To be presented to the AAAI 2021 Spring Symposium on
Combining Machine Learning and Knowledge Engineering, March 2021

# Guiding Parameter Estimation of Agent-Based Modeling Through Knowledge-based Function Approximation

William Broniec[a], Sungeun An[a], Spencer Rugaber[a,b] and Ashok Goel[a]

[a]School of Interactive Computing, Georgia Institute of Technology, 85 Fifth Street NW, Atlanta, GA 30308, USA
[b]School of Computer Science, Georgia Institute of Technology, 85 Fifth Street NW, Atlanta, GA 30308, USA

**Abstract**

Parameter estimation is a common challenge that arises in the domain of computational scientific modeling. Agent-based models offer particular challenges in this regard, and many solutions are too computationally intense and scale with the number of parameters. In this paper, we propose knowledge-based function approximation methods to deal with this problem in agent-based modeling. Our method is implemented within the VERA modeling system, and we show the validity of our methods using an internal model as well as an external model.

**Keywords**

Parameter estimation, Agent-based modeling, Genetic algorithms, Optimization, Scientific modeling

## 1. Introduction

VERA, the Virtual Experimentation Research Assistant, is an online modeling and simulation tool for the domains of ecology and epidemiology. VERA is the product of three fields: artificial intelligence, computational scientific modeling, and educational technology. The broad goal is to use AI to empower scientific research and provide a platform where novice scientists, such as students or citizen scientists, can enact and understand scientific thinking [1]. In VERA, users build conceptual models of observed phenomena and can rapidly iterate between stages of hypothesis, experiment, evaluation, and revision using agent-based simulations.

In the domain of computational scientific modeling, one common challenge that arises is the generation of a model that explains a set of data. Using a model, scientists can forecast future data and evaluate hypothetical "what-if" scenarios by altering the parameters of the simulation [2]. With the proliferation of machine learning technologies, many methods have been employed to automate the process of scientific modeling [3]. However, estimating and

*In A. Martin, K. Hinkelmann, H.-G. Fill, A. Gerber, D. Lenat, R. Stolle, F. van Harmelen (Eds.), Proceedings of the AAAI 2021 Spring Symposium on Combining Machine Learning and Knowledge Engineering (AAAI-MAKE 2021) - Stanford University, Palo Alto, California, USA, March 22-24, 2021.*
williambroniec@gmail.com (W. Broniec); sungeun.an@gatech.edu (S. An); spencer@cc.gatech.edu (S. Rugaber); ashok.goel@cc.gatech.edu (A. Goel)
http://dilab.gatech.edu (A. Goel)
0000-0002-0877-7063 (W. Broniec); 0000-0001-7116-9338 (S. An); 0000-0001-7116-9338 (S. Rugaber); 0000-0001-7116-9338 (A. Goel)

optimizing the parameters into the right set of values is a challenging problem because the high dimensionality of the parameter space typically requires a very large number of iterations [4, 5, 6, 7]. Given the large dimensionality of the problem, optimization techniques such as genetic algorithms (GA) can be used to explore the parameter space and find the best parameter set with respect to the optimization function.

VERA automatically generates agent-based simulation, which are formulations of macroscopic processes through the behavior definition of the individual components or "agents" that make up the system [8]. One major benefit of agent-based modeling is its expressiveness; no knowledge of differential equations is required to model complex real-world behaviors. However, an unavoidable drawback of agent-based simulation is time complexity [9]. Interactions between agents will introduce at least polynomial time complexity with regard to the number of agents, and interactions with even higher complexity may also be introduced. Therefore, parameter estimation of agent-based modeling may become long and tedious if we do not have an accurate, automatic, and systematic strategy to explore the parameter search space.

In this paper, we propose a method to guide the genetic algorithm to convergence in agent-based simulations for the parameter estimation problem (also called parameter optimization, "fit to model" or calibrate the model). The research question associated with this effort is:

- How can AI techniques be applied to agent-based simulations (ABS) in order to 1) given existing model $m$, induce process model $m^*$ such that $m^*$ better explains some underlying data? 2) analyze, ground, and provide an understanding of the emergent properties of that simulation?

To deal with a computational intensity problem, we propose a knowledge-based function approximation method that can optimize agent-based simulations. First, we developed a parameter ontology combining and grouping certain simulation parameters into four types according to their functions (e.g., trivial, isolated, per-behavior, or per-agent) in order to understand the underlying parameter characteristics and their behaviors involved in simulation and simplify the overall structure. Second, we use random variables and polynomial functions which could give a close approximation of the agent-based simulations while being much faster.

Our parameter optimization method is implemented within VERA. We illustrate the utility of the proposed method on an ecological model. Using synthetically generated data, the method successfully recovers the simulation outputs. While our function approximation is specialized towards VERA, we put forth an ontology for classifying simulation parameters in a way that can be used by other agent-based simulations, and we demonstrate this use with a simulation taken from the NetLogo standard library to ensure some external validity of the utilized method.

## 2. Related Work

### 2.1. Agent-based modeling

Agent-based modeling is a powerful simulation modeling technique that has seen a number of applications in the last few years, including applications to understand complex systems

and solve real-world problems [8]. In agent-based modeling (ABM), a system is modeled as a collection of autonomous individual components or "agents" that simulate real systems by interacting with each other within the environment. ABM serves as a “virtual laboratory” where alternative traits for key behaviors can be tested by plugging them into the ABM and testing how well the ABM then reproduces patterns observed in the real system. However, an important drawback of ABM is time complexity [9]. Agent-Based simulations most often operate on discrete-time units and further often employ parameters that are discrete and exceedingly difficult to formalize as differential attributes. Interactions between agents will introduce at least polynomial time complexity with regard to the number of agents, and interactions with even higher complexity may also be introduced. Regardless of optimization techniques employed, we necessarily will need to repeatedly make comparisons between our target data and our proposed simulation.

## 2.2. Inductive process modeling

Parameter estimation of a model given some data has been the topic of interest in the domain of computational scientific modeling. Using a model, scientists can forecast future data and evaluate hypothetical “what-if” scenarios by altering the parameters of the simulation. With the proliferation of machine learning technologies in the 90s, a new technique arose known as Inductive Process Modeling (IPM) with the goal of automating this process [3, 10]. Typically, machine learning models aim to implicitly replicate some unknown functions (e.g., neural network). That is, the “true” function may be very difficult to formalize, so it is approximated via an algorithm. In IPM, constraints for the algorithm and function space to map the problem are already in place. The goal is to determine a set of processes and the optimal parameters for those processes to fit the model to the data. There are a number of trade-offs between this approach and more common variants of machine learning. By mapping data to an explicit function, a level of transparency can be achieved to be interpreted and categorized very easily by scientists. However, the strong stipulation is that the model class must be pre-defined and sufficiently expressive. Because of these challenges, IPM is not a universal technique, and it has been used most successfully in modeling natural scientific phenomena [11].

## 2.3. Optimization in ABM

Optimization approaches including genetic algorithms have previously been applied to ABMs to reach global or near-global optima. However, the use of such metaheuristics in the context of ABM brings specific difficulties [5, 4, 6, 7]. First, the computation of the fitness function requires the execution of the interactions between agents, which implies a high time complexity. Second, although the property of emergence in ABM is powerful, it doesn’t naturally provide an explanation for how the result ties back to the parameters. Otherwise, the approach to understanding parameters come from statistical “Sensitivity Analysis” that can be used to determine the most important input variables to an output behavior within the model [6, 8, 12]. It is thus necessary to develop strategies to accelerate the convergence of the algorithm and to understand the parameters. In this paper, we are using a knowledge-based approach to guide the generic algorithm to address these issues.

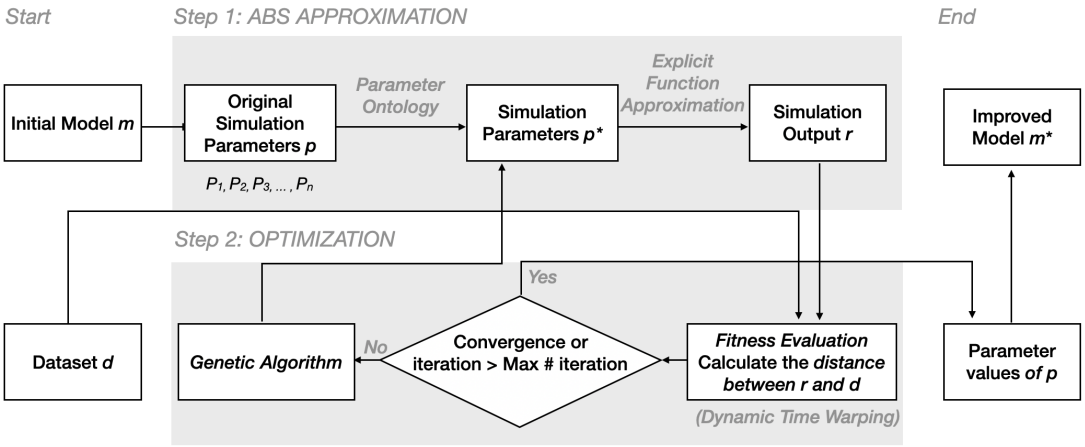



**Figure 1:** Overview of the proposed method for ABS approximation and parameter optimization.

## 3. Method

We propose a method for guiding parameter optimization of agent-based modeling by using knowledge-based explicit function approximation and genetic algorithms. Since the search space for optimizing an agent-based simulation is large, a parameter ontology is proposed to simplify the structure and reduce the computation while preserving their semantics, and then various functions are applied to approximate the agent-based simulation output. After the ABM approximation process, genetic algorithms are used to solve a combinatorial problem - finding the right sets of parameter values for different components. Our proposed method is implemented within VERA as Figure 1.

### 3.1. VERA

Our method for parameter optimization is implemented within VERA. VERA enables users to build a conceptual model by adding biotic or abiotic components and drawing relationships among them on the model canvas. VERA translates the visual conceptual model into the NetLogo simulation language to generate a simulation specification, which, when executed, produces an agent-based simulation graph. This allows users to explore ecological systems and perform "what if" experiments to either explain an existing ecological system or attempt to predict the outcome of future changes to one.

Table 1 provides templates of components and relationships for modeling ecological systems. VERA's templates including simulation parameters and relationship ontology are based on large-scale domain knowledge obtained from Smithsonian's Encyclopedia of Life (EOL) [13]. The template components include biotic organisms, abiotic substances, and habitat. A biotic component represents a population of organisms, each specified using eleven numerical simulation parameters (see param section of the biotic specification in Table 1). The abiotic components represent some non-biotic substances that are introduced into the ecosystem that have effects on the biotic populations. The habitat components represent physical regions in which

**Table 1**
Templates of components and relationships for modeling ecological systems.

```
Component biotic {
    Symbol: b
    Param: lifespan, body mass, offspring count, reproductive maturity, reproductive interval, respiratory rate (if applicable), carbon biomass (if applicable), assimilation efficiency (if applicable), photosynthesis rate (if applicable), move direction, move velocity
}

Component abiotic {
    Symbol: a
    Param: amount, growth rate
}
Component habitat {
    Symbol: h
    Param: drift direction, drift velocity
}
Relationship consumes {
    Direction: b -> b or a
    Param: consumption rate, interaction probability
}
Relationship destroy {
    Direction: b or a -> b or a
    Param: destruction rate, interaction probability
}
Relationship affects {
    Direction: b or a -> b or a
    Param: growth rate, interaction probability
}
Relationship produces {
    Direction: b -> a
    Param: production rate
}
Relationship becomes on death {
    Direction: b -> a
    Param: percent body mass
}
Relationship migrates {
    Direction: b or a -> h
    Param: consumption rate, interaction probability
}
```

the biotic and/or abiotic components interact.

A relationship template also includes a way in which various components can interact in a directed manner. The direction section of the relationship specification in Table 1 specifies the source and destination components of the relationship, which is an element of the Relations Ontology supported by EOL (e.g., "component X relates to component Y").

## 3.2. Explicit approximation

Given a dataset and an existing VERA model, we want to assist users in finding the parameter values that, when used to generate a simulation, yield results closest to that dataset. This can be formalized as an optimization problem where the inputs are the simulation parameters of a model and error is the distance between the simulation output and the initial dataset. However,

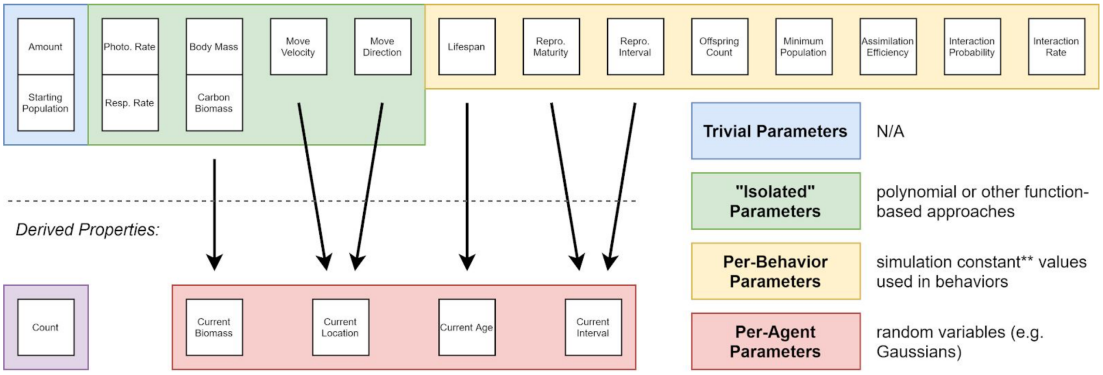



**Figure 2:** Parameter ontology (Trivial, isolated, per-behavior, or per-agent).

optimization of agent-based simulations is a computationally intensive problem as documented in other research. To support end-users in interactive scenarios, such as educational technology, we used two strategies to address the computational intensity of this problem. First, we developed a parameter ontology that reduces and maps out the parameter space. Second, we developed an explicit function approximation using random variables and polynomial functions to approximate the agent-based simulation results.

### 3.2.1. Parameter ontology

To effectively model an agent-based simulation, we developed a parameter ontology–combining certain parameters according to their function in the simulation. Understanding how the parameters are used is the essential part. First, a distinction needs to be made between agent properties and simulation properties. Agent properties are concerned with each agent in the simulation, and their values change each tick of the simulation's clock based on the agents' behaviors (e.g., age, location, etc.). On the other hand, simulation properties are constant values used to set up the simulation (e.g., starting population, lifespan, body mass, etc.) The top row of Figure 2 below shows the original parameters used in the agent-based simulation (e.g., trivial, isolated, per-behavior parameters) and the derived properties from the original parameters (e.g., per-agent parameters), color-coded by their category. Here are the descriptions of each parameter category:

- Trivial Parameters: Simulation values which set up the simulation's starting state
- Isolated Parameters: Parameters describing behaviors that only affect an individual agent and no others
- Per-Behavior Parameters: Parameters affecting interactions between different agents
- Per-Agent Value: Each agent tracks these core values internally
- Count: The output value in the simulation

The "stacked" parameters with pairs of blocks connected shown in the trivial and isolated parameters mean that these pairs of parameters are treated as a single parameter from the eyes of

the simulation. For example, "starting population" is used for biotic component (e.g., chicken) whereas "amount" is used for abiotic component (e.g., phosphorus); while the terms are different, they are semantically identical. In the simulation, the combinations of these parameters are calculated and shown in the graph. While all of these parameters are useful, the output graph only shows "count." Thus, "count" is what matters in the end result, not the individual parameters. Instead of optimizing each parameter individually and calculating them repeatedly (e.g., "lifespan" doesn't have to be calculated over and over), parameters are classified and simplified using explicit function approximation.

### 3.2.2. Random variables and polynomial functions

Using the parameter ontology, the approximation of the agent-based simulation output can be derived using random variables to model populations of agents and polynomial functions to model agent behaviors. Using distribution functions as stand-ins for population groups, we drastically reduce the number of computations performed and the memory used. Different populations may be more accurately modeled by specific distribution functions, but the Normal distribution serves as the best stand-in with an unknown distribution due to the central limit theorem. Therefore, rather than storing biomass for thousands of individual agents, a Gaussian can be represented using two variables, the mean and the variance, to describe the biomass for each age. The same process is applied to represent reproductive interval Gaussians as well.

When the simulation initiates (e.g., tick 0), VERA assigns each of the starting populations a random age from 0 to max age (e.g., lifespan-1) and sets the initial biomass value for each population, and the biomass follows a uniform distribution with the mean of initial biomass value and the variance of 0. Each tick of the simulation, the polynomial functions are applied to these populations to skew the distribution. For example, every simulation tick, a certain amount of biomass is lost from every agent due to its metabolism as determined by its respiratory rate, which will subtract from the mean while the variance doesn't change. However, when there is a relationship between two populations, such as predation, the corresponding consumption events will increase the average biomass for the predator agents, which changes the variance of the biomass. When the simulation is complete, a Gaussian distribution of the average biomass for a given population is generated where some agents will have more biomass than others within the same age.

## 3.3. Optimization

To obtain the closest values possible to the target dataset, an optimization algorithm is necessary to test and evaluate different parameter sets. Scientific models based on differential equations can rely on regression analysis to achieve this, but agent-based models typically lack such representations. Heuristic search is needed to explore the space, and due to the highly combinatorial nature of estimating parameters, genetic algorithms were selected. Figure 3 shows a standard genetic algorithm representation. The process begins with a set of individual members of a species which is called a Population. A species is characterized by a set of parameters (also known as Genes) that together determine the dynamics of the individuals of the species count (also known as a Chromosome).

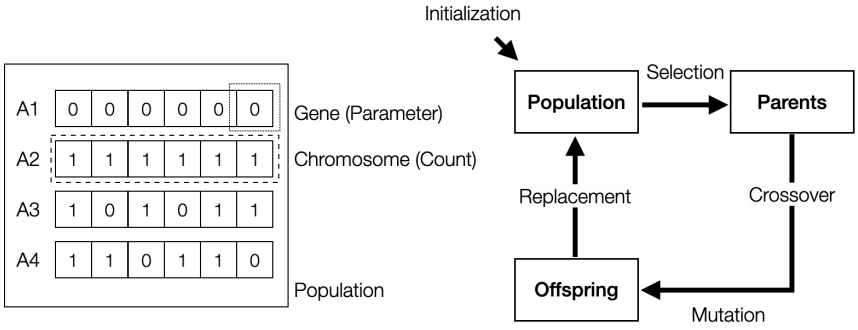



**Figure 3:** A standard genetic algorithm representation.

Figure 4 illustrates the pseudocode of our optimization method. The population of chromosomes is initialized randomly. Each chromosome is then evaluated using a fitness function that calculates the distance between the simulation output and the dataset: it comes to give a score to each chromosome. A selection is made among the population of chromosomes: we obtain a new population named parent population. Recombination and mutation operators are then applied to this population: we obtain a new population named intermediate population. The recombination consists in swapping parts between two chromosomes. With this operation, we obtain two new chromosomes. Intuitively the role of this operator is to pick up the best part of chromosomes to obtain a better chromosome. The mutation consists in changing a part of a chromosome. This operation avoids converging prematurely to a local solution. The new chromosomes of the intermediate population are evaluated. A new population is finally created from the initial population and the intermediate population, before starting again the whole process.

### 3.3.1. Fitness function

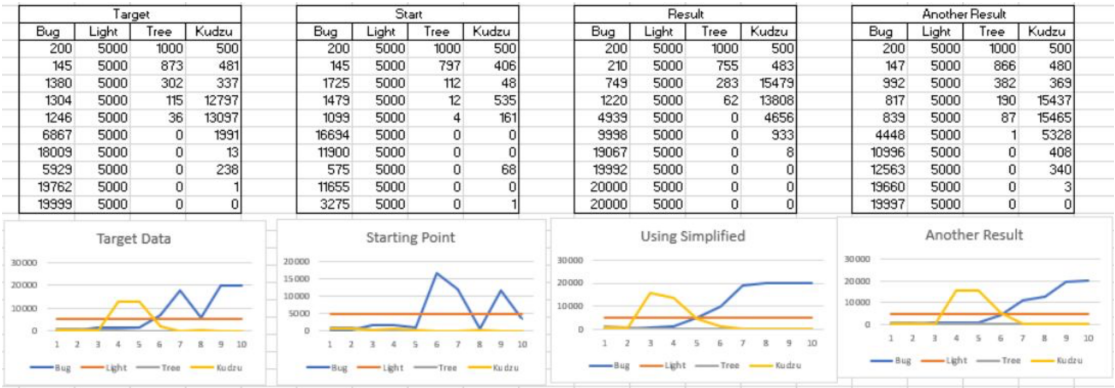
To evaluate how "fit" the simulation output r is with respect to the dataset d, we compare the similarity between the two sets of output graphs. Multiple methods including simply Euclidean distance can be used, but we used dynamic time warping (DTW), which is a robust, simple, and efficient measure for computing the dissimilarity between two time-series data [14, 15, 16]. DTW belongs to the group of so-called elastic dissimilarity measures, and works by optimally aligning (or 'warping') the time series in the temporal dimension so that the accumulated cost of this alignment is minimal. In its most basic form, this cost can be obtained by dynamic programming, recursively applying

$$D_{i,j} = \delta(x_i, y_j) + min(D_{i,j-1}, D_{i-1,j}, D_{i-1,j-1}) \quad (1)$$

for i = 1,...,M and j = 1,...,N, being M and N the lengths of time series x and y, respectively. As we are using distance as a fitness measure, we used negative distance to represent the fitness of the solution (larger fitness measure means better solutions).

```
START
Set up our initial population
Compute distance
REPEAT
    Selection
    Crossover
    Mutation
        Simulation Chromosome (Function Approximation)
    Compute distance
UNTIL it reaches maximum iterations
STOP
```

**Figure 4:** Pseudocode of our optimization process.



| Target | | | |
|---|---|---|---|
| Bug | Light | Tree | Kudzu |
| 200 | 5000 | 1000 | 500 |
| 145 | 5000 | 873 | 481 |
| 1380 | 5000 | 302 | 337 |
| 1304 | 5000 | 115 | 12797 |
| 1246 | 5000 | 36 | 13097 |
| 6867 | 5000 | 0 | 1991 |
| 18009 | 5000 | 0 | 13 |
| 5929 | 5000 | 0 | 238 |
| 19762 | 5000 | 0 | 1 |
| 19999 | 5000 | 0 | 0 |

| Start | | | |
|---|---|---|---|
| Bug | Light | Tree | Kudzu |
| 200 | 5000 | 1000 | 500 |
| 145 | 5000 | 797 | 406 |
| 1725 | 5000 | 112 | 48 |
| 1479 | 5000 | 12 | 535 |
| 1099 | 5000 | 4 | 161 |
| 16694 | 5000 | 0 | 0 |
| 11900 | 5000 | 0 | 0 |
| 575 | 5000 | 0 | 68 |
| 11655 | 5000 | 0 | 0 |
| 3275 | 5000 | 0 | 1 |

| Result | | | |
|---|---|---|---|
| Bug | Light | Tree | Kudzu |
| 200 | 5000 | 1000 | 500 |
| 210 | 5000 | 755 | 483 |
| 749 | 5000 | 283 | 15479 |
| 1220 | 5000 | 62 | 13808 |
| 4939 | 5000 | 0 | 4656 |
| 9998 | 5000 | 0 | 933 |
| 19067 | 5000 | 0 | 8 |
| 19992 | 5000 | 0 | 0 |
| 20000 | 5000 | 0 | 0 |
| 20000 | 5000 | 0 | 0 |

| Another Result | | | |
|---|---|---|---|
| Bug | Light | Tree | Kudzu |
| 200 | 5000 | 1000 | 500 |
| 147 | 5000 | 866 | 480 |
| 992 | 5000 | 382 | 369 |
| 817 | 5000 | 190 | 15437 |
| 839 | 5000 | 87 | 15465 |
| 4448 | 5000 | 1 | 5328 |
| 10996 | 5000 | 0 | 408 |
| 12563 | 5000 | 0 | 340 |
| 19660 | 5000 | 0 | 3 |
| 19997 | 5000 | 0 | 0 |



**Figure 5:** Results of using our methods on the agent-based simulation of VERA. (A) Left-most graph–Target data d. (B) Middle-left graph–Initial model m. (C) Middle-right graph–Improved model m* (< 2 minutes). (D) Right-most graph–Another improved model m* (= 2 minutes).

## 3.4. Results

Using the explicit function approximation in conjunction with genetic algorithms, we get results faster by orders of magnitude at the cost of some accuracy. Figure 5 shows the results of our methods. The left-most graph shows the target dataset (A), while the middle-left graph shows the output from simulating the initial model m (B). Between these two graphs, while the bug populations (indicated as blue lines) show a similar pattern where it starts to increase after 5 months, decreases and increases again at approximately 8 months, creating the valley. However, the kudzu populations (indicated as yellow lines) show very different patterns where the kudzu population in the initial model manifests as a horizontal line.

The middle-right (C) and right-most (D) graphs show two improved model m* using our function approximation methods twice. Since the mutation is random, each run of the simulation yields different results. In graph C (running time < 2 minutes), the kudzu bug population (indicated as the yellow line) is improved to resemble the target data while the valley in the bug population (indicated as blue lines) was absent due to compounding error in our approximation. In graph D (running time = 2 minutes), the kudzu bug population is further improved, and the bug population was improved creating the slight valley around 8 months.

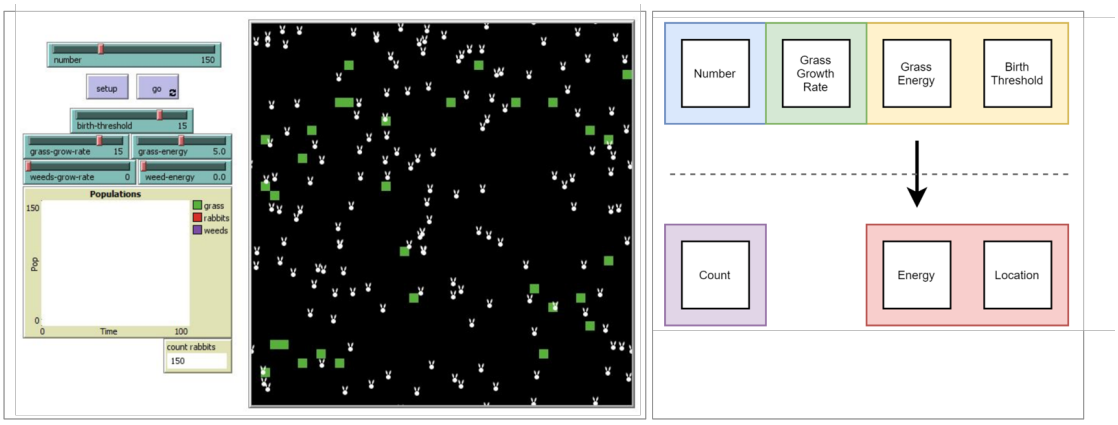



**Figure 6:** (A) Left: The "Rabbits, Grass, Weeds" simulation from the NetLogo example library. This screenshot shows the simulation interface in action with variable sliders on the left controlling the different simulation parameters. (B) Right: Parameter map in the "Rabbits, Grass, Weeds" simulation.

## 3.5. External validity

VERA is simply one engine for producing agent-based models. Being able to apply our techniques to different types of agent-based simulations would inforce the external validity of our methods. The “Rabbits, Grass, Weeds” simulation from the NetLogo example library was selected for two main reasons [17]. First, the example was also in the domain of ecology and posited a scenario (rabbits foraging for food) which could be replicated in VERA if desired but was written with entirely different simulation codes. Second, the example possessed only a handful of parameters, providing an example simulation more basic than VERA’s to work with. The "Rabbits, Grass, Weeds" simulation is a simplified model of a predator and prey between the rabbits, grass, and weeds. When a rabbit bumps into some grass or weeds, it eats the grass to gain its energy (see Figure 6). If the rabbit gains enough energy, it reproduces. Otherwise, it dies. This simulation consists of six parameters–starting number, birth threshold, grass growth rate, grass energy, weeds growth rate, and weeds energy. Each individual rabbit agent has two variable values associated with it–current energy and location. If a rabbit finds some grass, it will consume the grass and gain energy. If the rabbit finds weeds, it will gain no energy. During each tick of the simulation clock, the rabbits expend a fixed amount of energy, and a rabbit that runs out of energy dies, removing it from the simulation.

Using the same ontology described in the previous section to break down the simulation parameters, we get the following map as shown in Figure 6. Grass growth rate, grass energy, and birth threshold are combined to describe energy using polynomial functions, and energy and location of each agent is represented as a set of Gaussian distributions. The grass parameters affect the energy Gaussian of the rabbit population. Location is also a Gaussian in the simulation, but no parameters in this simulation control the location.

Figure 7 shows four graphs with sensitivity analysis of the different parameters–the blue line being the sensitivity analysis of the actual simulation and the orange line being that of the approximation. The x axis for these four graphs are the attempted parameter values, and the y axis is the difference in distance between the outputs. In other words, it shows how much

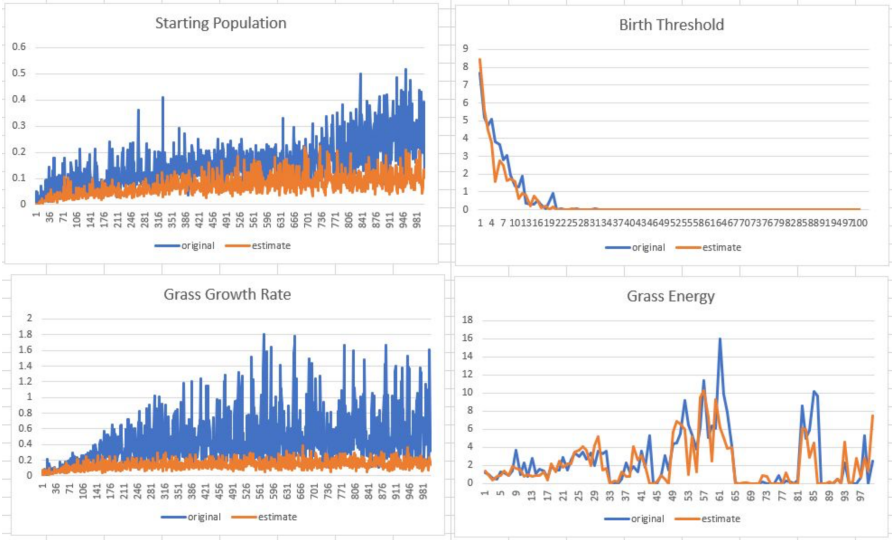



**Figure 7:** Results of Sensitivity Analysis of the different Parameters in the "Rabbits, Grass, Weeds" simulation. Blue line–Original. Orange line–Estimation. (A) Upper-left graph: Size of the rabbit population. (B) Upper-right graph: Birth threshold. (C) Lower-left graph: Grass growth rate. (D) Lower-right graph: Grass energy.

each parameter affects the simulation results. For example, starting population (A) and grass growth rate (C) have minor linear impacts on the output whereas birth threshold (B) is a sharp cutoff (e.g. if it's too high, rabbits will die before they have a chance to reproduce), and grass energy (D) is all over the place, having a stair step effect.

## 4. Conclusion

This paper presents a method as a route for speeding up the test phase of genetic algorithms of agent-based simulations as the use of genetic algorithms is computationally expensive for exploring the multiple parameter space. Specifically, we put forth an ontology for classifying simulation parameters in a way that can be used by other agent-based simulations. The validity of our methods was shown by the application examples in the internal VERA model as well as the external NetLogo predation model. Overall, our system works well for decomposing and understanding the semantic characteristics of the agent-based simulation parameters with exponentially faster results than optimization over the simulation itself. This affords the possibility of rapid, easy in end-user facing scenarios. The conclusions from these studies are a good starting point to investigate parameter estimation in the context of agent-based modeling tasks.

The primary drawback here was an error rate. With one species, the simulation was near-exact. With many more species over periods of time, it slowly began to deviate. Therefore, some important information may be missing, which can take the simulation to a completely different course afterward. Therefore, the next step is to develop additional strategies to reduce the compounding error in our approximation and to apply the method to more complex

examples. Another direction for further work is to conduct a user study to better understand how parameter estimation can facilitate model-based reasoning.

## Acknowledgments

This research is supported by an US NSF grant #1636848 (Big Data Spokes: Collaborative: Using Big Data for Environmental Sustainability: Big Data + AI Technology =Accessible, Usable, Useful Knowledge!) and the NSF South BigData Hub.

## References

[1] S. An, R. Bates, J. Hammock, S. Rugaber, A. Goel, Vera: Popularizing science through ai, in: International Conference on Artificial Intelligence in Education, Springer, 2018, pp. 31–35.

[2] W. Bridewell, J. N. Sánchez, P. Langley, D. Billman, An interactive environment for the modeling and discovery of scientific knowledge, International journal of human-computer studies 64 (2006) 1099–1114.

[3] W. Bridewell, P. Langley, L. Todorovski, S. Džeroski, Inductive process modeling, Machine learning 71 (2008) 1–32.

[4] E. Cabrera, M. Taboada, M. L. Iglesias, F. Epelde, E. Luque, Optimization of healthcare emergency departments by agent-based simulation, Procedia computer science 4 (2011) 1880–1889.

[5] B. Calvez, G. Hutzler, Automatic tuning of agent-based models using genetic algorithms, in: International Workshop on Multi-Agent Systems and Agent-Based Simulation, Springer, 2005, pp. 41–57.

[6] J. S. Lee, T. Filatova, A. Ligmann-Zielinska, B. Hassani-Mahmooei, F. Stonedahl, I. Lorscheid, A. Voinov, G. Polhill, Z. Sun, D. C. Parker, The complexities of agent-based modeling output analysis, The journal of artificial societies and social simulation 18 (2015).

[7] Z. Wang, J. Zhang, Agent-based modeling and genetic algorithm simulation for the climate game problem, Mathematical Problems in Engineering 2012 (2012).

[8] V. Grimm, U. Berger, F. Bastiansen, S. Eliassen, V. Ginot, J. Giske, J. Goss-Custard, T. Grand, S. K. Heinz, G. Huse, et al., A standard protocol for describing individual-based and agent-based models, Ecological modelling 198 (2006) 115–126.

[9] S. F. Railsback, V. Grimm, Agent-based and individual-based modeling: a practical introduction, Princeton university press, 2019.

[10] P. Langley, W. Bridewell, Combining data-driven and knowledge-guided methods to induce interpretable physiological models, in: 2011 AAAI Spring Symposium Series, 2011.

[11] P. Langley, A. Arvay, Scientific discovery, process models, and the social sciences, in: Scientific Discovery in the Social Sciences, Springer, 2019, pp. 173–190.

[12] B. Iooss, P. Lemaître, A review on global sensitivity analysis methods, in: Uncertainty management in simulation-optimization of complex systems, Springer, 2015, pp. 101–122.

[13] C. S. Parr, M. N. Wilson, M. P. Leary, K. S. Schulz, M. K. Lans, M. L. Walley, J. A. Hammock,

M. A. Goddard, M. J. Rice, M. M. Studer, et al., The encyclopedia of life v2: providing global access to knowledge about life on earth, Biodiversity data journal (2014).

[14] L. R. Rabiner, B.-H. Juang, Fundamentals of speech recognition, Tsinghua University Press, 1999.

[15] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, IEEE transactions on acoustics, speech, and signal processing 26 (1978) 43–49.

[16] J. Serra, J. L. Arcos, A competitive measure to assess the similarity between two time series, in: International Conference on Case-Based Reasoning, Springer, 2012, pp. 414–427.

[17] U. Wilensky, Netlogo rabbits grass weeds model, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. http://ccl. northwestern. edu/netlogo/models/RabbitsGrassWeeds (2001).