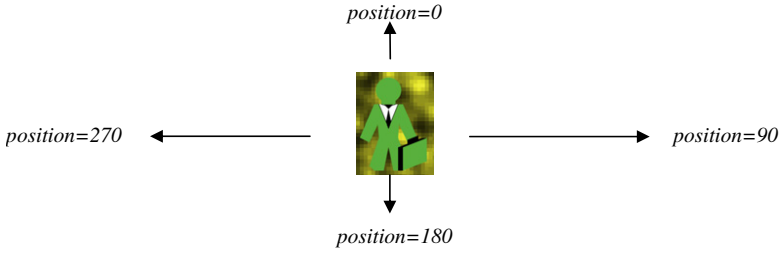

Fig. 2. The four directions of searching economic resources.

3. The computational experiments

We will do eleven computational experiments that will simulate the wealth distribution when the resources are nonrenewable and, respectively, renewable. The first experiment N01 will simulate the case when the resources are nonrenewable. The other ten will simulate the cases when resources are renewable in function of resource growth interval. To make a comparative experimental analysis, all eleven experiments will have the same initial variables and parameters with exception of control parameters *renewable-resources* = {*true*, *false*}, and resource-growth-interval that will have values ranging from 1 to 10. In addition, the simulation period will be the same: from *clock* = 1 to *clock* = 1000. In Table 1, we have the results of the computational experiments.

In order to make a quantitative analyze of the economic resources distribution, we are going to calculate an economic indicator called *Wealth* using the next formula:

$Wealth_{EXP} = N_{low} \cdot 1 + N_{middle} \cdot 2 + N_{up} \cdot 3$, where N_{low} is the number of turtles of lower class, N_{middle} is the number of turtles of middle class and N_{up} is the number of turtles of upper class.

Analyzing data of Table 1, we can see that the richest economy is that of *R02* with *Wealth* = 1247. The control parameters of this experiment have the values: *renewable-resources* = *true*, *resource-growth-interval* = 2. The second place is obtained in the case of *R01* with *Wealth* = 1243. For this experiment the control parameters have the values: *renewable-resources* = *true*, *resource-growth-interval* = 1. The poorest economy is that of experiment *N01* with *Wealth* = 510 when *renewable-resources* = *false*.

In addition, Table 1 shows us that the indicator *Wealth* has values that are descending in function of the growing value of the control parameter *resource-growth-interval* as we can see in Fig. 3.

Table 1
The results of computational experiments

Experiments	Number of turtles			Wealth
	Lower class	Middle class	Upper class	
N01	491	7	2	510
R01	13	231	256	1243
R02	18	217	265	1247
R03	24	275	201	1176
R04	40	305	155	1114
R05	61	327	112	1051
R06	103	323	74	972
R07	158	289	53	896
R08	203	250	47	844
R09	287	186	27	740
R10	349	134	16	667

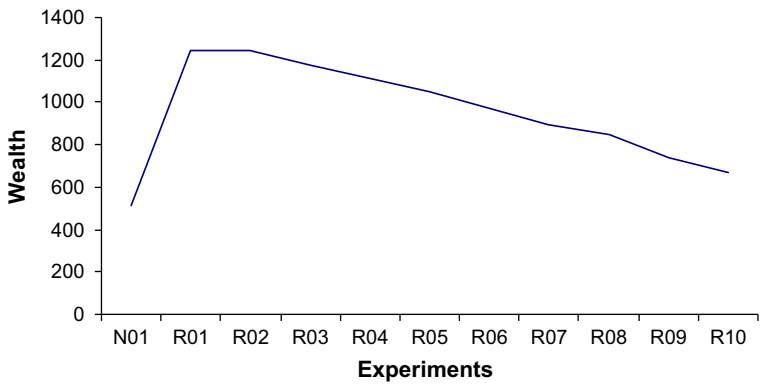

Fig. 3. The evolution of wealth in function of resource growth interval.

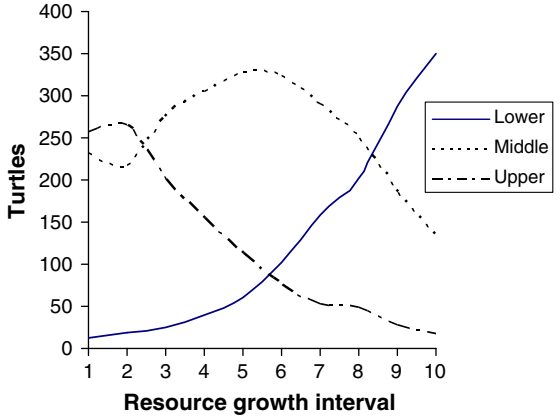

Fig. 4. The evolution of upper, middle and lower class in function of resource growth interval.

4. Conclusions

The experimental data show us that: (1) the wealth of economies based on renewable resources is increasing if the resource growth interval is decreasing with the condition that the other factors remained unchanged; (2) the wealth of economies based on renewable resources is higher in comparison with the wealth of economies based on nonrenewable resources. This evolution is explained by the distribution of turtles in the three considered classes: lower, middle and upper – see Fig. 4. Based on this figure, we observe next:

- the upper class has a descending evolution in function of resource growth interval;
- the middle class has an ascending evolution for *resource-growth-interval* $= \{1, 2, 3, 4, 5\}$ and an descending evolution for *resource-growth-interval* $= \{6, 7, 8, 9, 10\}$;
- the lower class has an ascending evolution in function of resource growth interval.

The conclusion of these computational experiments shows us that if we want to remove poverty from our present global economy, we must make efforts in order to discover ways that permit the economic development based on renewable resources. In our days, the global economy depends a lot on nonrenewable resource like petroleum, gas, coal, metals etc. If this resources will finish than the global economy will enter in collapse if the scientific community do not discover viable alternative technologies that will permit the economic development based on resources such as: the sun energy, the wind energy, bio-combustibles etc. These alternatives are used even in our days, but the technologies used are in incipient stages and must be improved. In the moment when these technologies will be perfected then the global economy will enter in a new age when we believe that the poverty will be reduced in comparison with our days.

References

[1] W.B. Arthur, S.N. Durlauf, D.A. Lane (Eds.), The economy as an evolving complex system II, in: SFI Studies in the Sciences of Complexity, Reading, Proceedings vol. XXVII, Addison-Wesley, Reading, MA, 1997.
[2] D. Batten, Discovering artificial economics: How Agents Learn and Economies Evolve, Westview Press, Boulder, CO, 2000.
[3] D. Day, P. Chen, Nonlinear Dynamics and Evolutionary Economics, Oxford University Press, Oxford, UK, 1993.
[4] J.M. Epstein, R. Axtell, Growing Artificial Societies: Social Science from the Bottom Up, MIT Press, Cambridge, MA, 1996.
[5] J. Holland, Adaptation in Natural and Artificial Systems, The MIT Press, Cambridge, MA, 1992.
[6] P. Krugman, The Self-organizing Economy, Blackwell Publishers, Cambridge, MA, 1996.
[7] T. Sargent, Bounded Rationality in Macroeconomics. The Arne Ryde Memorial Lectures, Clarendon Press, Oxford, UK, 1993.
[8] H.P. Young, Individual Strategy and Social Structure, Princeton University Press, Princeton, NJ, 1998.
[9] L. Walras, Elements of Pure Economics, Irwin, Homewood, 1954.
[10] K.J. Arrow, G. Debreu, Existence of an equilibrium for a competitive economy, Econometrica (22) (1964) 265–290.

[11] For an explanation of the problems and for the computer coding needed to derive a numerical solution even for the most simple general equilibrium computable models, see C.L. Dinwiddy, F.J. Teal, The Two-Sector General Equilibrium Model: A New Approach, Philip Allan, Oxford, 1988.
[12] For a discussion of the computational problems involved in solving computational general equilibrium, see J.B. Shoven, J. Whaley, Applying General Equilibrium, Cambridge University Press, Cambridge, 1992.
[13] L. Tesfatsion, J. Kenneth (Eds.), Handbook of Computational Economics, Agent-Based Computational Economics, vol. 2, Elsevier/North-Holland, 2006 ((Handbooks in Economics Series), see for a handbook regarding agent-based computational economics).
[14] For details about how to use NetLogo see: http://ccl.northwestern.edu/netlogo/docs/NetLogo%20User%20Manual.pdf.
[15] U. Wilensky, NetLogo Wealth Distribution model. http://ccl.northwestern.edu/netlogo/models/WealthDistribution. Center for Connected Learning and Computer-Based Modeling, 1998.