Replication of the Demographic Prisoner's Dilemma

Wolfgang Radax* and Bernhard Rengs

Vienna University of Technology, Austria

Abstract. This paper documents our efforts in replicating Epstein's (1998) demographic prisoner's dilemma model. While, qualitatively speaking, our replicated model resembles the results of the original model reasonably well, statistical testing reveals that in quantitative terms our endeavor was only partially successful. This fact hints towards some unstated assumptions regarding the original model. Confronted with a number of ambiguous descriptions of model features we introduce a method for systematically generating a large number of model replications and testing for their equivalence to the original model. With the help of this approach we show that the original model was probably based on a number of dubious assumptions. Finally we conduct a number of statistical tests with respect to the influence of certain design choices like the method of updating, the timing of events and the randomization of the activation order. The results of these tests highlight the importance of an explicit documentation of design choices and especially of the timing of events.
Key Words: Demographic, Prisoner's Dilemma, Replication, Simulation, Complex Adaptive Systems, Social Science Models

1 Introduction

The evolution of cooperation between egoistic individuals has attracted the attention of researchers from a great number of disciplines ranging from biology over political science to economics. Out of this effort the branch of cooperation theory came into existence, spearheaded by Robert Axelrod's groundbreaking works. Typically formulated as a prisoner's dilemma, the quest usually consists of discovering sufficient circumstances to allow for the emergence of cooperation or of deriving strategies, which generate high utility in a variety of situations.

While the common one shot-game leads to the outcome of mutual defection despite being an inferior solution, for repeated games Axelrod has shown Anatol Rapoport's Tit-For-Tat (TFT) to be a highly efficient strategy since it fairs very well against cooperators and defectors alike. In another interesting twist Nowak and May [6] tried to show within a spatial context that it's possible to arrive at

* Corresponding author. Wolfgang Radax contribution to this paper is funded by research grant #P19973 of the Austrian Science Fund FWF. We would like to thank two anonymous referees for their insightful comments.

a cooperative solution with zero-memory-strategies as well. Their results were later shown, however, to be critically dependent on the timing of events [5]. In the more plausible setting of asynchronous updating no clusters of cooperation were able to evolve.

Epstein's demographic prisoner's dilemma (DPD henceforth) [3] was yet another interesting take on the topic. Agents arranged on a torus are allowed to move freely and propagate. When they encounter another individual a round of prisoner's dilemma is played. This comparably simple setup serves as a proof of existence for the evolution of cooperative clusters, although only zero memory-strategies are employed and agents cannot distinguish cooperators from defectors, i.e. they are tag-less.

The goal of our work was to replicate the results of this model using the Repast framework for Java, but as the following report shows we succeeded only partially. Nevertheless, our efforts are instructive for a number of reasons, laid out in later sections.

In the next section, Epstein's DPD is presented in detail. Section 3 deals with our replication efforts while Sect. 4 summarizes our insights. In the closing section we discuss the implications of our results.

2 Original Model

The remarks in this section are completely based on Epstein's model description in [3]. The DPD is played on a 30x30 matrix with wrapped-around borders, which topographically corresponds to a torus. Initially 100 agents are placed on random locations of the torus. Each of these agents is born with an initial endowment of resources and a fixed strategy; either Cooperate (C) or Defect (D). This strategy is randomly assigned during initialization with equal probabilities. Each turn every agent is allowed to move randomly to an unoccupied site within its Von Neumann-neighborhood. If all neighboring sites are occupied, no movement takes place.

If there happen to be other agents in the Von Neumann-neighborhood after the movement, the currently active agent plays one game of prisoner's dilemma against each of them. As usual the payoff for mutual cooperation is R (reward) for both participants, P (punishment) for both agents in the case of mutual defection and if one agent cooperates while the other defects, the defector receives T (temptation) and the cooperator gets S (sucker's payoff). The payoffs follow $T > R > 0 > P > S$ and $R > (T + S)/2$.

Payoffs accumulate and since some payoffs of the game form are negative, the total amount of an agent's resources may turn negative. In this case, the agent instantly dies and is removed from the game. If, however, an agent's resources exceed a given threshold, this agent may give birth to a new agent in its Von Neumann-neighborhood which is born in a random vacant neighboring site of the agent. The newborn agent inherits its parent's strategy and is endowed with the afore mentioned amount of initial resources. Should all sites within the neighborhood be occupied, giving birth is not possible. After an agent has

completed all these steps, it is the next agent's turn, and so forth, until all agents have been active. All agents having been activated once corresponds to one time period.

This schedule resembles what is called asynchronous updating. Instead of assuming some kind of external timer, which synchronizes the individual actions, an agent takes all actions as soon as it is its turn. The choice of updating schedule has been shown to be of the utmost importance by Huberman and Glance [5]. In their own words "if a computer simulation is to mimic a real world system with no global clock, it should contain procedures that ensure that the updating of the interacting entities is continuous and asynchronous. This entails choosing an interval of time small enough so that *at each step at most one individual entity is chosen at random to interact with its neighbors. During this update, the state of the rest of the system is held constant. This procedure is then repeated throughout the array for one player at a time, in contrast to a synchronous simulation in which all the entities are updated at once*" (emphasis added). To avoid artifacts the order of activation is shuffled at the end of each period.

Given these basic assumptions Epstein investigates the behavior of the model for five different settings. For Run 1 he assumes no maximum age so that agents may die only from the consequences of playing the prisoner's dilemma. This first setting already proves his basic point that "*cooperation can emerge and flourish in a population of tagless agents playing zero-memory fixed strategies of cooperate or defect in this demographic setting*" (emphasis in the original paper). After only a few periods a stable pattern emerges and cooperators dominate the landscape counting nearly 90 percent (800 out of 900 agents at the maximum on a 30x30 torus), while the defectors fill up the rest of the space.

In Run 2 a maximum age is introduced so that agents may die of age as well. The maximum lifetime is set to 100 periods. This change leads to slight oscillations in the time series of numbers of cooperators and defectors but the mean values are not affected much.

Runs 3 and 4 change the payoff for mutual cooperation. In Run 3 R is decreased from 5 to 2. The effect of this change is an accentuation of the oscillatory dynamics. Furthermore, defectors fare comparatively better (on average counting about 200 agents) and cooperators do worse, ranging from 250 to 450 agents. Run 4 decreases R further down to 1 which pronounces the oscillatory dynamics even more, resembling predator-prey-cycles between the defectors and the cooperators. Because of these extreme oscillations, Run 4 leads to a number of different outcomes depending on the random seed. In some runs, cooperators dominate the scene while in others they die out (soon followed by the defectors who then have no prey and "feed" on each other until extinction).

Finally, in Run 5 Epstein introduces mutation while setting R to its original value of 5 to investigate the stability of the emergence of cooperation. Until now offspring inherited the fixed strategy from its parent. Mutation is defined "as the probability that an agent will have a strategy different from its parent's." The mutation rate is set to 50 percent. Still, cooperation persists despite pronounced oscillatory dynamics.

3 Replication

Although agent-based models are clearly on the rise as a modeling tool, with but a few exceptions most of these models have not been replicated or replications haven't been published. It is only in recent years that the importance of the replication of agent-based models is recognized and the problems associated with it are acknowledged. The original model and the replicated model may differ along many dimensions which complicates the process of replication. In [7] Wilensky and Rand have suggested a list of items to be included in publications of replication. We follow their suggestions and list our details in Table 1. In the case of multiple choice-issues we highlighted our choice with bold typeset.

Table 1. Details of replication

Standard	Numerical identity **Distributional equivalence** Relational alignment
Focal measures	Number of cooperators, Number of defectors
Level of communication	**None (original author didn't answer our request)** Brief email contact Rich discussion and personal meetings
Familiarity with language/toolkit of original model (C++)	None **Surface understanding** Have built other models in this language/toolkit
Examination of source code	**None** Referred to for particular questions Studied in-depth
Exposure to original implemented model	None **Run**[1] Re-ran original experiments Ran experiments other than the original ones
Exploration of parameters space	**Only examined results from original paper** Examined other areas of the parameter space

[1] We ran a reimplementation of the model provided with the accompanying CD of [4] a few times. The CD only contained executable versions of the model, source code was not provided.

The original model was written in C++ (a reimplementation for AScape is available in [4] – cf. footnote 1). Our replication was realized with the Repast 3.1 framework for Java.

As stated in Table 1 we aimed for distributional equivalence which [1] defined as two models producing distributions of results that cannot be distinguished statistically. We compared the results of both models in respect to the numbers of cooperators and defectors for Runs 1 and 2 by using t-tests for the equality of means of two samples with the same unknown variance – an approach already used, for instance, by [7].

Since we didn't get into touch with the author of the original model we had no exposure to the original source code and had to base our replication efforts solely on the published description of the model given in [3].

The first version of our reimplementation matched the reported results reasonably well with respect to the qualitative behavior of the original model for all five runs given in the original paper, although our model showed much more pronounced oscillatory dynamics. Statistical testing revealed that our results didn't reproduce the ones of the original model. So we went back to the published description and looked for clues where we could have gone wrong or possibly misinterpreted Epstein's assumptions. We identified a number of issues that we were not able to draw clear conclusions from. Additionally we wanted to test for a number of assumptions which we a priori assumed to be inconsequential (either because it has been stated so explicitly in the original article or because in fact they shouldn't matter anyway), but regarded as interesting tests nevertheless. We arrived at seven assumptions we wanted to test in a systematical way:

1. **Timing of the removal of dead agents:** In our first naive implementation of the DPD we assumed that dead agents are removed from the torus at the end of each period. This contradicts the assumption of asynchronous updating and may have considerable influence on the results, since the dead agents may fill up the space where other agents try to give birth to offspring. So we introduced the option to remove a dead agent exactly at the moment of its death.
2. **Timing of the death of agents:** Also connected with the issue of the death of agents was the question whether an agent may die although it is not its turn. This may happen if the active agent plays a game of prisoner's dilemma against the agent in question and, as a result of this game, the latter agent's accumulated payoff drops below zero. Although our first implementation already considered this "passive death" we allowed for an option that an agent doesn't die until it is activated next time.
3. **Origin of initial endowment:** When an agent's accumulated payoff exceeds a certain threshold, it may give birth to an offspring. The newborn agent starts with an initial endowment of six resources. We asked whether this initial endowment is inherited directly from the parent (i.e. subtracted from its accumulated payoff) or if the new agent receives this amount of resources from an exogenous source.

4. **Birth age:** Here, the original article was a little bit ambiguous stating that "[a]n agent's initial age is a random integer between one and the maximum age." We were not quite sure whether this only concerned the initial population of 100 agents or if offspring born during the simulation started with a random birth age as well. So, although it seems counter-intuitive, we included an option for random birth age as well.
5. **Updating mechanism:** Although the article explicitly emphasizes the use of asynchronous updating, we thought it to be an instructive lesson to investigate the extent of differences in the results when alternatively allowing for synchronous updating. The inclusion of this option was additionally motivated by the fact that the Ascape-reimplementation of this model provided by [4], allowed for "execution by agent" as well as "execution by rule", which seem to be labels for synchronous and asynchronous updating, respectively.
6. **Random number generator:** When coding in Repast for Java you have the choice between two random number generators, Repast's CERN Random Library and Java's own random library. We were quite curious if the choice of the random number generator might have an effect on the results and therefore included an option to choose one of these two libraries.
7. **Randomization of the order of activation:** Epstein explicitly describes his method of shuffling the activation order of agents: "Agent objects are held in a doubly linked list and are processed serially. If there are N agents, a pair of agents is selected at random and the agents swap positions in the list. This random swapping is done N/2 times after each cycle." Our first implementation disregarded this explicit description and for matters of convenience made use of Repast's own method for shuffling lists which, according to the Repast documentation, shuffles a list "... by iterating backwards through the list and swapping the current item with a randomly chosen item. This randomly chosen item will occur before the current item in the list." We thought that this might have as well been a reason for the divergence in results and included an option to switch between Repast's shuffling method and the one described by Epstein.

We formulated each of these points as a binary parameter for our model being either true or false. The exact meaning of each value of the parameters is given in Table 2.

Testing for all possible combinations of these seven binary options leads to $2^7 = 128$ different settings to be investigated or more precisely to 128*2*30=7680 runs of the model (2 because of testing Epstein's settings called Runs 1 and 2 and 30 because in the original model each setting was repeated 30 times to eliminate the role of the random seed. We adopted this measure.) These 128 different candidate models are then tested by means of t-tests. By process of elimination of those cases where the equality of means-hypothesis can be rejected, we arrive at those solutions which approximate the original model reasonably well. The pseudo code of our replication is given in Table A-1 and Table A-2 for asynchronous and synchronous updating, respectively.

Table 2. Description of the binary parameters

No.	Name in the model	TRUE	FALSE
1	Remove dead agents immediately	An agent is removed at the moment it dies either of age or as a result of playing the prisoner's dilemma.	Dead agents are removed at the end of each period after all agents have been active.
2	Die immediately	An agent dies immediately when its accumulated resources drop below zero. This can also happen when it's not the agent's turn as a result of another active agent playing the prisoner's dilemma with the former.	An agent can only die while being active. In consequence, if its resources drop below zero when it's not its turn, it dies not immediately but only the next time after taking its turn.
3	Initial endowment inherited	The initial endowment of a new offspring is subtracted from its parent.	The initial endowment of a new offspring is independent from its parent's and not subtracted from the latter's.
4	Random birth age	A new born agent's initial age is a random integer between one and the maximum age.	A new born agent's initial age is set to one.
5	Asynchronous updating	If an agent is active it performs all possible steps before it's the next agent's turn.	In each period, first all agents move, then all agents play against all of their neighbors. Afterwards all agents give birth to offspring if possible.
6	CERN Random	Repast's own random library is used.	Java's own random library is used.
7	Repast List-Shuffle	Repast's own method for shuffling lists is used for shuffling the activation order of agents at the end of each period.	The activation order of agents is shuffled according to Epstein's algorithm.

To level out the influence of the random element we conducted 30 runs per combination of true/false-values for the binary parameters; each time using different random seeds for the random number generator for each model corresponding to the parameter settings of Runs 1 and 2. As in the original model we sampled the numbers of cooperators and defectors at t=500 and calculated the mean and the standard deviation which we then tested against the values of the original model by means of a t-test. The results of this endeavor are summarized in the following section.

4 Results

Regarding Run 1, for 127 of the 128 cases tested, the null hypotheses of equality of means could be rejected for the number of cooperators or the number of defectors at $\alpha = 0.05$. So only one parameter combination remains that isn't statistically distinguishable from the original model with respect to both focal measures. We, however, are very confident to claim, that this candidate solution is radically different from the original model, for it assumes synchronous updating, while Epstein explicitly emphasizes the use of asynchronous updating. Details for this case are given in Table 3. The table reports the averaged values over 30 runs and additionally the respective standard deviations in parentheses. The results of the replicated model are rounded. Furthermore, the results of the t-tests on the equality of means are reported.[2]

Table 3. Details of statistical testing for Run 1

Binary parameter No.							**Original Model (Run 1)**			
							No. Coop.		No. Def.	
							779 (15)		121 (15)	
							Replicated Model (Run 1)			
1	2	3	4	5	6	7	No. Coop.	t-Value	No. Def.	t-Value
F	F	F	T	**F**	F	T	785 (16)	-1.63	114 (16)	1.65

Although having discovered one candidate solution whose results can't be distinguished from the original model, we can say that our goal of achieving distributional equivalence was not attained with respect to Run 1.

For Run 2 we were able to reject the null hypothesis in 121 of 128 cases, leaving seven cases where the results are statistically indistinguishable from the original model. The details of these seven cases are presented in Table 4.

Since we are interested in finding a model with a close fit to the original model, lower t-values are desirable. By far the best result is achieved by the parameter combination reported in the fifth row of the table (highlighted in bold typeset). The respective solution shares some features with the majority of

[2] Detailed results of all 128 cases are available upon request.

Table 4. Details of statistical testing for Run 2

Binary parameter No.							Original Model (Run 2)			
							No. Coop.		No. Def.	
							784 (29)		99 (25)	
							Replicated Model (Run 2)			
1	2	3	4	5	6	7	No. Coop.	t-Value	No. Def.	t-Value
T	T	F	T	T	T	T	796 (24)	-1.70	110 (24)	-1.67
T	F	T	F	T	T	T	788 (27)	-0.50	111 (27)	-1.77
F	T	T	**T**	**T**	T	F	789 (24)	-0.79	88 (22)	1.86
F	T	T	**T**	**T**	F	F	787 (29)	-0.44	90 (26)	1.30
F	T	F	**T**	**T**	F	F	**780 (25)**	**0.53**	**97 (22)**	**0.28**
F	F	T	**T**	**T**	F	T	773 (23)	1.61	103 (21)	-0.66
F	F	F	**T**	**T**	T	T	769 (31)	1.98	108 (27)	-1.28

candidate solutions making it even more probable that the respective parameters resemble the choices undertaken in the original model.

As expected, all cases resembling the results of the original model employ asynchronous updating. A little bit more surprising is the result that in the majority of the candidate solutions agents are born with a random birth age. The meaningfulness of this assumption is very dubious if the propagation process is to resemble giving birth[3]. Another problematic result of our extensive testing is that in the majority of successful replications of Run 2, dead agents are not removed immediately at the time of death but are removed only at the end of each period collectively. This result is problematic insofar as it contradicts the approach of asynchronous updating to some degree.

Unfortunately, we were not able to find a single parameter combination which is capable of reproducing the results of Runs 1 and 2 of the original model at once. This fact hints to additional assumptions regarding Run 1 which were not stated in [3] or to put it differently that the results of Run 1 were achieved by a slightly different model than those of Run 2.

The large quantity of data produced in the course of replicating the model led us to the idea to conduct further series of tests regarding the influence of each binary parameter ceteris paribus on the outcomes. Holding six of the seven parameters constant, we compared the two models with the seventh parameter being true and false, respectively with the t-test introduced above. This procedure was repeated for all seven parameters and all combinations of the six remaining parameters amounting to 64 tests per Run and parameter. To illustrate this more vividly, let's consider one specific test on the importance of the parameter *remove dead immediately*. Adopting the order of parameters employed

[3] In his Ascape-Reimplementation [4], Epstein no longer calls the propagation process giving birth but calls it fissioning instead (Model Settings Screen, Rules Section). One anonymous referee suggested that "another plausible variation for birth age would be to assign the new child the same as the parent." In this case, however, each simulation for Run 2 would end after 100 periods. Therefore, this option can't account for the irregularties concerning birth age, either.

in the tables above we have to test $2^6 = 64$ different combinations of binary parameters having *remove dead immediately* = true against their counterparts having *remove dead immediately* = false.

Table 5. Summary of statistical tests on the influence of parameters

Parameter	Times H_0 is rejected	
	Run 1	Run 2
Remove dead agents immediately	57 (89.06%)	60 (93.75%)
Die immediately	45 (70.31%)	51 (79.69%)
Initial endowment inherited	61 (95.31%)	38 (59.38%)
Random birth age	3 (4.69%)	54 (84.39%)
Asynchronous updating	64 (100.00%)	64 (100.00%)
CERN random	7 (10.94%)	4 (6.25%)
Repast list-shuffle	14 (21.88%)	32 (50.00%)

We present only a number of insights gained from this series of tests which are summarized in Table 5. For instance, the parameter *random birth age* confirms what could have been expected a priori. It has no influence at all on the outcomes of Run 1 (rejecting only 3 of 64 tests) since this run assumes no maximum age and therefore the birth age doesn't matter at all. For Run 2, however, the results vary significantly and produce different outcomes for 54 of the 64 cases. This serves as an interesting example of model validation by means of statistical testing.

A similarly clear picture emerges from the tests on the influence of the updating mechanism. In this instance, the equality of means-null hypotheses is rejected in all cases of Run 1 and Run 2 giving additional weight to the importance of choosing the right updating mechanism for the modeling problem at hand. The case is similar with respect to the parameter *remove dead immediately*. For the vast majority of parameter combinations (57 of 64 for Run 1, 60 of 64 for Run 2) the samples show a significant difference. The effect is a little bit less pronounced in the case of the parameter *die immediately*. Still, 45 of 64 parameter combinations show a significant difference for Run 1 and even 51 of 64 parameter combinations do so for Run 2. What these three parameters have in common is that they all deal with the timing of events within the model. All of these three assumptions concern only the exact point in time during the same global time step when a given procedure should be executed and yet the results vary dramatically. This points out the high importance of explicitly stating the course of events in an agent-based model – for instance by means of a detailed pseudo code[4] or flow charts – in order to be replicable.

[4] While writing this paper we realized that pseudo code is as prone to ambiguities as plain verbal description, if it is not used with great care for details. However, pseudo code forces the modeler to state the order of events in a strictly sequential way. As one anonymous referee correctly suggested, plain verbal description of the model can

For the assumption about the origin of an offspring's initial endowment the picture is not as clear as in the above cases. For Run 2, 26 out of 64 cases show no significant differences in the results, depending on the origin of the initial endowment and therefore it is hard to draw some decisive conclusions. For Run 1, however, the vast majority (61 of 64 cases) varies significantly when changing the parameter.

As also might have been expected, the choice of random library bears no influence on the results. While this might seem common sense, it is nevertheless reassuring that extensive statistical testing confirms this assumption. The case is a little bit different with the choice of a shuffling-algorithm. While for Run 1, this choice has a significant influence on the results in only a minority of cases (14 of 64 cases), Run 2 is affected 32 out of 64 times by the choice of the algorithm showing that the choice of the shuffling algorithm may be consequential to the outcomes of the model and should therefore be well documented.

5 Conclusion

While we haven't achieved our goal of distributional equivalence for Runs 1 and 2 at once, we were able to replicate the DPD reasonably well regarding Run 2. This outcome hints to unstated assumptions regarding the description of the original setting of Run 1, without which it is not possible to realize a successful replication of the original model.

By systematic testing of various parameter settings we confirmed that the original model employs asynchronous updating. On the other hand it turned out that it is highly likely, that in the original implementation not only the initial 100 agents start with a random age between one and the given maximum age but also new born agents are initialized with a random age. We doubt the meaningfulness of this assumption within the context of the model and furthermore showed that setting the birth age to one for all new born agents produces significantly different results for Run 2.

Another problematic insight is that in the original model dead agents probably were removed from the torus collectively at the end of each period and not at the immediate moment of their death. Not only have we shown that the timing of removal has significant influence on the results, removing the dead agents at the end of the period is also a breach of the assumption of asynchronous updating. Nevertheless, we have been able to verify the qualitative results of the original model that cooperation prevails under a wide variety of circumstances.

Further testing of our results revealed the importance of the timing of events in an agent-based model, highlighting the usefulness of explicitly stating the course of events, for instance by means of pseudo code documenting all critical aspects of the model. Furthermore, we confirmed the assumption that the choice of random library has no influences whatsoever on the average results of our

do the trick as well if it is complete, but contrary to the other two methods proposed it does not force you to be as explicit, thereby usually not arriving at the same level of clarity and often needing more space.

replication. The choice of shuffling algorithm, however, does have significant influence in a non-negligible number of cases.

We think that our approach of modeling ambiguous assumptions as binary parameters and systematically testing them is a valuable method for the replication of agent-based models which makes extensive use of the verbal description of the model to be replicated as well as the available data. In combination with statistical testing this procedure allows for some kind of reverse engineering when detailed information on the original model is not readily available.

Nevertheless, we are aware that this method is not free of shortcomings. First, turning ambiguous features into a binary parameter may not always be possible. Second, and even more important, the number of cases to be tested increases exponentially with the binary parameters and therefore our method can only be applied with respect to a selected number of model features, before the evaluation of the generated data turns into an arduous task. Objections might also be raised against our reliance on t-tests refraining from a more detailed comparison between the results of the model and of the replication. But for the goal at hand and for the process of elimination of candidate solutions, a test on the equality of means is by all means adequate. Despite these objections we believe this approach to be a helpful guide in the course of model replication.

References

1. Axtell, R., Axelrod, R., Epstein, J., Cohen, M.: Aligning Simulation Models: A Case Study and Results. Computational and Mathematical Organization Theory 1, 315–333 (1996)
2. Edmonds, B., Hales, D.: Replication, Replication and Replication: Some Hard Lessons from Model Alignment. Journal of Artificial Societies and Social Simulation 6(4)11 (http://jasss.soc.surrey.ac.uk/6/4/11.html) (2003)
3. Epstein, J.: Zones of Cooperation in Demographic Prisoner's Dilemma. Complexity 4, 36–48 (1998)
4. Epstein, J.: Generative Social Science: Studies in Agent-Based Computational Modeling (Princeton Studies of Complexity). Princeton University Press, Princeton, NJ. (2007)
5. Huberman, B., Glance, N.: Evolutionary games and computer simulations. Proceedings of the National Academy of Sciences USA 90, 7716–7718 (1993)
6. Nowak, M., May, R.: Evolutionary games and spatial chaos. Nature 359, 826–829 (1992)
7. Wilensky, U., Rand, W.: Making Models Match: Replicating an Agent-Based Model. Journal of Artificial Societies and Social Simulation 10(4)2 (http://jasss.soc.surrey.ac.uk/10/4/2.html) (2007)
8. Will, O., Hegselmann, R.: A Replication That Failed - on the Computational Model in 'Michael W. Macy and Yoshimichi Saito: Trust, Cooperation and Market Formation in the U.S. and Japan. Proceedings of the National Academy of Sciences, May 2002'. Journal of Artificial Societies and Social Simulation 11(3)3 (http://jasss.soc.surrey.ac.uk/11/3/3.html) (2008)

Appendix: Pseudo Codes

The presented pseudo codes assume the parameter *remove dead agents immediately* to be false. For the case of this parameter being true, the removal of agents occurs as soon as an agent dies, whether of age or from the result of playing the prisoner's dilemma.

Table A-1. Pseudo code in the case of asynchronous updating

```
Initialize model
DO t times
    FOR EACH agent DO
        Move
        Play against all Von Neumann-neighbors in random order
        IF resources < 0 THEN
            Die
        END IF
        FOR EACH neighbor of agent DO
            IF resources < 0 THEN
                Die
            END IF
        END FOR EACH
        Give birth to offspring if possible
        IF age >= maximum age THEN
            Die
        END IF
    END FOR EACH
    Remove dead agents from the space
    FOR EACH agent DO
        Age increases by 1
    END FOR EACH
    Shuffle activation order of agents
END DO
```

Table A-2. Pseudo code in the case of synchronous updating

```
Initialize model
DO t times
    FOR EACH agent DO
        Move
    END FOR EACH
    FOR EACH agent DO
        Play against all Von Neumann-neighbors in random order
        IF resources < 0 THEN
            Die
        END IF
        FOR EACH neighbor of agent DO
            IF resources < 0 THEN
                Die
            END IF
        END FOR EACH
    END FOR EACH
    FOR EACH agent DO
        Give birth to offspring if possible
    END FOR EACH
    FOR EACH agent DO
        If age >= maximum age THEN
            DIE
        END IF
    END FOR EACH
    Remove dead agents from the space
    FOR EACH agent DO
        Age increases by 1
    END FOR EACH
    Shuffle activation order of agents
END DO
```