Replacing Sugarscape: A Comprehensive, Expansive, and Transparent Reimplementation

Nathaniel Kremer-Herman[1(✉)] and Ankur Gupta[2]

[1] Seattle University, Seattle, WA 98122, USA
nkh@seattleu.edu
[2] Butler University, Indianapolis, IN 46208, USA
agupta@butler.edu

Abstract. We provide the definitive implementation of the seminal agent-based societal simulation *Sugarscape*. Our implementation is fully-featured open source software [5] that aims to make Sugarscape available for use by researchers across disciplines. It includes an extensive validation of all results shown in *Growing Artificial Societies* [2]. We also discuss the significant challenges in modernizing this groundbreaking body of work.

Keywords: sugarscape · computational social science · simulation validation · agent-based simulation

1 Introduction

Sugarscape was originally developed in 1996 and evaluated in *Growing Artificial Societies* [2]. It is an agent-based simulation platform that explored how societies might express *emergent behavior* based solely on the actions of individual agents. Sugarscape is a seminal work that has wide reach across many disciplines that require a carefully considered and feature-rich agent-based simulation.

Sugarscape is a two-dimensional $n \times m$ grid[1] landscape, where each grid cell contains some amount of the two available resources *sugar* and *spice*. These resources are metaphors for wealth and are gathered by agents who then consume sugar and spice over time. In Sugarscape, agents can lead long, rich lives where they interact with the environment and other agents.[2]

We present the following results:

1. We introduce the definitive implementation of Sugarscape and provides the full functionality displayed in [2].
2. Our implementation uses modern software development standards, is modular, transparently reproducible, and is highly configurable and customizable.
3. We tested all of the emergent behaviors described in [2]. We showcase several instances of our validated Sugarscape features.

[1] The grid is a *torus*. For example, an agent walking too far west ends up on the eastern side of the grid.

[2] Or, life can be *solitary, poor, nasty, brutish, and short* [4].

J.-L. Guisado-Lizar et al. (Eds.): SIMUtools 2023, LNICST 519, pp. 79–92, 2024.
https://doi.org/10.1007/978-3-031-57523-5_7

2 Related Work

The modeling of artificial life has a long lineage. John von Neumann's foundational work on self-reproducing automata [10] is insightful in the myriad ways he presaged then-future considerations of artificial life but also in his noting of the computational power necessary to meaningfully compute realistic, incredibly complex automata. An early workshop held by Langton et al. [8] helped jumpstart the study of artificial life. The impact of this initial workshop is still felt as the conference of the same name still convenes today. One of the initial and most significant attempts at using agent-based simulation for studying *social* phenomena was made by Thomas Schelling [11]. Epstein and Axtell extended the fundamentals from Schelling and took inspiration from John Conway's Game of Life [3] to create Sugarscape [2]: an agent-based simulation to observe emergent social behaviors. These early works were constrained by the computational power available at the time.

The Sugarscape model remains relevant decades after its creation. One recent work uses Sugarscape to explore implementing computational ethics [9]. Lasquety-Reyes provides an initial implementation of the Aristotelian virtue *temperance* as a quantitative formulation using the PECS (physical, emotional, cognitive, and social) model to represent the moderation of an agent's internal desires. Serrano and Satoh introduce a different extension to Sugarscape that models the effects differing pension and social security policies may have on society [12]. Kurakin [6] provides yet another extension called Technoscape to model a phenomenon studied by economic historians: the uneven coalescing of resources, trade power, and technology into the hands of a select few economic powerhouse states (represented as *cities* of agents in Sugarscape). The applications of Sugarscape are diverse, but none of these papers were able to leverage the true power of the fully-featured Sugarscape implementation.

3 Brief Overview of Sugarscape

The Sugarscape environment is a two-dimensional $n \times m$ grid. An initial allocation of the two resources *sugar* and *spice* is placed in each grid cell. These resources regenerate per-cell at preconfigured rates. Based on these user-defined regeneration rates, one can simulate particularly harsh or bountiful environments. Vanilla Sugarscape agents are born inherently greedy. Their sole aim is to live as long as possible. Every simulation timestep, an agent consumes their held sugar and spice according to their respective *metabolisms* for each. They also move a number of cells according to their *vision* to gather more resources. A faithful reimplementation of Vanilla Sugarscape would include additional, toggleable features for the environment (e.g. seasons, pollution, or disease) and for agents (e.g. trade, combat, or reproduction).

Vanilla Sugarscape is an elegant baseline for studying agent-based behavior because it demonstrates rich social dynamics ranging from simple, short-term behavioral rules to elaborate, long-term agent planning. It does a remarkable

job reinforcing our understanding of real world social phenomena. The level of abstraction in the simulation provides strong metaphors for the real world while remaining readily computable as it is driven by discrete numerical values [2].

Multiple versions of Sugarscape exist today. However, to the best of our knowledge, all publicly available versions remain incomplete. A popular implementation [13] of Sugarscape is provided with NetLogo[3], but it only replicates experiments from the first two chapters of *Growing Artificial Societies*. Most literature we found uses the NetLogo version (including the recent works noted previously [6,9,12]). Lange [7] implemented Sugarscape in Python with more functionality, but it was still not a complete representation of the book. His work inspired our own efforts in this area.

From private conversation with Epstein and Axtell, they consider the Bigbee [1] implementation (written in Java) to be the definitive edition of Sugarscape. However, even though it implements the majority of the features in [2], it *also* remains incomplete. We offer the first *complete* version of Sugarscape since the original implementation. Our software is freely available (along with validations of all experiments presented in the source book) in our code repository [5].

Table 1 compares the previously mentioned Sugarscape implementations and our own based on which features can be found in the software. We use the chapters of the source book represented in the code as a shorthand. Since the first and last chapter of the book do not introduce new features to the simulation, the experiments we reference are only found in Chaps. 2–5.

Table 1. Sugarscape Versions and Implemented Book Chapters

Implementation	Chapter 2	Chapter 3	Chapter 4	Chapter 5
NetLogo [13] Sugarscape	✓	✗	✗	✗
Python 2 Sugarscape [7]	✓	✓	✗	✗
MASON-Sugarscape [1]	✓	✓[a]	✓	✗
Our Implementation	✓	✓	✓	✓

[a] MASON-Sugarscape does not implement the combat feature found in Chap. 3.

4 Vanilla Sugarscape Features

We provide a brief overview of *Vanilla Sugarscape*, the original implementation of Sugarscape described in *Growing Artificial Societies* [2]. Vanilla Sugarscape has various features which can be selectively turned on or off. We categorize these features as those which affect the environment and those which affect agents. Many of these options are not simply Boolean switches; rather, one can assign a magnitude to the effect (such as setting the agents' maximum age range).

[3] NetLogo is a programming model and development environment for agent-based modeling which comes preloaded with various examples.

Agents are born, live in, interact with, and eventually die in the environmental landscape. The set of configured features enabled at the start of the simulation have quite pronounced effects on the resulting emergent societal behaviors.

4.1 Environment Features

The environment has toggleable features which may be enabled in a given simulated instance. Each has profound impacts for agents as they interact with the environment. These impacts are also expressed at the societal level.

Replacement. If an agent dies, a new agent is generated and placed randomly in the environment. By default, agents are *not* replaced in Sugarscape.

Seasons. Without loss of generality, seasons periodically shift between the northern and southern hemisphere of the environment. Hemispheres cycle between dry and wet seasons, which affect the resource regeneration rate of cells.

Pollution. Pollution is generated at a cell when an agent harvests sugar and spice from the cell and when they consume resources at the cell. This forever harms the cell's regeneration rate. Diffusion of pollution is also toggleable and governs how severely the pollution spreads to surrounding cells.

4.2 Agent Features

Agents consume sugar and spice according to their *metabolism* at each timestep. They die if they do not have enough resources to consume. In Vanilla Sugarscape, agents can move up to their *vision* to gather additional resources. We describe the following configurable agent features, which could have profound impacts on agent behavior:

Reproduction. Neighboring agents of appropriate age with sufficient resources can each expend some of their resources to produce *offspring*.

Trading. Agents can trade their sugar for spice and *vice versa*. Trading in Vanilla Sugarscape is always fair, meaning no agent can make a profit from trading.

Loans. Agents can loan resources to other agents with a specified repayment schedule and interest rate.

Inheritance. When an agent dies, their wealth and debt is passed along to another agent (usually their offspring). Inheriting debt provides more security for lenders should an agent starve or die of old age.

Culture. Agents have a bit vector of tags which represents their tribal affiliation. When agents meet in Vanilla Sugarscape, they exert cultural pressure upon each other that may change their tribal affiliations.

Combat. Only possible when *culture* is active. Agents may kill others from different tribes to take their resources. Agents consider the risk of retaliation before they kill.

Disease. Agents are born with an immune system, represented as a bit vector. Vanilla Sugarscape seeds some initial agents with starting diseases. Agents have a chance to contract diseases from neighboring agents; agents that recover from a disease are more resistant to reinfection.

4.3 Emergent Behavior in Vanilla Sugarscape

Emergent behavior refers to the behavior of groups of agents (a *society*) that transcend the available actions of any individual agent in the group. Vanilla Sugarscape agents exhibit observable collective social behaviors.

Group Migration. Since agents can only move orthogonally in the grid, societies that make diagonal migratory movements over time are exhibiting emergent behavior. These societies are incentivized to move toward a resource rich area of the grid when their current location becomes barren. Agents that cannot migrate instead hibernate, attempting to wait out the dry spell in hopes that the landscape recovers. Migrators bunch up and behave similar to a *flyer* in Conway's Game of Life [3].

With the season environment feature turned on, half the environment's sugar peaks do not replenish for a certain number of timesteps. This feature intensifies the need for agents to migrate or hibernate: agents with high vision flee during offseasons, whereas agents with low vision typically hibernate.

Formation of Tribes. Tribal formation is another example of emergent behavior that is best seen through the movement patterns of individual agents. Tribes are a consequence of agent tags. There are no explicit rules determining how agents *form* tribes. Rather, there is a simple mechanism for agents to influence each other's tags. When agents with very similar tags interact, they may stick together and become a tribe to avoid being attacked by agents with dissimilar tags. Aside from the tribal value as a protective group, they may instead become a hunting party and use their numbers to pick off lonely agents with dissimilar tags.

Stable Economy. We also observe the creation of an *ad hoc* (yet functional) economy as an emergent behavior. When the trading feature is enabled, agents can exchange goods with one another, and their valuation of sugar and spice stabilizes over time to create a healthy, dependable market for a fair exchange of goods. This free market value is synthesized from the bottom up as opposed to being dictated by some centralized arbiter. The evolution of this system is another example of emergent behavior. Thematically, this pattern is true of many other agent behaviors: introduce a simple rule, agents adhere to it collectively, and an emergent *social* behavior develops corresponding to that rule.

5 Validation of Vanilla Sugarscape

We validate Vanilla Sugarscape by recreating the experiments from the source material [2] in our implementation. Due to space constraints, it is not realistic to present all of our validation efforts, though we did test every experiment from Vanilla Sugarscape. All resources needed to reproduce our validations may be found in our source code repository [5].

Table 4 (found in Appendix A) lists the full set of validated examples found in the repository. We showcase three experiments from Vanilla Sugarscape here: seasonal migration, constrained reproduction, and disease transmission. Table 2 can be used when interpreting our Sugarscape visualizations; more saturated cell colors indicate the presence of a higher number of its associated resources:

Table 2. Visualization Color Legend

Cell Color	Meaning
Yellow	Sugar
Brown	Spice
Tan	Both sugar and spice
White	No sugar or spice
Other	Agent properties under study

Vanilla Sugarscape, although insightful, is not inherently easy to validate, in part since its documentation is incomplete. As a result, it is impossible to know the initial configuration for any particular result or rationale behind underlying design decisions for a given experiment. We address that deficiency in our reimplementation of Sugarscape and provide details as we go along. We parameterize our pseudo-random number generator with the fixed seed value of `12345` to establish a uniform baseline for initial conditions across experiments. We further organize all configurable parameters (such as agent max age ranges, maximum combat rewards, and resource growback) into a single configuration file. In this way, we produce a completely deterministic instance of Sugarscape.

Our descriptions of agent behavior in this paper are conveyed in qualitative, not quantitative terms, which mirrors the source text [2]. When considering the results of any experiment on any given seed, there will always be a few agents whose behavior reflects a blend of the influences that inform a particular experiment. As a result, agents may migrate more slowly or express a reproductive behavior that appears to oscillate between two relatively equal paths towards optimal resource collection. Our observational goal when considering *emergent*

behavior is to focus more on *majority trends* that govern agents, rather than the much lower frequency of actions of agents in the minority. In some sense, if we incorporate this minority behavior to a larger extent, we risk overfitting societal behavior based on this minority noise.

5.1 Seasonal Migration

The *seasons* environmental feature splits the world into northern and southern halves. At any given timestep, half of the world is experiencing a dry season where the sugar regeneration rate is severely reduced. Simultaneously, the other half of the world experiences a wet season where the sugar regeneration rate is normal. The two seasons switch at a regular timestep interval.

We describe the activity in this scenario in Fig. 1. Agents (indicated by red cells) are initially placed at random across the landscape. The south begins in a wet season, and we see the agents concentrate at the southern sugar peak. The northern agents either migrate south (given high enough vision to explore) or hibernate (given low enough metabolism to survive). Once the seasons change, we see mass migration from south to north.

Southern agents who can explore (having high vision) migrate north and find plentiful resources. Those who cannot (having low vision *and* low metabolism) hibernate through the dry season. Agents with both lower vision and high metabolism become scavengers, picking off the few remaining resources in the south. Due to natural selection, scavengers are the most likely to perish in the seasonal change. Figure 1 shows the environment in the initial stages of the southern wet season, the beginning of the seasonal transition, the agents migrating and hibernating, and finally a return to a southern wet season. This result verifies the behavior explained and demonstrated in pages 44–46 of *Growing Artificial Societies*.

5.2 Constrained Reproduction

Agents have multiple factors which impact their ability to reproduce. They have a fertility window that dictates the age range they can reproduce. Reproduction also has a configurable resource cost (in sugar and spice) that parents share. The cost to reproduce is based on their starting sugar and spice holds. In pages 64–66 of the source book, the authors study the effect of reducing an agent's fertility window by 10 timesteps. This change led to wild oscillations in population which, over the course of many generations, eventually stabilized.

Figure 2 shows a similar effect happening under the initial conditions. The population explodes in an initial boom due to plentiful access to resources and

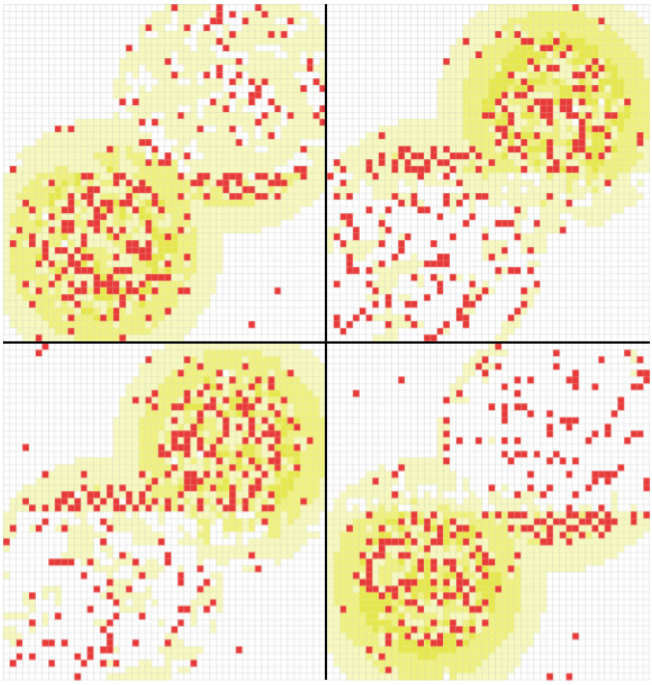

Fig. 1. Seasonal Migration in Sugarscape (Color figure online)

a population that reaches fertility age at roughly the same time. Soon after, the starting generation gets old and begins to die. We see our first big dip in population between timesteps 60 and 100, which corresponds to the maximum age range for agents.

The next generation experiences a similar pattern, though the population boom is not as pronounced. The dampened effect happens because resources are more scarce, since they are spread across more overall agents; also fewer agents have a synced fertility window. Additionally, there are fewer agents who are coming of age for this second boom. After three generations, we see the population has become more or less stable. Society is now less prone to generational death, since agents are more evenly distributed across all ages. At the end of the simulation, we find that the population has reached a steady state, which validates the results in pages 64–66 of *Growing Artificial Societies*.

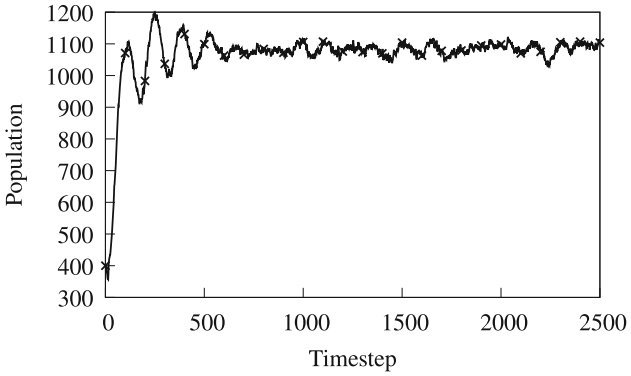

Fig. 2. Constrained Reproduction in Sugarscape

5.3 Disease Transmission

Diseases impact agent behavior by negatively affecting their attributes: increasing metabolism, reducing vision, rendering agents temporarily infertile, or other harmful effects. Diseases are also *highly* contagious since neighboring agents contract a random disease that the acting agent is carrying. Agents either die or recover from a given disease and become immune to it for a time. Diseases may be eradicated (most likely due to herd immunity), may become endemic to the society, or they may kill the society.

We compare Fig. 3 to the experiment on page 148 of [2], which shows an initial death event, eventual eradication of disease, and a final population recovery. An initial set of 10 diseases are placed on random agents. Red colored agents represent the sick while blue represents the healthy. At the beginning of the simulation, disease spreads *rapidly*, which causes an initial mass death. At this stage, a majority of the living population (a mere 165 agents) had contracted at least one new disease. The population battles disease over time, eventually leading to roughly 20 surviving agents either immune to infection or far enough away from a disease carrier to avoid further infection. Once rid of disease, society rapidly recovers and the population booms.

5.4 Further Validation with Random Seeds

In all the previous experiments, we use seed `12345` to validate our results; however, it is possible that this seed is an anomaly and does not represent the average case behavior. To that end, Table 3 shows abbreviated results on an additional 20 randomly generated seeds. A ✓ indicates the same or similar behavior as seed `12345`, whereas an `X` indicates behavior that was meaningfully different. For example, all seeds successfully demonstrated the migrators and hibernators behavior of the seasons experiment.

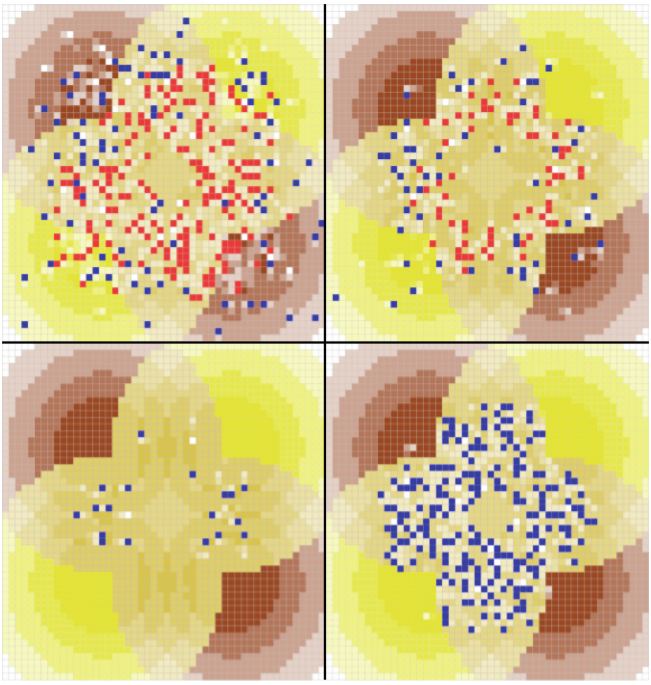

Fig. 3. Disease Transmission in Sugarscape

For the constrained reproduction experiment, seeds that differed from `12345` had especially resilient initial conditions and did not see the large population swings indicated in *Growing Artificial Societies*. The changes to the fertility window in these seeds appear to have made no meaningful difference.

For the disease transmission experiment, the authors of *Growing Artificial Societies* suggest the outcome as either an eradication of disease or endemic disease with a stable population. However, the two seeds with different behavior (`12881` and `22744`) completely died out from sickness. Seed `24022` nearly had endemic disease (as page 149 of the source book indicates), but the population eventually eradicated all disease.

5.5 Deviations from Vanilla Sugarscape

Although our implementation of Sugarscape largely matches the behavior of the original, there are cases where we deviate. The two most apparent deviations come from the trade and lending mechanisms which have uplifting effects on the number of agents the environment can support. This concept is called the society's *carrying capacity*. In both cases, on pages 111–112, the authors of the source book share data showing that trade and lending increase carrying capacity by 10% to 25%.

Table 3. Validating Vanilla Sugarscape with Random Seeds

Seed	Seasons	Reproduction	Disease
14133	✓	✓	✓
5172	✓	✓	✓
32209	✓	X	✓
23128	✓	✓	✓
15362	✓	✓	✓
12881	✓	✓	X
1484	✓	X	✓
19522	✓	✓	✓
23231	✓	✓	✓
24022	✓	✓	✓
5568	✓	✓	✓
13461	✓	X	✓
31844	✓	✓	✓
8885	✓	X	✓
21500	✓	✓	✓
28307	✓	✓	✓
28548	✓	✓	✓
22744	✓	✓	X
27167	✓	✓	✓
26341	✓	✓	✓

Magnitudes aside, this concept seems intuitively apparent: if an agent has a need for a resource, they could satisfy that need by trading with or borrowing from a neighbor, rather than depending on what is currently available in the environment. With more opportunities for agents to address their needs, we would *expect* a corresponding higher carrying capacity and higher reproduction to meet it.

In contrast, we were not able to find *any* seeds that demonstrate this significant uplift in carrying capacity. Our experiments show that increases from trade and lending average around 3%, with a maximum increase of 5% for all seeds we tested. Worse still, a few seeds saw a marginal *decrease* in carrying capacity!

We offer some possible reasoning for this mismatch. Epstein and Axtell focused their exposition on presenting the conceptual ideas clearly, and likely omitted many design and configuration details in the pursuit of a more accessible description of the complex and interconnected nature of the economic features of Vanilla Sugarscape. As experts in economic theory, Epstein and Axtell may have also written for an experienced audience, and we were unable to reconstitute that domain expertise. Finally, the authors of Vanilla Sugarscape were computationally limited by the hardware and software of the day, leading to potentially quite different design decisions from our implementation.

5.6 Additional Features in Our Implementation

We make many fundamental improvements to the underlying engine for Vanilla Sugarscape. For example, nearly all agent features (such as base interest rate or loan duration) now use a numeric range rather than a single value which allows for more granular behavioral variation. We also provide wholly new configurable parameters, such as `aggressionFactor` and `tradeFactor`, which indicate the likelihood an agent will engage in combat or trade respectively. A value of zero means the behavior is not present in the agent, with larger values indicating increasing prioritization of the associated action.

We also enhance the tracking of most agent interactions. Each agent keeps detailed records of their parents, children, neighbors, friends, lenders, borrowers, and traders throughout their lifetimes. We also maintain how often agents visit, trade, reproduce with, lend, etc. with each agent they meet. Though we do not use these tracking improvements in our current work, they provide us with a treasure trove of data we can investigate for future work.

The most significant, albeit simple, change is to consider resources as real numbers rather than integers. Trading and lending more cleanly map to exchanges of real number values rather than integer representations since rounding can punish agents when considering a loan or commencing a trade. The granularity of the trade or loan impacts its viability. In particular, it is more difficult to make a fair trade when agents cannot use fractional resource values to balance transactions. This change to the engine has profound implications on the realism for the Sugarscape simulation, making it a higher fidelity model for real world computations. Nevertheless, our observations on emergent behavior closely match the experiments in [2] and validate our approach.

5.7 Technical Details in Our Implementation

Our version of Sugarscape is written in Python 3. We chose this language to lower the barrier to entry for domain researchers who likely have more Python experience than other programming languages. The software repository [5] provides a `Makefile` to automate much of the setup and running process. The software requires a local `Bash` installation. Additionally, the Python code uses the `Tkinter` module *only* if it is run using the GUI as the module is conditionally imported.

The simulation takes a JSON (JavaScript Object Notation) configuration file as an optional input parameter and optionally produces a JSON log file as output. The code repository comes with a default configuration with most features enabled as well as the full set of examples from the source book [2] we have validated. The full details for running the software can be found in the included `README` in the repository.

6 Conclusions

Given its myriad applications across many different disciplines, Sugarscape serves as an important and feature-rich cornerstone to inspiring new and innovative

work that requires a carefully considered agent-based simulation. It is therefore incumbent upon the research community to maintain a modern, extendable, and robust version.

We humbly submit that we have created the *definitive version* of Sugarscape, together with the tools necessary to further research in the area. Our goals were to increase verification standards, integrate transparency, and provide a modern, modular software style that supports deep and meaningful customization. Along the way, we verify many of the results of [2], which are more easily reproduced in our software. Our code and validations are publicly available [5], and we invite the community to explore and use this seminal framework to study societal behavior.

Acknowledgements. We would like to thank Willem Hueffed, Maria Milkowski, and Joshua Palicka for their contributions to our Sugarscape software repository. This work was supported in part by the Seattle University Undergraduate Summer Research Award and the Holcolm Award Research Fund at Butler University.

A Appendix: Table of Validated Examples

Table 4. Full Set of Validated Examples

Example	Book Page Number(s)
Resource Collection with Immediate Growback	21–26
Resource Collection with Constant Growback	28–30
Seasonal Migration	43–45
Pollution	45–50
Reproduction	55–58
Constrained Reproduction	64–66
Inheritance	67–68
Cultural Tagging	72–79
Combat with Unlimited Reward	82–83
Combat with Limited Reward	86–90
Sugar & Spice	96–99
Trading	101–107
Trading & Agent Replacement	120–122
Trading & Pollution	127–129
Agent Foresight	129–130
Lending	131–133
Disease Transmission	141–147

References

1. Bigbee, A., Cioffi-Revilla, C., Luke, S.: Replication of Sugarscape using MASON. In: Terano, T., Kita, H., Deguchi, H., Kijima, K. (eds.) Agent-Based Approaches in Economic and Social Complex Systems IV. ASS, vol. 3, pp. 183–190. Springer, Tokyo (2007). https://doi.org/10.1007/978-4-431-71307-4_20
2. Epstein, J.M., Axtell, R.: Growing Artificial Societies: Social Science From the Bottom Up. Brookings Institution Press, Washington, D.C. (1996)
3. Games, M.: The fantastic combinations of John Conway's new solitaire game "life" by Martin Gardner. Sci. Am. **223**, 120–123 (1970)
4. Hobbes, T.: Leviathan: Or the Matter, Forme and Power of a Commonwealth Ecclesiasticall and Civil. A. Crooke, London (1651)
5. Kremer-Herman, N., Gupta, A., Palicka, J.: Sugarscape (2023). https://github.com/digital-terraria-lab/sugarscape
6. Kurakin, P.: Technoscape: a multi-agent model of all-human global web. In: 2022 15th International Conference Management of Large-Scale System Development (MLSD), pp. 1–5. IEEE (2022)
7. Lange, H.: Sugarscape (2022). https://github.com/langerv/sugarscape
8. Langton, C.G.: Artificial life. In: Langton, C.G. (ed.) Artificial Life: The Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems, held September, 1987, Los Alamos, New Mexico, pp. 1–48. Addison-Wesley, Redwood City (1989)
9. Lasquety-Reyes, J.A.: Towards computer simulations of virtue ethics. Open Philos. **2**(1), 399–413 (2019)
10. von Neumann, J.: Theory of Self-Reproducing Automata. University of Illinois Press, Champaign (1966)
11. Schelling, T.C.: Micromotives and Macrobehavior. W. W. Norton & Company, October 1978
12. Serrano, E., Satoh, K.: An agent-based model for exploring pension law and social security policies. In: Sakamoto, M., Okazaki, N., Mineshima, K., Satoh, K. (eds.) JSAI-isAI 2019. LNCS (LNAI), vol. 12331, pp. 50–63. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58790-1_4
13. Tisue, S., Wilensky, U.: NetLogo: design and implementation of a multi-agent modeling environment. In: Proceedings of Agent, vol. 2004, pp. 7–9 (2004)