ROBUST REPRODUCIBILITY OF AGENT BASED MODELS

Joseph Kehoe
Institute of Technology, Carlow
email: joseph.kehoe@ITCarlow.ie

KEYWORDS

Asynchronous Updating, Synchronous Updating, Agent Based Models, Reproducibility

ABSTRACT

Reproducibility is a well known problem in Agent Based Modelling. Most approaches to this problem replicate the original updating strategy but this leaves open the problem of results being "artefacts" of the specific updating strategy. We propose new synchronous updating algorithms that can be applied to Agent Based Models. Employing different updating strategies can show results are independent of any specific strategy. A C++ framework has implemented that allows different updating strategies to be applied in Agent Based Models, including novel algorithms for synchronous updating. Using this framework the original results of Sugarscape are reproduced and Sugarscape is shown to be robust with respect to asynchronous and synchronous updating.

INTRODUCTION

An Agent Based Model (ABM) is one where individuals (agents) within the system and their interactions are explicitly represented. Global properties are not explicitly modelled but emerge from the local interactions between agents. The asynchronous approach to simulating ABMs is espoused in (21). This paper distinguishes between Synchronous Updating (SU) and Asynchronous Updating (AU) and defines the differences between them. The SU approach is defined as having a global clock that synchronises the updating of all agent states so that all updates occur in unison. AU approaches, on the other hand, have no global clock and updates of agent state do not occur simultaneously. In effect agent interactions are applied sequentially in some random order. There are various approaches that determine how this random ordering is decided (7). It has been demonstrated that AU and SU implementations of the same simulation can result in widely differing behaviours. Today AU is used almost exclusively in ABM and Agent Based Social Simulations (ABSS). This leaves ABM open to the suspicion that its results might depend on the updating strategy used. We propose that when reproducing ABM/ABSS both AU and SU strategies be tried and compared, as done with Cellular Automata (CA) based simulations (3, 1). Results shown to be independent of the updating strategy used are more robust than those only demonstrated with one specific updating strategy. We produced a framework that implements many different updating strategies including new SU algorithms. These SU algorithms can handle the full range of interactions that occur in ABM. The practicality of these algorithms is demonstrated by using them to implement Sugarscape. Although issues with reproducing the results of Sugarscape have been reported (4) we show that the properties of Sugarscape are robust with regard to the updating strategy. In the next section we look at the updating strategies used in ABM and the differences between them. In section three we review different approaches to the replication problem in ABM and attempts to implement SU in ABM. Section four gives a brief overview of our framework. In Section five we show results suggesting that Sugarscape is robust to the type of updating strategy used. Finally we finish with a summary of our conclusions and briefly state how our work will progress in the future.

UPDATING STRATEGIES

Synchronous simulations are defined as those in which all the agents are updated simultaneously and instantaneously at each time step. Each step is a discrete quantum of time and the simulation progresses as a sequence of discrete states, one per time step. The state at step n is dependent solely on the state at step $n-1$. The best known simulation of this type is Conway's Game of Life (13). The AU model, on the other hand, is implemented by choosing an interval of time small enough so that at each step at most one individual agent is chosen at random to interact with its neighbours. During this update, the state of the rest of the system is held constant. The procedure is then repeated throughout the array for each agent in turn. Using AU, (33) developed a synchronous simulation of the Spatial Iterated Prisoners' Dilemma and showed that the simulation generates chaotically changing spatial patterns, in which *cooperators* and *defectors* both persist indefinitely.(21) used AU on the same simulation to instead show that the simulation always evolves, within 100 generations into a steady state where all agents become *defectors*. Thus we have two clearly contradictory results deriving from the application of the same rule and differing only in the updating technique. (6, 31) claim that AU is more realistic (i.e. more closely resembles the real world). Similarly (7) claims that synchronous behaviour is rare in the real world . There is not universal agreement on these points with (10) countering that AU is suitable only for instantaneous events which do not occur in biological systems. (36, 34) shown that the precise ordering used by AU affects the outcomes as well. (34) states this is often ignored in ABM even when showing results are reproducible. It has become standard practice to use AU for ABM and ABSS. AU has been adopted as standard by the major agent toolkits such as NetLogo, Repast, Mason and Swarm (32, 29, 2). ABSS, such as (9)'s Sugarscape, assume AU. The ability to execute a sequence of agent actions in a random order, an essential part of AU, forms part of the (35) *StupidModel* , a suite of models designed to test the suitability of any toolkit for ABM development. The preference for AU has been accentuated by a lack of synchronous algorithms that can handle the complex interactions that occur in ABM. This contrasts with Cellular Automata (CA) based simulation where, because the interactions are simpler SU algorithms exist. For example, CA-based Simulations of traffic flow, a real world system, employ SU (5) as standard. It is not uncommon to see CA based simulations employ both updating methods and comparisons made from the results as in (3, 1). Even in CA (14) states that the effects of the two updating techniques are not well understood. It is clear that the case for choosing AU over SU has not been convincingly made and, given that the outcomes of a simulation can depend on the approach taken, we need to take updating strategies into account when reporting results. An approach similar to that used in CA should be used where different strategies are used and the outcomes compared. We have produced new SU algorithms that can be employed on ABM and ABSS. These new SU algorithms provide modellers with the ability to run AU and SU based ABMs side by side for comparison.

RELATED WORK

The replication problem is simply stated: although simulations are developed in quantity and their results peer reviewed and published it has proven difficult for researchers to replicate these results: (18, 38, 8). Given the nature of ABM, where behaviour is defined in terms of simple agent interactions it might appear that reproducing an ABM defined by another researcher is easy but it has proven difficult. If we wish to check that the published results are accurate we must re-implement the simulation ourselves, seldom an easy task. If our results do not match the original we cannot be sure which implementation is correct. Modellers need to provide a clear and precise description of the model to work from. There have been a number of attempts to overcome these problems but none have been completely successful. If the published results of a simulation cannot be replicated then we can have little trust in these results. Overview, Design concepts and Details (ODD) (15, 16) is a protocol for specifying Individual Based Models (IBM). ODD was introduced to overcome shortcomings in published IBM definitions, specifically the issue that IBM are not specified precisely enough to allow for replication of the simulations. The ODD protocol consists of three blocks (Overview, Design concepts and Details) divided into seven sections:*Purpose; State variables and Scales; Process overview and scheduling; Design Concepts; Initialisation; Inputs and Submodels.* (12) has criticised ODD for not being formal enough. ODD has been extended by (17) who added an algorithmic model to specify the behaviours in the IBM .

(8) concluded that we cannot trust simulation results that have not been replicated. They illustrate this by taking a published simulation model and producing two independent implementations of it. They compared the results of their independent implementations against the original to find differences. The process of re-implementing a model is called *model alignment*. Following from the conclusion of (8) that experimentation is the only route to verification, (38) list two closely related approaches: *Re-implementation* where the model is rewritten following the original authors instructions and *Alignment or Docking* where a conceptually identical model is produced using different tools and different methodologies.

(20, 19, 18) identified the problem as belonging to the entire computational science field. The proposed solution is to produce *executable papers*, termed *ActivePapers*. That is, published papers will, as well as containing the usual text, also contain the full simulation model, executable code, source code, etc. all in the correct formats in a single HDF format. A similar proposal to *ActivePapers* is made by (26) for embedding data with existing formats (pdf, etc). Here the suggestion is that no new formats are required and what is missing is only the toolset to allow easy insertion of different data attachments to existing formats.

All these approaches are useful and give more information to experimenters but all have the same weakness. They only allow us to repeat an experiment exactly as it was done originally. Therefore such approaches do not deal with errors that are hidden in the original experiment. For example, if the updating approach used is the source of the problem then simply copying this approach will give us the same erroneous results. A higher level approach proposed in (23) uses formal methods to specify the model precisely. We build upon this work to replicate the results of Sugarscape.

Implementing SU in ABM

There have been a number of attempts at incorporating SU into ABMs. None of these has been completely successful. It is not clear how these approaches could handle interactions in Sugarscape such as *Combat*. The example ABMs that have been implemented using these approaches contained only simple interactions. The most complex interaction implemented is *Movement*, one of the simplest interactions available within Sugarscape.

In (11) a model of situated multi agent systems, *Influences and Reaction*, is derived from (28)'s *Situation Calculus*. It extends Situation Calculus to enable it to deal with simultaneous interactions. This is achieved by splitting state into two components. The first represents the state of the system (as per Situation Calculus) while the second represents any *influences* on the system that might cause the state to change. No algorithm to implement this approach is given so it is unclear how collisions (e.g. two agents try to move to the same location or take the same resource) can be detected and resolved.

Following on from (11), the *Influences and Reaction* model was used as a basis for a framework that converts ABM into equivalent synchronous CA (39). The resulting CA is called a *Transactional CA*. In the Transactional CA model updates occur using a three step approach outlined in (39). They used this to implement *Diffusion-Limited Aggregation*, a very simple model. It is not clear if it to could handle more complex interaction types.

(6) developed a model in the field of ecology that uses both SU and AU and compared the results. It was shown that the outcomes differed for AU and SU with the changes being more continuous in nature (smoother) in the AU version. They noted that the differences were more pronounced at higher population densities. It must be pointed out that although the changes in state (resource updates) were handled with SU the agent movement is handled with AU. A SU model of movement is required before this model is not fully synchronous.

In (5) a stochastic CA is used to simulate pedestrian traffic. The simulation uses a two dimensional lattice where each location can hold a maximum of one agent. Agents have a preferred direction of travel and from this a *matrix of preferences* is produced containing the probabilities of a move in each direction. Updating in this model is synchronous. This model displays oscillations in agent movement and other overall behaviours that do not appear in other models.

In (10) a SU implementation of the multi-turmite model is presented. It is used to demonstrate that different updating schemes have large effects on simulation outcomes. The model itself is very simple with only one type of agent and only one type of behaviour (27).

In (24, 25) a SU model of programming was developed for Computer Games. There are many similarities between ABM and computer games. The model used is based on many interacting heterogeneous agents with overall system properties emerging from their interactions. It employs the dual state common in SU along with message passing similar to that proposed by (39). However it allows for multiple passes of message between objects during each step.

In (37) a model that combined agents using SU alongside agents that used AU was introduced. This model of a supply chain used the SU/AU distinction to model the flow of information, or rather the delay in information propagation, within a network of agents. It is not clear how this approach could handle more complex interactions.

SYNCHRONOUS SUGARSCAPE

In order to demonstrate how our algorithms cope with the types of complex interaction that can occur in an ABSS we will use examples from Sugarscape, a canonical ABSS. (9) developed *Sugarscape* in order to investigate how individual behaviour can influence and cause different social dynamics within large populations. It has been used to show how, for example, inheritance of wealth affects resource distribution in populations and how disease can spread through a population. The rules of agent interaction defined in Sugarscape cover a wide range of interaction types and complexity. This makes Sugarscape an ideal benchmark. We used a formal specification of Sugarscape from (22) as our reference.

Sugarscape consists of 13 rules that range in complexity from simple independent rules (e.g. *Growback*) to the more complex read (e.g. $PollutionDiffusion$) and write dependent (e.g. *Combat*). It uses an $N \times N$ lattice (or grid) of locations upon which agents reside. In every location a food resource (known as sugar) grows up to some specified maximum amount. Each location can hold at most one agent at a time. Agents are mobile and can move across the lattice of locations in the four cardinal directions. When an agent arrives at a location it immediately consumes all the sugar at that location. Agents metabolise sugar during each step of the simulation and die if their sugar levels reach zero. Some of the rules of Sugarscape determine the rate of sugar growth (*Growback*, *SeasonalGrowback*) or pollution ($PollutionFormation$, $PollutionDiffusion$) at each location. The remaining rules determine how the agents move and interact with each other ($Movement$, *Combat*, *Disease*, *Culture*, *Reproduction*, etc.).

While the original version of Sugarscape assumes AU, the large range of rules employed in this ABSS makes it an excellent testbed for SU. The version of Sugarscape developed here is, to the best of our knowledge, the only SU implementation in existence and the most complex ABM/ABSS we know of using SU. The simplest rule combination of Sugarscape ($Movement$ and *Growback*) is equal to the most complex rules used by alternative SU algorithms and it is not clear how or if they could cope with many of the more complex rules. This imple-

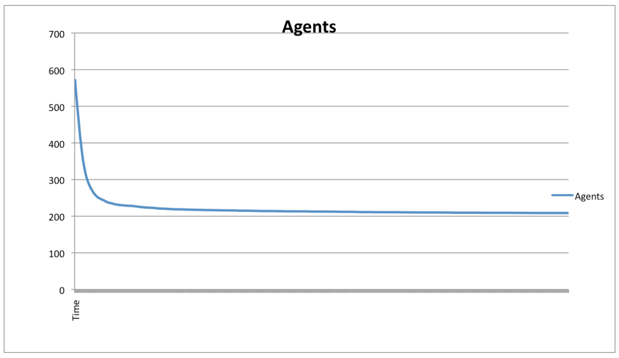

Figure 1: Lattice Carrying Capacity with Synchronous Updating

mentation demonstrates that our algorithms can handle the full range of possible agent interactions in any ABM. It also enables us to make comparisons between the original AU implementation of Sugarscape and our SU implementation.
At present we have only fully developed the single resource version of the simulation. The single resource version employs 12 of the 13 rules. The final rule *Trade* requires a second resource type, known as *spice*, so that *sugar* can be traded against *spice*. Although this means that the *Trade* rule is not implemented, the 12 implemented rules cover the full range of interaction types including 8 rules with a complexity equivalent to the *Trade* rule (e.g. *Combat* and *Culture*).

REPLICATING SUGARSCAPE

Due to ambiguities in the original Sugarscape definition, discovered in the specification process (23), it is not possible to know the exact values of all the simulation parameters. We have made our definitions match the original as closely as possible. Where the information is precise we replicated it and where there is ambiguity we have made a best guess. Even given these ambiguities we can still see if we can replicate the general emergent properties of the simulation[1]. One of the first emergent properties identified and measured was the carrying capacity[2] of the lattice.
In all cases the resulting graph (see Figure 1) of the carrying capacity matched the characteristic shape obtained by (9). Figure 1 shows the carrying capacity over time for a 50×50 lattice. In this case individual location maximum sugar carrying capacities ranged from 0 to 3 and were distributed evenly throughout the lattice. The initial agent population size was 600 and all agents were randomly assigned metabolisms ranging from 1 to 3 sugar units per step and a vision of 1. The results closely match the original outcomes and so cannot just be an artefact of the updating strategy. A stable population size is reached in the same manner independent of updating strategy.

[1]Trends in population growth, spread of culture over time, etc.
[2]The number of agents the lattice can support.

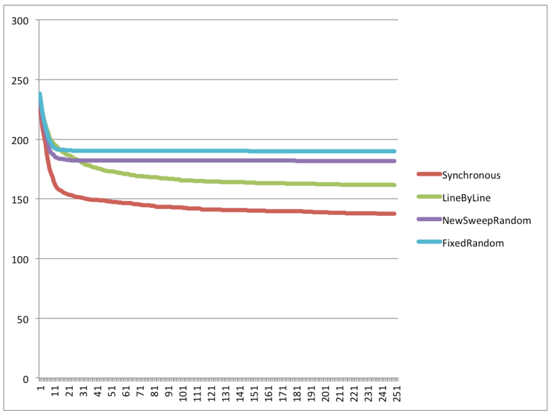

Figure 2: Lattice Carrying Capacity : Updating Strategies Compared

Table 1: **Carrying Capacity**

Strategy	Minimum	Maximum	Average
Synchronous	127	145	137
Line By Line	150	169	161
Random New Sweep	167	199	181
Fixed Random	172	198	190

To see if there is any difference in the stable carrying capacity when different updating strategies are used we compared the carrying capacities under each updating strategy while keeping all other aspects of the simulation identical. When we compare the carrying capacity of the simulation under the different updating strategies (Figure 2) we can see that although they all have the same distinctive shape there are some differences. The original AU strategy was *Random New Sweep*. This is the most commonly employed updating strategy in ABM and ABSS.
There are two points of interest.

1. The carrying capacity of the simulation under SU is noticeably lower than it is under *any* AU strategy. It is not clear why this is the case;

2. As we can see from Table 1, there is less variation in carrying capacity under SU. This is perhaps less surprising as SU is more deterministic that AU. Given the same initial state, the outcome of applying AU will depend on the random order applied to the agents during each step. Under SU this randomness is not present.

A question that arrises is whether the collision detection and resolution mechanism employed by the SU algorithm causes these differences. The synchronous *Move*

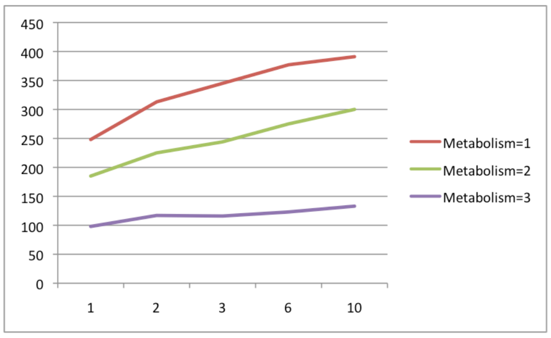

Figure 3: Carrying Capacity with varying Metabolism and Vision

rule resolves collisions between agents moving to the same destination by favouring the agent closest to the destination. This is more deterministic than the AU approach where the winner is chosen by running the moves in a random sequential order.

To test the hypothesis that the collision resolution mechanism is responsible for the differences in outcomes we compared two versions of the SU version of *Move*. The first applies the standard resolution mechanism as already discussed while the second determines the winners of conflicts at random. Each version was run with identical starting parameters so that the only differences between them is the collision resolution mechanism. For these simulations vision was set to seven for all agents so as to ensure overlap occurs when choosing destinations to move to. Each version was run 300 times and the resulting carrying capacities checked for any differences.

Table 2: **Carrying Capacity with different synchronous Movement Rules**

Strategy	Minimum	Maximum	Average
Closest Wins	210	267	210
Random Choice	211	272	211

As we can see from table 2 there is no discernible differences between either version. Our hypothesis that collision resolution causes these differences turns out to be false.

Effect of Metabolism and Vision on Carrying Capacity

It was originally demonstrated that carrying capacity is affected by both agent metabolism and agent vision. Agents with high vision and low metabolism are more likely to survive on the lattice. Thus carrying capacity is directly proportional to the average vision of the agents and inversely proportional to the average agent metabolism. These results were replicated using SU. The initial setup is as before but with vision and

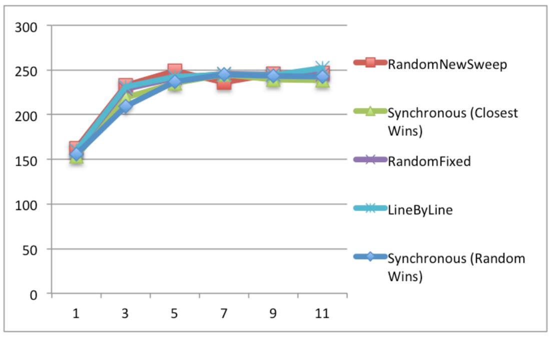

Figure 4: Carrying Capacity with varying Vision and Metabolism=1

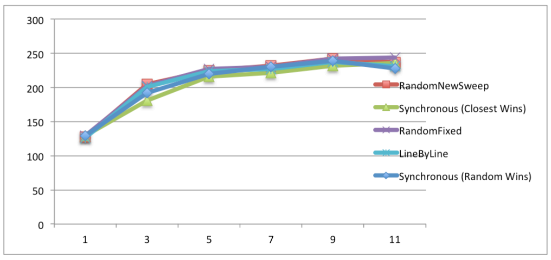

Figure 5: Carrying Capacity with varying Vision and Metabolism=2

metabolism fixed to a a constant value throughout the population. Figure 3 shows the effect of metabolism and vision on carrying capacity and this matches the original findings. To produce this graph the carrying capacity of the simulation was calculated when all agents were assigned fixed metabolism and vision values. The simulation was tested for metabolisms between 1 and 3 and vision from 1 to 11. It shows that carrying capacity is inversely proportional to metabolism as expected. It also demonstrates that for any fixed metabolism rate an increase in vision improves carrying capacity. The differences in carrying capacity when metabolism is fixed at 3 is due to the fact that the maximum sugar carrying capacity of any location is never more than 3. Therefore only agents reaching these high capacity locations can survive. The findings again show the robust nature of the general trends and properties in Sugarscape.

The differences in the specific values attained for carrying capacity can be accounted for by the differences in simulation set-up between the original lattice and this one. To see what specific differences exist between the SU and SU strategies the measurements were repeated on the simulations where the only difference was the updating strategy used. Figures 4 and 5 show that there is little discernible difference between the AU and SU approaches when metabolism is fixed at one or two sugar units per move. Figure 6, on the other hand, shows larger differences between AU and SU when the metabolism for agents is fixed at three units per move. This marked difference when metabolism is set to three

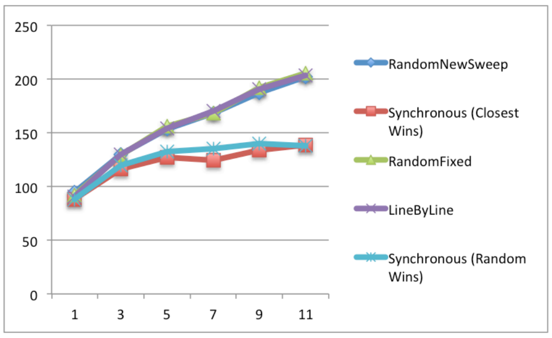

Figure 6: Carrying Capacity with varying Vision and Metabolism=3

is not entirely unexpected. The lattice is set up so that each location can contain at most three sugar units. This is a world of scarcity. An agent moving in the world can, at best, consume just enough sugar units to cover its costs. In the best case, by moving to a location with three sugar units, an agent can only cover its costs. It can never build up any sugar reserves. Every time it moves to a location with less than three sugar units its sugar reserves will deplete. The only locations that have the maximum capacity of three sugar units are based in two regions, the northeast and the southwest. Any agent not reaching these regions will die. Any agent forced to move away from these peaks (due to, say, overcrowding) will start losing sugar and probably die. There is little to no margin of error. Therefore any small differences between AU and SU will effect the population dynamics as the graph shows. With a lower metabolism, as in the other two scenarios, agents can build up reserves of sugar that allow them to compensate for not finding an optimal destination.

It is also the case that if agents vision is larger there will be more overlap between agents when they move and more frequent collisions (two or more agents attempting to move to the same destination). As collisions are handled differently by SU and AU we would expect any differences between the two approaches to be emphasised.

It is clear that the differences between AU and SU are subtle and affected by many different factors. Our previous graph of carrying capacity (Figure 2) showed differences between the two updating strategies with a different simulation setup.

Convergence of Culture

An agent's culture is determined by an attribute consisting of a bit string containing an odd number of bits. If an agent's bit string contains more 1's than 0's then we say they are members of the red group (or culture) and if they have more 0's than 1's they belong to the blue group. To simulate agents influencing each others culture or group membership we make neighbours swap a randomly selected bit. Agents can still move across the lattice and seek out the best sugar resources. This movement causes the agents to come into contact with different agents as the simulation progresses. The simulation employs the rules:*Metabolism*, *Growback*, *Movement* and *Culture*. Simulations were run until the agent culture completely converged to one colour. In the original experiment after approximately 2,700 steps the agents converge. We found little difference in the average times for convergence to occur regardless of updating strategy (approximately 2,000 steps using synchronous updating and 2,200 steps with asynchronous updating). The convergence time is highly dependent on the initial agent distribution. Occasionally convergence occurs quickly (approximately 1,200 steps) but it can take much longer to converge (about 4,000 steps) and even oscillate between the two cultures before convergence. This large variance makes it difficult to draw any conclusions about differences between the two approaches but in all cases convergence does occur as predicted. Our initial investigations have found no great differences between the AU and SU implementations. That is not to say that there will not be differences between the approaches. We would expect, based on previous work that differences will be present in the simulation. An example of the differences that can occur can be seen with the *Culture* rule. It also shows how difficult it is to predict how the updating strategies will affect ABMs.

CONCLUSIONS

In the CA literature it is not unusual to see the same CA implemented using AU and SU. The lack of SU algorithms for ABM has made this impractical. Our SU algorithm means that it is possible for these types of comparison to be made with ABMs. We have shown that it is possible to apply SU to complex ABM/ABSS by replicating Sugarscape. This is the only SU implementation of Sugarscape that we are aware of and the most complex ABM implemented with SU. We have shown that despite the disparate results obtained by applying AU and SU to Spatial Iterated Prisoners Dilemma these approaches do not always produce divergent results. In order to increase confidence of repeatability and reliability of ABM simulations different updating strategies should be used. This is the only way to insure that the results are not biased by updating strategy. More work is required on understanding how the choice of updating algorithm can affect simulations. Side by side comparisons of the two approaches we can help identify and explain the circumstances under which they diverge. Our analysis suggests that:

1. AU, by interfering with the speed of transmission of information across the simulation space, will produce less periodicity in the system than SU;
2. High density populations within systems are more

likely to enhance the differences in outcomes between synchronous and asynchronous implementations of the system.

Both seem to be borne out in the literature (6, 30). It would be a useful next step to have synchronous updating incorporated into simulation toolkits.

REFERENCES

[1] Lars Arve Bach*, David JT Sumpter, Jan Alsner, and Volker Loeschcke. Spatial evolutionary games of interaction among generic cancer cells. *Journal of Theoretical Medicine*, 5(1):47–58, 2003.

[2] Matthew Berryman. Review of software platforms for agent based models. Technical report, DTIC Document, 2008.

[3] Marija Bezbradica, Heather J Ruskin, and Martin Crane. Comparative analysis of asynchronous cellular automata in stochastic pharmaceutical modelling. *Journal of Computational Science*, 5(5):834–840, 2014.

[4] Anthony Bigbee, Claudio Cioffi-Revilla, and Sean Luke. Replication of sugarscape using mason. In Takao Terano, Hajime Kita, Hiroshi Deguchi, and Kyoichi Kijima, editors, *Agent-Based Approaches in Economic and Social Complex Systems IV: Post-Proceedings of The AESCS International Workshop 2005*, pages 183–190. Springer Japan, Tokyo, 2007.

[5] C Burstedde, K Klauck, A Schadschneider, and J Zittartz. Simulation of pedestrian dynamics using a two-dimensional cellular automaton. *Physica A: Statistical Mechanics and its Applications*, 295(3):507–525, 2001.

[6] Geoffrey Caron-Lormier, Roger W. Humphry, David A. Bohan, Cathy Hawes, and Pernille Thorbek. Asynchronous and synchronous updating in individual-based models. *Ecological Modelling*, 212(3 4):522 – 527, 2008.

[7] David Cornforth, David G. Green, and David Newth. Ordered asynchronous processes in multi-agent systems. *Physica D: Nonlinear Phenomena*, 204(1 2):70 – 82, 2005.

[8] Bruce Edmonds and David Hales. Replication, replication and replication: Some hard lessons from model alignment. *Journal of Artificial Societies and Social Simulation*, 6(4), 2003.

[9] Joshua M. Epstein and Robert Axtell. *Growing Artificial Societies: Social Science from the Bottom Up*. The Brookings Institution, Washington, DC, USA, 1996.

[10] Nazim Fatès and Vincent Chevrier. How important are updating schemes in multi-agent systems? an illustration on a multi-turmite model. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 533–540. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[11] Jacques Ferber and Jean-Pierre Müller. Influences and reaction: a model of situated multiagent systems. In *Proceedings of Second International Conference on Multi-Agent Systems (ICMAS-96)*, pages 72–79, 1996.

[12] Jose Manuel Galan, Luis R. Izquierdo, Segismundo S. Izquierdo, Jose Ignacio Santos, Ricardo Del Olmo, Adolfo Lopez-Paredes, and Bruce Edmonds. Errors and artefacts in agent-based modelling, 2009.

[13] Martin Gardner. Mathematical games: The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*, pages 120–123, October 1970.

[14] Carlos Grilo and Lu s Correia. Effects of asynchronism on evolutionary games. *Journal of Theoretical Biology*, 269(1):109 – 122, 2011.

[15] Volker Grimm, Uta Berger, Finn Bastiansen, Sigrunn Eliassen, Vincent Ginot, Jarl Giske, John Goss-Custard, Tamara Grand, Simone K. Heinz, Geir Huse, Andreas Huth, Jane U. Jepsen, Christian J¿rgensen, Wolf M. Mooij, Birgit M ller, Guy Pe er, Cyril Piou, Steven F. Railsback, Andrew M. Robbins, Martha M. Robbins, Eva Rossmanith, Nadja R ger, Espen Strand, Sami Souissi, Richard A. Stillman, Rune Vab¿, Ute Visser, and Donald L. DeAngelis. A standard protocol for describing individual-based and agent-based models. *Ecological Modelling*, 198(1 2):115 – 126, 2006.

[16] Volker Grimm, Uta Berger, Donald L. DeAngelis, J. Gary Polhill, Jarl Giske, and Steven F. Railsback. The {ODD} protocol: A review and first update. *Ecological Modelling*, 221(23):2760 – 2768, 2010.

[17] Franziska Hinkelmann, David Murrugarra, Abdul-Salam Jarrah, and Reinhard Laubenbacher. A mathematical framework for agent based models of complex biological networks. *Bulletin of Mathematical Biology*, 73(7):1583–1602, 2011.

[18] Konrad Hinsen. A data and code model for reproducible research and executable papers. *Procedia Computer Science*, 4(0):579 – 588, 2011. Proceedings of the International Conference on Computational Science, {ICCS} 2011.

[19] Konrad Hinsen. Computational science: shifting the focus from tools to models, 2014.

[20] Konrad Hinsen. Activepapers: a platform for publishing and archiving computer-aided research. *F1000Research*, 3, 2015.

[21] Bernardo A Huberman and Natalie S Glance. Evolutionary games and computer simulations. *Proceedings of the National Academy of Sciences*, 90(16):7716–7718, 1993.

[22] Joseph Kehoe. The specification of sugarscape. `http://arxiv.org/abs/1505.06012`, 2015.

[23] Joseph Kehoe. Creating reproducible agent based models using formal methods. In *17th International Workshop on Multi-Agent-Based Simulation*. Springer, May 2016.

[24] Joseph Kehoe and Joseph Morris. A concurrency model for game scripting. In *12th International Conference on Intelligent Games and Simulation*, pages 10–16. EUROSIS, Aug 2011.

[25] Joseph Kehoe and Joseph Morris. Scalable parallelism in games. Sixth York Doctoral Symposium, Oct 2013.

[26] John R. Kitchin. Examples of effective data sharing in scientific publishing. *ACS Catalysis*, 5(6), Jun 2015.

[27] C G Langton. Studying artificial life with cellular automata. *Phys. D*, 2(1-3):120–149, October 1986.

[28] Yves Lespérance, Hector J Levesque, Fangzhen Lin, Daniel Marcu, Raymond Reiter, and Richard B Scherl. Foundations of a logical approach to agent programming. In *Intelligent Agents II Agent Theories, Architectures, and Languages*, pages 331–346. Springer, 1996.

[29] Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, Keith Sullivan, and Gabriel Balan. Mason: A multiagent simulation environment. *Simulation*, 81(7):517–527, jul 2005.

[30] Robert M. Robert McCredie May. *Stability and complexity in model ecosystems*. Monographs in population biology. Princeton, N.J. Princeton University Press, 1973.

[31] David Newth and David Cornforth. Asynchronous spatial evolutionary games. *Biosystems*, 95(2):120 – 129, 2009.

[32] Michael J North, Nicholson T Collier, Jonathan Ozik, Eric R Tatara, Charles M Macal, Mark Bragen, and Pam Sydelko. Complex adaptive systems modeling with repast simphony. *Complex Adaptive Systems Modeling*, 1(1):1–26, 2013.

[33] Martin A. Nowak and Robert M. May. Evolutionary games and spatial chaos. *Nature*, 359(6398):826–829, October 1992.

[34] Wolfgang Radax and Bernhard Rengs. Timing matters: Lessons From The CA Literature On Updating. *Third Word Congress of Social Simulation*, abs/1008.0941, 2010.

[35] Steve Railsback, Steve Lytinen, and Volker Grimm. Stupidmodel and extensions: A template and teaching tool for agent-based modeling platforms, 2005.

[36] Graeme D Ruxton and Leonardo A Saravia. The need for biological realism in the updating of cellular automata models. *Ecological Modelling*, 107(2 3):105 – 112, 1998.

[37] Nihar Sahay, Marianthi Ierapetritou, and John Wassick. Synchronous and asynchronous decision making strategies in supply chains. *Computers and Chemical Engineering*, 71:116 – 129, 2014.

[38] Candelaria Sansores and Juan Pavn. Agent-based simulation replication: A model driven architecture approach. In Alexander F. Gelbukh, Alvaro de Albornoz, and Hugo Terashima-Mar n, editors, *MICAI*, volume 3789 of *Lecture Notes in Computer Science*, pages 244–253. Springer, 2005.

[39] Antoine Spicher, Nazim Fatès, and Olivier Simonin. Translating discrete multi-agents systems into cellular automata: Application to diffusion-limited aggregation. In Joaquim Filipe, Ana Fred, and Bernadette Sharp, editors, *Agents and Artificial Intelligence*, volume 67 of *Communications in Computer and Information Science*, pages 270–282. Springer Berlin Heidelberg, 2010.