A COMPARATIVE ANALYSIS OF THREE MULTI-AGENT COMPUTATIONAL ALGORITHMS USED TO HARVEST GRAIN

R. C. DĂMĂCEANU[1]

[1] "Petre Andrei" University, Iasi
e-mail: romulus_catalin_damaceanu@yahoo.com

The development of theory and applications of multi-agent systems determined in the last years a real revolution regarding the modeling of complex systems. The structure of any agent-based computational model contains the next elements: parameters, variables, algorithms and agents. We make a comparative analysis of three multi-agent computational algorithms (RAND, NMEM and WMEM) used to harvest grain in a bi-dimensional lattice. The first algorithm RAND is the simplest because the agent sets randomly a certain harvesting direction and harvests all the grain he can find. The second algorithm NMEM uses 8 searching directions. From these 8 variants, the agent selects the alternative that gives the maximum amount of grain. The searching is repeated for vision times. If this search fails to find any grain than the agent pass to RAND algorithm. The third algorithm WMEM uses the same searching algorithm as NMEM algorithm, but this one memorizes the patches that the agents have discovered. When this searching algorithm fails to find grain, than the agent uses its memory to find the nearest patch with available grain. The algorithms are implemented using NetLogo. This software platform was designed by Uri Wilensky it in the year 1999. NetLogo 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. We did three computational experiments and we observe that the best results are obtained when we used WMEM algorithm. In this case, the grain was harvested in a period of 1141 simulation steps. On the second place was NMEM algorithm with a harvesting period of 6982 simulation steps and on the last place was RAND algorithm with a harvesting period of 18183 simulation steps.

***Key words**: agent-based computational economics, computational algorithms used to harvest grain, computational experiments*

The development of theory and applications of multi-agent systems determined in the last years a real revolution regarding the modeling of complex systems [1-10]. The structure of any agent-based computational model contains the next elements:

a) *Parameters* have numerical values that do not modify on the entire period of simulation. Normally, parameters are initialized before simulation. However, in some situations they can be changed during the simulation.

b) Variables are labels that have a number of values during the simulation.

c) *Algorithms* are a finite list of well-defined instructions for accomplishing some task that, given an initial state, will proceed through a well-defined series of successive states, possibly eventually terminating in an end-state.

d) *Agents* encapsulate all three elements discussed above and are actors in a model that (generally) solve an optimization problem.

In order to implement an agent-based computational model you need a software platform. A simple and accessible platform for creating agent-based models is NetLogo [11]. A similar platform, StarLogo, has also been released with similar functionality - see http://education.mit.edu/starlogo-tng/. For Java programmers, we have Ascape - see http://ascape.sourceforge.net/. Another software package is LSD that has user-friend interface and can be used with very good results - see http://www.business.aau.dk/lsd/ [5].

The main construction blocks of any agent-based computational model are the next: the set of agents (*A*), the initializations (*I*) and simulation specifications (*R*). The set of agents *A* contain all *agents* defined as artificial entities encapsulating the next elements: parameters, variables and algorithms. Initializations *I* are a set of identities that have in left side the variable or parameter name and in the right side the associated value. The simulation specifications *R* are a set of identities that have in the left side the control parameter name and in the right side the associated value.

Under these circumstances an agent-based computational model (*ACM*) can be defined as a list of three of three arguments: the set of agents (*A*), the initializations (*I*) and simulation specifications (*R*). Shortly, *ACM = (A, I, R)*.

In the frame of this paper, we are going to use three multi agent-based computational algorithms of searching grain. The first algorithm *RAND* is the simplest because the agent sets randomly a certain harvesting direction and harvests all the grain he can find. The second algorithm *NMEM* uses 8 searching directions. From these 8 variants, the agent selects the alternative that gives the maximum amount of grain. The searching is repeated for *vision* times. If this search fails to find any grain than the agent pass to *RAND* algorithm. The third algorithm *WMEM* uses the same searching algorithm as *NMEM* algorithm, but this one memorizes the patches that the agents have discovered. When this searching algorithm fails to find grain, than the agent uses its memory to find the nearest patch with available grain.

All these three searching algorithms are implemented using NetLogo. This software platform was designed by Uri Wilensky it in the year 1999. NetLogo 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 [11].

Netlogo uses three types of agents: turtles, patches and observer. For details about how to use NetLogo see [11]. Turtles are agents that are moving inside the world. The world is bi-dimensional and it is 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.

MATERIAL AND METHOD

The algorithm used by the observer has 6 steps. Step (1) of the observer algorithm declares the next global variables and parameters:

(i) *clock* – is a variable that keeps the number of simulation steps;

(ii) *vision* – is a parameter that measure how far the agent can see and evaluate the available resources that surrounds him

(iii) *proc-res* – is a parameter that keeps the percentage of resources of the bi-dimensional world;

(iv) *mean-res* – is a variable that keeps the average level of resources;

(v) *wealth* - the amount of resources harvested by the turtle;

(vi) algorithm – the algorithm used by turtles: “random”, “no memory” and “with memory”;

(vii) *mode* - the mode of the turtle: random, finding not using memory and finding with memory.

In addition, the same step (1) declares the next patch-own and turtles-own variables:

(i) *res* - keeps the amount of resource that the patch holds;

(ii) *memory* - a variable that has the next possible values: “false" if the turtle did not memorize the amount of res variable, "true" if the the turtle memorize the amount of res variable;

(iii) *best-direction* - the best direction of turtle computed using algorithm *A1*;

(iv) *element* - this variable has 8 possible values 0, 1, 2, 3, 4, 5, 6, 7 and is used by the algorithm *A2*

(v) *best-element* - this variable keeps the best variant from the eight possible and is used by the algorithms “no memory” and “with memory”.

The same step (1) sets the initial values for parameters and variables of the computational model – see *Table 1*.

Table 1

The initial values for parameters and variables

Label	Type	Initial value
clock	global variable	0
random-seed	system variable	0
proc-res	global parameter	30
mean-res	global variable	0
wealth	global variable	0
res	patch variable	random value between 1 and 100
memory	patch variable	“false”
best-direction	turtle variable	0
element	turtle variable	0
best-element	turtle variable	0

The observer algorithm continues with updating the variable clock (step (2)) and asks the turtle to harvest resources (step (3)) using one of the three available algorithms.

The observer algorithm goes to the step (4) that computes the average level of resources *mean-res* using the formula:

mean-res = $\frac{\sum_{i \in patches} res_i}{count(patches)}$, where *count(patches)* computes the total number of patches of the bi-dimensional lattice.

The step (5) of the observer algorithm checks if the global variable *mean-res* is equal with zero. If this condition is fulfilled then the simulation is stopped using step (6). Otherwise the algorithm goes to step (2).

RESULTS AND DISCUSSIONS

We will do 3 computational experiments that will simulate the harvesting of resources in a bi-dimensional lattice of 51x51 patches. The total amount of *GRAIN* for all 30 experiments is the same and is computed using the next formula: *RES* = $\sum_{i \in patches} grain_i$ =608537. The first computational experiment will use RAND algorithm, the second will use NMEM algorithm and the third will use WMEM algorithm. We are going to use five turtles that have the task to harvest all the grain from the bi-dimensional lattice. All the turtles will start the harvesting from the origin point of lattice. The turtle 0 will have a vision of 6 patches, turtle 1 will have a vision of 7 patches, turtle 2 will have a vision of 8 patches, turtle 3 will have a vision of 9 patches and turtle 4 will have a vision of 10 patches. The results of computational experiments are presented in fig. 1, 2 and 3.

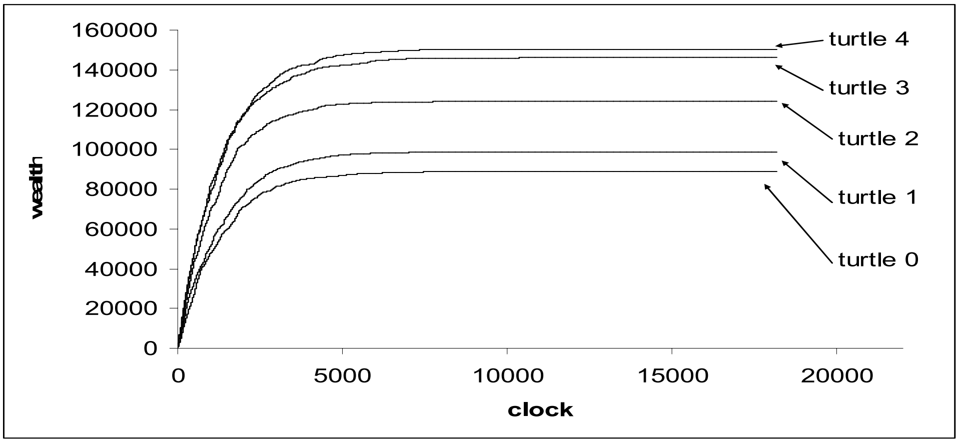

Figure 1 **The evolution of wealth in the case of RAND algorithm**

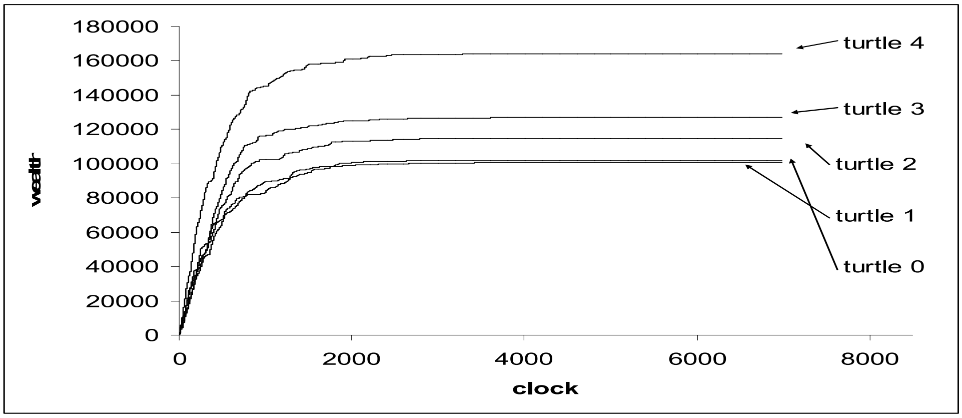

Figure 2 **The evolution of wealth in the case of NMEM algorithm**

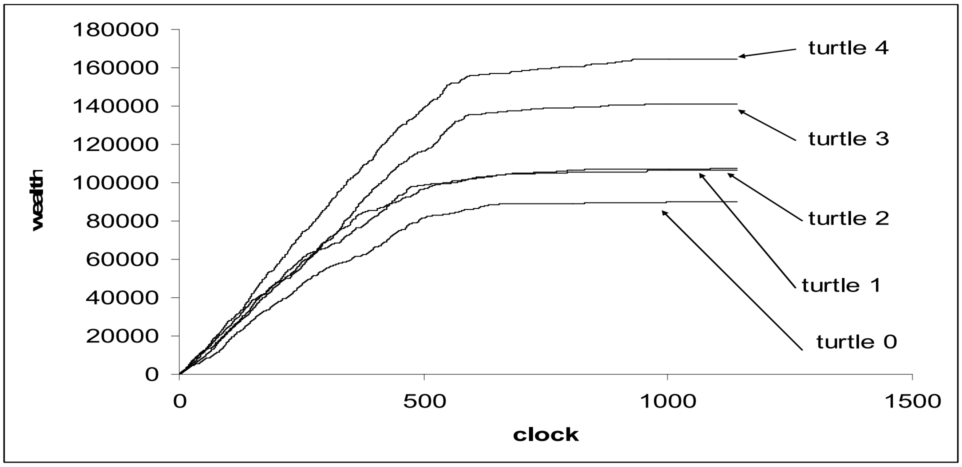

Figure 3 **The evolution of wealth in the case of WMEM algorithm**

CONCLUSIONS

As *Fig. 1, 2* and *3* show us, we observe that the best results are obtained when we used WMEM algorithm. In this case, the grain was harvested in a period of 1141 simulation steps. On the second place was NMEM algorithm with a harvesting period of 6982 simulation steps and on the last place was RAND algorithm with a harvesting period of 18183 simulation steps.

BIBLIOGRAPHY

1. Arthur ,W. B., Durlauf, S. N., Lane, D. A., (Eds.), 1997 - *The economy as an evolving complex system II*, in *The Sciences of Complexity, Reading, Proceedings Vol. XXVII*, Addison-Wesley, Reading, MA.
2. Batten, D., 2000 - *Discovering artificial economics: How Agents Learn and Economies Evolve*, Westview Press, Boulder, CO.
3. Day, D. , Chen, P., 1993 - *Nonlinear Dynamics and Evolutionary Economics*, Oxford University Press, Oxford, UK.

4. Damaceanu ,R. C., 2007 - *An agent-based computational study of wealth distribution in function of resource growth interval using NetLogo*, Applied Mathematics and Computation, (201), 371-377.
5. Damaceanu, R. C.,2007- *Implementation of simulation model of world economy using LSD*, Applied Mathematics and Computation (189), 1011-1024.
6. Epstein ,J. M.,, Axtel,l R., 1996 - *Growing Artificial Societies: Social Science from the Bottom Up*, MIT Press, Cambridge, MA.
7. Holland, J.,1992 – *Adaptation in Natural and Artificial Systems*, The MIT Press, Cambridge, MA.
8. Krugman, P., 1996 – *The self-organizing economy, Blackwell Publishers*, Cambridge, MA, 1996
9. Sargent, T., 1993 – *Bounded Rationality in Macroeconomics. The Arne Ryde Memorial Lectures*, Clarendon Press, Oxford, UK.
10. Young, H. P., 1998 – *Individual Strategy and Social Structure*, Princeton University Press, Princeton, NJ.
11. Wilensky, U., 1999 – *NetLogo. http://ccl.northwestern.edu/netlogo*/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.