A General *N*-Person Game Solver for Pavlovian Agents

Miklos N. Szilagyi
Department of Electrical and Computer Engineering
University of Arizona, Tucson, AZ 85721

A new *N*-person game solver is presented in this paper. It has superior capabilities in at least two respects: (1) the payoff functions can be represented by arbitrary functions, even in parameterized form; and (2) it is very easy to investigate the multidimensional parameter space of any model. Several interesting experiments are also presented.

1. Introduction

Game theory [1] is a well-developed discipline, but relatively little effort has been devoted to the theoretical investigation of *N*-person games [2–7]. "The Tragedy of the Commons" [8] pointed to the practical importance of such games.

Axelrod's famous tournaments [9] have generated an enormous interest in *N*-person games. Unfortunately, however, these tournaments are not really *N*-person games but series of two-person games among *N* participants. Nevertheless, the interest is still alive and many papers are devoted to the investigation of various two-person game strategies that could be used in such tournaments.

Agent-based simulation is a powerful tool for the study of simultaneous operation of a large number of participants to investigate complex phenomena. Therefore, it is quite suitable for the investigation of *N*-person games.

We represent the participating players as autonomous agents that interact with all the other agents directly or indirectly (through their neighbors) [10]. In the latter case, the agents are simply cells in a cellular automaton [11–13].

We developed an agent-based simulation tool [14] that has been successfully applied to the solution of a large number of various *N*-person games and practical problems [10]. Our software is a genuine multi-agent model and not a model for repeated two-person games. It is a general framework for inquiry in which the properties of the environment as well as those of the agents are user-defined parameters, and the number of interacting agents is theoretically unlimited. There are, however, two issues that need attention. First, the

Complex Systems, **24** © 2015 Complex Systems Publications, Inc.
https://doi.org/10.25088/ComplexSystems.24.3.261

software is only capable of handling linear and quadratic payoff functions. Practical problems sometimes require more sophisticated inputs, for example, exponential, sigmoid, or periodic functions. Second, in order to investigate the multidimensional parameter space of a model, the user must modify the configuration file and run each configuration separately, which is time-consuming work.

We present a new software tool here that solves these problems.

2. The Model

We assume that the game is uniform and iterated and that the agents have no goals, know nothing about each other, and cannot refuse participation in any iteration. They are distributed in and fully occupy a finite two-dimensional space, and the updates are simultaneous. They must choose between two options. In accordance with the accepted notation, we call these options cooperation and defection, but they can mean anything else as well.

The Pavlovian model is chosen for the representation of agents: the probability of choosing the previously chosen action again changes by an amount proportional to the reward/penalty for the previous action; the coefficient of proportionality is called the learning rate (of course, the probabilities always remain in the interval between 0 and 1). This decision heuristic is based on Pavlov's experiments and Thorndike's law [15]: if an action is followed by a satisfactory state of affairs, then the tendency of the agent to produce that particular action is reinforced. These agents are primitive enough not to know anything about their rational choices, but they have enough "intelligence" to learn a behavior according to this law. Their behavior is not determined but only shaped by its consequences; that is, an action of the agent will be more probable but still not certain after a favorable response from the environment. Kraines and Kraines [16], Macy [17], Flache and Hegselmann [18], and others used such agents for the investigation of iterated two-person games. Indeed, rationality assumes that incredibly smart participants make decisions in unbelievably simple situations. In real life, just the opposite is true. Therefore, the simple assumption that what works well is likely to be repeated and what turns out poorly is likely to be changed is a much better representation of reality.

The probabilities of the agents' actions are updated by the reward/penalty received from the environment based on their and other agents' behavior. Actions are taken according to these probabilities. The outputs of the stochastic environment are influenced by the actions of all participating agents. Behavior is learned by adjusting the

https://doi.org/10.25088/ComplexSystems.24.3.261

action probabilities to the responses of the environment. The learning capability alters the agents' behavior as they make repeated decisions. The aggregate behavior of the society of agents usually converges to a stable or oscillating state.

Our simulation environment is a two-dimensional array of the participating agents. The size of this environment is a user-defined parameter. The limiting case of just two agents makes the investigation of two-person games possible.

The agents may interact with all other agents simultaneously or through their immediate neighbors as cellular automata.

It is possible to wrap the array of agents in either the horizontal or vertical direction or both directions. If there is no wrapping, it is possible to simulate cities or other confined environments.

The payoff (reward/penalty) functions are given as two curves: $C(x)$ for cooperators and $D(x)$ for defectors. The payoff to each agent depends on its choice and on the distribution of other players among cooperators and defectors. The payoff curves are functions of the ratio x of cooperators to the total number of agents or neighbors.

In an iterative game, the aggregate cooperation proportion changes in time, that is, over subsequent iterations. The outcome of the game depends on the values of at least the following parameters:

- number of rows and columns in the array of agents
- depth of the neighborhood for each agent
- payoff curves for cooperators and defectors
- initial actions of individual agents at various locations in the array
- initial states of individual agents at various locations in the array

If the parameters are selected appropriately, the simulations will exhibit behavior that is close enough to the behavior of real agents when they are placed in a similar situation.

3. Implementation

We have chosen NetLogo Version 5.0.5 [19] for the implementation of this project. NetLogo is a programmable modeling environment for simulating natural and social phenomena. It was authored by Uri Wilensky in 1999 and has been in continuous development ever since at the Center for Connected Learning and Computer-Based Modeling at Northwestern University. It is particularly well suited for modeling complex systems developing over time. Modelers can give instructions to a large number of agents all operating independently. This makes it

https://doi.org/10.25088/ComplexSystems.24.3.261

possible to explore the connection between the micro-level behavior of individuals and the macro-level patterns that emerge from their interactions.

NetLogo was previously used for the simulation of a prisoner's dilemma game of greedy agents with $C(x) = x$ and $D(x) = px$, where p is the only parameter of the simulation except for the initial cooperation ratio [20]. This software cannot be used for any other simulation.

The most important feature of NetLogo is that user interfaces are built with the software. First we choose a computational grid and determine the number of rows and columns, the size of each cell, and horizontal or vertical wrapping if we need them. Then we create buttons, sliders, plots, switches, and other interface tools by elementary commands.

We used four buttons, two switches, five sliders, one monitor, and one plot in this model. The *setup* button sets the program up according to the setting of the initial cooperation ratio x_0 with its *slider* and its alterations by using the mouse after clicking the *manual* button. The parameters of the payoff functions are set by the other four sliders (more about this later). The *cellular* switch controls whether the agents interact with all other agents simultaneously or through their immediate neighbors as cellular automata. The user can ask for a four-color display, which shows the current and previous action for each agent, or a black-and-white display, which shows only their current actions by using the *colors* switch. We start the program by clicking either the *go* or the *go once* button. The former continuously runs the program until we stop it by clicking the button again. The latter produces just one iteration at a time so that it is possible to watch the result of each subsequent iteration. The cooperation ratio x as a function of the number of iterations is shown in the *plot*. We can also watch the change of the numerical value of x in the *monitor*. This makes it possible to immediately see the final value of x when we stop the simulation. The interface is shown in Figure 1 for the case of $41 * 41 = 1681$ agents with $C(x) = \sin(5x)$, $D(x) = \cos(5x)$, $x_0 = 0.902$ after the 1062^{nd} iteration. The equilibrium of the cooperation ratio was reached before the 100^{th} iteration at $x_{\text{final}} = 0.572$. The figure also shows the graphics output at this iteration. Cooperator agents are black; defector agents are white.

The code consists of procedures that contain commands and reporters. Once we define a procedure, we can use it anywhere in the code. Each button has a corresponding procedure. The entire code does not exceed 100 lines.

https://doi.org/10.25088/ComplexSystems.24.3.261

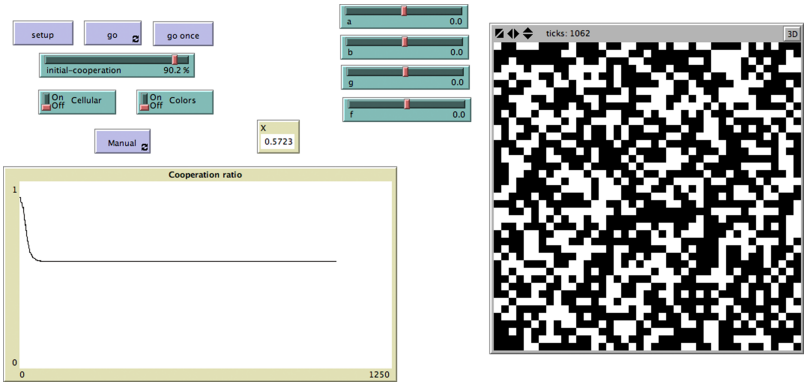

Figure 1. The NetLogo interface for the case of $41 * 41 = 1681$ agents with $C(x) = \sin(5x)$, $D(x) = \cos(5x)$, $x_0 = 0.902$ after the 1062nd iteration. Cooperator agents are black; defector agents are white.

The payoff functions are determined by any formula using combinations of basic mathematical functions. For example, $\sin\{\exp[5x / \cos(2x)]\}$ is an acceptable representation. The freedom of using arbitrary functions for the determination of the payoff functions makes it possible to simulate any N-person game. In addition, the payoff functions can be represented in parameterized form. The parameters can easily be changed by their sliders, and they can also be used for the investigation of the multidimensional parameter space of any model. The experimenter simply determines which parameters should be varied within given ranges and increments. The software will run the simulation with each set of possible values and record the results. It can perform sophisticated experiments. The results are presented in the form of a table or spreadsheet. It is also possible to vary any other parameter, including the random seed used in the code.

The updated probabilities lead to new decisions by the agents. With each iteration, the software tool draws the array of agents in the graphics output, with each agent in the array colored according to its most recent (and previous) action. The experimenter can view and record the evolution of the behavior of any agent and of the society of agents as they change in time.

We can obtain more detailed information about individual agents. The experimenter selects the agent to be examined in detail by pointing at it with the mouse. This information includes the agent's coordinates in the array, its color, its two most recent actions, its probabilities of cooperation, and the last reward or punishment that the agent received. When the experimenter stops the simulation, the history

https://doi.org/10.25088/ComplexSystems.24.3.261

of aggregate cooperation proportions for each iteration is presented as a list of numerical values as well as an automatically generated plot. The iterations are usually stopped by the user when the cooperation ratio reaches equilibrium or starts to oscillate around a constant value. The speed of the iteration can be controlled by the built-in speed slider. Naturally, the running time depends on many factors, mostly on the number of agents chosen. An iteration takes only a fraction of a second, even for tens of thousands of agents.

4. Verification

The proper operation of the software was verified by repeating the simulations published in papers [14, 21–26]. We obtained the same results with the new software.

5. Experiments

5.1 Refinement of a Prisoner's Dilemma Simulation

We investigated in detail a prisoner's dilemma with $C(x) = -1 + 2x$ and $D(x) = -0.5 + 2x$ in [22]. If our formula

$$xC(x) = \big(1 - x\big)D(x) \tag{1}$$

has two solutions in the region $0 < x < 1$, then the smaller (x_1) is a stable attractor, and the greater (x_2) is an unstable repulsor. In this case, $x_1 = 0.180$ and $x_2 = 0.695$. Accordingly, when $x_0 < x_2$, the solution of the game converges toward x_1 as an oscillation while it stabilizes at different x_{final} values when $x_0 > x_2$. This was proven by simulations with the values of x_0 chosen as 0.00, 0.65, 0.69, 0.71, 0.73, 0.75, 0.80, and 0.90. The most interesting region, however, is between $x_0 = 0.69$ and $x_0 = 0.71$, where the drastic change occurs.

It is very easy to investigate this region with the new software by automatically incrementing the value of x_0 by 0.001 between 0.69 and 0.71. The values of x_{final} as a function of x_0 are shown in Figure 2. We see that the drastic change occurs at $x_0 = 0.694$. Figures 3 and 4 show the evolution of the game for the first 100 iterations when $x_0 = 0.69$ and 0.71, respectively.

https://doi.org/10.25088/ComplexSystems.24.3.261

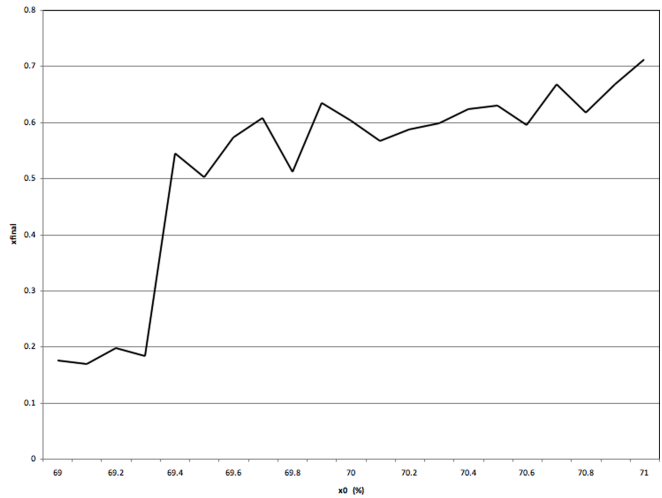

Figure 2. The values of x_{final} as a function of x_0 in the region $0.69 < x_0 < 0.71$ for the game with $C(x) = -1 + 2x$ and $D(x) = -0.5 + 2x$.

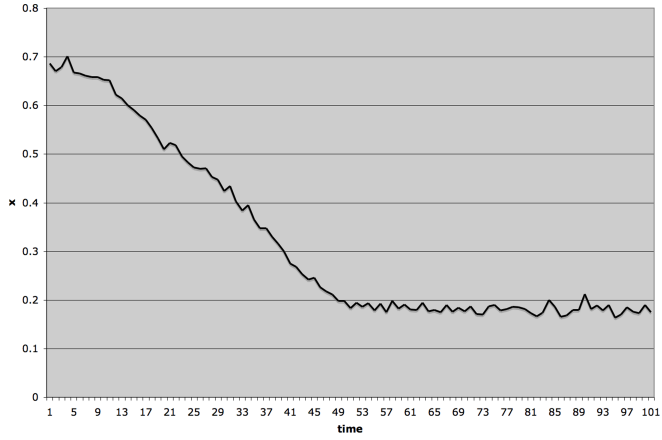

Figure 3. The evolution of the game with $C(x) = -1+2x$ and $D(x) = -0.5+2x$ for the first 100 iterations when $x_0 = 0.69$.

https://doi.org/10.25088/ComplexSystems.24.3.261

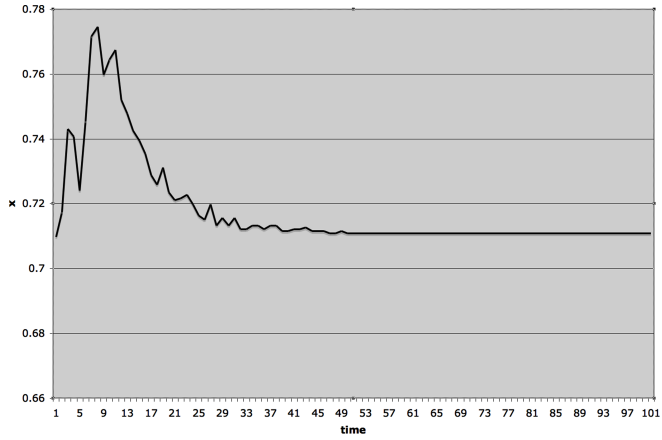

Figure 4. The evolution of the game with $C(x) = -1 + 2x$ and $D(x) = -0.5 + 2x$ for the first 100 iterations when $x_0 = 0.71$.

5.2 Periodic Payoff Functions

In this experiment we chose the payoff functions as $C(x) = \sin(5x)$ and $D(x) = \cos(5x)$, as shown in Figure 5.

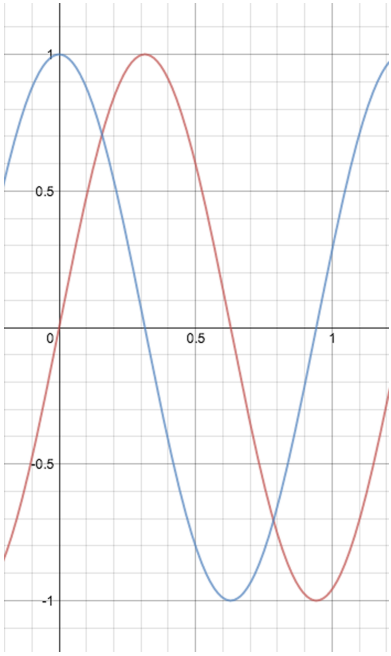

Figure 5. The payoff functions $C(x) = \sin(5x)$ and $D(x) = \cos(5x)$.

https://doi.org/10.25088/ComplexSystems.24.3.261

When x_0 is less than 0.85, this game ends with universal defection. However, at $x_0 = 0.85$ it starts growing with x_0 and eventually reaches total cooperation. The values of x_{final} as a function of x_0 in the region $0.85 < x_0 < 1.00$ for this game are shown in Figure 6.

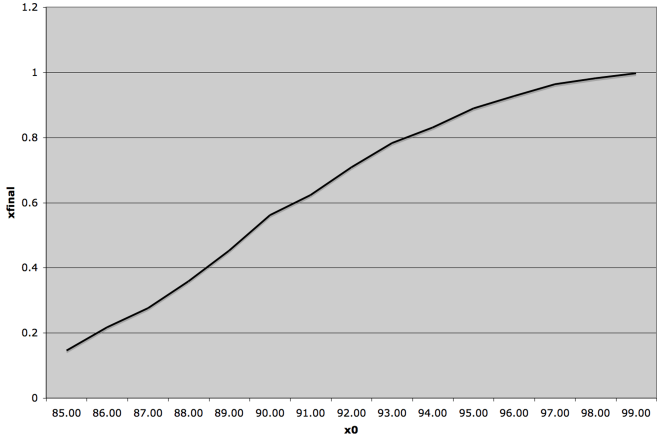

Figure 6. The values of x_{final} as a function of x_0 in the region $0.85 < x_0 < 1.00$ for the game with $C(x) = \sin(5x)$ and $D(x) = \cos(5x)$.

5.3 Exponential Payoff Functions

When the payoff functions are expressed as $C(x) = \exp(x)$ and $D(x) = \exp(-x)$ (Figure 7), the values of x_{final} grow with x_0 and reach total cooperation at $x_0 = 0.6$. Interestingly, if we exchange $C(x)$ and $D(x)$, the trend remains the same, with the only difference that total cooperation is reached only when x_0 reaches 1.0.

5.4 Sigmoid Payoff Functions

Sigmoid functions are important for the investigation of neural networks. We chose $C(x) = 1 \Big/ \{1 + \exp[-5(x - 0.5)]\} - 0.5$ and $D(x) = -C(x)$ (Figure 8) and found that the result is total defection when x_0 is less than 0.507 and total cooperation when it is greater than 0.507.

If we exchange $C(x)$ and $D(x)$, the final result is always about 0.5, independent of the initial conditions. This is a unique situation because the solution of N-person games usually very strongly depends on the initial condition.

https://doi.org/10.25088/ComplexSystems.24.3.261

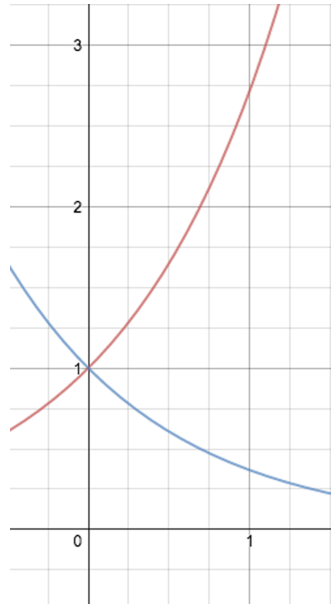

Figure 7. The payoff functions $C(x) = \exp(x)$ and $D(x) = \exp(-x)$.

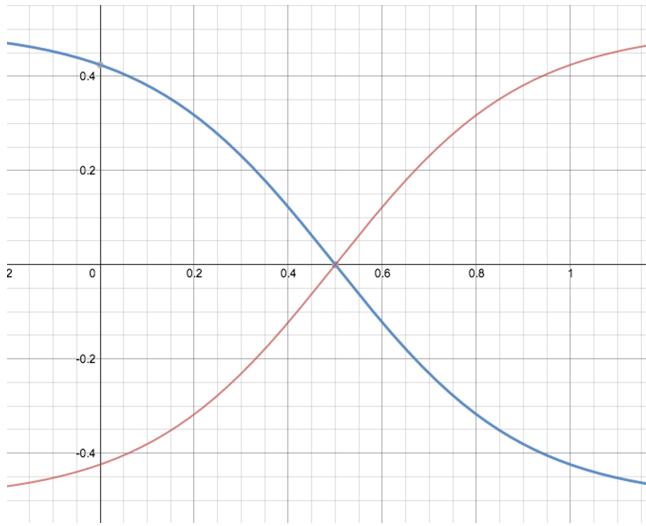

Figure 8. The sigmoid payoff functions $C(x) = 1 / \{1 + \exp[-5(x - 0.5)]\} - 0.5$ and $D(x) = -C(x)$.

https://doi.org/10.25088/ComplexSystems.24.3.261

5.5 Parameterized Payoff Functions

One of the most important features of this software is that the payoff functions can be represented in parameterized form. The parameters can easily be changed manually by moving their sliders, but they can also be used for the investigation of the multidimensional parameter space of a model. In this case, the sliders are moved automatically.

We chose linear payoff functions in the form of $C(x) = ax + b$ and $D(x) = gx + f$. This is the most general representation of linear payoff functions because the parameters a, b, g, and f can have any values. For simplicity, we present here only the cases when a and g are equal to either 1 or 2, while b and f can be -0.5 or -1. The value of x_0 is set to zero (total initial defection). This yields 16 different games. We ran this experiment for 200 iterations in each case. The results are shown in Table 1.

Run Number	b	a	f	g	Initial Cooperation Ratio	Final Cooperation Ratio	x_1	Steps
1	−0.5	1	−0.5	1	0	0.466389	0.5	200
2	−0.5	1	−0.5	2	0	0.208209	0.211	200
3	−0.5	1	−1	1	0	1	complex	200
4	−0.5	1	−1	2	0	0.502082	0.5	200
5	−0.5	2	−0.5	1	0	0.988102	complex	200
6	−0.5	2	−0.5	2	0	0.240928	0.25	200
7	−0.5	2	−1	1	0	1	complex	200
8	−0.5	2	−1	2	0	0.941701	complex	200
9	−1	1	−0.5	1	0	0.246282	0.25	200
10	−1	1	−0.5	2	0	0.168947	0.167	200
11	−1	1	−1	1	0	0.506246	0.5	200
12	−1	1	−1	2	0	0.338489	0.333	200
13	−1	2	−0.5	1	0	0.326591	0.333	200
14	−1	2	−0.5	2	0	0.182035	0.18	200
15	−1	2	−1	1	0	1	complex	200
16	−1	2	−1	2	0	0.48602	0.5	200

Table 1. Results of the parametrized payoff functions experiment.

Equation (1) yields two solutions in the region $0 < x < 1$ in 11 cases out of 16. As we can see, the values of x_1 and x_{final} are indeed very close to each other.

In the other five cases, the solutions of equation (1) are complex. Interestingly, the game ends at or very close to total cooperation in these cases.

https://doi.org/10.25088/ComplexSystems.24.3.261

The actual output is a spreadsheet that shows the values of x for each iteration and their minimum, maximum, and mean values for each case.

6. Conclusion

Our new N-person game solver has superior capabilities in at least two respects: (1) the payoff functions can be represented by arbitrary functions, even in parameterized form; and (2) it is very easy to investigate the multidimensional parameter space of any model. The experiments presented confirm these properties. The results are interesting, and they very strongly depend on the initial conditions.

Acknowledgments

The author appreciates the interest of Chao Meng, Rohit Chacko Philip, and Marcos Zuzuárregui in this work.

References

[1] J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*, Princeton: Princeton University Press, 1944.

[2] R. L. Weil, "The N-Person Prisoner's Dilemma: Some Theory and a Computer-Oriented Approach," *Behavioral Science*, **11**(3), 1966 pp. 227–234. doi:10.1002/bs.3830110310.

[3] A. Rapoport, *N-Person Game Theory*, Ann Arbor, MI: University of Michigan Press, 1970.

[4] H. Hamburger, "N-Person Prisoner's Dilemma," *Journal of Mathematical Sociology*, **3**(1), 1973 pp. 27–48. doi:10.1080/0022250X .1973.9989822.

[5] T. C. Schelling, "Hockey Helmets, Concealed Weapons, and Daylight Saving: A Study of Binary Choices with Externalities," *The Journal of Conflict Resolution*, **17**(3), 1973 pp. 381–428.

[6] D. J. Goehring and J. P. Kahan, "The Uniform N-Person Prisoner's Dilemma Game: Construction and Test of an Index of Cooperation," *The Journal of Conflict Resolution*, **20**(1), 1976 pp. 111–128.

[7] S. Govindan and R. Wilson, "A Decomposition Algorithm for N-Player Games," *Economic Theory*, **42**(1), 2010 pp. 97–117. doi:10.1007/s00199-009-0434-4.

https://doi.org/10.25088/ComplexSystems.24.3.261

[8] G. Hardin, "The Tragedy of the Commons," *Science*, **162**(3859), 1968 pp. 1243–1248. doi:10.1126/science.162.3859.1243.

[9] R. Axelrod, *The Evolution of Cooperation*, New York: Basic Books, 1984.

[10] M. N. Szilagyi, "Investigation of *N*-Person Games by Agent-Based Modeling," *Complex Systems*, **21**(3), 2012 pp. 201–243. http://www.complex-systems.com/pdf/21-3-4.pdf.

[11] T. Toffoli, "Cellular Automata as an Alternative to (Rather than an Approximation of) Differential Equations in Modeling Physics," *Physica D: Nonlinear Phenomena*, **10**(1–2), 1984 pp. 117–127. doi:10.1016/0167-2789(84)90254-9.

[12] B. Chopard and M. Droz, *Cellular Automata Modeling of Physical Systems*, New York: Cambridge University Press, 1998.

[13] S. Wolfram, *A New Kind of Science*, Champaign, IL: Wolfram Media, Inc., 2002.

[14] M. N. Szilagyi and Z. C. Szilagyi, "A Tool for Simulated Social Experiments," *Simulation*, **74**(1), 2000 pp. 4–10. doi:10.1177/003754970007400101.

[15] E. L. Thorndike, *Animal Intelligence: Experimental Studies*, New York: The Macmillan Company, 1911.

[16] D. Kraines and V. Kraines, "Pavlov and the Prisoner's Dilemma," *Theory and Decision*, **26**(1), 1989 pp. 47–79. doi:10.1007/BF00134056.

[17] M. W. Macy, "PAVLOV and the Evolution of Cooperation: An Experimental Test," *Social Psychology Quarterly*, **58**(2), 1995 pp. 74–87. doi:10.2307/2787147.

[18] A. Flache and R. Hegselmann, "Rationality vs. Learning in the Evolution of Solidarity Networks: A Theoretical Comparison," *Computational and Mathematical Theory*, **5**(2), 1999 pp. 97–127. doi:10.1023/A:1009662602975.

[19] U. Wilensky. "NetLogo." Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. (Jun 19, 2015). http://ccl.northwestern.edu/netlogo.

[20] U. Wilensky. "NetLogo PD Basic Evolutionary Model." Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. (Jun 19, 2015). http://ccl.northwestern.edu/netlogo/models/PDBasicEvolutionary.

[21] M. N. Szilagyi and Z. C. Szilagyi, "Non-trivial Solutions to the *N*-Person Prisoners' Dilemma," *Systems Research and Behavioral Science*, **19**(3), 2002 pp. 281–290. doi:10.1002/sres.435.

[22] M. N. Szilagyi, "An Investigation of *N*-Person Prisoners' Dilemmas," *Complex Systems*, **14**(2), 2003 pp. 155–174. http://www.complex-systems.com/pdf/14-2-3.pdf.

https://doi.org/10.25088/ComplexSystems.24.3.261

[23] M. N. Szilagyi, "Simulation of Multi-Agent Prisoners' Dilemmas," *Systems Analysis, Modelling, Simulation*, **43**(6), 2003 pp. 829–846. doi:10.1080/0232929021000055488.

[24] M. N. Szilagyi, "Cars or Buses: Computer Simulation of a Social and Economic Dilemma," *International Journal of Internet and Enterprise Management,* **6**(1), 2009 pp. 23–30. doi:10.1504/IJIEM.2009.022932.

[25] M. N. Szilagyi, "Analytical Solutions of *N*-Person Games," *Complexity*, **17**(4), 2012 pp. 54–62. doi:10.1002/cplx.21385.

[26] M. N. Szilagyi, "The El Farol Bar Problem as an Iterated *N*-Person Game," *Complex Systems*, **21**(2), 2012 pp. 153–164. http://www.complex-systems.com/pdf/21-2-4.pdf.

https://doi.org/10.25088/ComplexSystems.24.3.261