IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 18, Issue 3, Ver. IV (May-Jun. 2016), PP 92-101
www.iosrjournals.org

A New Multi-Agent Game Solver

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

Abstract: *A new N-person game solver is presented in this work. The new software is an advance in the whole sate of art for the investigation of N-person games. It is now possible to investigate entire classes of games almost automatically. Moreover, this software makes it trivial to perform a systematic investigation of any game with a multidimensional parameter space. In addition, the payoff functions are no longer limited to quadratic expressions; any elementary mathematical function can easily be used, even in parameterized form. The experimental results are interesting and they very strongly depend on the initial conditions.*
Keywords: *N-person games, Netlogo, Agent-based simulation, Prisoners' Dilemma, Chicken Dilemma, Complexity*.

I. Introduction

Many complex phenomena can be represented as N-person games in which the participating agents all interact with all the other agents directly or indirectly [1-4]. In the latter case the agents immediately interact only with their immediate neighbors. In this case the agents are simply cells in a cellular automaton.
Cellular automata [5-7] are powerful tools for the study of simultaneous operation of a large number of agents, i. e. for the investigation of complex systems and phenomena. Therefore, they are quite suitable for the study of spatial N-person games. We developed an agent-based simulation tool [8] that has been successfully applied to the solution of a large number of various N-person games [9]. In this software 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 limitations that need attention: 1) The software is only capable of handling linear and quadratic payoff functions. Practical problems sometimes require more sophisticated inputs represented by arbitrary mathematical expressions. 2) In order to investigate the multidimensional parameter space of a model, the user must modify a configuration file and run each configuration separately, which is a time-consuming work.

These two problems were solved by the present author with a new software tool [10] but it is limited to Pavlovian agents only. For Pavlovian agents the probability of choosing the previously chosen action again changes by an amount proportional to the reward/penalty for the previous action [11]. This decision heuristic is based on Pavlov's experiments and Thorndike's law [12]: if an action is followed by a satisfactory state of affairs, then the tendency of the agent to produce that particular action is reinforced.

We present a new software tool here that solves these problems for more traditional agents that directly interact with their immediate neighbors including themselves according to the following update rules: they copy the behavior of the neighbor with the highest reward for their previous action and do not change their behavior in case of a tie among two or more agents with different behaviors [13]. This is a typical case of a two-state cellular automaton.

II. The Model and its Implementation

The agents are distributed in and fully occupy a finite two-dimensional cellular automaton and the updates are simultaneous. They must choose between two choices. In accordance with the accepted notation, we call these choices cooperation and defection. In an iterative game the aggregate cooperation proportion changes in time, i.e., over subsequent iterations. This proportion usually converges to a stable or oscillating state. It is possible to wrap the cellular automaton in either horizontal or vertical or both directions. If there is no wrapping, one can simulate 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 in the neighborhood plus the agent itself. For the Moore neighborhood this number is always 9 inside the automaton or if wrapping is present in both directions. On the borders of a confined automaton it is equal to 6, and it is equal to 4 if the agent is in the corner. For Neumann neighborhood these numbers are 5, 4, and 3, respectively.

The outcome of the game depends on the values of at least the following user-defined parameters:

- number of rows and columns in the cellular automaton;
- payoff curves for cooperators and defectors;
- initial states of individual agents at various locations in the automaton.

We have chosen NetLogo Version 5.2 [14] for the implementation of this project. Netlogo is a programmable modeling environment authored by Uri Wilensky of the Center for Connected Learning and Computer-Based Modeling of Northwestern University in 1999. It is based on a system of patches that constitute a cellular automaton. Therefore, it is particularly well suited for our task.

The most important feature of Netlogo is that user interfaces are built in the software. The user can create buttons, sliders, plots, switches, monitors, and other interface tools by elementary commands to set the parameters of the particular project.

We used one graphics output, five buttons, two switches, several sliders, two input boxes, one monitor, and one plot in this model. We start with setting up the number of rows and columns in the cellular automaton as well as wrapping in vertical, horizontal, or both directions. The *setup* button sets the program up according to the value of the *initial cooperation ratio* determined by its slider. It can be modified or completely overwritten by using the mouse after clicking the *manual* button. The *payoff functions* are determined by any combination of basic mathematical expressions. This makes it possible to simulate N-person games with arbitrary payoff functions. They are entered by the user in two input boxes. In addition, the payoff functions can be represented in parameterized form. The *parameters* can easily be changed by the necessary number of their sliders A, B, C, D, etc. 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 records the results. Thus, sophisticated experiments can be performed. The results are presented in the form of a table or a spreadsheet. It is also possible to vary any other parameter used in the code.

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 *color?* 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 running of the program. The total cooperation ratio as a function of the number of iterations is shown in the *plot*. We can also watch the change of its numerical value in the *monitor*. This makes it possible to immediately see its final value when we stop the simulation. The *demo?* switch sets up the demonstration of the software.

As an alternative, we can start the simulation with the *make-movie* button that runs the program for any number of iterations determined by the setting of the *number* slider and creates a movie of the changing graphics output in a separate file.

At 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 or of the entire 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, and the last reward or punishment that the agent received. When the experimenter stops the simulation, the history 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 when the cooperation ratio reaches equilibrium or starts to oscillate around a constant value. The speed of the iteration can be controlled by the in-built speed slider.

Netlogo was previously used in an attempt to simulate a Prisoners' Dilemma game where there is only one parameter of the simulation except for the initial cooperation ratio [15]. This software cannot be used for any other simulation. In addition, in case of a tie among two or more agents with different behaviors, it chooses the behavior of one of such agents randomly. This immediately destroys the symmetry of any symmetric problem. Our new software makes it possible to investigate entire classes of games almost automatically. Moreover, the multidimensional character of the parameter space is no longer an obstacle to perform a systematic investigation of any game. In addition, any combination of elementary mathematical expressions can easily be used to express the user-defined parameterized payoff functions.

The proper operation of the software was verified by repeating the simulations published in papers [16, 17]. We obtained the same results with the new software.

III. Demonstration

To demonstrate the working of this software tool, we performed experiments with a cellular automaton formed by 101x101=10,201 agents that initially all cooperate except one in the geometrical center of the automaton. This corresponds to a 99.99% cooperation ratio (Figure 1). The payoff functions are C(x) = x and D(x) = 1.85 x. This seemingly trivial case exhibits remarkably beautiful fractal behavior. The user interface is shown in Figure 2 after the 500th iteration. The figure also shows the *graphics output* at this iteration.

During the initial stage of the simulation the number of defectors starts drastically grow showing increasingly interesting patterns. Cooperator agents are black, defector agents are white. After the 10^{th} iteration the total cooperation ratio is 97.5% (Figure 3), after the 50^{th} iteration it is 58.2% (Figure 4), and after the 74^{th} iteration it reaches its minimum at 15.2% (Figure 5). In subsequent iterations it is rising again for a while but at about the 100^{th} iteration it starts wildly fluctuate between 19% and 42%. Figures 6, 7, 8, and 9 show the graphics outputs for the 100^{th}, 200^{th}, 300^{th}, and 400^{th} iterations, respectively. The graphics outputs change like a kaleidoscope, never repeating their patterns. This fact is demonstrated by Figures 10, 11, and 12 that show the output patterns after the 499^{th}, 500^{th}, and 501st iterations, respectively. Figure 13 shows the total ratio of cooperators as a function of subsequent iterations. If we switch wrapping on, the pattern will change. Figure 14 shows it for the 501st iteration. Note the difference between Figures 12 and 14.

The kaleidoscopic pictures are even more interesting if we use a four-color display. The cooperators are now colored blue if they also cooperated in the previous iteration and green if they changed from defection to cooperation. The defectors are colored red if they also defected in the previous iteration and yellow if they changed from cooperation to defection. Therefore, the black cooperators are now divided into blue and green; the white defectors are now red or yellow. This coloring scheme results in truly beautiful patterns. Color Plate 1 shows it after the 500th iteration. Compare it with Figure 11. The color fluctuations are demonstrated by Color Plates 2, 3, and 4 for the 1049^{th}, 1050^{th}, and 1051^{st} iterations, respectively.

These results are even more impressive if we watch the changing of output patterns in a movie: https://www.youtube.com/watch?v=UidY8ST58qA Naturally, the result will be totally different even if we just slightly change the payoff functions. For example, Figure 15 shows the output pattern after the 52^{nd} iteration for the case of C(x) = x and D(x) = 2 x. This pattern remains unchanged in the subsequent iterations. This fact shows the chaotic nature of N-person games: even totally deterministic initial conditions produce an unpredictable behavior that is the result of even the slightest change of those conditions.

IV. Conclusion

Our new software is an advance in the whole state of art for the investigation of N-person games. It is now possible to investigate entire classes of games almost automatically. The new software makes it trivial to perform a systematic investigation of any game. In addition, the payoff functions are no longer limited to quadratic expressions; any elementary mathematical function can easily be used. The experimental results are interesting and they very strongly depend on the initial conditions.

Acknowledgment

The author appreciates the participation of Mr. Ryan Province in the initial stage of this work.

References

[1]. A. Rapoport, *N-Person Game Theory*. University of Michigan Press, Ann Arbor, MI, 1970.
[2]. H. Hamburger, N-person Prisoner's Dilemma, *Journal of Mathematical Sociology* 1973, **3**, 27–48.
[3]. T. C. Schelling, Hockey helmets, concealed weapons, and daylight saving. *Journal of Conflict Resolution* 1973, **17**(3), 381-428.
[4]. G. Hardin, The Tragedy of the Commons. *Science* 1968, **162,** 1243– 1248.
[5]. T. Toffoli, Cellular automata as an alternative to (rather than an approximation of) differential equations in modeling physics, *Physica D* 1984, **10**, 117-127.
[6]. A. Ilachinski, *Cellular Automata: A Discrete Universe*, World Scientific, 2001.
[7]. S. Wolfram, *A New Kind of Science*, Wolfram Media, Inc,, Champaign, IL, 2002.
[8]. M. N. Szilagyi and Z. C. Szilagyi, A tool for simulated social experiments. *Simulation* 2000, **74**, 1, 4-10.
[9]. M. N. Szilagyi, Investigation of N-person games by agent-based modeling. *Complex Systems* 2012, **21**, 201-243.
[10]. M. N. Szilagyi, A general N-person game solver for Pavlovian agents. *Complex Systems* 2015, **24**, 261-274.
[11]. D. Kraines and V. Kraines, Pavlov and the Prisoner's Dilemma. *Theory and Decision* 1989, **26**, 47-79.
[12]. E. L. Thorndike, *Animal Intelligence*, Hafner, Darien, CT, 1911.
[13]. M. A. Nowak and R. M. May, Evolutionary games and spatial chaos, *Nature*, 1992, 359, 826-829.
[14]. U. Wilensky, NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1999.
[15]. U. Wilensky, NetLogo PD Basic Evolutionary model. http://ccl.northwestern.edu/netlogo/models/PDBasicEvolutionary. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 2002.
[16]. A. L. Lloyd, Computing bouts of the Prisoner's Dilemma, *Scientific American* 1995, 272, 6, 110-115.
[17]. M. N. Szilagyi and I. Somogyi, Agent-based simulation of an N-Person game with parabolic payoff functions, *Complexity* 2010, **15**, 3, 50-60.

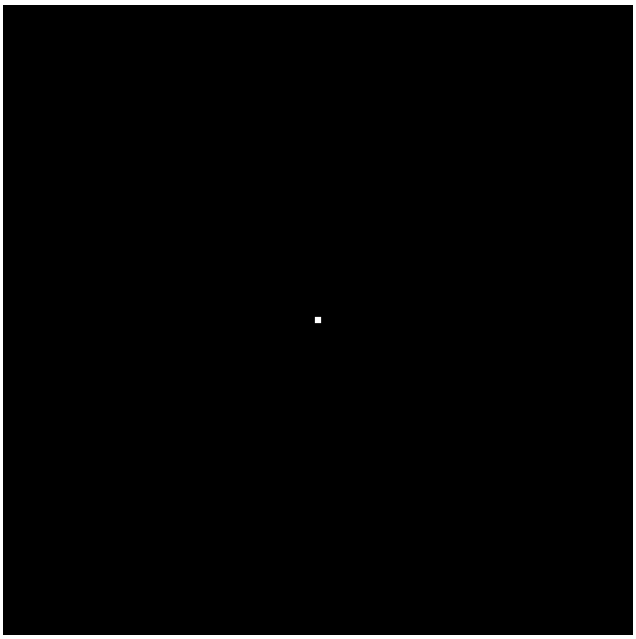


Figure 1

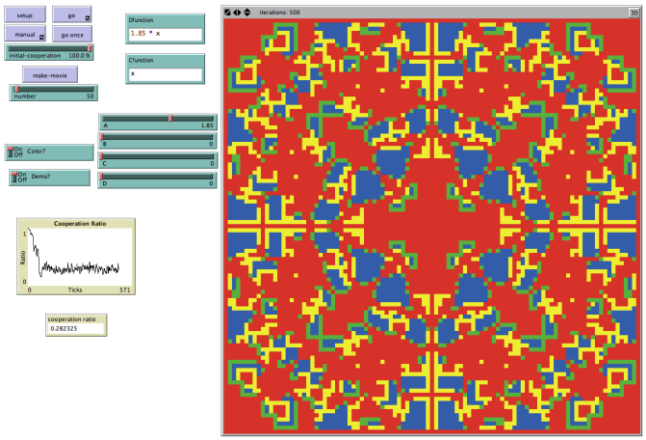


Figure 2

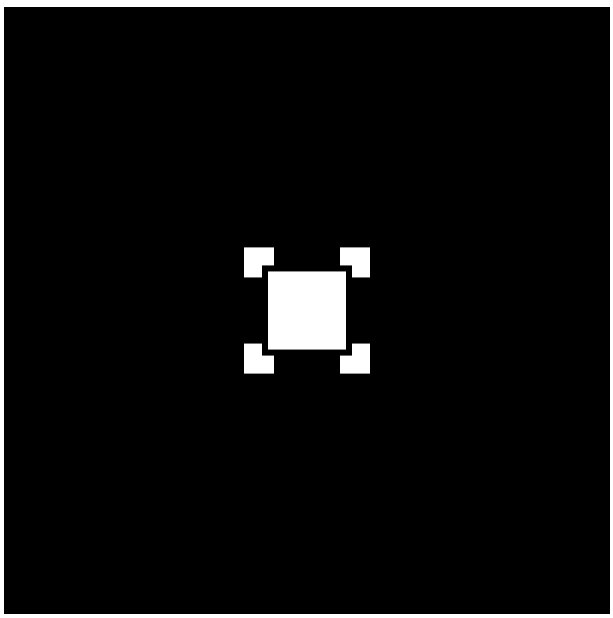


Figure 3

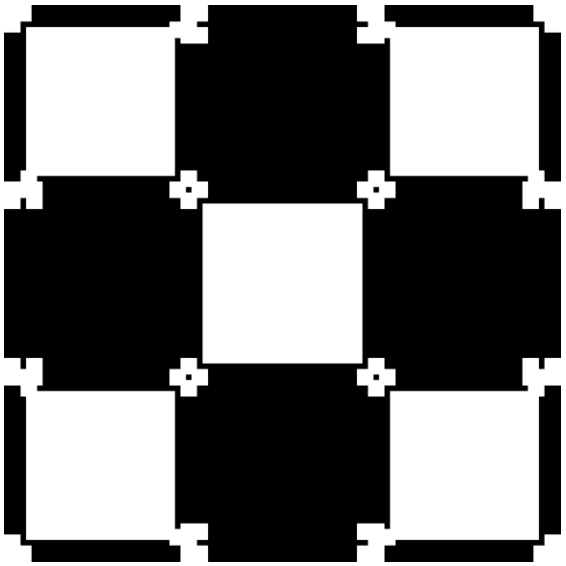


Figure 4

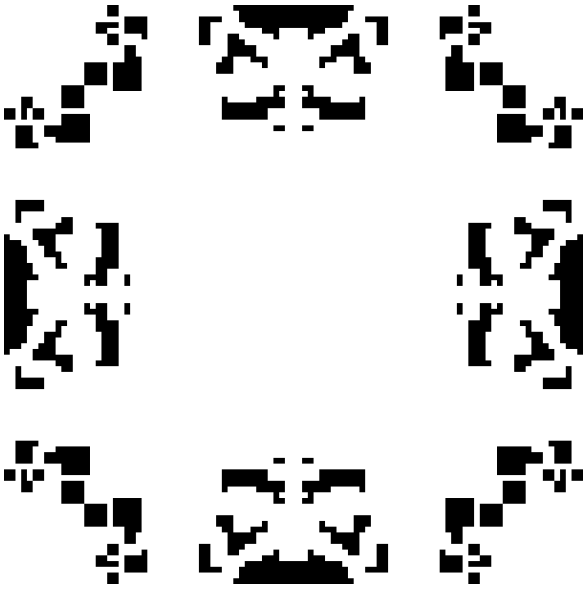


Figure 5

Figure 6

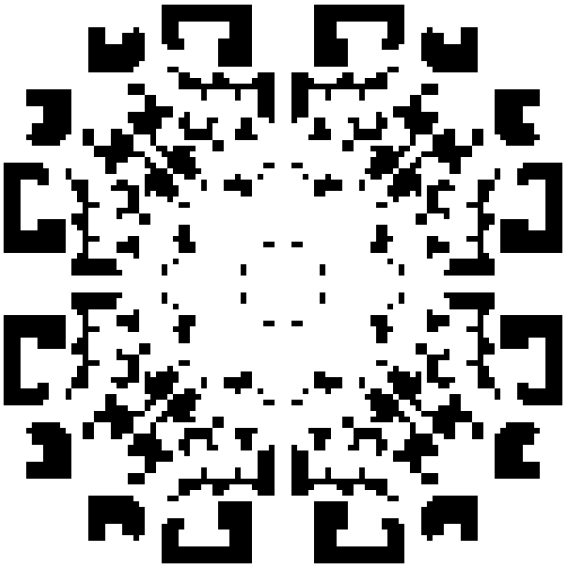


Figure 7

Figure 8

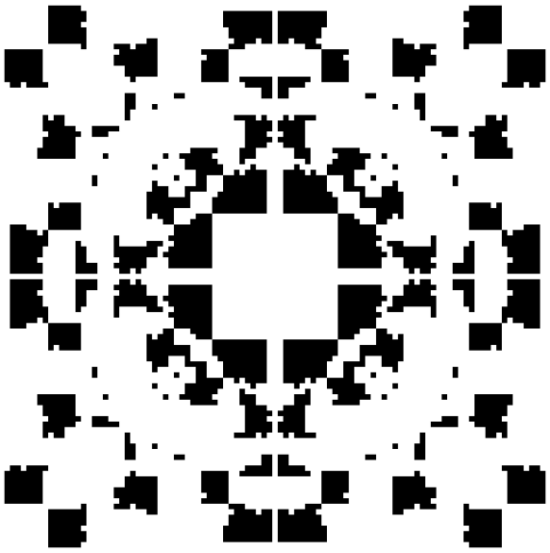


Figure 9

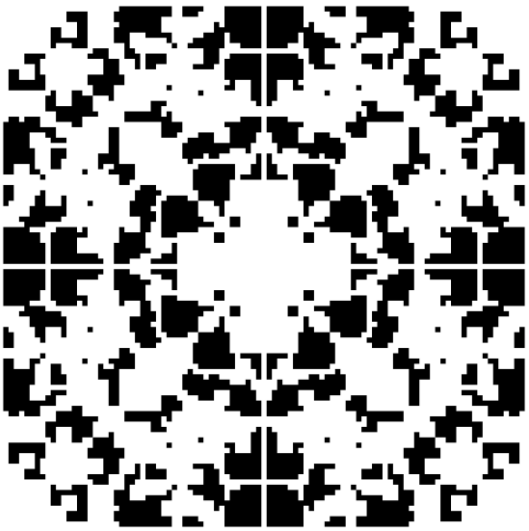


Figure 10

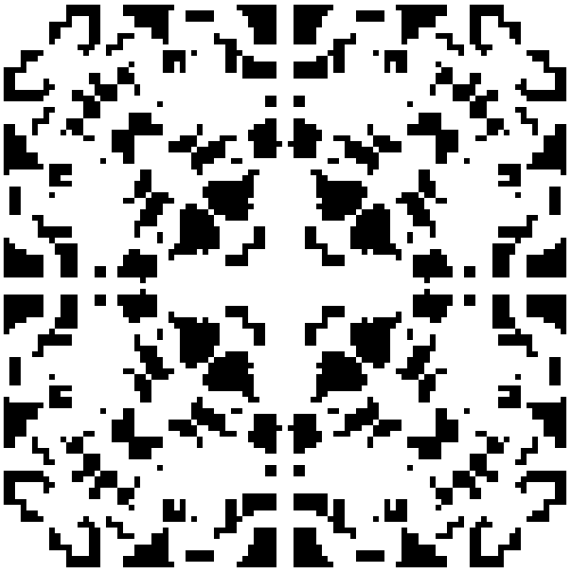


Figure 11

Figure 12

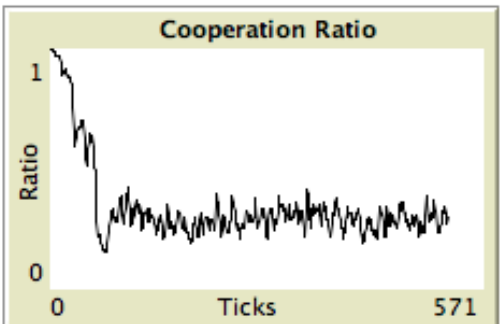


Figure 13

Figure 14

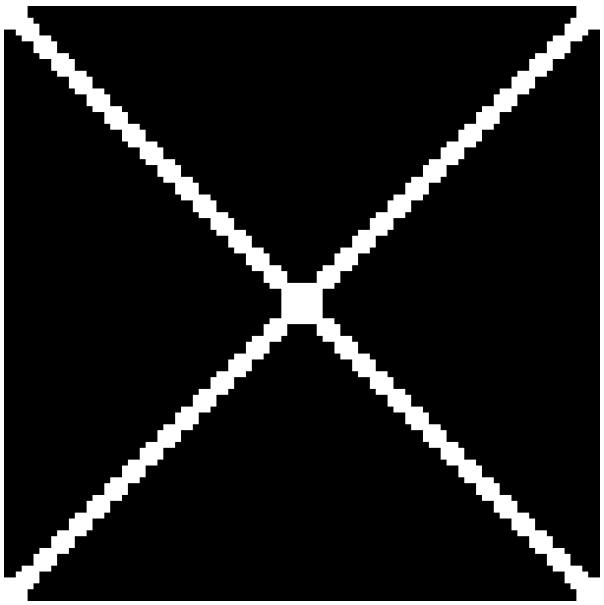
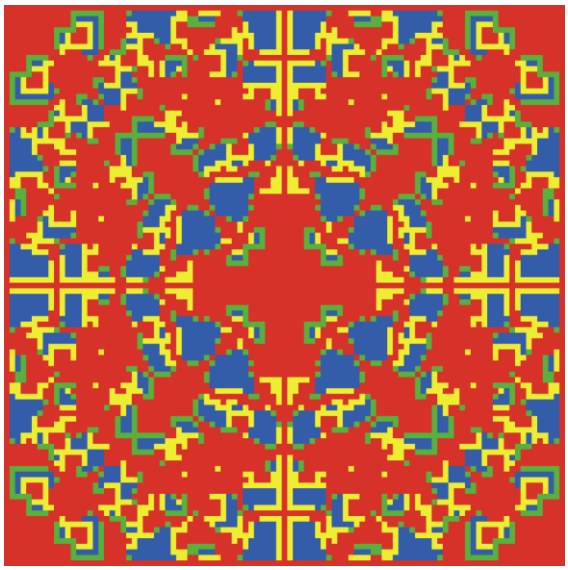
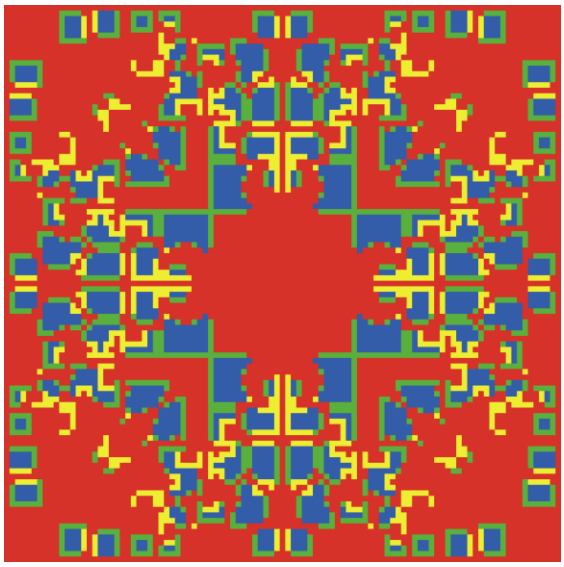


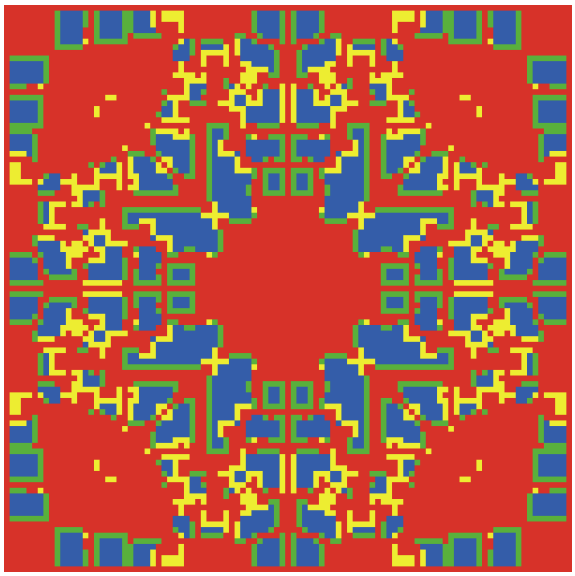
Figure 15



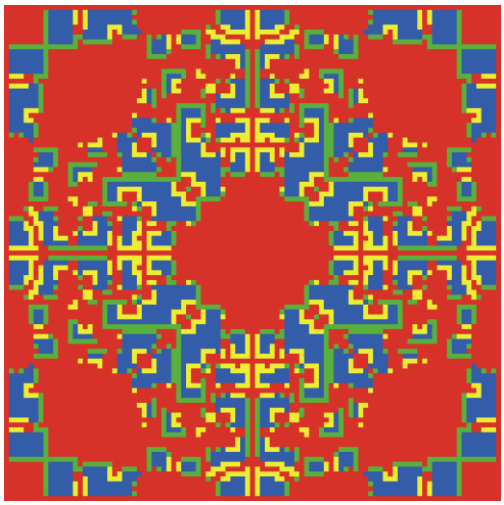
Color Plate 1



Color Plate 2



Color Plate 3



Color Plate 4