Techniques for Mathematical Analysis and Optimization of Agent-based Models

Matthew S. Oremland

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Mathematics

December 6, 2013
Blacksburg, Virginia

Keywords: agent-based models, optimization, heuristic algorithms, model reduction, discrete dynamical systems

Techniques for Mathematical Analysis and Optimization of Agent-based Models

Matthew S. Oremland

ABSTRACT

Agent-based models are computer simulations in which entities (agents) interact with each other and their environment according to local update rules. Local interactions give rise to global dynamics. These models can be thought of as *in silico* laboratories that can be used to investigate the system being modeled. Optimization problems for agent-based models are problems concerning the optimal way of steering a particular model to a desired state. Given that agent-based models have no rigorous mathematical formulation, standard analysis is difficult, and traditional mathematical approaches are often intractable.

This work presents techniques for the analysis of agent-based models and for solving optimization problems with such models. Techniques include model reduction, simulation optimization, conversion to systems of discrete difference equations, and a variety of heuristic methods. The proposed strategies are novel in their application; results show that for a large class of models, these strategies are more effective than existing methods.

This work was funded by U.S. Army Research Office grant # W911NF-09-1-0538. Results were in no way influenced by the funding agency.

Dedicated to 芳芳,
and all of my family.

Acknowledgments

I am grateful to many people for their contributions, both personally and professionally, to the work presented in this document. First and foremost I would like to thank Reinhard Laubenbacher, my advisor, for his continued support and guidance throughout my graduate studies. Dr. Laubenbacher was instrumental in my coming to work at the Virginia Bioinformatics Institute (VBI) and in addition to his research contributions, he has helped me navigate endless conferences, talks, and applications. I would also like to thank my commitee, Dr. Ciupe, Dr. Hoops, and Dr. Lawrence. As professors, colleagues, and collaborators, each have contributed meaningfully in the course of my studies.

I thank my fellow students at VBI, including Seda Arat, Madison Brandon, Cory Brunson, Franziska Hinkelmann, Claus Kadelka, Shernita Lee, David Murrugarra, and Kasia Swirydowicz, for making studying and working a continually interesting and engaging experience. I also thank professional mentors and colleagues Jennifer Galovich, Kathy O'Hara, Anael Verdugo, and Betsy Williams for their guidance and advice. I would also like to acknowledge my colleagues from a working group sponsored by the National Institute for Mathematical and Biological Synthesis entitled Optimal Control for Agent-based Models; these include Scott Christley, Paula Federico, Suzanne Lenhart, Rachael Neilan, and Rene Salinas. Many of the ideas presented in this work have come from ongoing discussions and meetings with the working group, and I am grateful to have been a part of it.

Several chapters in this document began as Research Experience for Undergraduates Summer projects, and I would like to thank Cara DeAngelis, Chris Gaskill, Eric Kernfeld, Herb Susmann, and Erin Twohy for being parts of those projects and contributing in a meaningful way to my own research. They were all a joy to work with, and I would gladly work with each of them again. Of course, research cannot be accomplished without funding, and I would like to acknowledge and thank the U.S. Army Research Office, NIMBioS, the National Science Foundation, and the Mathematics Department at Virginia Tech for supporting me and my work in various ways throughout my graduate studies.

Finally, I would like to thank 芳芳, my parents, and the rest of my family. They have always encouraged and supported me in everything I've chosen to pursue, and none of this could have been accomplished without them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Agent-based models (ABMs) are computer simulations consisting of entities (agents) that act according to local update rules. From these local rules, global dynamics emerge. Often times, the relationship between the local rules and the emergent global behavior is of primary interest – what are the local mechanisms which drive the larger system? Understanding this interplay is highly non-trivial: in some cases, global phenomena are robust with respect to changes in local rules, but in other cases, even the most subtle changes can have a substantial impact.

While some ABMs are quite abstract, many are rooted in some real-world system. A model may represent an ecology, a molecular interaction, a social network, or a citywide traffic grid, for example. The focus of this dissertation is on models of this type. Rather than focusing on how closely model dynamics hew to real systems, this work emphasizes techniques for analysis of ABMs: given an ABM of some system, what deductions can be made from the model, and what type of problems can be solved? The inherent stochasticity in many ABMs and the lack of a formal mathematical structure make standard mathematical approaches intractable – hence, there is a great need for the development of a rigorous mathematical framework for analysis of ABMs. The work contained here is meant to address that need, particularly in terms of solving optimization problems.

The meaning of *optimization* in this context is perhaps best explained via several examples. In a traffic network, what is the optimal timing schedule for stoplights in order to minimize the total amount of time spent waiting at red lights? In a cancer model, what is the best drug target, or the best delivery protocol for a given treatment? In a social rumor network, what is the best way to transmit information to the greatest number of people in the shortest amount of time? These are optimization questions because they ask about the optimal way to achieve a particular goal. ABMs are well-suited to questions of this sort: since simulation is (relatively) quick and inexpensive, ABMs act as *in silico* laboratories in which questions such as these can be investigated. Solving optimization questions for ABMs is not a trivial proposition. Stochasticity makes data unreliable, and the lack of mathematical formulation renders traditional optimal control theory inapplicable. The focus of this dissertation is on the development of techniques both for analysis of ABMs and for solving optimization problems.

This document consists of five manuscripts each emphasizing a different aspect of these goals. Background and related work are presented at the beginning of each chapter and hence omitted here. Throughout Chapters 2 – 5, variations of two running examples are considered. The first is an ABM of rabbits in a field. The rabbits hop randomly around the field, gaining energy from eating grass and losing energy by moving (and via reproduction). Considering these rabbits to be an invasive species, the optimization problem is to determine a poison schedule which minimizes the number of rabbits while also minimizing the amount of poison required. This model was chosen because it represents a general predator-prey system (with grass acting as prey); such systems hold interest for researchers from a variety of fields. Additionally, the model is spatially homogeneous – thus, the Rabbits and Grass model serves as a simple baseline example of the techniques introduced in the manuscripts.

The second model was one of the first large-scale agent-based models. It is known as SugarScape: in this model, ants traverse a landscape comprised of various amounts of sugar. Some ants can see farther than others, and some have slower metabolism; these advantageous features allow certain ants to survive longer than others. SugarScape represents a step up in complexity from the Rabbits and Grass model because the agents have more traits and because the model is spatially heterogeneous. While not present in the original formulation of SugarScape, this dissertation introduces a tax structure into the model. Periodically, ants are taxed for their sugar; tax rates are based on physical location and/or agent traits such as vision and metabolism. The optimization problem is to determine the optimal tax structure in order to minimize the number of deaths (ants die if their sugar is taken away) while maximizing the amount of tax collected. SugarScape was chosen as a representative model due to its popularity, complexity, and wide applicability – minor modifications have framed SugarScape as a model of economics, trade, culture, and even combat.

Each manuscript focuses on a different aspect of model analysis. Chapter 2 deals with simulation optimization techniques: that is, strategies that can be used directly on ABMs by obtaining results from repeated simulation. Since simulation optimization is often time-intensive, the issue of model reduction is of natural interest. Hence, scaling strategies are introduced in this chapter, in addition to heuristic methods for solving optimization problems via simulation alone. A metric known as Cohen's weighted κ is applied as a statistical measure of model similarity. In Chapter 3, the Rabbits and Grass model is translated into a system of discrete difference equations. In particular, the system of equations is derived analytically from the ABM rules. In the course of deriving the equations, this chapter emphasizes the effect that subtle rule changes can have on global dynamics. Specifically, the manuscript shows how changing the way agents move can substantially affect population dynamics, even when no other changes are made. Additionally, some mathematical analysis is performed on the equations that is not possible with the ABM alone – particularly, steady state and bifurcation analysis. Finally, the optimization problem is solved using the equations as a surrogate for the ABM. This precludes the need for lengthy simulation and provides a rigorous mathematical formulation to which a wealth of mathematical tools can be applied. Chapter 4 performs a similar procedure for a modified version of SugarScape: a system of difference equations are formulated that capture the pertinent dynamics of the ABM. This chapter focuses on the issue of heterogeneity, which is not present in Chapter 3. In particular, there are different agent categories and different spatial regions; Chapter 4 shows how these types of heterogeneity are addressed in the equation formulation. In Chapter 5, a more complicated version of SugarScape is analyzed. The landscape is considerably more heterogeneous than that examined in Chapter 4, and agent behavior is more complicated as well. In this case, the equations are not derived analytically. Rather, a technique known as *symbolic regression* is used to fit difference equations to ABM data. This manuscript lays out a framework for solving optimization problems for ABMs, using a variety of the techniques introduced in previous chapters. Finally, Chapter 6 introduces an agent-based model of invasive aspergillosis in the lung. This model was originally developed as a two-dimensional simulation during the course of a Research Experience for Undergraduates (REU) program in Summer 2012. In Chapter 6, a three-dimensional version is introduced. Rather than focusing on optimization techniques, this chapter shows how a complicated agent-based model can be built using results from literature and laboratories. While the Rabbits and Grass model and SugarScape are pre-existing models chosen for their wide applicability, the lung model in Chapter 6 is an original creation. This model serves multiple purposes: it shows how an ABM is built, and in doing so, emphasizes the layers of complexity that build up from relatively straightforward local interactions; at the same time, it serves as a template model to which the framework established in previous chapters can be applied.

As mentioned above, each chapter in this document is its own manuscript, meant to stand alone outside the dissertation. As such, there may be noticeable overlap in the subject matter among chapter introductions and in certain descriptions. This is because the chapters were prepared with journal submission in mind. The appendices contain formal descriptions of the ABMs examined in each chapter; these descriptions provide enough detail that all models and results can be replicated by independent research. Chapter 2 was submitted to a peer-reviewed journal in Summer 2013, and the remaining chapters will be submitted Fall 2013 or Spring 2014. Names of collaborators, acknowledgements, and the author's contribution to each manuscript are provided at the beginning of each chapter, where applicable.

Chapter 2

Scaling Methods and Heuristic Algorithms

2.1 Introduction

Agent-based models (ABMs) are often created in order to simulate real-world systems. In many cases, ABMs act as *in silico* laboratories wherein questions can be posed and investigated; such questions often arise naturally in the context of the system in question. For example, an ABM of a financial network might be used to determine which policies lead to maximized profit, while an ABM modeling social networks might be studied to determine the most effective means of transmitting information. Questions concerning how one can influence an ABM in order to best achieve some specific goal are *optimization* problems. In other contexts, optimization may refer to parameters or model design. It is important to reiterate that the meaning of the term in this study is different – it refers to the optimal choice of a sequence of external inputs to a model in order to achieve a particular goal. The stochasticity inherent in many ABMs means that care must be taken when attempting to solve optimization problems. Under fixed initial conditions, data from individual simulation replications often vary. Thus, careful analysis of ABM dynamics is a prerequisite for the development of optimization techniques. In particular, statistical methods must be brought to bear on the interpretation of simulation results.

In this study, statistical and optimization techniques are presented which can be applied directly to ABMs: translation of the model to an equation-based form is not necessary. There are several advantages to this approach – such techniques can be applied to virtually any ABM, and there is no need for transformation of either the model or the controls. Repeated simulation is used to obtain reliable results, and controls are applied directly to the ABMs. While there may be models for which this approach fails, the sufficiently broad examples provide good evidence that for large classes of ABMs, meaningful results can be obtained by direct analysis and optimization.

The goal of this paper is to introduce and illustrate a framework for solving optimization problems using agent-based models. In general, the number of possible solutions to an optimization problem is far too large for enumeration. Thus, heuristic methods must be employed to answer such questions. Computational efficiency is a key factor in this process; as such, the use of scaled approximations can be invaluable. As long as a scaled model faithfully maintains the dynamics of the original, it can be used to solve the optimization problem, resulting in a reduction of run time and computational complexity.

The paper is organized as follows: standards for data analysis are established and a statistical measure for model similarity is proposed. A heuristic technique known as Pareto optimization is proposed as a means for solving optimization problems. The framework is presented via the use of two models acting as representatives of large classes of ABMs, which ought to hold interest for researchers from a wide variety of disciplines. Brief model descriptions are outlined in the text, and detailed model descriptions following the Overview, Design Concepts, and Details (ODD) protocol for agent-based models [47, 48] are provided in the appendices. These descriptions ought to provide enough detail that the model (and results) can be reconstructed and verified by independent research.

2.1.1 Related work

Optimization problems of the type presented here have been studied in models of influenza and epidemics [67, 138], cancer treatment [82, 121], and the human immune system [6, 107], to name a few. Previous studies have investigated the effect of various model features on outcomes – for example, subway travel on the spread of epidemics [24], mobility and location in a molecular model [70], molecular components in a cancer model [129], and strategies for mitigating influenza outbreaks [86] – while not quite posing formal optimization problems. A study on the effect of ABMs in determining malaria elimination strategies [39] suggests that results from agent-based models are invaluable in the analysis of interventions.

In other studies, ABMs have been transformed into systems of differential equations [68] and polynomial dynamical systems [55, 125], among others. The importance of spatial heterogeneity has been examined in specific [51] and more general [50] cases, and predator-prey ABMs have been analyzed using statistical methods [135, 136].

2.1.2 A framework for solving optimization problems

The framework is summarized in Figure 2.1; subsequent sections motivate and explain the process in detail.

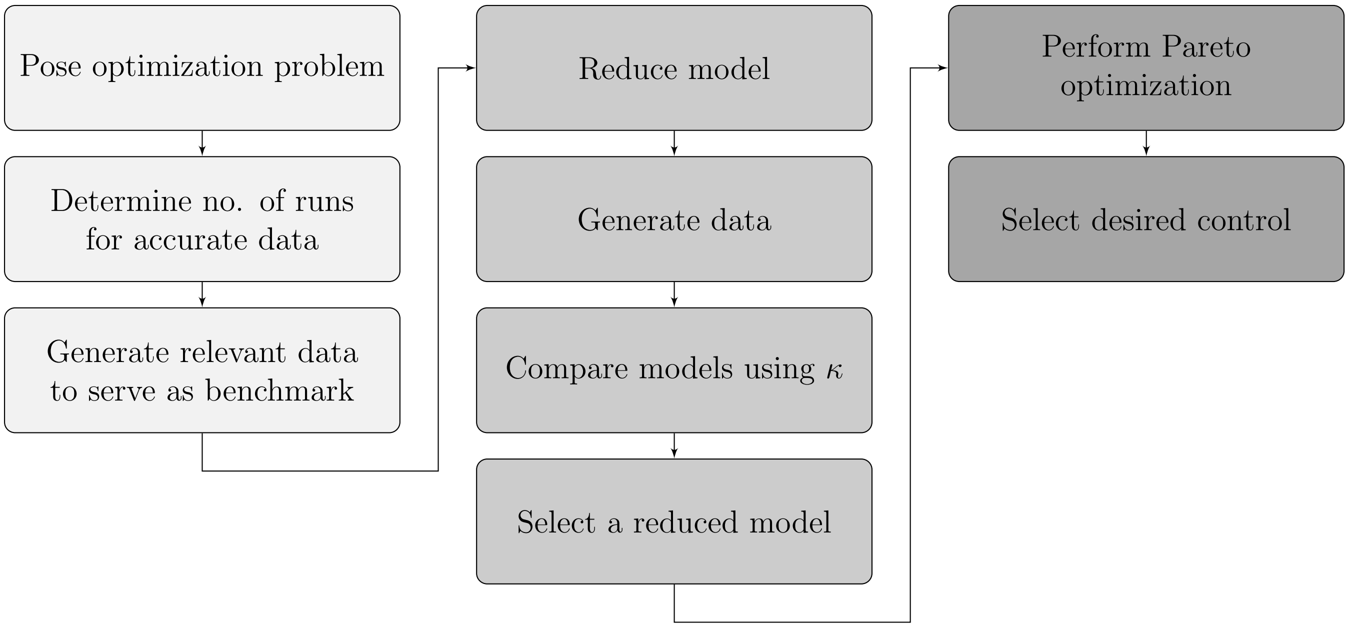

Pose optimization problem
Determine no. of runs for accurate data
Generate relevant data to serve as benchmark
Reduce model
Generate data
Compare models using κ
Select a reduced model
Perform Pareto optimization
Select desired control

Figure 2.1: An overview of the framework presented in this work. The three shadings represent the phases of analysis, scaling, and optimization.

2.2 Data reliability

A key factor in analysis of agent-based models is stochasticity. The approach suggested here is to examine how data averages change as the number of simulations (runs) increases. In many cases, the data will settle in on some average that is not improved upon by increasing the number of runs. Determining a sufficient number of runs is the first step in obtaining reliable results. The emphasis on solving optimization problems necessitates this process: while some of the stochasticity inherent to an individual run is lost when averaging over repeated runs, it is necessary in order to determine the general efficacy of one control versus another.

Agent-based models are often implemented on a grid, representing the 'space' of the model (often times, the grid indeed represents some physical space). Treating the original size and scope of the model as true, the goal of *scaling* is to determine the extent to which a model can be reduced without altering pertinent dynamics. The models examined here contain physical agents traversing physical landscapes. In this setting, the strategy is to gradually scale down the model until the dynamics no longer faithfully represent the original model. When applicable, this strategy results in reduced run time – in many cases substantially so – reducing the computational requirements for the solving of optimization problems and allowing access to a wider range of analytical tools.

Determining to what degree a reduced model is a faithful representation of the original is an important question. In terms of optimization, it is necessary to determine the extent to which models can be reduced *for the purpose of optimal control.* In order to accomplish this, a sample of the control space is implemented in both the original model and reduced versions. For each reduced version, the controls are ranked according to their effectiveness in regards to the optimization or control objective. The aim is to use a reduced model as a proxy for the original; thus, the ranking of the controls on the reduced model must be compared to the ranking of the same controls applied to the original.

We propose Cohen's weighted κ [21] as a measure of concordance of rankings for different model sizes. Let p_{obs} be the observed proportion of agreement in the two lists and let p_{exp} be the proportion of agreement expected by random chance. Then $\kappa = \frac{p_{obs}-p_{exp}}{1-p_{exp}}$. Hence if the lists are in perfect agreement, $\kappa = 1$; if the lists are no more similar than what is expected purely by chance, $\kappa = 0$. This similarity metric for ranked lists determines penalties based on the magnitude of disagreement. For details of how to calculate p_{obs}, p_{exp}, and weighted penalties, see [21].

For examples of the use of this statistic as a measure of agreement, see [41] and [37]. Cohen's weighted κ is chosen because of its wide documentation and implementation in a variety of studies; as such, there is precedent for this measure. There is no objective way to determine a benchmark value for κ. Several studies propose a κ value greater than 0.75 as being very good [3, 40], while others recommend a value of 0.8 or higher [74, 73]. In this study no benchmark is set; rather, κ values are assessed *a posteriori.* For more details on setting a benchmark for κ, see [116] and [35].

It is of course not guaranteed that all ABMs will be amenable to the strategies presented here (for discussion on this issue see [32]). In fact, models may exist for which no reduction is possible – nevertheless, reduction strategies are frequently useful and invariably informative. In particular, the investigation of differences in qualitative behavior can be served by these (and other) methods of model reduction. For examples of model reduction strategies applied to ABMs, see [144], [111], and [139]. It is also worth noting that 'model reduction' is a phrase whose meaning may be discipline-dependent: the extent to which a model can be reduced is dependent on which meaning is taken and which model details one wishes to preserve.

2.3 Pareto optimization

Once a suitable reduction has been made, an optimization problem can be solved using the reduced model as a surrogate for the original. Perhaps the most explored method for optimal control of ABMs has come in the form of heuristic algorithms. Given that enumeration of the solution space is often infeasible, heuristic algorithms are used to conduct a guided search of the solution space in order to determine locally optimal controls.

Several heuristic algorithms have been utilized in solving optimization problems for ABMs. Examples include simulated annealing [101], tabu search [127], and squeaky wheel optimization [79]. In this study, attention is focused on a certain type of genetic algorithm (GA). These algorithms, first brought to general attention in 1989 [46], are designed to mimic evolution: solutions that are more fit are used to 'breed' new solutions. GAs have been used in conjunction with ABMs to find optimal vaccination schedules for influenza [100], cancer [81], and in determining optimal anti-retroviral schedules for HIV treatment [15]. Vaccination schedule optimization results obtained from simulated annealing and genetic algorithms have even been compared and contrasted [96]. As the primary focus of this paper is to introduce a general framework for solving optimization problems for ABMs, a comparison of various heuristic methods is outside the scope of this study. For a more comprehensive look at heuristic control of ABMs, see [93].

The control problems described here have multiple objectives – this necessitates assigning weights to each objective. Determination of weights in multi-objective optimization problems can be problematic because *a priori*, the appropriate weights may be unknown – in particular, the assignment is at the discretion of the investigator. While there have been various proposals for these assignments (for an example, see [45]), any method which does not require weights has particular appeal.

Pareto optimization is just such a heuristic method: instead of a focusing on a single solution, the algorithm returns a suite of solutions. Solutions on the *Pareto frontier* represent those that cannot be improved upon in terms of one objective without some sacrifice in another. In this sense, each solution on the Pareto frontier is optimal with respect to some choice of weights. Thus, the 'managerial' decision of how to assign weights can be settled after the search has concluded.

An extensive list of references on multi-objective optimization techniques can be found in [20]. Pareto optimization has been selected for this study as it is novel in its application to ABMs. The algorithm adopted here is a minor variant of that described in [60]: it is a heuristic algorithm which searches the control space in an attempt to find the Pareto frontier. Pseudocode for the algorithm is presented in Algorithm 1.

2.4 Software

The proposed framework requires two types of software: modeling, and statistical analysis. Agent-based models can be implemented in a variety of software packages. Some of these, such as NetLogo [133], Repast [91], and MASON [84] have been designed for general agent-

Algorithm 1 Pseudocode for Pareto optimization algorithm.

```
generate random initial population of solutions
while $generation < max\_gens$ do
    evaluate current generation, call it $current\_pop$                         ▷ tally multiple objective data
    determine Pareto frontier; call it $current\_frontier$
    let $next\_pop = current\_frontier$                                         ▷ save frontiers between generations
    while size of $next\_pop <$ size of $generation\_size$ do                   ▷ each generation contains a fixed number of solutions
        repeat
            choose two candidate solutions from $current\_pop$
            let $comp\_set$ = random subset of 5 solutions from $current\_pop$
            if exactly one candidate is Pareto dominant over $comp\_set$ then
                that candidate becomes parent                                   ▷ add candidates more likely to be on frontier
            else if one candidate has fewer neighbors in solution space then   ▷ give preference to isolated solutions
                that candidate becomes parent
            else
                choose candidate at random to become parent
            end if
        until two parents have been chosen
        Breed two new solutions ($a$ and $b$) using parents:
            for all components in parent solutions do                           ▷ uniform crossover
                select component from random parent
                add this component to $a$
                add corresponding component from other parent to $b$
            end for
            $mutation\_rate = 0.20((max\_gens - generation)/max\_gens)$         ▷ mutation decreases over time
            for all components in $a$ and $b$ do                                ▷ perform mutation
                change component with probability $mutation\_rate$
            end for
        end routine
        add $a$ and $b$ to $next\_pop$
    end while
    increase $generation$ by 1
end while
```

based modeling. Other software has been developed for agent-based modeling in specific fields – these include C-ImmSim, VaccImm, and SIMMUNE for the human immune system [13, 137, 89], FluTE for influenza epidemiology [17], and SnAP for public health studies [10]. While one can always implement one's own toolkit for examining agent-based models, the use of established software can reduce both the variability between researchers' implementations and the learning curve for conducting research in this field. NetLogo was chosen as the modeling platform in this study, though there is no reason why the study could not have been undertaken using a number of different software packages. A standard NetLogo installation contains an extensive library of models from a variety of fields; the models discussed here are adaptations of popular models from this built-in library. Statistical analysis can be performed by virtually any statistical software package; in this study, Microsoft Excel was used. Due to the fact that simulation data was needed in order to perform Pareto optimization, this process was implemented in NetLogo as well. In general, the techniques described here are sufficiently straight-forward that highly specialized software is not needed, and the framework is not limited to any particular software choice.

2.5 Two models

2.5.1 Rabbits and grass

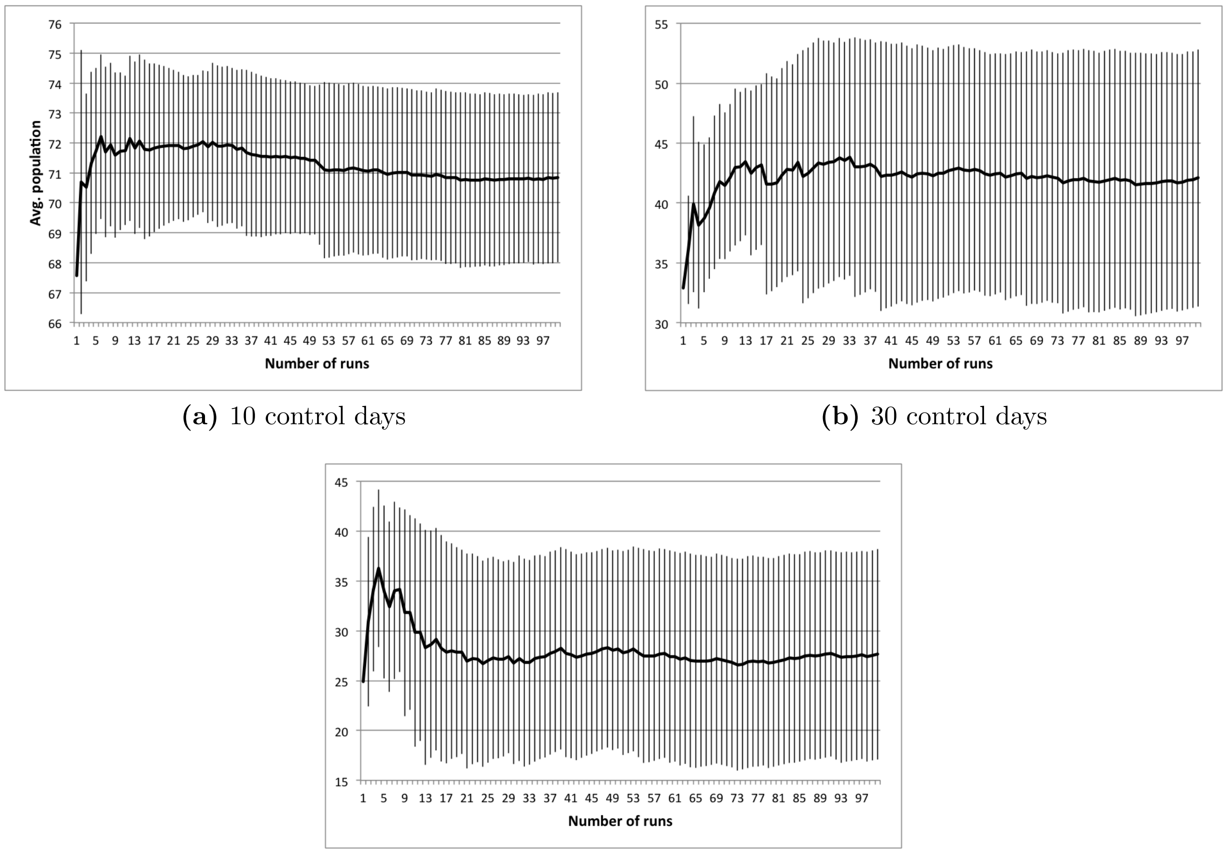
The first model to be examined is based on a sample model from the NetLogo library [133] involving rabbits in a field. At each time step, each rabbit moves, eats grass (if there is grass at its location), and then possibly reproduces or dies, based on its energy level. There are several compelling reasons for the use of this model as a test case for the proposed framework. One is that the model is sufficiently simple to describe, so results can be obtained, interpreted, and understood with minimal overhead. A more important reason is that this model represents the category of general predator-prey systems (with grass functioning as prey). Such models are commonly used in ecology and have been widely studied. Thus, the framework can be presented through an example that appeals to a broad community of researchers. Indeed, this model illustrates many concepts common in ABMs containing interacting species. A detailed description of the model and parameter values are provided in Appendix A.

Control consists of deciding (each day) whether or not to apply poison to the grid (i.e., uniformly to all grid cells). Specifically, the control objective is to determine a poison schedule u that minimizes the number of rabbits alive throughout the course of a simulation while also minimizing the number of days on which poison is used. Note that it is unlikely that one control schedule will minimize both objectives simultaneously – for example, the control wherein no poison is used certainly minimizes the second objective, but not the first. Thus, this problem is a good candidate for Pareto optimization: a suite of solutions can be found, each member of which is optimal depending on the weights assigned to the two objectives.

Scaling results

One of the control objectives concerns the average number of rabbits alive over the course of a simulation; thus, this is the pertinent metric in terms of model reduction (given that the other control objective – minimizing days on which control is used – is entirely preserved at any model size and for any number of runs).

As noted in Section 2.2, the first consideration when attempting to scale the model is determining the number of runs necessary in order to achieve reliable results. To this end, several control schedules were selected at random. Each was applied to the original 50×50 model, and results were tallied up to 100 runs. Population dynamics for three randomly selected control schedules are presented in Figure 2.2: plots show how the average number of rabbits alive over the course of the schedule change as the number of runs increases, with error bars representing one standard deviation. These three schedules represent three distinct regions of the control space, in that each contains a different number of control days. Note that in all cases, there is little change in the mean or the standard deviation beyond 50 runs – this suggests that there is no advantage in averaging over more than 50 runs. It is important to note that if the control objectives were altered (for example, if grass levels were of interest rather than rabbit levels) then this conclusion may not hold. In particular, the necessary number of runs depends upon the model dynamics of interest – in this case, the average number of rabbits.



(a) 10 control days

(b) 30 control days

(c) 50 control days

Figure 2.2: Average population values are not improved upon by using more than 50 runs.

Once a benchmark for reliable results has been established, various model reductions can be investigated with respect to control. In the original model, there are $50 \times 50 = 2500$ patches and 150 rabbits initially. A **model size** of M means that the world width and height are both M. Hence, when reducing the model to size M, the initial number of rabbits ought to be $150 \times \frac{M^2}{2500}$, in order to maintain the same proportion of rabbits to model size. All other state variable values remain the same.

For each of a set of controls applied to the original model, average rabbit numbers are obtained via simulation; the controls can then be ranked by these numbers. These same controls can be applied to a reduced model, resulting in a (potentially) different ranked list. The κ statistic (see Section 2.2) measures the similarity between the two rankings, thereby serving as a measure of the extent to which the reduced model serves as a substitute for the original. It is important that these rankings are maintained over a wide range of control schedules, since solving the optimal control problem will involve the potential examination of the entire solution space.

Generating a set of controls randomly results in a normal distribution centered on solutions with fifty zeros and fifty ones. To avoid focusing on too narrow a portion of the control space, a stratified random sample was taken: 24 values $N_1, \ldots, N_{24}$ were chosen as **frequency numbers**, representing the number of ones (i.e., poison days) in the schedule. These values were chosen at random within the following scheme: three values were chosen between 1 and 10, three between 10 and 20, and so on, with the final three chosen between 70 and 80. Four control schedules were then randomly generated, each containing N_k ones and $100 - N_k$ zeros (distributed randomly throughout the schedule), for $k \in \{1, \ldots, 24\}$. Thus, for each trial, a total of 96 schedules were evaluated, chosen as a stratified random sample of the solution space. Note that schedules with more than 80 non-zero entries were not considered, as preliminary investigation showed that such schedules were quickly eliminated from any heuristic optimal control search.

One trial is defined as follows: 96 control schedules (chosen according to the above description) were run using the original $M = 50$ model, and then again on the model at each of the following model sizes: 50, 40, 30, 20, 10, 5, and 3. Note that the schedules are run *twice* on the original $M = 50$ model: this is done in order to establish how consistent the rankings are when evaluated twice on the same-sized model. In some sense, this serves as validation of the choice of 50 simulations as being sufficient for reliable results, and also provides insight into the analysis of an appropriate benchmark for κ, as will be seen below.

Evaluation of 150 schedules (each averaged over 50 simulations) at model size $M = 50$ requires approximately 3 seconds. Table 2.1 gives the number of simulations that can be run for the reduced models in approximately 3 seconds. Given that the primary advantage

World size	Simulations	Avg. run time (sec.)
50	50	3.04
40	75	3.00
30	135	3.05
20	290	3.03
10	1100	3.03
5	3500	3.01
3	5700	3.03

Table 2.1: Number of simulations in equivalent run time.

for scaling models is to reduce run time, it is more appropriate to compare data based on equivalent run time rather than using a fixed number of simulations for each size. This data aids in scaling analysis: if one wishes to reduce the run time by 50%, the number of runs that can be performed is easily calculated.

Figure 2.3 summarizes κ values for various world sizes and run times. Each data point represents the mean taken over ten trials, with error bars representing one standard deviation. A benchmark value of $\kappa = 0.8$ is plotted as well – it is presented to serve as a preliminary gauge of how well the reduced models capture the dynamics of the original. Each line on the graph connects data points of equivalent run time. Figure 2.3 helps identify

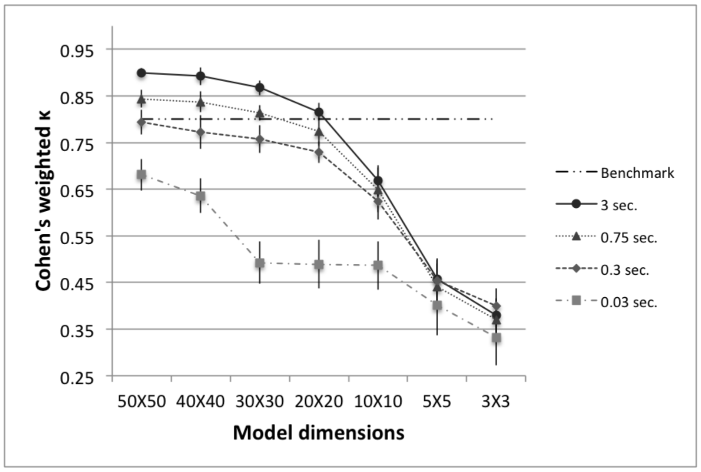


Figure 2.3: Cohen's weighted κ for various world sizes and run times.

unviable reductions: accepting a benchmark of $\kappa = 0.8$, world sizes below 20 are not sufficiently accurate representatives of the original model (and the size 20 model is only sufficient at 100% run time). The data also show that if one insists on using the size 3 model, the benchmark for κ will have to be lowered. It further shows that if one wishes to use the size 3 model *and* insists on a κ value higher than 0.8, it will certainly require an increase in the run time of the model (and even then, may not be possible).

Several important conclusions can be drawn from this data: one is that if the priority is achieving the highest possible value for κ, then the original size 50 model is always the best choice for any fixed run time. This is perhaps unsurprising, as one can only expect to lose some accuracy as model size decreases. Another important conclusion is that if the only priority is decreased run time, it is always better to use fewer runs of the size 50 model rather than more runs of a smaller model. This follows because each line represents a fixed run time, and for any fixed run time, the size 50 model results in the highest value for κ. A fixed benchmark for κ further informs a researcher with a priority of reduced run time: as the data show, if one wishes to keep κ above 0.8, then it is possible to reduce the run time by 90%, but not further (as indicated by the data for the size 50 model at 0.3 seconds). Thus, not only can one determine which world size should be used in order to obtain minimum run time, but also the minimum run time that can be achieved in order to maintain a pre-set benchmark for κ.

There may be cases where a reduced model is of particular interest – for example, [55] describe methods for translating ABMs into polynomial dynamical systems (PDS), offering advantages such as steady state and bifurcation analysis. The number of required equations may be too large for the size 50 model, but not so for the size 10 model. A similar concern applies to differential equations approximations. Examples of these considerations are discussed in [69] and [68]. Hence, depending on the priority of the modeler, the data here shows which world sizes may be used and what κ values can be expected when doing so. Furthermore, note that for smaller world sizes the range of κ values is decreased. In particular, in order to achieve 1% run time on the 50×50 model, κ decreases from 0.90 to 0.68. However, in order to achieve a 1% run time on the 3×3 model, κ decreases from 0.38 to 0.33. Thus, for smaller models κ may be less affected by a decrease in run time.

In addition to the above conclusions, the data informs the study of points at which the dynamics of the model undergo a qualitative change. There is a larger change in reducing from world size 10 to 5 than there is in going from size 20 to 10; this indicates the dynamics are more rapidly changing between world sizes 10 and 5. In particular, the data seem to suggest that the pertinent dynamics are not drastically altered between the original size 50 model and the size 20 model, but change rather quickly at smaller sizes. This is of particular interest in light of the fact that the original world size was chosen more or less arbitrarily. If one began with a size 10 model, it may not be possible to reduce it to the same extent that one can reduce a size 50 model.

As mentioned in Section 2.2, several studies suggest a κ value of 0.8 as a benchmark for sufficient similarity. While largely cited and used, the applicability of this value ought to be examined in light of the results obtained by heuristic algorithms. In particular, for each model size an appropriate κ value can be determined *a posteriori* based on said results. The goal of this model reduction analysis is not to prescribe which model size one 'should' use; rather, given that the process depends on the priority of the modeler, the goal is to present κ values and run times one can expect when using a particular reduced model.

Results from Pareto Optimization

As discussed in Section 2.3, the goal of a Pareto optimization algorithm is to return a suite of solutions, each of which is optimal for a particular choice of objective weights. For this model, the objectives are to minimize the number of days on which control is used and to minimize the number of rabbits alive during the course of a simulation. Recall that a control is a vector of length 100 with entries in $\{0, 1\}$. Figure 2.4 shows an example of the Pareto frontier. Each dot corresponds to one control, plotted according to the values on the axes. The $\times$'s make up the Pareto frontier of this data set: for every point on this frontier, one of the objectives cannot be improved upon without some sacrifice in the other. On the other hand, for each point not on the frontier, there exists some point in the set that improves upon *both* objectives: in particular, for every square (i.e., non-Pareto frontier) data point, there exists at least one other point with fewer control days and a lower average number of rabbits. The goal of the heuristic Pareto optimization algorithm is to determine, as near

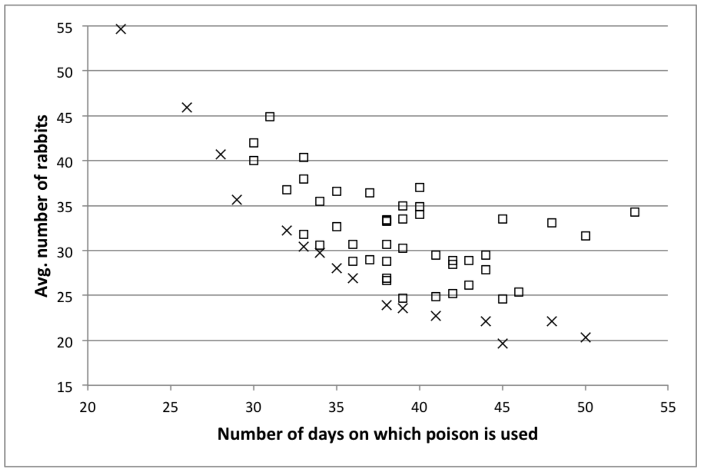


Figure 2.4: An example Pareto frontier for the Rabbits and Grass model. Frontier points are marked with an $\times$ and non-frontier points with a square.

as possible, the true Pareto frontier of the control space. Thus, remaining figures consist of Pareto frontiers only and not the entire data sets. In order to investigate a variety of κ values as determined in Section 2.5.1, several representative model sizes and run times were chosen. For each representative model the Pareto optimization algorithm was run and a Pareto frontier obtained. If a reduced model is a suitable substitute for the original then the Pareto frontier for the reduced model ought to be the same as the Pareto frontier of the original model. For each reduced model, the controls making up the frontier are implemented in the original model in order to determine if they are actually Pareto optimal (as the reduced model results has suggested). Note that Pareto optimization has been performed on the original model as well in order to serve as a basis for comparison.

Figure 2.5 summarizes results for representative models with lower κ values. Each shape corresponds to one representative model, with results coming from the implementation of these controls in the *original* model. These results suggest that models with κ values below 0.5 are not very good surrogates for the original model. In particular, there are fewer data points, and they tend to cluster near certain regions of the frontier. In short, very few of the controls determined to be Pareto optimal by these representative models are in fact Pareto optimal in the original model. Figure 2.6 shows similar results for models with higher κ values. The representative model with a κ score of 0.89 produces a Pareto frontier very near to the frontier of the original model, suggesting that a κ value of 0.89 is sufficiently high – hence, a reduced model with a κ value of 0.89 can likely be used as a surrogate for the original model. The data for the model with a κ value of 0.76 is also near to the Pareto

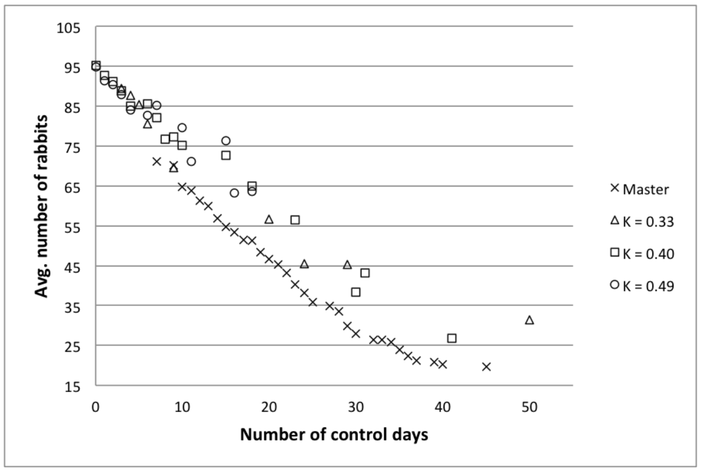


Figure 2.5: Pareto frontiers for models with lower κ values.

frontier of the original model, though not to the same extent. For the model with $\kappa = 0.65$, there are fewer data points, and these are a bit further from the true frontier.

Finally, Figure 2.7 suggests that there is a mildly linear relationship between the κ value of a reduced model and the number of points on the Pareto frontier. The same settings were used for each Pareto optimization algorithm; yet, in general, this data shows that models with lower κ values tend to produce smaller Pareto frontiers (even within the reduced model itself). One possible explanation for this is that for a reduced model, there is a narrower range in the possible dynamics of a model, and thus the true Pareto frontier for a reduced model may indeed be smaller. Thus, again, κ values indicate the extent to which model dynamics are preserved. A qualitative examination of the data presented here suggests that a κ benchmark in the region of $0.75 - 0.80$ is in fact a good benchmark for this example. Once again, the final decision rests with the researcher and is ultimately determined by the level of desired accuracy.

2.5.2 SugarScape model

The second model is a modified version of SugarScape [36], in which a population of agents traverse a landscape in search of sugar. This model was chosen for several reasons: first, it is spatially heterogeneous. This is a common feature of many ABMs and thus it is important to demonstrate how the framework presented here can be applied to models wherein space is an issue. Second, the model has been examined by researchers in a variety of fields – studies based on SugarScape have focused on migration and culture [27], distribution of wealth [105], and trade [26]. As such, it is of broad general interest as a test case. Thus, future work may build on the framework presented here as a means of conducting research in areas as diverse as social science, biology, and economics.

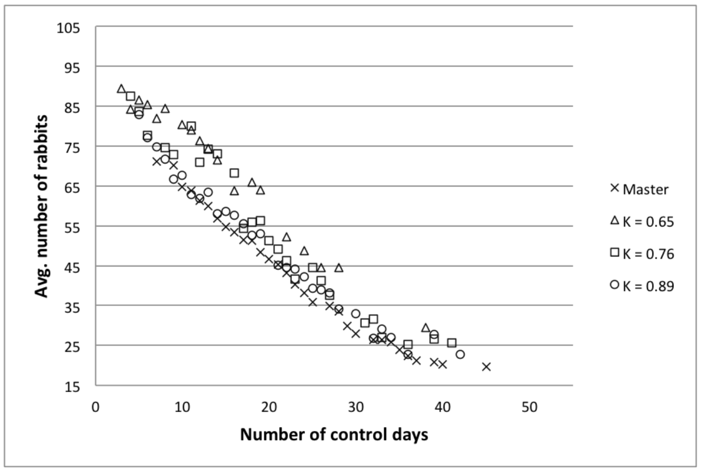


Figure 2.6: Pareto frontiers for models with higher κ values.

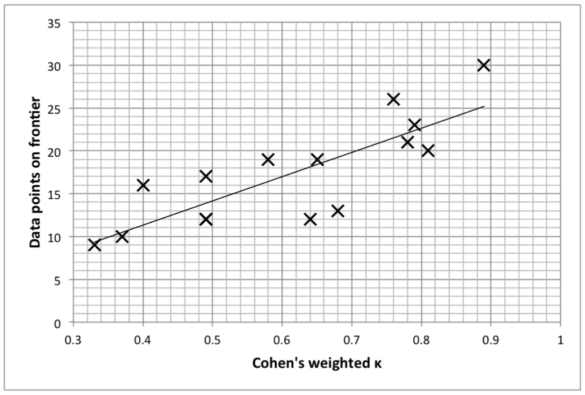


Figure 2.7: Plot of κ values vs. size of frontier, with line of best fit (Pearson's $r^2 = 0.66$).

The basis of the model used here is included with the standard NetLogo distribution [133]. The landscape consists of fixed regions containing different amounts of sugar; as such, this model contains a spatial heterogeneity not present in the rabbits and grass model. The original landscape is presented in Figure 2.8; darker regions represent areas with more sugar. Periodically, ants are taxed based on their vision, metabolism, and location (e.g., high-vision ants in sugar-rich regions may be taxed at higher rates than low-vision ants in regions with less sugar). The optimization problem is to determine the tax schedule which maximizes the total tax income collected while minimizing the number of deaths. Full model details, including those pertaining to taxation, are provided in Appendix B. Note that certain parameter values are altered when considering reduced models; parameter values given in Appendix B refer to the 50×50 model.

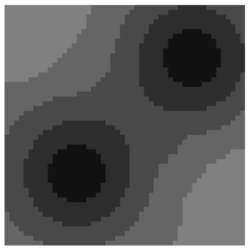


Figure 2.8: Landscape of the 50 × 50 Sugarscape model – darker regions contain more sugar.

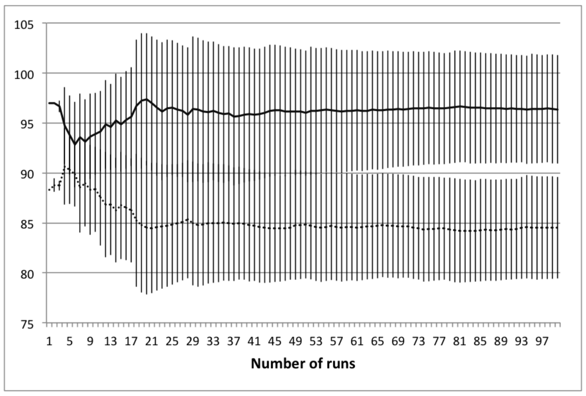
Scaling results

A total of 100 controls were generated, consisting of three different average tax rates. The number of deaths and tax income for each was collected over a total of 100 runs. Representative data is presented in Figure 2.9, with error bars representing one standard deviation. As seen in the figure, there is very little change in the mean and standard deviation of the data beyond 50 runs; hence, there is no benefit to averaging over more than 50 simulations.

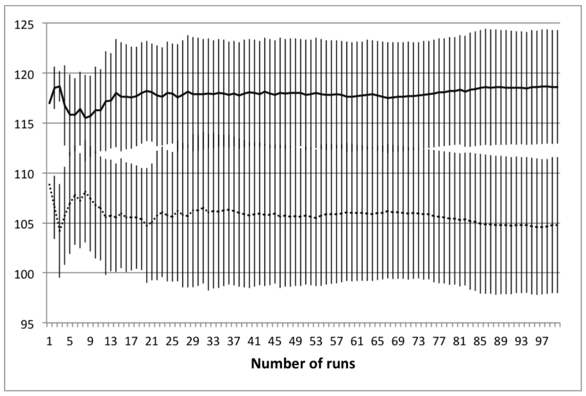
Given the importance of the spatial layout of the SugarScape model, it is necessary to preserve this layout as nearly as possible in any reduced version. Landscape reduction was determined by the **nearest-neighbor** algorithm, a means of re-sampling the original landscape in order to determine the layout of a reduced version.

In addition to scaling the map, the number of agents was also scaled. While low vision is defined to be 1 at any model size, high vision depends on the size of the grid: an agent with vision v on a 50×50 grid has vision $v_r = v \cdot \frac{n}{50}$ on an $n \times n$ grid. For grid sizes 10 and 5, this would result in high vision being equivalent to low; thus in these two cases high vision was defined to be 2. The metabolism of each agent is *not* scaled: at each model size, it was randomly set between 1 and 4 (inclusive).

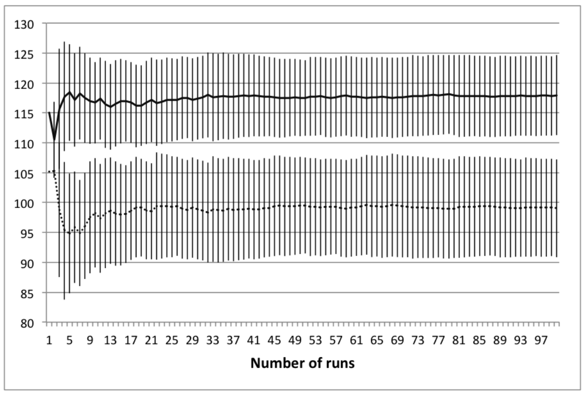
To run the simulation 50 times at model size $M = 50$ (meaning a 50×50 grid) takes approximately 8.5 seconds. Table 2.2 shows the number of simulations which can be run in equivalent run time for reduced model sizes. Figure 2.10 shows κ values for various model reductions. Since there are two variables in this case (deaths and tax income), controls can be ranked according to either, resulting in two different κ values. Note that when ranked according to the number of deaths, κ values are extremely low – in fact, close to being completely random. While the rankings according to tax income result in higher κ values, they still fall short of the proposed minimum benchmark of 0.8. An interesting feature of this data is that there appears to be no great difference in the κ values obtained from models run at 2% of the original run time versus those obtained from 100% run time. This may indicate that the number of runs (50) used for reliable data was originally set too high, or it may indicate that κ values at or below 0.45 are equally unreliable. Nevertheless, there is



(a) Avg. tax rate of 0.125



(b) Avg. tax rate of 0.25



(c) Avg. tax rate of 0.375

Figure 2.9: Average values for deaths (solid) and tax income (dashed) are not improved upon beyond 50 runs. Tax income values have been scaled linearly to fit on the plots.

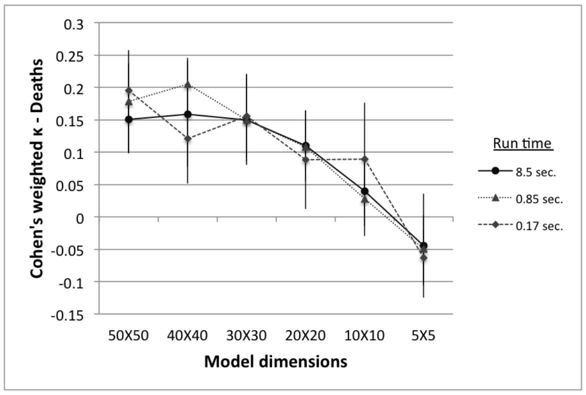
a clear trend showing that as the model size decreases the κ values decrease as well. As in the previous example, this may be an indication of qualitative changes in model dynamics as the model size is reduced.

Results from Pareto Optimization

Although κ values appear lower in this case, it is necessary to again examine the performance of reduced models with respect to control. While the suggested benchmark of $0.75 - 0.8$ proved fitting for the previous model, it may be that a lower κ benchmark is acceptable in this case. Results are presented in Figure 2.11. Pareto frontiers from three models are presented here: those with κ values of $0.43,\ 0.27,$ and 0.15 (with respect to tax income). The shape of the data from the three models follow the same basic shape of the true Pareto frontier (labeled 'Master' in the figure), but there is a distinct difference in performance. In particular, none of the controls found by any of these models are on the Pareto frontier of the original.

World size	Simulations	Avg. run time (sec.)
50	50	8.51
40	88	8.48
30	173	8.49
20	427	8.52
10	1375	8.53
5	3900	8.48

Table 2.2: Number of simulations of SugarScape in equivalent run time.



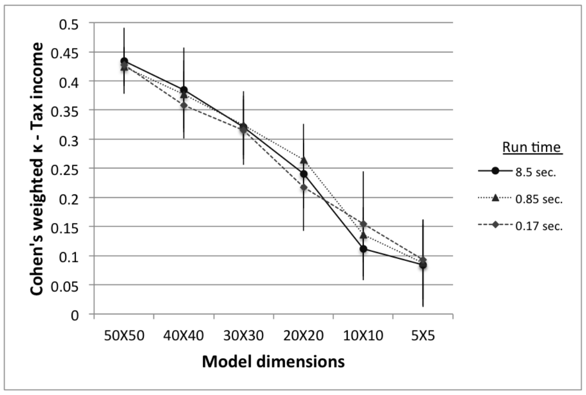


Figure 2.10: Cohen's weighted κ when controls are ranked by deaths and by tax income.

Furthermore, there does not appear to be any significant difference between solutions found using the model with $\kappa = 0.43$ and those found using the model with $\kappa = 0.15$. This indicates that not only are none of these models appropriate as replacements for the original, but that there may in fact be a *minimum* κ value, below which all models are unsuitable. In other words, the downward trend indicated in Figure 2.10 may be misleading: while it seems to suggest that as model size decreases, the models become less representative of the dynamics of the original, the results in Figure 2.11 suggest that this isn't actually the case. On the contrary, models with a κ value of 0.15 may be no worse than those with κ values of 0.43. Given that no models attained a κ value higher than 0.45, it is impossible to judge the benchmark of 0.75 as appropriately high. On the other hand, it *is* possible to conclude that κ values at or below 0.45 are certainly too low.

2.6 Conclusions

The goal of this paper is to introduce a framework for optimization of agent-based models. Once reliable data is obtained, reduced models can be compared to the original. Similarity can be measured using Cohen's weighted κ. Pareto optimization was implemented in order

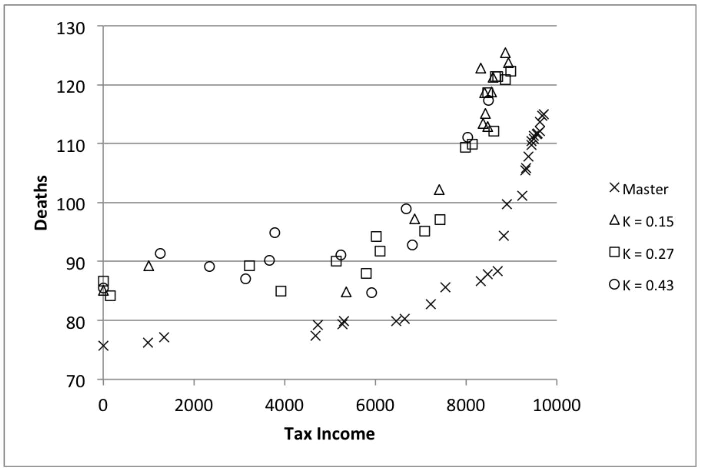


Figure 2.11: Pareto optimization results for the SugarScape model.

to solve control problems in both cases, allowing for *a posteriori* analysis of the κ benchmark. Results presented here show that in one example, the established benchmark in the range of $0.75 - 0.8$ was indeed sufficient for model reduction, while the second example showed that values below 0.45 were too low. These results suggest κ can be a meaningful measure for model reduction.

The models presented here were selected for their universality and popularity – as such, they act as standard models to which any attempt at analysis should be applied. In principle there is no reason why the methodology would not apply to extensions of these models or to other ABMs. In the future this collection of models should be expanded to include a wider variety of models of increasing complexity. This methodology has been applied here to models wherein space and agent location are key features; it may require some modification in order to generalize to non-spatial models. In addition, other methods for analysis of agent-based models include transformation to equation models. Such work (using the same models presented here) is underway.

As ABMs are used more and more to investigate real-world systems, optimization and optimal control problems will naturally arise in the context of ABMs. Heuristic methods have several advantages: they are easy to implement on a computer and they can be applied to virtually any ABM. This is particularly important for models that are too complex for conversion to other mathematical forms, e.g., in cases where differential equations are insufficient. The user has direct control over how each algorithm runs, and can fine-tune parameters and settings to better suit the model. However, there are drawbacks to these methods. For those interested in the certainty of finding globally optimal solutions, heuristic methods are lacking. On the other hand, one may obtain sufficient controls using these methods, and that is a step in the right direction for control of ABMs – in particular when one's goal is to obtain controls that are either sufficient or simply better than any previously known.

It is possible that the complexity of agent-based models will make formulaic translation to rigorous mathematical models intractable – in that case, heuristic methods provide the only means for optimization and optimal control of agent-based models. Coupled with the model reduction techniques and analysis introduced here, this technique provides valuable methodology for solving control problems with agent-based models.

Chapter 3

Optimal Harvesting for a Predator-Prey Agent-based Model using Difference Equations

Acknowledgements are due to Claus Kadelka and Kristin Deger, who helped in the development of ideas in this paper. MSO is responsible for the movement scheme analysis, parameter estimation algorithm, derivation of the equation model, and implementation of the Pareto optimization algorithm.

3.1 Introduction

Agent-based models (ABMs) are computer simulations wherein entities follow update rules based on their local environment. These local interactions give rise to global dynamics – in many cases, the global dynamics are directly determined by the choice of local rules. The ability to investigate the nature of this relationship is a key aspect of agent-based modeling. In this sense, ABMs function as *in silico* laboratories, wherein experiments can be conducted, analyzed, and used to suggest further investigation, either into the ABM itself or into the system being modeled. Beyond standard analysis of model dynamics, one frequently wishes to use an ABM to answer some specific question (or set of questions). External inputs may be implemented which allow formulation of an *optimization* problem: what is the best way to steer the model to a particular state?

Currently, there is no established framework for solving optimization problems for agent-based models, despite the obvious value that a universal framework would provide. In this study, a predator-prey ABM is used as a representative model in order to investigate how global dynamics are affected by changes to local update rules, and how optimization problems might be approached. In particular, discrete difference equations are implemented to serve as a mathematical tool for ABM analysis. The equations are initially developed analytically from the rules of the model, and subsequently altered numerically in order to track changes to the ABM ruleset. Results suggest that equations can be useful in investigating the importance of local rules as well as in solving optimization problems. As such, this mathematical approach has the potential to be a powerful tool for analysis of a broad class of models.

This work presented in the following order: first, related work is presented in order to brief the reader on what has been done in this area, both by the authors and by others in the field. Next, a representative predator-prey model is described, followed by details on how the model was converted to a system of equations. This is followed by some analysis using the equation model and how it compares to ABM dynamics. Finally, the optimization problem is implemented, the equations are updated to incorporate it, and a heuristic algorithm is used to solve the problem. The paper concludes with a discussion and thoughts on future work.

3.2 Background and related work

Many studies have investigated the differences and similarities between agent-based and equation-based modeling. A 1998 study compared an ordinary differential equation (ODE) model with an ABM, concluding that ABMs are preferable if the system is highly localized [33]; this point is reiterated in a 2008 study which notes that equation models may be more appropriate in cases where sensitivity analysis is of particular interest [106]. A 1992 study contrasts local and global population dynamics [88]; a similar study on mean-field approximations emphasizes the importance of local densities [94]. All of this work suggests that local dynamics are a critical feature of agent-based models, and that care must be taken in any attempt to analyze or describe ABMs with equations. This is echoed in the conclusions of [63] and [44], wherein equations were found to better fit ABMs containing large populations. A study wherein spatial dynamics were shown to match well with mean-field aggregation [34] was expanded upon and improved by the inclusion of neighborhoods of varying sizes [61]. Given that the model used here is a predator-prey system, studies of similar systems are of particular interest. One study compared linear stability analysis between an ABM and a differential equations model [134], highlighting the key differences of each system. Another study focused on the oscillatory nature of predator and prey populations in an ABM [83], noting that it can be difficult to capture this feature using ODEs. In previous studies, we have shown how ABM dynamics can be translated into systems of polynomial dynamical systems [76, 55].

A second focus of the work presented here concerns methods for solving optimization problems for agent-based models. It is worth noting that the term 'optimization' may refer to model design or parameter choice; the meaning implied here is neither of these. Rather, solving an optimization problem refers to determining the best sequence of external inputs to a model in order to steer the model to a particular state, or to achieve a certain goal. For example, ABMs have been used in this context in order to investigate influenza mitigation strategies [138, 86], malaria outbreaks [92], general disease control [10], and resource allocation [67], among others. We have examined the model presented here in a textbook chapter aimed at undergraduate students [77]; this work contains results using simulation optimization techniques. One of the advantages of converting ABM dynamics to a system of equations is that the equation model can be used in order to solve optimization problems. A pair of studies on leukemia used partial differential equations to do precisely that [69, 68]. A separate study conducted on a model very similar to that presented here (and with a similar optimization problem) was carried out in [23]; there, ordinary differential equations are used. The combination of discrete dynamical systems and heuristic methods for optimization as described below provide a novel approach.

3.3 The model: rabbits and grass

The model we examine in this study simulates a population of rabbits in a field. The rabbits jump around throughout the field, gaining energy by eating grass (if it is available) and losing energy by moving and via reproduction. Grass is removed from a location when eaten and grows back at each time step with some fixed probability. Rabbits die when they run out of energy. A version of the model is available as part of the model library in NetLogo [133], free software available for download from http://ccl.northwestern.edu/netlogo/. Note that the version of the model included with NetLogo differs slightly from the formulation presented here – in particular, in this paper *poison* is implemented as an external input. Viewing the rabbits as an invasive species, this gives rise to a natural optimization problem: what is the best poison strategy in order to minimize the rabbit population while also minimizing the amount of poison used? For a detailed description of the model examined here, including details on how poisoning is implemented, see Appendix C. A screenshot of the model interface is provided in Figure 3.1.

3.4 Deriving the equations

Table 3.1 provides a description of all terms used in the derivation of the equations. Note that the equations are discrete difference equations – time jumps from t to $t+1$, corresponding to one time step in the ABM. First we note that $\frac{r(t)}{N}$ represents the density of the rabbits. The proportion of empty (i.e., grassless) patches is $(1-g(t))$. Given that the initial distribution of rabbits and grass is random, we expect the proportion of grass that gets eaten to be

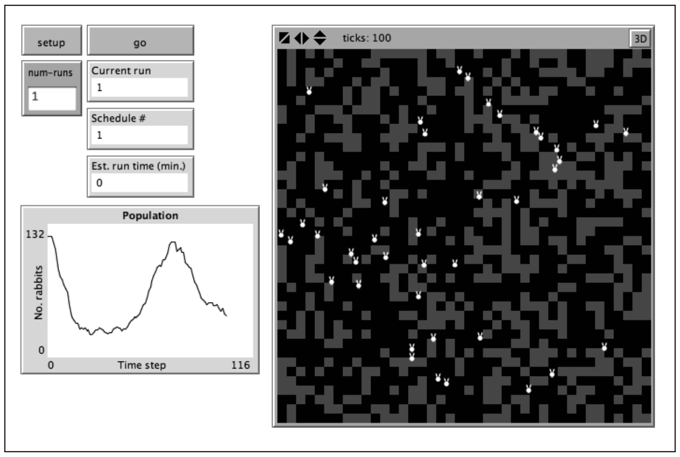


Figure 3.1: A view of the NetLogo interface for the Rabbits and Grass model.

Term	Description
$r(t)$	Number of rabbits alive at time t
$g(t)$	Proportion of grid cells containing grass at time t
N	Total number of grid cells
γ	Probability of grass growing back in an empty grid cell at each time step
$p_k(t)$	Number of rabbits with energy k at time t
en	Energy gained by eating grass
m	Energy lost by moving (i.e., movement cost)
b_t	Birth threshold
b_c	Energy lost by reproduction (i.e., birth cost)

Table 3.1: A description of the terms used in the difference equation system.

$\frac{r(t)}{N}g(t)$. Then the difference equation for the grass is:

$$g(t+1) = g(t) + \gamma(1 - g(t)) - \frac{r(t)}{N}g(t) = \left(1 - \gamma - \frac{r(t)}{N}\right) g(t) + \gamma.$$

Because food energy, movement cost, and birth cost all have integer values, rabbits can be tracked by their precise energy level. Note that since all rabbits with energy above b_t give birth, the maximum energy of a rabbit at the end of each day is b_t.
Any rabbit with energy zero or less dies, so given that rabbits lose m energy each time they move, $p_0(t+1) = (1 - g(t))\sum_{i=1}^{m} p_i(t)$. Here, $1 - g(t)$ indicates the proportion of patches not containing grass – in other words, the proportion of rabbits that fail to eat.

For $k > 0$, there are several ways a rabbit can end each day with energy k:

- a rabbit moves and fails to eat, thereby losing energy,
- a rabbit moves and eats, thereby gaining energy,
- a rabbit moves, eats, and gives birth, thus reducing its (and its offspring's) energy.

The first case is described by the term $(1-g(t))p_{k+m}(t)$, the second by the term $g(t)p_{k+m-en}(t)$, and the third by the term $2g(t)p_{k+m-en+b_c}(t)$. Note that $k+m-en \leq 0 \implies k \leq en-m$ and by assumption, $en-m>0$. Thus the second term is not present for values of k such that $k \leq en-m$.

Next note that in order for birth to have occurred, $k+b_c > b_t \implies k > b_t-b_c$. Also, since all rabbits above energy level b_t give birth, there are no rabbits with energy greater than b_t, so the third term will only be present when $k+m-en+b_c \leq b_t \implies k \leq b_t-m+en-b_c$. Thus the third term (the 'birth' term) only applies for k satisfying $b_t-b_c < k \leq b_t-m+en-b_c$ (the coefficient 2 appears because the parent's energy level is reduced, and the offspring is spawned with this reduced energy level). The above description allows for the model to be written in terms of parameters m, en, b_c, and b_t. Section D presents the entire discrete model, using state variable values provided in Appendix C.

The next step is to validate the discrete model by comparing data obtained from the ABM with data predicted by the difference equation (DE) model. Figure 3.2 confirms that the equations capture the average population dynamics of the ABM. However, one important adjustment to the ABM has been made: the equations do not account for agent dispersal or local densities. In particular, the ABM data shown in figure 3.2 was generated under the *random jump* movement rule (see section C.6). Under this movement scheme, rabbits lose the same amount of energy regardless of the actual distance moved. *Random jump* ensures that every rabbit has the same probability of coming into contact with each grid cell. In light of this, it is perhaps not so surprising that the equations do not account for space: *random jump* has the effect of eliminating localization from the model.

The original movement rule implemented in the ABM is *wiggle* movement, which has the effect of rabbits moving in a more or less straight direction, with some wiggling; see details in section C.6. Figure 3.3a shows the equation data plotted alongside ABM data generated using *wiggle* movement. While the same basic shape of the data remains the same, it is now clear that the data from the equation model is no longer a good fit – this is entirely due to changes in the movement rule. The next step, then, is to introduce movement parameters $m_0, \ldots, m_8$ that adjust for changes to the movement rule. The updated model is presented in figure D.2.

A heuristic technique known as 'squeaky wheel' optimization was used in order to determine parameters values; see section E for details. Figure 3.3b shows how well the equations fit the ABM data generated using *wiggle* movement; parameter values are given in table 3.2.

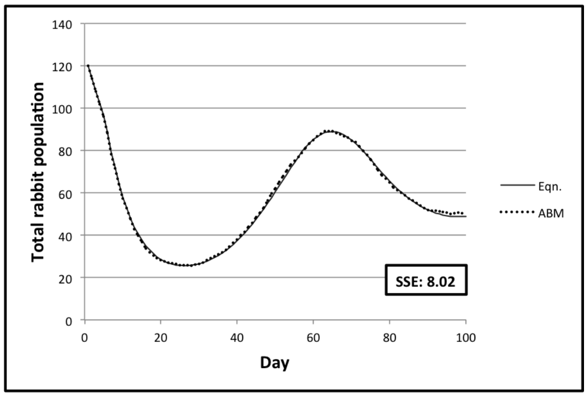
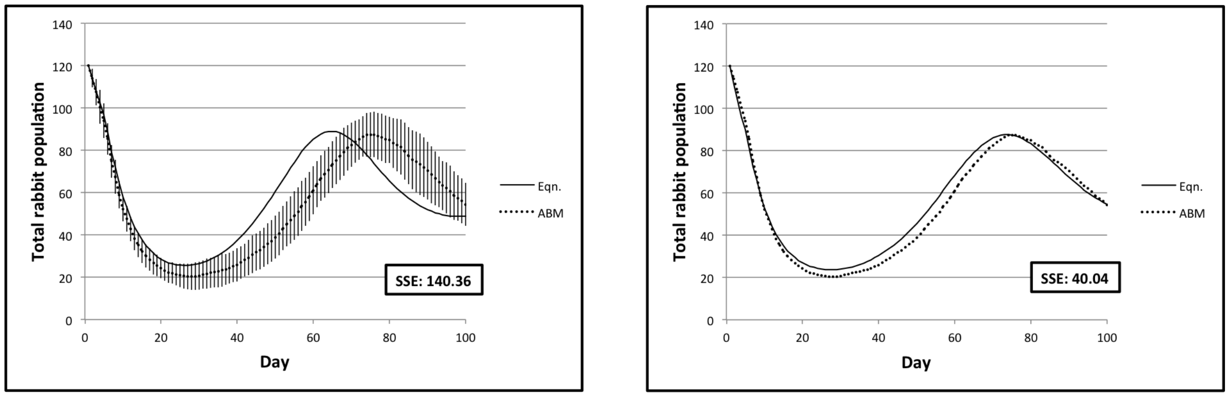


Figure 3.2: Data from the ABM and the equation model, using *random jump* movement.



(a) Unadjusted data – bars represent one standard deviation).

(b) Equation data after parameter adjustment.

Figure 3.3: Population data from the ABM vs. data predicted by the equations, using the *wiggle* movement rule.

Finally, to illustrate the effectiveness of the movement parameters, a third movement scheme entitled *Neighbor 8* was also considered (see section C.6) – in this scheme, rabbits move to the center of one of the neighboring 8 grid cells. Movement parameters adjusted to match *neighbor 8* ABM data are presented graphically in figure 3.4. *Neighbor 8* movement is included here for an important reason: it highlights how a subtle change in the movement rule can produce qualitatively different dynamics. While the difference between *random jump* and *wiggle* is notable, the difference between *neighbor 8* and either of those is dramatically more so. Nevertheless, the heuristic algorithm for parameter estimation is able to match these dynamics, lending credibility both to the form of the model and to the algorithm itself.

Recall that one of the main uses of the equation model is to solve an optimization problem – in light of that, having accurate equations is critical. One commonly noted drawback of

Movement rule	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8
Random jump	1	1	1	1	1	1	1	1	1
Wiggle	0.954	1.182	1.194	0.960	0.856	1.094	0.918	0.695	1.009
Neighbor 8	0.818	1.583	1.139	0.849	0.804	0.913	0.908	0.908	1

Table 3.2: Movement parameters for various movement rules.

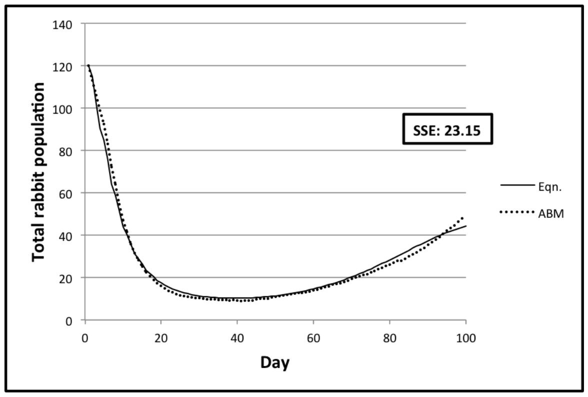


Figure 3.4: Data from the ABM and the equation model using *neighbor 8* movement. Parameter values are provided in table 3.2.

aggregate models such as the one presented here is that they do not offer any particular insight; they are phenomenological rather than analytical or explanatory. This remains an issue here, as we can see that the parameters have something to do with agent movement but we certainly can't interpret them in the same way as γ, b_c, or m, for example. However, this issue doesn't take away from the usefulness of the equation model: in order to solve optimization problems with the equations, our main priority is that the equations respond to various controls in the same way as the ABM. As long as they retain these qualitative features, the equations remain quite useful indeed. We next use the DE model to analyze long-term behavior.

3.5 Steady-state analysis of the DE model

The main motivation for using equations to describe the ABM is to solve a particular optimization problem. However, the usefulness of the equations is not limited to this setting. Understanding the long-term limiting behavior of any ABM can be quite difficult due to the stochastic nature of many models. The DE model makes such analysis possible, providing insight into the dynamics of the ABM while precluding the need for extensive simulation. In particular, equations $p_0(t), \ldots, p_8(t)$, and $g(t)$ track the rabbit population and the grass

Equation	Steady state 1	Steady state 2
p_0	0	2.53
p_1	0	4.44
p_2	0	6.31
p_3	0	8.87
p_4	0	12.52
p_5	0	10.31
p_6	0	7.98
p_7	0	4.80
p_8	0	2.96
$r(t)$	0	58.20
g	1.79	0.40

Table 3.3: Steady states for *wiggle* movement.

level. Solving for steady states requires solving the system $f(t+1) = f(t)$ simultaneously for each equation f in the system. The `FindRoot()` function in Mathematica was used to solve this nonlinear system, though many other software packages and alternatives exist. For the analysis presented here, *wiggle* movement was implemented in the ABM and the equation parameters were set according to table 3.2. For a wide variety of initial conditions, only two steady states were found – these are presented in table 3.3. Steady state 1 (SS1) is the *extinction* state; all rabbits die out and the grass fills up the field (the value of g being higher than 1 is an artifact of the DE model). Steady state 2 (SS2) is the *co-existence* state: a total of approximately 58 rabbits are alive at each time step, with the distribution of energy levels given in the table. The proportion of patches containing grass goes to 0.40. In addition to yielding these steady states, mathematical analysis enables us to determine the stability of each by examining the DE model. The Jacobian of the system is $\mathcal{J} =$

$$\begin{bmatrix}
0 & m_0(1-g) & \dots & 0 & 0 & 0 & 0 & -m_0p_1 \\
0 & 0 & \dots & 0 & 0 & 0 & 0 & -m_1p_2 \\
0 & 0 & \dots & 0 & 0 & 0 & 0 & -m_2p_3 \\
0 & m_0g & \dots & 0 & 0 & 0 & 0 & m_0p1 - m_3p_4 \\
0 & 0 & \dots & m_4(1-g) & 0 & 2m_5g & 0 & m_1p_2 - m_4p_5 + 2m_5p_7 \\
0 & 0 & \dots & 0 & m_6(1-g) & 0 & 2m_7g & m_2p_3 - m_6p_6 + 2m_7p_8 \\
0 & 0 & \dots & 0 & 0 & m_5(1-g) & 0 & m_3p_4 - m_5p_7 \\
0 & 0 & \dots & m_4g & 0 & 0 & m_7(1-g) & m_4p_5 - m_7p_8 \\
0 & 0 & \dots & 0 & m_6g & 0 & 0 & m_6p_6 \\
0 & \frac{-m_8g}{1600} & \dots & \frac{-m_8g}{1600} & \frac{-m_8g}{1600} & \frac{-m_8g}{1600} & \frac{-m_8g}{1600} & m_8\left(0.98 - \frac{p_1+\ldots+p_8}{1600}\right)
\end{bmatrix}.$$

Recall that the DE model provides expected values over a large number of runs rather than results for an individual run. Thus, the steady states presented here must be interpreted in the same way: not as values that any particular simulation will tend to, but rather as

Energy	SS2 value	ABM value
1	4.44	4.34
2	6.31	6.29
3	8.87	9.16
4	12.52	12.09
5	10.31	9.81
6	7.98	7.50
7	4.80	5.15
8	2.96	2.67

Table 3.4: Population at each energy level. ABM values are averaged over a single 1000-tick simulation.

averages of the represented values over many time steps.

Evaluating $\mathcal{J}$ at the extinction steady state and solving for eigenvalues, we see that the maximum magnitude of the eigenvalues is 2.72. Since this is greater than 1, we may conclude that SS1 is *unstable*. This means that as long as the initial number of rabbits is non-zero, the probability of extinction is highly unlikely (though still technically possible in the ABM due to stochasticity). On the other hand, the maximum magnitude of the eigenvalues for SS2 is 0.983. In particular, this is less than 1, so SS2 is *stable*. We may verify these results by initializing the ABM with a distribution of rabbits near SS2 (exact values are not possible since the number of rabbits must be an integer value) and examining long-term behavior. Our observation is that for each individual run, the distribution is not closely followed. The population at each energy level undergoes frequent oscillations, though the amplitude and frequency are quite regular. However, the *mean* number of rabbits at each energy level hews closely to the distribution of SS2, as shown in table 3.4. Similar values were observed for a wide variety of initial conditions, including an initialization of only one rabbit at each energy level. This confirms the validity of the steady state analysis conducted above. It is important to reiterate that the DE model is unable to capture the oscillatory effects of the ABM; as such, it is less applicable for studying the behavior of individual replications. Nevertheless, it remains effective for an accurate description of aggregate behavior. In the next section we describe the optimization problem and compare results from the DE and ABM models in order to confirm that the DE model faithfully captures the aggregate dynamics of the ABM *with respect to control.*

3.6 Cohen's weighted κ: a similarity measure

In order for the DE model to serve as a surrogate for the ABM in terms of solving the optimization problem, it is necessary that it responds to different controls in the same way as the ABM. In this model, a control is a poison schedule: specifically, a binary vector indicating whether or not poison is used on each day of the simulation (for details see Appendix C). In order to obtain the parameters given in table 3.2, a set of 100 poison schedules was used as training data. The details of how this training set was selected and of the parameter estimation process are provided in Appendix E. A different set of 500 poison schedules was generated in order to validate that the DE model responds to these schedules in the same way as the ABM. These validation schedules were implemented both in the ABM and the DE models. Since the optimization problem has two objectives and one of these (the sum of the control schedule) is identical in the two models, the only objective that differs is the rabbit population data. This data was tallied and ordinal lists were created – in ascending order by number of rabbits – representing rankings of the validation schedules for each of the models. In order to determine if the DE model is an accurate representation of the ABM with respect to control, it is necessary that the two models rank poison schedules in the same way – thus the problem reduces to examining the similarity of the ordinal lists.

Cohen's weighted κ provides just such a measure. The original κ metric was introduced in 1960 [22]. The weighted version, introduced in 1968 [21], can be used to measure the similarity of two ranked lists of the same set of objects. Let p_{obs} be the observed proportion of agreement and p_{exp} be the proportion of agreement expected by chance. Then $\kappa = \frac{p_{obs}-p_{exp}}{1-p_{exp}}$. Hence for rankings only as similar as those expected by chance, $\kappa = 0$, and for two lists in perfect agreement, $\kappa = 1$. κ is weighted in the sense that dissimilarities are penalized based on their magnitude; for details of calculating p_{obs} and p_{exp} (and assigning weights), see [21]. For examples of studies using weighted κ in a manner similar to that presented here, see [41] and [37].

Of course, it is unclear exactly how high the κ value needs to be in order to determine that two lists are sufficiently similar. In the literature, common benchmarks for acceptable κ values generally range from 0.75 [73] to 0.8 [3]. In this case, the ordinal lists from the ABM and DE models result in $\kappa = 0.9023$ – an extremely high value indicating that the lists are very nearly identical. Hence, κ serves as a rigorous statistical means of determining the extent to which the equation model matches the ABM with respect to control, and subsequently how it may be used as a surrogate for the ABM in solving the optimization problem. The next section describes how the system of equations might be further reduced.

3.7 The two-equation model

Given that the control problem is only concerned with the average number of rabbits alive over the course of a simulation, it is not necessary (from the perspective of the control problem) to keep track of how many rabbits there are at each energy level. In principle, a system of two equations is sufficient: one equation to track the grass levels and another to track the total rabbit population. Note that the optimization problem does not depend on the grass level, leading one to perhaps conclude that a one-equation model might be sufficient. A simple thought experiment reveals the necessity of the grass equation: it is not enough to know, for example, that there were only twenty rabbits at the previous time step. If the field was empty this number will go down, but if the field is full of grass, the number will go up. Thus a one-equation model is not possible. We begin the assumption that the rabbit equation $r(t)$ has the following form:

$$r(t+1) = a \cdot r(t) + b \cdot r(t)g(t).$$

Here, the number of rabbits at the next time step depends on the number of rabbits alive during the previous time step, along with an interaction term representing how the population depends on the grass level. The grass equation $g(t)$ remains the same as in figure D.2. Parameter estimation is performed as described in Appendix E; the full two-equation model is presented in figure 3.5. While it would seem that fewer equations result in a loss

$$\begin{aligned} r(t+1) &= 0.7213 \cdot r(t) + 1.1355 \cdot r(t)g(t), \\ g(t+1) &= 0.9679 \left(\left(0.98 - \frac{r(t)}{1600} \right) g(t) + 0.02 \right). \end{aligned}$$

Figure 3.5: The two-equation DE model.

of accuracy, the validation process yields $\kappa = 0.9019$ in this case – practically the same as the κ value for the model in figure D.2. This makes sense in light of the control objective, which does not require information about rabbits at different energy levels. In particular, this result suggests that the two-equation DE model is a suitable surrogate for the ABM in terms of solving the optimization problem. The trade-off is a loss of analytical power: the meaning of parameters a and b are not as easily understood as the parameters b_t, b_c, m, and en in the original model, for example. It is worth noting that were the control objective altered – for example, if we poisoned only rabbits with a certain energy level – then the two-equation model would not be sufficient. However, the fact that the κ value is the same for the two-equation model emphasizes an important consideration: an equation model need not describe every feature of an ABM. As long as pertinent model details are preserved, it may be possible to work with a simplified version without sacrificing accuracy. To that end, the two-equation model was used to generate all further results. In the following section, we describe how the optimization problem is implemented in the equations.

3.8 The optimization problem

Recall that a poison schedule is a binary vector u of length equal to the total simulation time: a 0 indicates that poison is not used on that day and a 1 indicates that it is (for more details see Appendix C). On any given day, the *strength* s of the poison depends on p_{deg}, p_{max}, and u; it is determined by the equation

$$s(t+1) = (1-p_{deg})\cdot u(t)\cdot s(t)+(1-u(t))\cdot(s(t)+p_{deg}(p_{max}-s(t))), \qquad s(0) = p_{max}. \tag{3.1}$$

The equation shows that if poison was not used on the previous day (i.e., $u(t) = 0$) then the strength grows back at a rate proportional to p_{deg}, while if poison was used on the previous day, the strength is reduced. Note that $s(t)$, the strength of the poison, determines the number of rabbits that we expect to die on each day. The poison schedule u is an input to the equation model, which now includes 3.1. Hence the two-equation DE model with control is the coupled system (X, u), where X is the system presented in figure 3.6. From the rabbit equation $r(t)$ we see that poison strength only affects the rabbit population when poison is used.

$$\begin{aligned}
r(t+1) &= (1-u(t)s(t))\cdot(0.7213\cdot r(t)+1.1355\cdot r(t)g(t))\,,\\
g(t+1) &= 0.9679\left(\left(0.98-\frac{r(t)}{1600}\right)g(t)+0.02\right),\\
s(t+1) &= 0.5\cdot u(t)\cdot s(t)+(1-u(t))\cdot(s(t)+0.5(0.3-s(t))),\\
\{r(0), g(0), s(0)\} &= \{120, 0.20, 0.3\}.
\end{aligned}$$

Figure 3.6: The two-equation DE model coupled with poison schedule u. The length of u indicates the length of the simulation.

3.9 Pareto optimization

There are several popular approaches to optimal control theory for discrete difference equations. Ding et al. [30] examine a stochastic model on a spatial grid and perform optimal control by forward-tracking. Lenhart and Workman [78] include a textbook chapter on the subject which focuses on a similar technique. A multi-objective problem dealing with an infinite time horizon is investigated by Hayek in [53]. In fact, many studies implement back-tracking or forward-tracking to solve optimal control problems in the discrete case. In our study the length of a simulation is finite but arbitrary; for the example here we set the length at 100 time steps. Due to computational complexity, traditional approaches are no longer applicable: hence, we focus on heuristic methods in order to solve the optimization problem.

Evolutionary algorithms (EA) mimic the natural process of evolution in that more fit solutions are used to 'breed' new solutions in the hopes that the new solutions inherit and improve upon the beneficial features of the virtual 'parents'. Various EAs – first brought to general attention by Goldberg in 1989 [46] – have been implemented in solving optimal control problems for agent-based models. In particular, they have been used in studies on optimal vaccination strategies in an influenza model [100] and cancer models [81, 101], as well as in a study on optimal drug treatment schedules for HIV patients [15]. A common approach to such problems – and the one taken in these studies – is to determine a fitness function by assigning weights to each objective in the optimization problem. This allows one to define an optimal solution by fine-tuning the balance of the objectives; see [45] for a discussion of assigning weights in objective functions.

A potential drawback of this approach is that optimal solutions may be highly sensitive to the choice of weights, and one must be careful that the weights are not assigned arbitrarily. Thus, any method which sidesteps this issue is of particular appeal. The method proposed here is *Pareto optimization*: rather than employing a fitness function, a set of solutions is determined, each of which may be optimal depending on the choice of weights. The method is named for ideas first put forth by Italian economist Vilfredo Pareto [90]. The *Pareto frontier* refers to solutions that cannot be improved upon in terms of one objective without a sacrifice in at least one other objective.

For this example, it is clear that using no poison at all minimizes the cost of the poison – then the only way to obtain a lower number of rabbits is by increasing the poison cost. Hence, the 'no poison' schedule is a *de facto* member of the Pareto frontier, as one objective (minimizing rabbits) cannot be improved upon without some sacrifice in the other (minimizing the amount of poison used). Since the solution space is finite, the existence of a Pareto frontier is guaranteed. The goal of Pareto optimization, then, is to determine the Pareto frontier of the solution space by employing a strategy similar to that of traditional EAs. Of course, there are many other approaches to multi-objective optimization; for an extensive list of resources see [20]. Pseudocode for the algorithm used here is presented in Algorithm 2; it is based on work introduced by Horn et al [60] in 1994. A key feature of Algorithm 2 is the method by which poison schedules are mutated, which encourages exploration of the entire Pareto frontier rather than focusing on a narrow section.

3.10 Results

Results were obtained using the two-equation model presented in figure 3.6 with $d = 0.5$ and $p_{max} = 0.3$. The Pareto EA was run for 80 generations with a schedule population size of 40.The initial and final generations of the algorithm are presented in figure 3.7. Each element in the figure corresponds to one poison schedule, which was evaluated using the

Algorithm 2 Pseudo-code for Pareto optimization algorithm.

```
1: generate random initial population of solutions
2: while $generation < max\_gens$ do
3:     evaluate current generation, call it $current\_pop$                ▷ store rabbit and poison levels
4:     determine Pareto frontier; call it $current\_frontier$
5:     let $next\_pop = current\_frontier$                                 ▷ save frontiers between generations
6:     while size of $next\_pop <$ size of $generation\_size$ do           ▷ each generation contains a fixed number of solutions
7:         repeat
8:             choose two candidate solutions from $current\_pop$
9:             let $comp\_set$ = random subset of 5 solutions from $current\_pop$
10:            if exactly one candidate is Pareto dominant over $comp\_set$ then
11:                that candidate becomes parent                          ▷ add candidates more likely to be on frontier
12:            else if one candidate has fewer neighbors in solution space then   ▷ give preference to isolated solutions
13:                that candidate becomes parent
14:            else
15:                choose candidate at random to become parent
16:            end if
17:        until two parents have been chosen
18:        Breed two new solutions ($a$ and $b$) using parents:
19:            for all components in parent solutions do                  ▷ uniform crossover
20:                select component from random parent
21:                add this component to $a$
22:                add corresponding component from other parent to $b$
23:            end for
24:            mutation:                                                  ▷ $a$ and $b$ are both subject to mutation
25:                $mutation\_rate = 15 \cdot ((max\_gens - generation)/max\_gens)$   ▷ mutation rate decreases over time
26:                randomly choose whether to increase or decrease poison
27:                if increasing poison then
28:                    replace $mutation\_rate$ 0's with 1's              ▷ increase poison days
29:                else
30:                    replace $mutation\_rate$ 1's with 0's              ▷ decrease poison days
31:                end if
32:            end routine
33:        end routine
34:        add $a$ and $b$ to $next\_pop$
35:    end while
36:    increase $generation$ by 1
37: end while
```

two-equation DE model and plotted according to the amount of poison used and the average number of rabbits alive over the course of evaluation. The wide range of schedules and

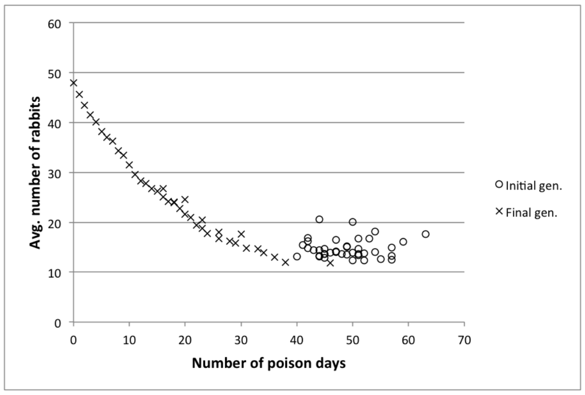


Figure 3.7: Results from Pareto optimization.

the shape of the final generation indicate that the Pareto optimization algorithm is quite successful. The data confirm our intuition that as the number of poison days increases, the average rabbit population decreases. However, this is true only to a point – note that in the final generation, there are no schedules with more than 50 poison days. This is likely due to the degradation of the poison – with increasing poison days, the effective strength of the poison is diminished and thus has less of an effect on the rabbit population. This figure highlights several advantages of Pareto optimization: since the algorithm returns a set of schedules, one can now choose the 'optimal' schedule from a managerial point of view. Additionally, is now possible to examine the trade-off between poison and population. For example, increasing poison days from 0 to 10 can substantially decrease the average rabbit population, but increasing poison days from 30 to 40 results in a much smaller population decrease. Several example schedules from the final generation are presented in Appendix F.

3.11 Discussion and future work

The equation models presented here illustrate how equations can be built analytically from the rules of an ABM and how they can be altered in order to better fit data. While the ABM investigated here is a predator-prey system, there is no *a priori* reason why the same methodology would not apply to a wide variety of ABMs; Cohen's weighted κ and Pareto optimization ought to apply just as well. Thus, the model used here is simply a representative example of an approach that ought to hold appeal to a variety of researchers, as it provides more rigorous mathematical and statistical grounds for ABM analysis. The application of Pareto optimization to ABMs is novel, and should be of wide interest as well due to the investigatory motivation behind many ABMs.

Ongoing work includes the application of this framework to a more complicated ABM as a means of validation. In particular, we are currently applying the approach described here to several spatially heterogeneous models; preliminary results are promising. While more complicated models may require more equations, the framework applies just as well. There are two areas of research which we believe are critical to development of mathematical analysis of agent-based models. The first is developing a rigorous formulation of the relationship between ABM model rules and equation parameters. This would help identify the precise mathematical mechanisms which drive model dynamics, and as such would be invaluable to future analysis. The second area of research is control theory for large, finite horizon, discrete systems. This would reduce uncertainty in solving optimization problems and would benefit both modeling and mathematics. We plan to pursue these areas in future work.

Chapter 4

Mathematical Conversion of SugarScape to Analytical Difference Equations

Acknowledgements for this chapter are due to Scott Christley, Rachael Neilan, and Rene Salinas, all of whom were consulted during the course of this work. MSO is responsible for the design and implementation of the SugarScape Gradient model, the difference equation derivation, and the Pareto optimization implementation.

4.1 Introduction

Agent-based models (ABMs) are computer simulations in which entities (i.e., agents) interact with each other and their environment according to local update rules; these local rules give rise to global dynamics. ABMs are a natural framework for investigation of systems consisting of heterogeneous populations wherein traditional modeling approaches are unwieldy, either due to computational constraints or lack of knowledge of what drives global dynamics. A key feature of agent-based modeling is the investigation of hypotheses of how local behavior affects global phenomena; hence a rigorous framework for analysis of ABMs is of particular interest.

Even the simplest local interactions can give rise to complicated and unpredictable global dynamics – indicating both the power of agent-based modeling and the care that must be taken in any attempted analysis. A mathematical description of an ABM is appealing because it brings the desired rigor to the form of the model, and allows access to a plethora of theory and tools. Once a system of equations is formulated describing pertinent model dynamics, steady state analysis and optimal control are possible, for example. For many

ABMs, questions amenable to investigation arise naturally: for example, in a cancer model, what is the best possible drug target? In an ecological model, what is the best harvesting strategy in order to avoid extinction? In a economic model, which policies maximize wealth? We refer to these types of question as *optimization* problems: what is the best way to achieve a particular goal? While 'optimization' in other contexts may refer to model design or parameter estimation, here it refers to questions of the sort in these examples.

There are methods for solving optimization problems for ABMs – this area of study is referred to (unsurprisingly) as simulation optimization. However, a mathematical formulation of ABM dynamics brings to light another advantage when examined in this context: rather than relying on results from simulation – which can be computationally expensive and unreliable – the equation system can be used to solve optimization problems. Furthermore, if the equations are analytical rather than phenomenological, the solving of such problems can offer additional insight into model dynamics, as results can be interpreted in terms of the model parameters.

To illustrate the power of this technique, we introduce a spatially explicit ABM, pose an optimization problem, and develop analytical equations to capture pertinent model dynamics. Finally, we use the equations to solve the optimization problem and then verify that the solutions faithfully translate back to the ABM. The results indicate both the power and the potential limitations of this approach, as well as informing directions for future research.

4.1.1 Related work

Many studies have examined similarities and differences between ABMs and equation-based models (EBMs). In their user's guide, Parunak et al contrast ordinary differential equation (ODE) models with ABMs [98], concluding that ABMs are more appropriate in models with highly local features. Fahse et al use an ABM of birds in an attempt to extract a population growth rate from ABM data [38]; a similar goal is examined in a zooplankton model by Duboz et al [31]. A spatially heterogeneous forest ABM is used to derive a series of equation-based models in a study by Picard and Franc [104]; here, local spatial effects are shown to be critically important. Mean-field aggregations of ABM dynamics are shown to work well by Edwards et al [34], while Ovaskainen and Cornell emphasize the importance of local densities in such models [94]. Huet et al examine the issue of local neighborhoods in the development of mean-field approximations [61]. A broader study on the use of ABMs versus EBMs by Rahmandad and Sterman [106] discusses circumstances for which a particular model type is preferable. Cecconi et al use a model from game theory to generate empirical data in order to fit an EBM to that data [16]; they emphasize the complementary nature of the two modeling paradigms. A formal methodology for translation of ABMs to polynomial dynamical systems (PDS) over a finite field is presented by Hinkelmann et al [55]; this technique demonstrates that in theory, every ABM can be represented as a PDS.

In addition to general studies on agent- and equation-based modeling, simulation optimization techniques have been investigated in order to solve problems similar to those posited in section 4.1. Okell et al examine the effects of therapy and treatment in an ABM of malaria transmission [92], while Kasaie et al focus on optimal resource allocation in epidemiology [67]. Strategies for mitigating influenza outbreaks are the focus of studies by Mao [86] and Yang [138]. A textbook chapter by Laubenbacher et al examines optimal control theory for ABMs [77], offering examples of simulation optimization strategies.

In this study, a system of discrete difference equations is developed analytically from ABM rules; the equations are subsequently used to solve an optimization problem. While optimal control theory exists for the mathematical system, it is often limited to cases of very few equations or a very short time horizon. Ding et al describe such theory for a model of raccoons [30] containing several equations, but simulations are limited to fewer than ten time steps. Lenhart and Workman dedicate a textbook chapter to the topic [78] but also focus on small systems. An infinite-time discrete optimal control problem is investigated by Hayek [53]; it is not clear if results are generalizable to finite-time systems.

Existing optimal control theory is intractable for models containing many equations over a long (but finite) time horizon, and translation of spatially heterogeneous ABMs often results in just such models. Hence, heuristic methods are employed in order to solve optimization problems. Heuristic methods have been used in conjunction with ABMs to solve optimization problems related to cancer vaccination [81, 101] and HIV treatment [15]. Different heuristic methods for determining general vaccination strategies are contrasted by Pappalardo et al [96]. Hence, there is precedent for the use of such methods in solving the same optimization problem in an equation-based model, which is precisely the approach taken in this study.

4.2 The model

In order to illustrate how analytical difference equations can be used to describe agent-based model dynamics, a representative model is used as a running example. The version of the model used here is based on SugarScape [36], a large-scale ABM wherein virtual ants patrol a landscape made up of sugar. SugarScape was chosen as a representative model because it was the first large-scale ABM and it can be studied in a variety of contexts. Although first introduced as an abstract study of generative social science, SugarScape has been used to study economic trade [26] and wealth distribution [105]; it even served as the basis for a study on the culture and migratory patterns of the Anasazi people [27]. We use a modified version of the model: while simplified in order to highlight how analytical equations can be built, our version maintains spatial heterogeneity and complicated local dynamics. In the following section we provide a brief description of the model; full details (including parameter values and pseudo-code) are provided in Appendix G. It should be noted that there are many free software platforms for agent-based modeling, including Repast [91], MASON [84], and NetLogo [133]. A comparison of these platforms is outside the scope of this text. Results for this study were generated using NetLogo – in fact, the SugarScape Gradient model is adapted from a model contained in the NetLogo model library – though by using the description in Appendix G, the model can be replicated in a variety of agent-based modeling platforms. NetLogo was chosen for its intuitive interface and ease of use.

4.2.1 The SugarScape Gradient model

The landscape in the version of SugarScape studied here consists of four vertical regions, increasing in sugar from left to right (see figure 4.1); hence, we refer to this as the SugarScape Gradient model. While edges wrap vertically, they are bound horizontally, so ants can only move to a higher sugar region by moving to the right. Each ant has its own metabolism, determining how much sugar it burns each time step. Ants collect sugar at their current location and lose sugar based on their metabolism; an ant dies if it runs out of sugar. In addition to the modified landscape, there are several changes to the original SugarScape model rules that bear mentioning:

- ants can collect up to a **maximum** of 50 sugar,
- upon being collected by an ant, the sugar on that grid cell is **not** depleted,
- multiple ants may occupy one grid cell at any given time,
- ants move to the grid cell within their vision with the most sugar; if there are multiple such locations, ants choose one **at random**.

In order to make the analytical structure of the equations clearer, all ants have metabolism of 1 or 2, and vision 1 (meaning they can see one grid cell in each of the four principal directions).

4.2.2 An optimization problem

In the SugarScape Gradient model, wealth (i.e., sugar) is of natural interest: it drives the migration of the ants and determines their survival. In order to pose an optimization problem, we impose *taxation* in the model: at regular intervals, ants are taxed a proportion of their sugar, depending on their location at the time of taxation. The purpose of taxation is to study its effect on the overall population dynamics: if taxes are always higher in a particular region, will agents tend to avoid that location? How high can taxes be before all of the ants die as a result? There are many optimization questions that arise from this line of thinking. We pose the following: what is the optimal tax structure in order to minimize

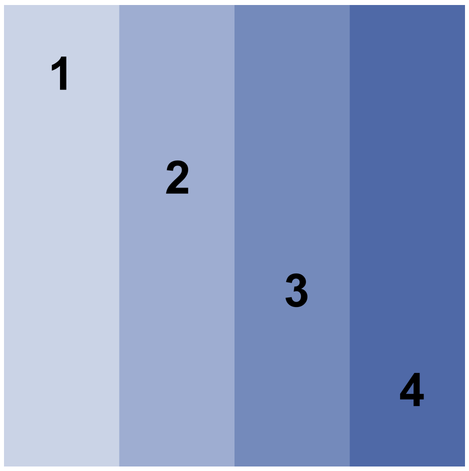


Figure 4.1: Landscape of the SugarScape Gradient model. Labels indicate the amount of sugar contained on each grid cell in that region.

the number of deaths and maximize the amount of tax collected? There are four regions, and five permissible tax rates: $\{0, 0.25, 0.5, 0.75, 1\}$. Tax is collected ten times, and the rates for each region may change each time it is collected. Hence there are $5^{4\cdot 10} \approx 9 \times 10^{27}$ possible tax structures. This is far too many for enumeration via simulation, highlighting the importance of both the equation system and the heuristic approach.

4.3 Analytical equations

In order to derive analytical equations describing ABM dynamics, it is important to consider the optimization problem. Since our optimization problem involves taxation, we need information about the location of the ants (since tax is location dependent) and the wealth distribution (so we know how much income we collect). Furthermore, the equation system is being used to find optimal tax structures for the ABM; hence it is preferable that a given tax structure can be implemented without alteration in both the ABM and the equation system. Hence, results obtained from the equations can be tested directly in the ABM without any modification. Several example equations are explained here; for the full set of equations see Appendix H. It should be noted that in the following explanations, 'wealth' and 'sugar level' are used interchangeably.

Population equations. Each of the four regions consists of 12 columns; since the map wraps in the vertical direction each column is homogeneous. Hence there are $4 \cdot 12 = 48$ population equations $p_1, \ldots, p_{48}$ tracking the number of ants in each column at each time step. We use discrete-time difference equations whose values are updated according to values at the previous time step. Consider the equation for $p_5(t)$, given in equation 4.1. The left half of figure 4.2 illustrates this equation: each ant can see exactly 5 grid cells, and in this case all of them have the same amount of sugar (since the ants are in the interior of the region). All ants have vision 1. In the interior of each region, there is a 60% chance at each time step that an ant will stay in that column and a 20% chance that it will move to the left or the right. Ants at the right edge of a region **all** move to the right. At the left edge, there are only 4 grid cells that an ant considers, so $\frac{3}{4}$ of them remain in that column (note that we do not need to track the vertical location of the ants).

$$p_5(t+1) = 0.2p_4(t) + 0.6p_5(t) + 0.2p_6(t). \tag{4.1}$$

If an ant is at the rightmost edge of a boundary, it can see into the next region and will

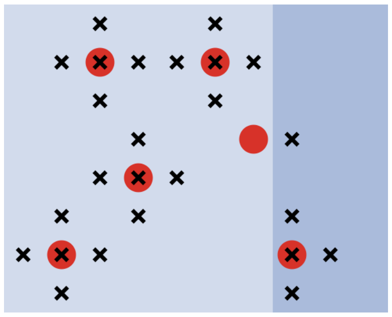


Figure 4.2: Ants are represented by circles; potential moves are represented with an '×'.

consequently move there. At the leftmost edge ants can see a column with lower sugar than their current location; in that case, they are guaranteed to **not** move there. The right half of figure 4.2 illustrates these scenarios; equations 4.2 and 4.3 show examples of how boundary column equations account for these effects.

$$p_{12}(t+1) = 0.20p_{11}(t), \tag{4.2}$$
$$p_{13}(t+1) = p_{12}(t) + 0.75p_{13}(t) + 0.2p_{14}(t). \tag{4.3}$$

Similar equations hold for each of the remaining columns; for the full set of equations see Appendix H.

Wealth distribution. Ants may accumulate up to 50 sugar; ants with 0 sugar die. Some ants have metabolism 1 and some have metabolism 2, and these are tracked separately. Finally, since taxation occurs by region, wealth distributions need to be tracked by region as well. Hence, since there are 51 sugar levels (0 to 50) for ants at each metabolism, there are $51 \cdot 2 = 102$ wealth equations per region, for a total of 408 wealth equations. Let $w_{r,m,s}(t)$ represent the number of ants in region r with metabolism m and sugar level s. The net sugar gain is $r - m$, so in general,

$$w_{r,m,s}(t+1) = w_{r,m,s-(r-m)}(t). \tag{4.4}$$

Some modification is needed for particular sugar levels in each region: for example, ants cannot accumulate more than 50 sugar, so this equation must absorb all ants whose sugar level would have reached 50 or higher:

$$w_{r,m,50}(t+1) = \sum_{i=50-(r-m)}^{50} w_{r,m,i}(t). \tag{4.5}$$

Similar modifications are necessary for equations tracking the number of ants with 0 sugar. The full system of equations is provided in Appendix H; explanations here are meant to provide some insight as to how the equations are formulated.

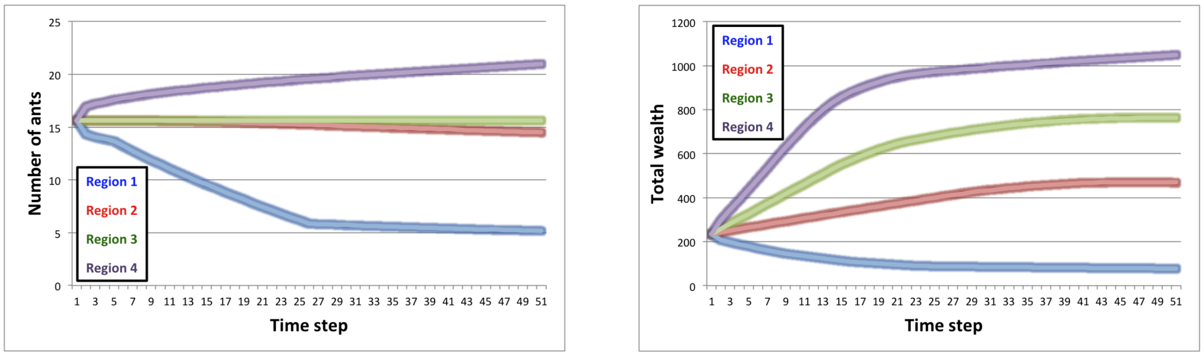
Migration and taxation. Wealth distributions are affected not only by eating and metabolism, but by migration and taxation as well. When ants move from one region to another, the wealth equations for each region are affected as the incoming (and outgoing) ants take their wealth with them. Furthermore, taxation causes additional wealth shifts, and these are also region-dependent. In order to account for this, the equations are updated in a certain order; the order is determined by the order of events in the ABM. Figure 4.3 explains the order in which the population and wealth equations are evaluated. Note that wealth equations are affected by migration (step 3) and population equations are in turn affected by taxation via the wealth equations (step 8).

```
1. Update population equations based on movement.
2. Store how many ants left each region.
3. Scale wealth equations based on how many ants left that region.
4. Update wealth equations based on eating and metabolism.
5. Scale wealth equations based on taxation (if applicable).
6. Store the amount of tax collected.
7. Store the number of deaths.
8. Scale population equations based on sum of wealth equations per
   region (taxation causes some ants to die; population equations
   need to reflect this).
```

Figure 4.3: The order in which the equation system is updated.

4.4 Results

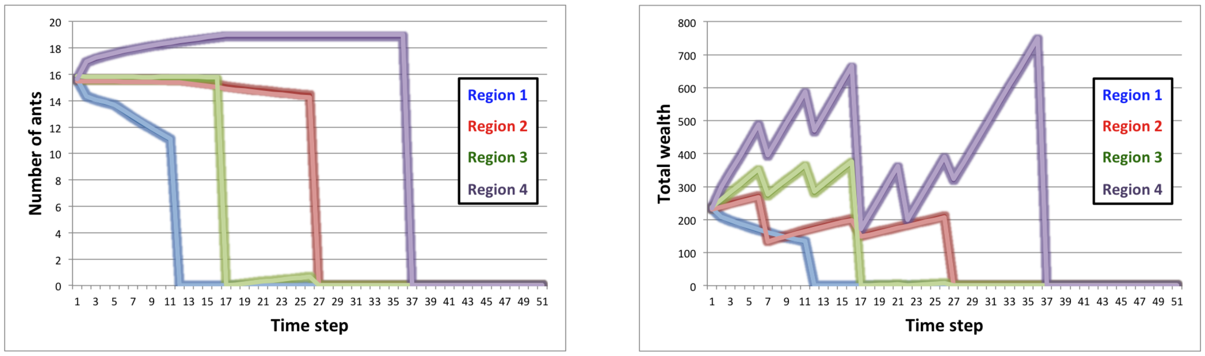
Population and wealth data. Figure 4.4 compares data from the ABM and the equation model; ABM data is averaged over 100 runs. Since the optimization problem concerns taxation by region, it is necessary that the dynamics match *at the regional level*; hence, regional populations are presented. Note that there does not appear to be any error propagation; that is, though the simulation time was fixed in this case, the equations ought to capture dynamics for arbitrary time horizons.



(a) Population by region (b) Total wealth by region

Figure 4.4: Population and wealth data from the ABM (light colors) and the equation model (dark colors) when no tax is applied.

Similarity with respect to control. Recall that a primary feature of the equation model is the ability to use it to solve the optimization problem. Thus, while the equations appear to be a near-perfect match for population and wealth levels without taxation, it is necessary to examine how the equations hold up when subjected to taxation. To that end, 100 tax schedules were generated at random and evaluated in both the ABM and the equation model. Figure 4.5 provides an example taken from these 100 schedules; the results for the other tax schedules are similar. Note that as in figure 4.4, ABM data is averaged over 100 runs. As in the no-taxation case, the equations appear to capture the dynamics of the ABM nearly perfectly, further indicating that the equation model is a suitable surrogate – with respect to the optimization problem – for the ABM.



(a) Population by region (b) Total wealth by region

Figure 4.5: Population and wealth data from the ABM (light colors) and the equation model (dark colors) for a randomly chosen tax structure.

4.4.1 Pareto optimization.

Having confirmed that the equation model truly captures the dynamics of the ABM with respect to the optimization problem, the equation model was used to solve the optimization problem. This was accomplished by a heuristic method known as *Pareto optimization.* Based on ideas first introduced by Italian economist Vilfredo Pareto in 1897 [90], this evolutionary algorithm is designed for multiobjective optimization problems. The *Pareto frontier* consists of the set of solutions with the following property: no objective can be improved upon without some tradeoff in another. For our example, this means that the Pareto frontier consists of solutions wherein if we wish to obtain fewer deaths, we must use a tax schedule that results in lower income; alternately, if we wish to increase tax income, we can only do so with schedules that cause more deaths. A typical evolutionary algorithm defines a fitness function that combines all of the objectives, perhaps by attaching weights to each, and then searches the solution space for the solution that maximizes fitness. Pareto optimization side-steps the issue of defining such a fitness function by searching for a set of solutions – each is optimal with respect to some choice of weights. There are several advantages to this approach: determining appropriate weights for a fitness function can be highly arbitrary, and may inadvertently focus attention on too narrow a region of the solution space. Additionally, heuristic algorithms typically examine many hundreds of solutions, but in the end return only one. By making use of a broader set of the solution space, Pareto optimization is more efficient – in fact, run time is similar to the case where only one solution is returned. A comprehensive review of heuristic methods for multiobjective optimization is outside the scope of this paper (for an extensive reference of other methods see [20]); Pareto optimization was chosen because it is intuitive, effective, and novel in its application to optimization for ABMs.

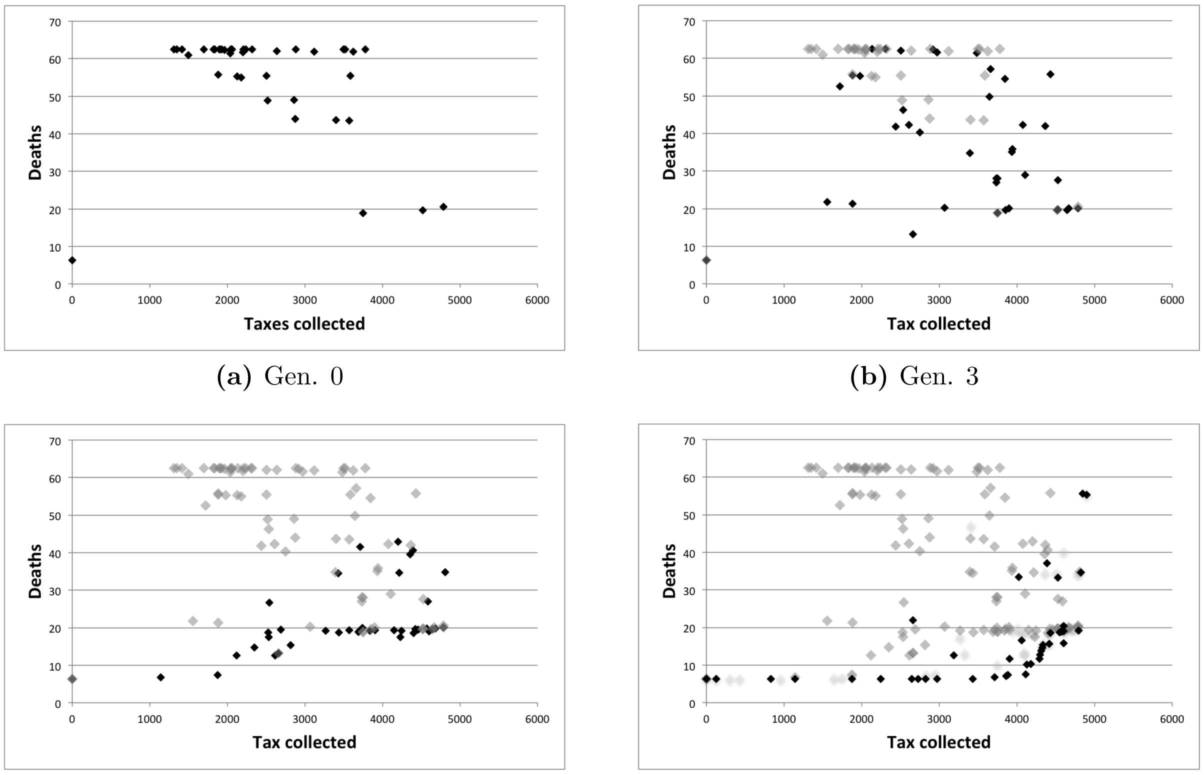
The algorithm. The Pareto optimization algorithm is based on a standard evolutionary algorithm: a population of solutions are generated at random and evaluated. Solutions from this population that contain advantageous features with respect to the optimization problem are then used to 'breed' subsequent solutions – child solutions are composed by selecting values from parent solutions at random. These solutions are then subjected to mutation, wherein taxation is either increased or decreased, in order to broaden the range of the Pareto frontier. The next generation of solutions consists of the Pareto optimal solutions from the prior generations plus the newly bred solutions. This process repeats for a fixed number of generations, or until some stopping condition is met. In this case, a solution is advantageous if it is either Pareto optimal with respect to the current generation of solutions, or if it has few neighbors in solution space. This latter trait rewards more unique solutions and encourages exploration of the solution space. The algorithm used here is based on work by Horn et al [60]; pseudo-code is provided in Algorithm 3.

Algorithm 3 Pareto optimization pseudo-code.

```
generate random initial population of 40 solutions                ▷ each solution is a tax structure
while $generation < 80$ do
    evaluate current generation, call it $current\_pop$                   ▷ tally deaths and income
    determine Pareto frontier; call it $current\_frontier$
    let $next\_pop = current\_frontier$                         ▷ save frontiers between generations
    while size of $next\_pop <$ size of $generation\_size$ do   ▷ each generation contains a fixed number of solutions
        repeat
            choose two candidate solutions from $current\_pop$
            let $comp\_set$ = random subset of 5 solutions from $current\_pop$
            if exactly one candidate is Pareto dominant over $comp\_set$ then
                that candidate becomes parent                   ▷ add candidates more likely to be on frontier
            else if one candidate has fewer neighbors in solution space then   ▷ give preference to isolated solutions
                that candidate becomes parent
            else
                choose candidate at random to become parent
            end if
        until two parents have been chosen
        breed two new solutions ($a$ and $b$) using parents:
            for all components in parent solutions do                          ▷ uniform crossover
                select component from random parent
                add this component to $a$
                add corresponding component from other parent to $b$
            end for
            $mutation\_rate = 0.20((max\_gens - generation)/max\_gens)$       ▷ mutation decreases over time
            for all components in $a$ and $b$ do                                  ▷ perform mutation
                change component with probability $mutation\_rate$   ▷ permissible values: {0, 0.25, 0.5, 0.75, 1}
            end for
        end routine
        add $a$ and $b$ to $next\_pop$
    end while
    increase $generation$ by 1
end while
```

Evolution of the Pareto frontier. There are two objectives in our optimization problem: minimizing death, and maximizing taxes collected. Each tax structure is evaluated in the equation model, and the values for these two objectives are stored; we can then visualize the solutions as data points on a scatter plot, where the axes correspond to the objectives. Examining the state of these plots at various times during the optimization algorithm gives some indication of its progress. Figure 4.6 provides four snapshots of the algorithm, taken from generations 0, 3, 8, and 25. Values from past generations remain on the plot in a lighter shade in order to accentuate the evolution of the solutions toward the Pareto frontier – as generations progress, we tend to see solutions with fewer deaths and higher income. A notable feature of the optimization algorithm is that by giving preference to solutions with fewer neighbors, a wide variety of Pareto optimal solutions are found. This is hinted at in figure 4.6d, where we see solutions spread across a range of values.

Validation. While the evolutionary features observed in figure 4.6 indicate that the algorithm is working, they do not serve as total validation of Pareto optimization. A better validation comes from the comparison of results from the algorithm with results from a random search, which serves as a baseline for heuristic methods. The algorithm was run for a total of 80 generations, each consisting of 40 solutions. Since the Pareto frontier carries over each generation, the total number of different schedules evaluated by the Pareto optimization algorithm was 821. Hence, 821 random solutions were generated and evaluated for comparison. The results are presented in figure 4.7. The searches are equivalent in run time, indicating that Pareto optimization offers a substantial improvement on a random search. Note that a comprehensive overview of multi-objective optimization methods is beyond the



(a) Gen. 0

(b) Gen. 3

(c) Gen. 8

(d) Gen. 25

Figure 4.6: Snapshots of the evolutionary Pareto optimization algorithm.

scope of this text; however, this evaluation indicates the potential advantage of at least one such method. It is particularly worth noting that the algorithm returns a wide variety of solutions rather than just focusing on one region of the Pareto frontier. Another advantage of Pareto optimization is the insight offered into the choice of fitness function. The solution wherein no tax is imposed is a *de facto* member of the frontier since it is guaranteed to minimize deaths; however, as seen in figure 4.7, it is possible to obtain more than 4000 sugar as tax without visibly increasing the number of deaths. On the other hand, the tradeoff between death and taxes increases drastically between 4000 and 5000 sugar collected, indicating that beyond 4500 sugar or so, the tax can only be increased by a substantial sacrifice in the number of deaths. The shape of the frontier seems to suggest that for most reasonable fitness functions, there should be no more than 10 deaths and no less than 4000 sugar collected.

The final step in the validation process is to check how the solutions obtained via the equation model actually track back to the ABM – after all, the equation model was built in order to aid investigation into an ABM optimization problem. To that end, the solutions comprising the final generation of the Pareto algorithm were implemented in the ABM; deaths and taxes were tallied for each. The results predicted by the equation model and those obtained

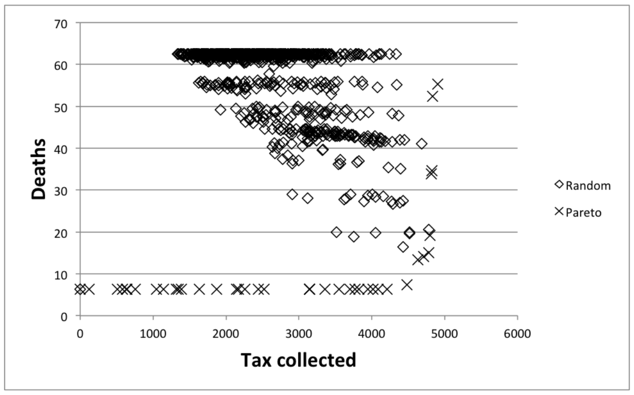


Figure 4.7: A comparison of results from a random search and those from the Pareto optimization algorithm.

via simulation in the ABM (with results averaged over multiple runs) are presented in figure 4.8. These results indicate that values obtained by the equation model are very similar to

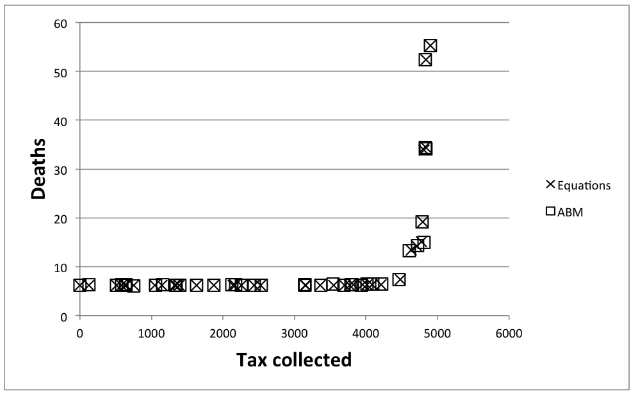


Figure 4.8: The Pareto frontier evaluated in the equation model and in the ABM.

those obtained via ABM simulation, further validating both the qualitative and quantitative accuracy of the equations with respect to the dynamics of interest.

4.5 Discussion

In this paper, a spatially heterogeneous agent-based model is approximated by a system of discrete difference equations, which are then used to solve an optimization problem. The example presented here illustrates an approach to solving optimization problems for agent-based models which ought to have wide appeal: ABMs are often created to investigate real-world systems, and optimization problems typically arise naturally in many such investigations. Here, the equation system was shown to capture pertinent ABM dynamics for a variety of conditions, and Pareto optimization was used as a novel means for solving the optimization problem. The results of this heuristic approach are encouraging, as they serve as an effective means for optimization when other theoretical approaches are not tractable in practice. Finally, the results obtained via the equations were shown to track faithfully back to the ABM, serving as a sort of *a posteriori* validation of the equation system. While there may be cases for which this approach is not suitable, this example serves as evidence that equation-based analysis of agent-based models holds the potential to be a powerful tool for a broad class of models.

FUTURE WORK. In this case, the equation system was built analytically from the rules of the ABM. However, in many ABMs this approach is not practical: either the rules are too complicated to express mathematically, or the dynamics of interest rely on too many factors. In such cases, phenomenological equations must be developed that fit ABM data. While principles from symbolic regression can be applied, they are not well-established in the framework of ABM analysis. This is an area of ongoing research and preliminary results are promising. It was mentioned earlier that there may be cases for which the approach presented here is inappropriate; hence, the ability to classify models based on susceptibility to equation-based analysis would be of particular interest. For any attempts at describing ABMs via equations, statistical methods need to be developed that can bring rigor to the discussion of model similarity; we are examining such methods in another ongoing project. Finally, a similar study to that presented here, using partial differential equations rather than difference equations, is currently underway. This will help contrast the advantages and disadvantages of the continuous and discrete modeling frameworks as applied to mathematical analysis of agent-based models.

Chapter 5

A Framework for a Mathematical Approach to Solving Optimization Problems for Agent-based Models

This project began as a Research Experience for Undergraduates (REU) project in Summer 2013; acknowledgements are due to Cara DeAngelis, Herb Susmann, and Erin Twohy. MSO is responsible for coding the SugarTax model, drafting the equation system, designing and implementing the Pareto algorithm in the ABM, and the comparison of equation and ABM results.

5.1 Introduction

Agent-based models (ABMs) provide a rich environment in which to model and study a wide variety of phenomena. At the macroscopic level they may involve interactions between planets, people, or traffic systems, while at the microscopic level they can be used to investigate interactions between proteins, molecules, or atoms. The visual component of ABMs makes them ideal for interdisciplinary research: experts in the domain being modeled need not be computer scientists in order to suggest experiments or to interpret results from a model. ABMs can save time, money, and resources by acting as *in silico* laboratories; in cases such as traffic scenarios and cancer studies they even have the potential to conduct otherwise infeasible or unethical experiments.

Once the desired system has been modeled, a natural follow-up question immediately arises: what should we *do* with it? In many cases, there are natural questions that can be posed and investigated using an ABM. Solving a posed question raised with regard to an ABM is an *optimization* problem. Various scenarios, inputs, and interventions are investigated in order to determine the best way to solve the problem. One drawback to these simulation models is their resistance to standard analytical approaches. A lack of rigorous mathematical formulation and inherent stochasticity makes consistent and reliable information more difficult to obtain.

One step in the process towards solving an optimization problem for an ABM comes from preliminary model analysis. A model is run under a variety of conditions and observations are made, giving some indication of the factors most pertinent to the question at hand. Studies concerning influenza [24, 75], malaria [39], cancer [129], and a variety of other biological systems [70, 107] have been conducted in this manner, as have studies in applied engineering [141] and economics [50]. A reasonable next step is to introduce interventions into an ABM in order to determine better protocol or policies. This has been investigated in studies focusing specifically on influenza [138, 86] and malaria [92] as well as more general disease control strategies [10] and resource-allocation problems [67]. These studies have all focused on ABMs directly, and rely on results obtained from simulation. Other studies have focused on differences between equation models and ABMs [106] and methods for translating ABMs to discrete [55, 76] and continuous [144, 134] mathematical formulations. An overview of simulation and mathematical techniques can be found in [93].

Perhaps the most critical issue in describing ABMs mathematically is space: while equations have been shown to describe population dynamics fairly well on a large scale [34, 44, 63], models with heterogeneous spatial structures have proven more difficult to approximate [61, 94, 99]. However, non-ABM-derived equation models are capable of handling spatial issues [8, 4, 88], indicating the importance of careful methodology in capturing spatially explicit ABM dynamics.

To summarize: ABMs have been used as simulation tools for investigation, and they have been described and analyzed mathematically via equation models, both with and without spatial heterogeneity. However, the use of equations to replicate simulation data as a means of solving optimization problems, while not unheard of (see [69, 68]), is not well-established. We propose a framework for this approach which combines discrete mathematical models, heuristic algorithms, and symbolic regression. The remainder of this section outlines the approach; in subsequent sections we introduce an ABM to use as a running example, pose an optimization problem, and apply the approach to solve it. Our results indicate that this approach can be used in a wide variety of ABMs, including those wherein space is critical.

5.1.1 A framework for solving optimization problems for ABMs

Before outlining the framework it is necessary to establish some terminology which will be used throughout the remainder of this work. ABMs often incorporate randomness into the local rules, so resetting the model and running the simulation again often produces variability in the data. Each simulation is referred to as a **run**. The set of potential answers to an optimization problem is the **solution space**, and each element of that set is a **solution** (note that a solution need not be optimal; it refers to any element of the solution space).

Figure 5.1 outlines the approach described in this work. First an ABM is selected and an optimization problem posed. Note that the problem posed determines the nature of the equations to be fit, so it is necessary to identify a problem prior to model conversion. Thus, even for a fixed ABM the conversion process may vary depending on the optimization problem under investigation. Once a problem is posed, pertinent data is generated using the ABM. The solution space for any optimization problem is typically too large for enumeration; thus a stratified sample is taken and data is generated for each solution in the sample. Given the stochasticity inherent in many ABMs, reliable data is obtained by averaging over many runs. The next phase is the conversion process. Here, discrete difference equations are fit to the data; these equations may be determined analytically or via numerical methods, depending on the complexity of the dynamics. The equations are deterministic and return expected values, so there is no stochasticity in the equation system. Note that the equations are fit over the entire stratified sample of the solution space. Next, the fit must be verified. Given that the sample data was used to generate the equations, one would expect a good fit and ought not use this data for validation. Thus, a *separate* stratified sample is chosen, evaluated both in the ABM and in the equations, and compared. This indicates the extent to which the equations can be expected to actually match the behavior of the ABM. While a variety of statistical methods may be used to quantify how well the equations fit the data, the final validation comes from comparison of solutions and thus the fit may only truly be established *a posteriori*. Once established, the equation system is used to solve the optimization problem. For certain models mathematical theory may allow for the employment of rigorous methodology; in our running example we employ heuristic methods.

The final steps of the framework are for validation purposes. The optimization problem is solved using the ABM itself via simulation optimization techniques in order to compare results from the equation system. Of course, the key advantage to using the equation system is precisely that one does not need to solve the problem using the ABM itself – indeed, by doing so the equations are not necessary at all. However, this part of the process actually serves an important purpose: this framework is only just being introduced, and as such it is necessary to establish its efficacy. Direct comparison with ABM results is the only way to accomplish this. Once the usefulness and validity of the framework is established, it will not be necessary to solve the optimization problem using the ABM directly.

5.2 SugarTax: A running example

The model we use as a subject for the framework presented here is a modified version of a model known as SugarScape [36]. SugarScape simulates a population of abstract agents

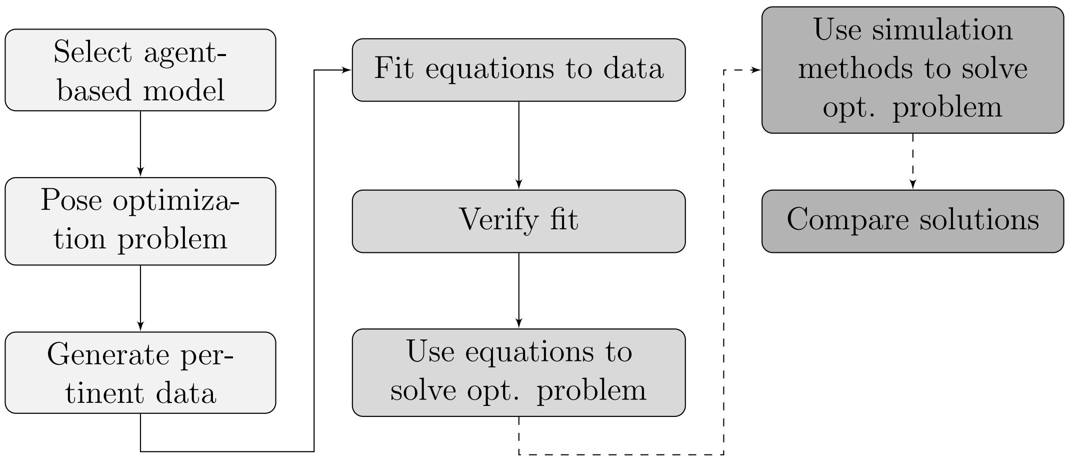

Select agent-based model
Pose optimization problem
Generate pertinent data
Fit equations to data
Verify fit
Use equations to solve opt. problem
Use simulation methods to solve opt. problem
Compare solutions

Figure 5.1: A flowchart of the framework presented in this work. Dashed lines indicate the validation process.

(hereafter referred to as 'ants') which patrol a fixed geographical landscape made up of sugar. The landscape contains peaks and valleys; as the simulation progresses one can observe the population clustering in high-sugar regions. Each time step, ants accumulate sugar based on their location and burn sugar based on their metabolism. Each ant has a fixed vision – that is, a distance they are able to see in the four principal directions – but vision levels vary from ant to ant.

As one of the earliest large-scale agent-based models, SugarScape is well known and well-studied in the modeling community. Originally introduced in the context of social science, the model has been used as the basis of studies on trade and economics [26], culture and migration [27], and wealth distribution [105]. SugarScape was chosen as a representative model for this framework for several reasons. One is its wide versatility: with only minor adjustments, the model can be studied in a number of different contexts, making it appealing to researchers from many areas. Another reason is the balance the model strikes between complexity and ease of use: SugarScape is spatially heterogeneous and contains agents with varying attributes, which makes traditional mean-field approximation problematic. At the same time, the model can be explained and understood relatively quickly, without need for specialization in any particular field.

In order to pose an optimization problem, we implemented taxation in the model: periodically, ants are taxed for their sugar based on their location (e.g., perhaps ants in regions with more sugar ought to be taxed more frequently than those in regions with less). Stated precisely, the optimization problem is the following: what is the ideal tax structure in order to maximize tax income while minimizing deaths? The more frequently ants are taxed, the more likely they are to run out of sugar and die, so the two objectives conflict. As a result, the answer to the optimization problem is not clear.

A detailed description of the model (with taxation) is provided in Appendix I. This description follows the Overview, Design concepts, and Details (ODD) protocol for agent-based models [47, 48]. The purpose of the ODD protocol is to to act as a template for describing agent-based models, and to provide sufficient detail for model dynamics and results to be reproduced. While SugarScape is explained in detail in [36], there are modifications in the model presented here – particularly, the addition of taxation – which are perhaps most easily explained via this detailed description. In order to emphasize the differences, and to avoid confusion with the original model, we refer to the model presented here as SugarTax. For those interested only in a general overview of the model, and in order to understand the framework outlined in Figure 5.1, the above description may suffice. Finally, it should be noted that there are many software platforms for agent-based modeling, many of which are free of charge. SugarTax was built using NetLogo [133] – in fact, the SugarScape code upon which it is based is included as a sample model in the NetLogo model library.

5.3 Deriving the equation model

In order to determine the fit of equations to ABM data, pertinent data must first be generated using the ABM. Of course, stochasticity is a key feature of most ABMs – in order to minimize discrepancy between simulations, data must be averaged over multiple runs. The extent to which the data can be fit by deterministic equations depends on the reliability of these averages and the associated standard deviations. The appropriate number of runs depends on the model; as such, it is not possible to prescribe a fixed number that is appropriate in every case. We have found 50 runs to be sufficient – indeed, one might argue that if the average over 50 runs of an ABM is not a good indicator of typical behavior, then the model is likely to be resistant to analysis via deterministic equations.

The optimization problem in SugarTax concerns minimizing death and maximizing income, and the tax rates are regional. Thus for every tax structure investigated, data and equations are needed for the following:

- wealth distribution in each region at each time step,
- income tax collected from each region at each time step,
- population in each region at each time step,
- deaths.

Table 5.1 summarizes the meaning of each term found in the equation model and figure 5.2 provides a labeled image of the landscape.

Term	Meaning
$w_{r,b}(t)$	Number of ants in region r and wealth bin b at time t
$p_r(t)$	Population in region r at time t
$m_{x,y}(t)$	Proportion of ants migrating from region x to region y at time t
$tax_r(t)$	Binary entry indicating taxation (0 corresponds to no taxation)

Table 5.1: Table of terms used in the equation model.

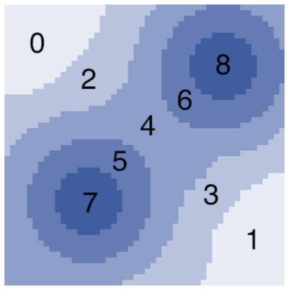


Figure 5.2: Spatial regions labeled by number.

5.3.1 Wealth and population

Rather than tracking the number of ants at each individual sugar level, they are tracked according to **bins**. With the exception of the first bin, all bins have size 10. Bin 1 contains the number of ants with sugar from $1-10$ and in general, the n^{th} bin contains the number of ants with sugar from $1+10(n-1)$ to $10n$. Note that in SugarTax, 'sugar' and 'wealth' are interchangeable terms. Bin 0 contains the number of ants with zero wealth – in other words, the number of deaths. Given the fixed metabolism, the maximum sugar level in the landscape, and the finite duration of a simulation, only fifteen bins are necessary, as no ant can accumulate more than 130 sugar. An example of a template wealth equation is provided in 5.1 – the number of ants in bin 1 and region 0.

$$\begin{aligned} w_{0,1}(t+1) = & (1-tax_0(t)) \cdot (0.8 \cdot (1-m_{0,2}(t)) \cdot w_{0,1}(t) + 0.2 \cdot (1-m_{0,2}(t)) \cdot w_{0,2}(t)) \\ & + tax_0(t) \cdot (0.8 \cdot (1-m_{0,2}(t)) \cdot w_{0,2}(t) + 0.2 \cdot (1-m_{0,2}(t)) \cdot w_{0,3}(t)). \end{aligned} \tag{5.1}$$

Note that if region 0 is not being taxed at this time step, then the first half of the equation is used; otherwise, the second half is used. Suppose that the region is not being taxed. Some ants will migrate into region 2, leaving $(1-m_{0,2}(t)) \cdot w_{0,1}(t)$. Since each ant has metabolism 2 and the bins have size 10, we expect 80% of the ants (specifically, those with sugar $3-10$) to remain in bin 1. Eating and metabolization occurs post-migration, so the amount that remains is thus $0.8(1-m_{0,2}(t))w_{0,1}(t)$. Similarly, of the ants that do not migrate into region 2 in bin 2, 20% of them (those with sugar $11-12$) will drop into bin 1; hence the term $0.2 \cdot (1-m_{0,2}(t) \cdot w_{0,2}(t)$. If, on the other hand, the region *is* being taxed, then the latter half of equation (5.1) comes into play: the terms are the same, but the bins are shifted by one. This is because the tax rate is 10 – precisely the same as the bin size – so every ant in that region is shifted down an extra bin when taxation occurs.

The equation for population in region 2 is provided in equation (5.2): it is the population that remains in the region after migration, minus deaths and plus those that migrated in from region 0. Note that migration does not occur from region 4 into region 2 (though migration from region 8 to region 6, for example, happens regularly).

$$p_2(t+1) = (1 - m_{2,4}(t))p_2(t) - w_{0,0}(t) + m_{0,2}(t)p_0(t). \tag{5.2}$$

5.3.2 Migration

While wealth and population equations are analytical, equations determining migration are not so easily captured. The difficulty stems from ABM features that are common to many models. As such, rather than altering SugarTax to simplify the equations, it is more beneficial to demonstrate a technique for dealing with these features. In particular, random execution order, asynchronous update, and the exclusion principle – that only one ant may occupy a grid cell at any time – make migration data difficult to fit. In this case, we do not know the *form* of the equations, but we do know the *inputs*. For example, consider region 2. We know that the migration rate depends on how many ants are in region 2, but it will also depend on the number of ants in region 4. This is because of the exclusion principle: if every space in region 4 is occupied, for example, then no migration can occur, regardless of how many ants are in region 2. On the other hand, if region 2 is densely occupied, fewer ants will migrate because many (or all) available spaces will be taken by the first ants to execute movement. The interplay of these two populations is not clear, and is difficult to determine probabilistically. Knowing the inputs allows us to use **symbolic regression** to determine the form of the equations. Symbolic regression builds on the idea of parameter estimation by allowing not only the parameters to change, but the terms in the equations as well. There are many software packages for symbolic regression; we used Eureqa [112], which contains many options for user customization. In order to fit migration data, we restricted the symbolic regression search to the four principal operations of addition, subtraction, multiplication, and divison (i.e., no trigonometric, exponential, or logarithmic functions were allowed). The algorithm was set to minimize the sum-squared error between time course data from the ABM and the equations. The migration equation out of region 2 is presented in (5.3).

$$m_{2,4}(t+1) = 6.39 \times 10^{-18} \cdot p_2(t)^4 \cdot p_4(t)^6. \tag{5.3}$$

The sample wealth equation in (5.1) is referred to as a **template** equation because once the entire system is formed, a parameter estimation algorithm is used to refine the fit – hence, exact parameter values are altered in the final system. The full model is provided in Appendix J.

5.4 Solving the optimization problem

The best way to solve an optimization problem using a system of equations depends on the details of the system and the preferences of the researcher. For sufficiently small systems, techniques from optimal control theory can be applied, both for continuous and discrete mathematical models. However, the number of equations necessary for a model the size of SugarTax (and many other ABMs) is typically too large for the practical application of control theory. At the same time, enumeration of the solution space is often computationally infeasible. Thus, in this study we employ heuristic methods to solve the optimization problem.

5.4.1 Pareto optimization

The two objectives in the SugarTax optimization problems conflict: by increasing tax rates, one expects to increase income but at the same time to cause more deaths. On the other hand, removing taxation entirely will certainly minimize deaths, but will minimize tax income as well. It is not likely that there is a tax structure that optimizes both variables: typically, any improvement in one of the objectives will require some sacrifice in the other. One strategy for multi-objective optimization problems is to weight the objectives separately and combine them into a fitness function, and then find the solution which optimizes this function. Though determination of appropriate weights has been investigated [45], it remains an inexact science – weights are frequently chosen arbitrarily and without justification.

Pareto optimization is a heuristic method that addresses this issue head-on by returning not just one solution, but a suite of solutions, each of which is optimal depending on some choice of weights. This method is based on (and named for) ideas posited by Italian economist Vilfredo Pareto [90]; it is a type of genetic algorithm [46] wherein a population of solutions is used to 'breed' better solutions in subsequent generations. In particular, a solution is *Pareto optimal* if it cannot be improved upon with respect to any objective without a sacrifice with respect to some other objective. The set of Pareto optimal solutions is known as the *Pareto frontier*. The goal of Pareto optimization is to determine this frontier as completely as possible. Given that there is no solution that is optimal with respect to every objective at the same time, the solution that is chosen from the frontier is a 'managerial' decision – it will depend on the priorities and interests of the researcher. Pseudo-code for the Pareto optimization algorithm used here is presented in 4; this algorithm has been adapted from [60]. A fairly comprehensive list of references on this and other methods of multi-objective optimization is maintained at [20].

Algorithm 4 SugarTax Pareto optimization pseudo-code.

```
generate random initial population of solutions    ▷ each solution is a tax structure
while $generation < max\_gens$ do
    evaluate current generation, call it $current\_pop$    ▷ tally deaths and income
    determine Pareto frontier; call it $current\_frontier$
    let $next\_pop = current\_frontier$    ▷ save frontiers between generations
    while size of $next\_pop <$ size of $generation\_size$ do    ▷ each generation contains a fixed number of solutions
        repeat
            choose two candidate solutions from $current\_pop$
            let $comp\_set$ = random subset of 5 solutions from $current\_pop$
            if exactly one candidate is Pareto dominant over $comp\_set$ then
                that candidate becomes parent    ▷ add candidates more likely to be on frontier
            else if one candidate has fewer neighbors in solution space then    ▷ give preference to isolated solutions
                that candidate becomes parent
            else
                choose candidate at random to become parent
            end if
        until two parents have been chosen
        Breed two new solutions ($a$ and $b$) using parents:
            for all components in parent solutions do    ▷ uniform crossover
                select component from random parent
                add this component to $a$
                add corresponding component from other parent to $b$
            end for
            $mutation\_rate = 0.20((max\_gens - generation)/max\_gens)$    ▷ mutation decreases over time
            for all components in $a$ and $b$ do    ▷ perform mutation
                change component with probability $mutation\_rate$    ▷ only permissible values are 0 and 1
            end for
        end routine
        add $a$ and $b$ to $next\_pop$
    end while
    increase $generation$ by 1
end while
```

5.4.2 Results

The final steps of the framework in Figure 5.1 concern validation: in order to determine whether or not the equation model can be used as a surrogate for the ABM, the results from the equation model must be validated in the ABM. In order to perform such a validation, the Pareto optimization outlined in Algorithm 4 was performed directly on the ABM, using repeated simulation in place of function evaluation. These results serve as a baseline for comparison – while not necessarily providing the global Pareto frontier, the results serve as the nearest attainable 'true' frontier. Hereafter we refer to these results as the *ABM frontier*.

The equation system described in section 5.3 fits the ABM data qualitatively, but in order to compare results with the ABM frontier, the frontier obtained by the equations must be re-evaluated in the ABM. This is not particularly time-intensive as the equation frontier consists of approximately 20 solutions only. The purpose of this re-evaluation is to correct for the qualitative nature of the fit: by evaluating the frontier directly in the ABM, results can be compared quantitatively with the ABM frontier, giving a better indication of the reliability of the equation model.

Figure 5.3 presents three sets of data: values predicted by the equation model, the values actually obtained when the equation frontier is evaluated in the ABM, and the values obtained by performing Pareto optimization on the ABM directly. It is worth first discussing the shape of the ABM frontier and what it implies about the optimization problem. Note that there are a cluster of solutions that all cause approximately 70 deaths, but for whom the tax income varies greatly. The shape of this subset implies that there isn't much justi-

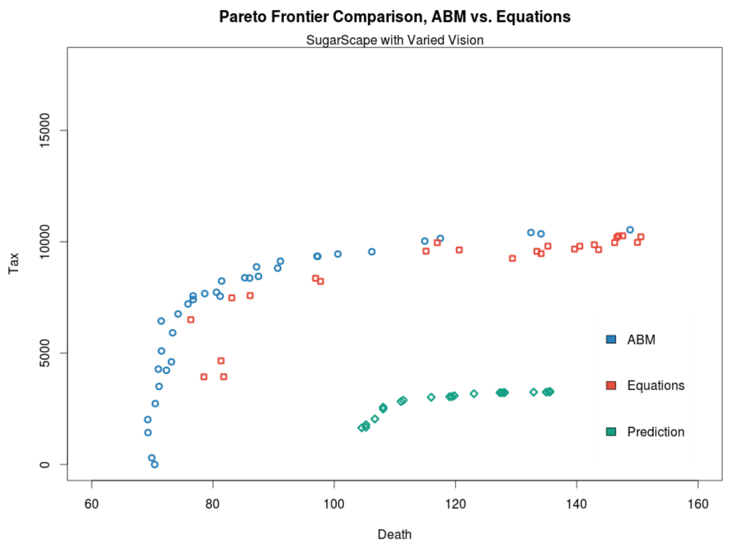


Figure 5.3: Results from Pareto optimization.

fication in choosing any solution that collects less than $5000 - 6000$ sugar in tax, because there are solutions in this income range that essentially minimize deaths. In contrast, the relative flatness of the remainder of the frontier suggests that there is not much justification for selecting any solution that causes more than $90 - 100$ deaths: there are solutions that certainly cause more deaths, but they do not substantially increase the amount of tax collected. Thus the shape of the frontier itself indicates that regardless of how the objectives are weighted, optimal solutions are likely to be those that collect at least 5000 sugar and cause no more than 100 deaths. This is information that is only obtainable by examining the entire Pareto frontier.

While the values predicted by the equation model are not a good quantitative fit, they do match ABM data qualitatively. In particular, the basic shape of the frontier is maintained upon re-evaluation, validating that model dynamics are qualitatively preserved in the equation model. Once the solutions are re-evaluated, the values are much closer to those found by the ABM, further supporting the equation model as a surrogate for the ABM. In particular, a wide range of the Pareto frontier is obtained via the equation model. These results are promising, as they demonstrate the possibility for equation systems to describe spatially heterogeneous ABMs. Specifically, the results encourage further investigation of the framework presented here as a viable means for solving optimization problems with agent-based models.

5.5 Discussion

There are many ways to perform the framework presented in Figure 5.1 – for example, one can use differential equations (stochastic or otherwise), discrete dynamical systems, polynomial dynamical systems, or other mathematical representations to fit data. The mechanism used to fit the data is up to the researcher, as are the statistical techniques used to evaluate the fit. The optimization problem can be solved in a variety of ways as well, including methods from optimal control theory for continuous and discrete systems, heuristic algorithms, and other simulation optimization strategies. The details presented in the running example merely provide one avenue for implementing this framework, but the implementation is ultimately up to the researcher. For that reason, this framework may be beneficial to researchers with a broad range of interests. It is a step towards a rigorous mathematical analysis of agent-based models, and it is our intention to continue to expand and develop the framework.

That said, there are a few issues that bear mentioning. While several steps presented here rely on heuristic methods, it would be preferable to have access to mathematical theory that would preclude such methods. Rigorous statistical verification, optimization, and analysis ought to be the next steps in developing this framework. In addition, it is likely that there are agent-based models for which this framework simply does not apply or would not work – models with many agent types and more complicated spatial dynamics, for example. It is not our intention to claim that the framework can be applied in all cases. On the contrary, a classification of models for which this approach would not and could not work would be of great benefit. Such a classification would be a boon to anyone interested in analysis of agent-based models, as it is difficult at present to classify models in terms of complexity. This is another issue we plan to investigate in future work. While there may be cases where this framework does not apply, it has the potential to be an indispensable tool in any mathematical approach to agent-based modeling. It is certainly worth pursuing as a viable means for solving optimization problems for agent-based models.

Chapter 6

A Computational Model of Invasive Aspergillosis in the Lung

Acknowledgements are due to Chris Gaskill and Eric Kernfeld, who were involved with the development of the two-dimensional version of this model. MSO contributed to the design of the agent-based model and is responsible for its implementation. Acknowledgements are also due to Reinhard Laubenbacher, Chris Lawrence, and Borna Mehrad, all of whom contributed to the design of the model.

6.1 Introduction

Invasive aspergillosis represents a major and growing health problem in the U.S. and around the world. The growing population of immunocompromised patients, including those with haematologic malignancies, and stem cell- or solid organ-transplant recipients are at highest risk for this disease [118]. In addition to conventionally immunosuppressed patients, other large populations are also at risk of this infection, including individuals with fibrocavitary tuberculosis in developing countries who develop chronic invasive aspergillosis as a secondary infection [43]; it is estimated that in 2007, $372,000$ of the 7.7 million new cases of pulmonary tuberculosis world-wide also developed chronic pulmonary aspergillosis [28]. The introduction of new antifungal drugs during the last decade, principally azole-based compounds capitalizing on new insights into the molecular structure of the fungal cell wall, has dramatically improved disease outcomes, but mortality rates remain approximately 30% in recent surveys [118, 54]. In addition, increased resistance to these new drugs [49] raises the specter of a "perfect storm," as it has been called in [28], combining a rapidly growing patient population with a diminished repertoire of treatment options.

To date, the focus of the search for new therapeutics has been largely on fungal targets. But more recent promising efforts have looked to the host, in particular host immunity [11, 12].

The host immune response to respiratory fungal pathogens is multilayered, involving the actions of several cell types, such as epithelial cells, mononuclear phagocytes, neutrophils, and dendritic cells, among others. However, a full exploration of the possibilities for antifungal therapeutics targeted at the host requires a better understanding of the innate host response. The complexity of the dynamic regulatory molecular networks and the multi-scale nature of the innate immune response strongly suggest taking a systems biology approach [59], as done in, e.g., [1, 80].

A substantial body of literature supports the critical role of iron homeostasis in *Aspergillus* biology. *Aspergillus* species adapt to iron-limited environments by activating a system of intracellular and secreted siderophores that scavenge from the environment and store it. In in vitro studies, *Aspergillus* siderophores remove iron from transferrin in human serum [56] and impair macrophage iron uptake [115]; conversely, neutrophil lactoferrin inhibits *Aspergillus* conidial growth by sequestering extracellular iron [142]. In animal models, mutant *Aspergillus* species with defective siderophore systems are avirulent [113], and therapeutic iron chelation has an additive benefit to antifungal antibiotics [62]. These mechanisms appear to be clinically important, since among immunocompromised stem cell transplant patients, clinically unsuspected iron overload is an independent risk factor with invasive aspergillosis [72, 2]. Taken together, these data suggest that the competition for iron is a key component of the pathogenesis of invasive aspergillosis.

The innate immune response to invasive aspergillosis is difficult to study. Studying dynamic molecular networks in a human host is, in most cases, impossible. In the study of the innate immune response to A.f. a number of in vitro and in vivo approaches have been used successfully. These include the in vitro interaction of A.f. with leukocytes and epithelial cells [52, 120]. In addition, animal models have been a valuable tool to investigate the complexities of cell-cell interactions and inflammatory pathways in a realistic system. These complementary approaches have led to recognition of neutrophils, macrophages, dendritic cells, and lung epithelial cells as key early players in host response to Aspergillus species [97].

Mathematical modeling related to respiratory pathogens and the host response has received only limited attention so far, and the field has not yet taken full advantage of this tool. As examples of existing studies, a mathematical model of the host response to pneumococcal lung infection [117] depicts a 3-stage process involving alveolar macrophages, neutrophils, and monocyte-derived macrophages. The model captures cell counts as well as the concentrations of certain cytokines, validated through data from mouse experiments. The goal of the model is to capture the effect of the initial inoculum on disease outcome. Also, a mathematical model of key regulatory networks in A.f. was published recently [80]. The work presented in this paper represents the first step toward a multi scale systems biology model of invasive aspergillosis in the lung, focused on the role of iron. Here, we present the tissue level component of the model, validated with in *in vivo* data from a mouse model of invasive aspergillosis.

6.1.1 Related work

A number of immune system simulators have been presented for general use. C-ImmSim [13, 14, 6] is a multi-purpose agent-based model of a section of lymph node tissue; a web version is available as well [108]. This model focuses on immune response to a range of pathogens, as well as HIV infection and cancer therapy. More recently, C-ImmSim has incorporated assessment of molecular binding [107]. An extension of this model, known as Vacc-Imm, focuses on peptide vaccination in cancer therapy [137]. SIMMUNE [89] is another general immune system simulator that can be adapted to simulate other systems as well. The Basic Immune Simulator [42] emphasizes interactions between innate and adaptive immunity. Another model, SIMISYS [66], is a cellular automata model which reproduces certain aspects of the human immune system. SimB16 [95] is a more specific simulation model focusing on immune response to B16 melanoma.

Studies with narrower biological focus include simulation models examining influenza [110], brain tumors [143], metastasis [19], Toxoplasma gondii [64], pancreatic macrophages [87], and cell motility [70]. Additionally, a number of studies discuss the usefulness of agent-based modeling in biological systems – for a review of host-pathogen ABMs see [5]; for a review of ABMs and the immune system see [18]. The versatility of agent-based models of biological systems is discussed in [57]; models at the intracellular, cellular, and organism levels are used to demonstrate this point.

The model presented here focuses on fungal infection in the lung. As such, it is worth briefly reviewing similar lung models and their respective emphases. A multi-scale ABM of cancer in the lung was introduced in 2007 [128], followed by a separate study emphasizing metastasis in 2009 [102]. Inflammation and fibrosis were the focus of a more recent study [9], wherein an ABM of the lung was subjected to particulate exposure. Certain players in immune response in the lung have been studied in detail: see [114] for a study of macrophage response to bacteria in the lung, including a discussion of chemotaxis (a key feature of the model introduced here). Agent movement is also examined in [123], wherein a model of neutrophil response to fungal presence in the lung is investigated. While various aspects of the model presented below have been examined in previous studies, there are no agent-based models tying all of these processes together in quite the same way. Related work serves as a precedent for simulation models of biological systems (in particular, the lung); however, the scope and focus of this particular model make it a novel entry to the field of *in silico* models.

6.2 The model

The agent-based model is a three-dimensional simulation of a section of lung tissue; see figure 6.1 for a labeled image of a simulation snapshot. *A. fumigatus* conidia begin at one end of

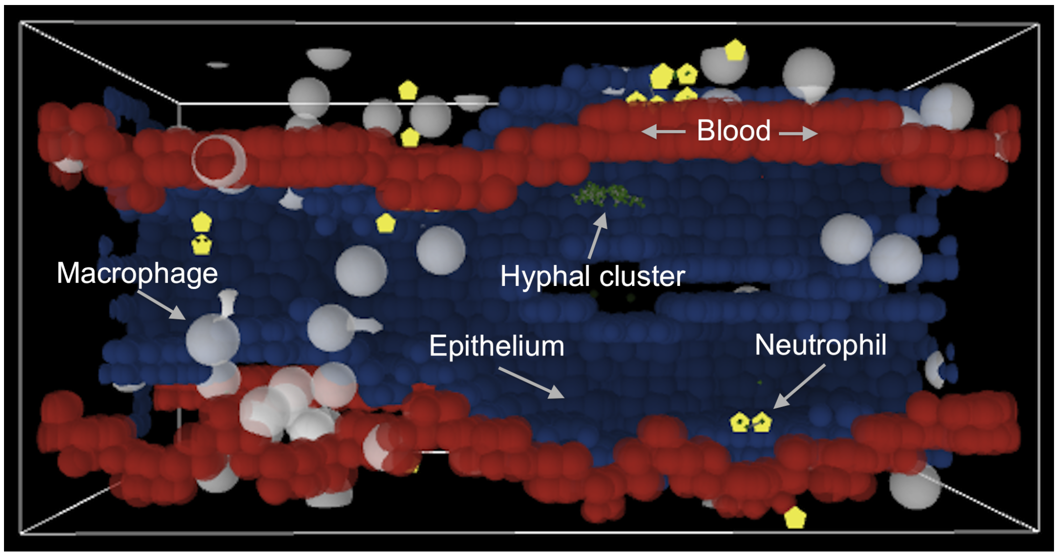


Figure 6.1: A labeled snapshot from the simulation window.

a branching airway. As the simulation progresses, the conidia drift through the airway (see figure 6.2). Upon encountering the epithelium, most conidia are swept away due to ciliary beating [122]; some, however, are not swept away and lodge themselves in the epithelial layer. Occasionally, these lodged conidia even enter an epithelial cell [131], though most remain outside the cell. Left undisturbed, conidial spores germinate and hyphal clusters grow in the interstitial space (see figure 6.3). The description of the modeling of the immune response to *A. fumigatus* provided here is meant to serve as an overview of the ABM and how the agents function; for a detailed description see Appendix K.

Epithelial cells act as the second line of defense against *A. fumigatus* (after the cilia). Epithelial cells recognize the presence of conidia and emit cytokines into the interstitial space in an effort to initiate an immune response. After a period of swelling, the conidia germinate and attempt to grow – either between epithelial cells and into the interstitial space, or directly through the cell if the conidia has been internalized by an epithelial cell. The cells that grow out of a germinated spore are fungal hyphae; these tend to form clusters as they grow.

There are two immune cell types in the ABM: macrophages and neutrophils. Epithelial cells can emit two different kinds of cytokines: one that attracts macrophages, and another that attracts neutrophils. These are tracked separately because certain cytokines attract neutrophils but not macrophages [140] while others (such as MCAF) do just the reverse [124]. Since macrophages are the primary defense against conidia and neutrophils are the primary defense against hyphae, macrophage and neutrophil cytokine production levels are determined by the presence of these fungal cell types, respectively. The cytokines diffuse through the interstitial tissue, eventually reaching the bloodstream. Once the cytokine level

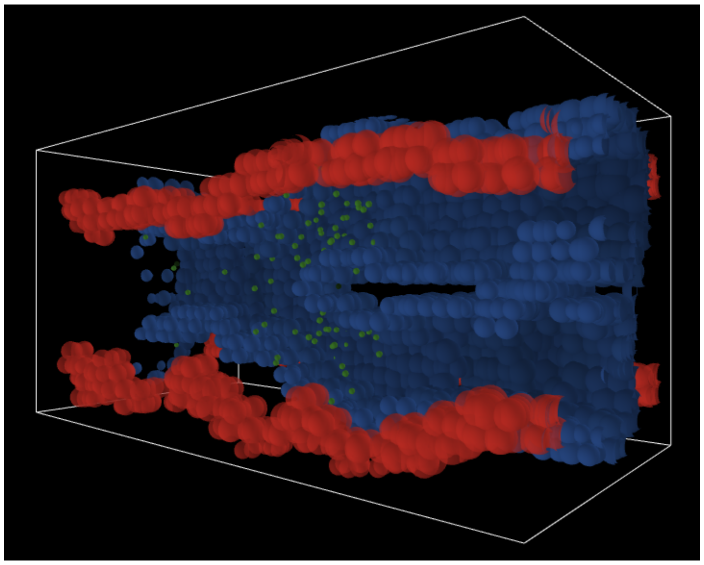


Figure 6.2: A. fumigatus spores drifting through the branching airway.

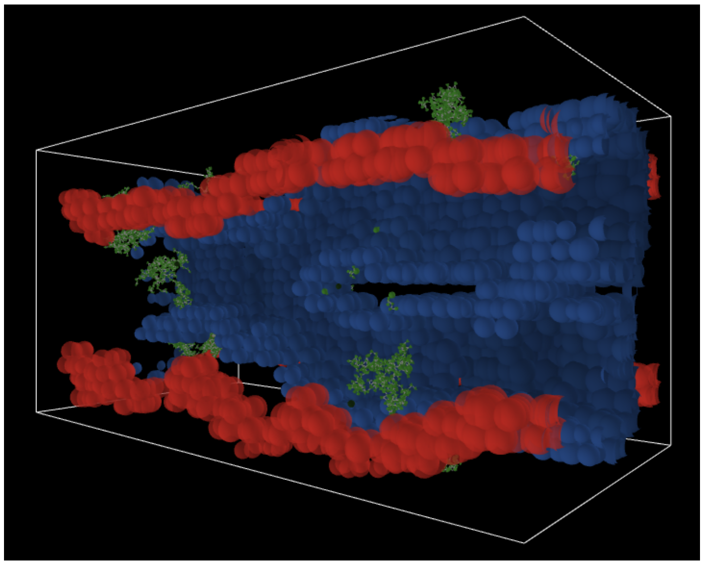


Figure 6.3: Clusters of *A. fumigatus* hyphae.

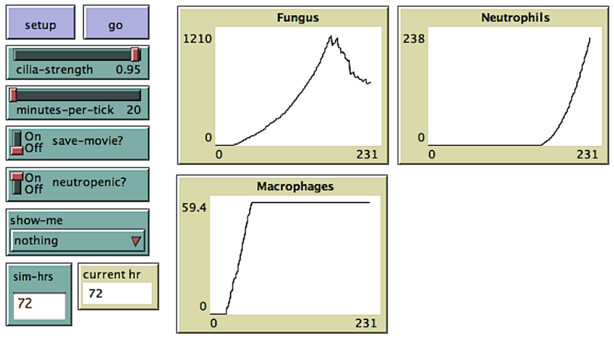
in the blood for a particular immune cell rises above a certain threshold, an immune cell of the appropriate type is recruited to that site via the blood.

Recruited macrophages and neutrophils enter the interstitial space via the bloodstream. There, chemotaxis is simulated as the immune cells follow the direction of highest cytokine concentration [25], leading to the area with highest fungal concentration. Along the way, the immune cells may encounter fungal cells, whereupon such cells are attacked. Macrophages may internalize up to five fungal cells. Once internalized by a macrophage, cells cannot escape and cannot germinate. The 'health' of the internalized fungal cells is reduced according to the time scale of the model. Upon encountering fungal hyphae, neutrophils destroy the cells via granulation: granules are deposited at that location until all fungal cells have been destroyed. Neutrophils have a finite number of granules – once these have all been deposited, the neutrophil may no longer affect fungal cells.

The cytokine level in the surrounding area of a macrophage or neutrophil must be above a certain threshold in order for chemotaxis to occur. If there aren't any such locations nearby, macrophages and neutrophils move through the tissue randomly. Since the lifetime of neutrophils is between 24 and 48 hours [119], neutrophils die after a certain amount of time has passed. Macrophages begin to drift out of the represented cross section once all conidial spores have been eliminated.

The focus of this simulation is on the battle for iron: while host cells require iron in order to maintain function, the fungal cells are competing for iron in order to grow. Fungal hyphae cannot grow if the parent cell does not contain enough iron, which they obtain from surrounding tissue. At each time step in the simulation, iron enters the tissue via the bloodstream. From there, iron diffuses into the interstitial tissue, where fungal cells attempt to acquire it. Macrophages halt iron export by the production of ferroportin, which binds to hepcidin and is subsequently degraded [130]. Hence, in the ABM, macrophages have no effect on the iron levels of their surrounding area.

The model interface (see 6.4) allows the user to investigate various scenarios: the time scale, simulation duration, cilia strength, whether or not the virtual patient is neutropenic, and the amount of iron that enters the blood are all settings which can be changed in the interface. Additionally, there is an option to save a movie of the simulation. Alongside the visualization there are plots tracking cell counts and iron levels; the user can easily configure custom plots. By building up the ABM from local rules based on individual cell behaviors, the ABM is capable of capturing subtle biological features. At the same time, the intuitive interface and three-dimensional visualization are conducive to innovative interdisciplinary research.



cilia-strength 0.95
minutes-per-tick 20
save-movie? On Off
neutropenic? On Off
show-me nothing
sim-hrs 72
current hr 72

Figure 6.4: The interface for the simulation.

6.3 Discussion

The development of this model is part of an ongoing collaboration with mathematicians, bioinformaticians, and experimental biologists. While the preliminary results indicate that this is a promising modeling paradigm for invasive aspergillosis, it will only continue to improve in detail and accuracy as the model is refined based on data from the laboratory and the literature. The model already exhibits several features observed *in vivo*. One of these features, the clusters of hyphae which tend to form, arises naturally as a result of local individual interactions. It is our hope to observe more emergent properties like this, and to validate simulation results with laboratory data. The next step for this model is to incorporate multiple scales: epithelial cells will have internal iron metabolism networks determining iron export and import, and this will tie in to the tissue-level dynamics as iron levels in the interstitial space are affected. Similar intracellular networks are being developed for fungal cells, and we hope to develop intracellular networks for neutrophils and macrophages as well.

Bibliography

[1] D. Albrecht, O. Kniemeyer, F. Mech, M. Gunzer, A. Brakhage, and R. Guthke. On the way toward systems biology of Aspergillus fumigatus infection. *Int. J. Med. Microbiol.*, 301(5):453–459, Jun 2011.

[2] A. Altes, A. F. Remacha, P. Sarda, F. J. Sancho, A. Sureda, R. Martino, J. Briones, S. Brunet, C. Canals, and J. Sierra. Frequent severe liver iron overload after stem cell transplantation and its possible association with invasive aspergillosis. *Bone Marrow Transplant.*, 34(6):505–509, Sep 2004.

[3] D. Altman. *Practical Statistics for Medical Research.* Chapman and Hall, London, 1991.

[4] F. Ball and P. Neal. Network epidemic models with two levels of mixing. *Mathematical Biosciences*, 212(1):69 – 87, 2008.

[5] A. L. Bauer, C. A. Beauchemin, and A. S. Perelson. Agent-based modeling of host-pathogen systems: The successes and challenges. *Inf Sci (Ny)*, 179(10):1379–1389, Apr 2009.

[6] M. Bernaschi and F. Castiglione. Design and implementation of an immune system simulator. *Comput. Biol. Med.*, 31:303–331, Sep 2001.

[7] F. Botterel, K. Gross, O. Ibrahim-Granet, K. Khoufache, V. Escabasse, A. Coste, C. Cordonnier, E. Escudier, and S. Bretagne. Phagocytosis of Aspergillus fumigatus conidia by primary nasal epithelial cells in vitro. *BMC Microbiol.*, 8:97, 2008.

[8] T. Britton, T. Kypraios, and P. O'Neill. Inference for epidemics with three levels of mixing: Methodology and application to a measles outbreak. *Scandinavian Journal of Statistics*, 38(3):578–599, 2011.

[9] B. N. Brown, I. M. Price, F. R. Toapanta, D. R. DeAlmeida, C. A. Wiley, T. M. Ross, T. D. Oury, and Y. Vodovotz. An agent-based model of inflammation and fibrosis following particulate exposure in the lung. *Math Biosci*, 231(2):186–196, Jun 2011.

[10] D. L. Buckeridge, C. Jauvin, A. Okhmatovskaia, and A. D. Verma. Simulation Analysis Platform (SnAP): a Tool for Evaluation of Public Health Surveillance and Disease Control Strategies. *AMIA Annu Symp Proc*, 2011:161–170, 2011.

[11] A. Carvalho, C. Cunha, F. Bistoni, and L. Romani. Immunotherapy of aspergillosis. *Clin. Microbiol. Infect.*, 18(2):120–125, Feb 2012.

[12] A. Carvalho, C. Cunha, R. G. Iannitti, A. Casagrande, F. Bistoni, F. Aversa, and L. Romani. Host defense pathways against fungi: the basis for vaccines and immunotherapy. *Front Microbiol*, 3:176, 2012.

[13] F. Castiglione. C-ImmSim Simulator. Institute for Computing Applications, National Research Council (CNR) of Italy, 1995. `http://www.iac.cnr.it/~filippo/C-ImmSim.html`.

[14] F. Castiglione, M. Bernaschi, and S. Succi. Simulating the immune response on a distributed parallel computer. *Int J Mod Phys C*, 8(3):527–545, 1997.

[15] F. Castiglione, F. Pappalardo, M. Bernaschi, and S. Motta. Optimization of HAART with genetic algorithms and agent-based models of HIV infection. *Bioinformatics*, 23(24):3350–3355, 2007.

[16] Federico Cecconi, Marco Campenni, Giulia Andrighetto, and Rosaria Conte. What do agent-based and equation-based modelling tell us about social conventions: The clash between abm and ebm in a congestion game framework. *Journal of Artificial Societies and Social Simulation*, 13(1):6, 2010.

[17] D. L. Chao, M. E. Halloran, V. J. Obenchain, and I. M. Longini. FluTE, a publicly available stochastic influenza epidemic simulation model. *PLoS Comput. Biol.*, 6:e1000656, Jan 2010.

[18] A. K. Chavali, E. P. Gianchandani, K. S. Tung, M. B. Lawrence, S. M. Peirce, and J. A. Papin. Characterizing emergent properties of immunological systems with multi-cellular rule-based computational modeling. *Trends Immunol.*, 29(12):589–599, Dec 2008.

[19] J. Chen, K. Sprouffske, Q. Huang, and C. C. Maley. Solving the puzzle of metastasis: the evolution of cell migration in neoplasms. *PLoS ONE*, 6:e17933, 2011.

[20] C.A. Coello. List of references on evolutionary multiobjective optimization, 2013. `http://www.lania.mx/ ccoello/EMOO/EMOObib.html`. Archived at `http://www.webcitation.org/6HhFo4K5H`.

[21] J. Cohen. Weighted kappa: nominal scale agreement with provision for scaled disagreement or partial credit. *Psychol Bull*, 70(4):213–220, Oct 1968.

[22] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[23] C. Collins, S. Lenhart, S. Nanda, Jie Xiong, K. Yakovlev, and J. Yong. Optimal control of harvesting in a stochastic metapopulation model. *Optimal Control Applications and Methods*, 33(2):127–142, 2012.

[24] P. Cooley, S. Brown, J. Cajka, B. Chasteen, L. Ganapathi, J. Grefenstette, C. R. Hollingsworth, B. Y. Lee, B. Levine, W. D. Wheaton, and D. K. Wagener. The role of subway travel in an influenza epidemic: a New York City simulation. *J Urban Health*, 88:982–995, Oct 2011.

[25] T. R. Dagenais and N. P. Keller. Pathogenesis of Aspergillus fumigatus in Invasive Aspergillosis. *Clin. Microbiol. Rev.*, 22(3):447–465, Jul 2009.

[26] Monica Dascàlu, Eduard Franti, and George Stefan. Modeling production with artificial societies: the emergence of social structure. In S. Bandini, R. Serra, and F.Suggi Liverani, editors, *Cellular Automata: Research Towards Industry*, pages 218–229. Springer London, 1998.

[27] Jeffrey S Dean, George J Gumerman, Joshua M Epstein, Robert L Axtell, Alan C Swedlund, Miles T Parker, and Stephen McCarroll. Understanding anasazi culture change through agent-based modeling. *Dynamics in human and primate societies. Oxford University Press, Oxford*, pages 179–206, 2000.

[28] David W Denning, Alex Pleuvry, and Donald C Cole. Global burden of chronic pulmonary aspergillosis as a sequel to pulmonary tuberculosis. *Bulletin of the World Health Organization*, 89:864 – 872, 12 2011.

[29] R. D. Diamond and R. A. Clark. Damage to Aspergillus fumigatus and Rhizopus oryzae hyphae by oxidative and nonoxidative microbicidal products of human neutrophils in vitro. *Infect. Immun.*, 38(2):487–495, Nov 1982.

[30] W. Ding, L. J. Gross, K. Langston, S. Lenhart, and L. A. Real. Rabies in raccoons: optimal control for a discrete time model on a spatial grid. *J Biol Dyn*, 1(4):379–393, Oct 2007.

[31] Raphaël Duboz, Éric Ramat, and Philippe Preux. Scale transfer modeling: Using emergent computation for coupling an ordinary differential equation system with a reactive agent model. *Systems Analysis Modelling Simulation*, 43(6):793–814, 2003.

[32] R. Durrett and S. Levin. The importance of being discrete (and spatial). *Theoretical Population Biology*, 46(3):363 – 394, 1994.

[33] H. Dyke Parunak, Robert Savit, and Rick L. Riolo. Agent-based modeling vs. equation-based modeling: A case study and users guide. In Jaime Simão Sichman, Rosaria Conte, and Nigel Gilbert, editors, *Multi-Agent Systems and Agent-Based Simulation*, volume 1534 of *Lecture Notes in Computer Science*, pages 10–25. Springer Berlin Heidelberg, 1998.

[34] M. Edwards, S. Huet, F. Goreaud, and G. Deffuant. Comparing an individual-based model of behavior diffusion with its mean field aggregate approximation. *J. Artificial Societies Soc. Simulation*, 6(4), 2003.

[35] Khaled El Emam. Benchmarking kappa: interrater agreement in software process assessments. *Empirical Software Engineering*, 4(2):113–133, 1999.

[36] Joshua M. Epstein and Robert Axtell. *Growing artificial societies: social science from the bottom up.* The Brookings Institution, Washington, DC, USA, 1996.

[37] Barbara Di Eugenio. On the usage of kappa to evaluate agreement on coding tasks. In *In Proceedings of the Second International Conference on Language Resources and Evaluation*, pages 441–444, 2000.

[38] L. Fahse, C. Wissel, and V. Grimm. Reconciling classical and individual-based approaches in theoretical population ecology: a protocol for extracting population parameters from individual-based models. *Am. Nat.*, 152(6):838–852, Dec 1998.

[39] Jordi Ferrer, Clara Prats, Daniel Lopez, Joaquim Valls, and Domingo Gargallo. Contribution of individual-based models in malaria elimination strategy design. *Malaria Journal*, 9(Suppl 2):P9, 2010.

[40] J. Fleiss. *Statistical Methods for Rates and Proportions.* John Wiley And Sons, Hoboken, NJ, 1981.

[41] Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382, 1971.

[42] V. A. Folcik, G. C. An, and C. G. Orosz. The Basic Immune Simulator: an agent-based model to study the interactions between innate and adaptive immunity. *Theor Biol Med Model*, 4:39, 2007.

[43] J. P. Gangneux, C. Camus, and B. Philippe. [Epidemiology of and risk factors for invasive aspergillosis in nonneutropenic patients]. *Rev Mal Respir*, 25(2):139–153, Feb 2008.

[44] J. Gani and S. Yakowitz. Error bounds for deterministic approximations to Markov processes, with applications to epidemic models. *J. Appl. Prob*, 32(4):1063–1076, 1995.

[45] Michael A. Gennert and A.L. Yuille. Determining the optimal weights in multiple objective function optimization. In *Computer Vision., Second International Conference on*, pages 87–89, 1988.

[46] D.E. Goldberg. *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley, Reading, MA, 1989.

[47] 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 Jørgensen, Wolf M. Mooij, Birgit Müller, Guy Pe'er, Cyril Piou, Steven F. Railsback, Andrew M. Robbins, Martha M. Robbins, Eva Rossmanith, Nadja Rüger, Espen Strand, Sami Souissi, Richard A. Stillman, Rune Vabø, 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.

[48] 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.

[49] I. Hadrich, F. Makni, S. Neji, S. Abbes, F. Cheikhrouhou, H. Trabelsi, H. Sellami, and A. Ayadi. Invasive aspergillosis: resistance to antifungal drugs. *Mycopathologia*, 174(2):131–141, Aug 2012.

[50] Kathrin Happe. Agent-based modelling and sensitivity analysis by experimental design and metamodelling: An application to modelling regional structural change. 2005 International Congress, August 23-27, 2005, Copenhagen, Denmark 24464, European Association of Agricultural Economists, August 2005.

[51] Y. Harada, H. Ezoe, Y. Iwasa, H. Matsuda, and K. Sato. Population persistence and spatially limited social interaction. *Theoretical Population Biology*, 48(1):65 – 91, 1995.

[52] M. Hasenberg, J. Behnsen, S. Krappmann, A. Brakhage, and M. Gunzer. Phagocyte responses towards Aspergillus fumigatus. *Int. J. Med. Microbiol.*, 301(5):436–444, Jun 2011.

[53] Naïla Hayek. Infinite horizon multiobjective optimal control problems in the discrete time case. *Optimization*, 60(4):509–529, 2011.

[54] R. Herbrecht, D. W. Denning, T. F. Patterson, J. E. Bennett, R. E. Greene, J. W. Oestmann, W. V. Kern, K. A. Marr, P. Ribaud, O. Lortholary, R. Sylvester, R. H. Rubin, J. R. Wingard, P. Stark, C. Durand, D. Caillot, E. Thiel, P. H. Chandrasekar, M. R. Hodges, H. T. Schlamm, P. F. Troke, and B. de Pauw. Voriconazole versus amphotericin B for primary therapy of invasive aspergillosis. *N. Engl. J. Med.*, 347(6):408–415, Aug 2002.

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

[56] A. H. Hissen, J. M. Chow, L. J. Pinto, and M. M. Moore. Survival of Aspergillus fumigatus in serum involves removal of iron from transferrin: the role of siderophores. *Infect. Immun.*, 72(3):1402–1408, Mar 2004.

[57] M. Holcombe, S. Adra, M. Bicak, S. Chin, S. Coakley, A. I. Graham, J. Green, C. Greenough, D. Jackson, M. Kiran, S. Macneil, A. Maleki-Dizaji, P. McMinn, M. Pogson, R. Poole, E. Qwarnstrom, F. Ratnieks, M. D. Rolfe, R. Smallwood, T. Sun, and D. Worth. Modelling complex biological systems using an agent-based approach. *Integr Biol (Camb)*, 4:53–64, Jan 2012.

[58] W. W. Hope, V. Petraitis, R. Petraitiene, T. Aghamolla, J. Bacher, and T. J. Walsh. The initial 96 hours of invasive pulmonary aspergillosis: histopathology, comparative kinetics of galactomannan and $(1- > 3)$ β-d-glucan and consequences of delayed antifungal therapy. *Antimicrob. Agents Chemother.*, 54(11):4879–4886, Nov 2010.

[59] F. Horn, T. Heinekamp, O. Kniemeyer, J. Pollmacher, V. Valiante, and A. A. Brakhage. Systems biology of fungal infection. *Front Microbiol*, 3:108, 2012.

[60] J. Horn, N. Nafpliotis, and D.E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 82–87 vol.1, 1994.

[61] S. Huet, M. Edwards, and G. Deffuant. Taking into account the variations of neighbourhood sizes in the mean-field approximation of the threshold model on a random network. *J. Artificial Societies Soc. Simulation*, 10(1), 2007.

[62] A. S. Ibrahim, T. Gebremariam, S. W. French, J. E. Edwards, and B. Spellberg. The iron chelator deferasirox enhances liposomal amphotericin B efficacy in treating murine invasive pulmonary aspergillosis. *J. Antimicrob. Chemother.*, 65(2):289–292, Feb 2010.

[63] J. A. Jacquez and C. P. Simon. The stochastic SI model with recruitment and deaths. I. Comparison with the closed SIS model. *Math Biosci*, 117(1-2):77–125, 1993.

[64] W. Jiang, A. M. Sullivan, C. Su, and X. Zhao. An agent-based model for the transmission dynamics of Toxoplasma gondii. *J. Theor. Biol.*, 293:15–26, Jan 2012.

[65] David E. Joslin and David P. Clements. "Squeaky wheel" optimization. *J. Artificial Intelligence Res.*, 10:353–373 (electronic), 1999.

[66] J. K. Kalita, K. Chandrashekar, R. Hans, and P. Selvam. Computational modelling and simulation of the immune system. *Int J Bioinform Res Appl*, 2:63–88, 2006.

[67] P. Kasaie, W.D. Kelton, A. Vaghefi, and S.G.R.J. Naini. Toward optimal resource-allocation for control of epidemics: An agent-based-simulation approach. In *Winter Simulation Conference (WSC), Proceedings of the 2010*, pages 2237 –2248, Dec. 2010.

[68] P. S. Kim, P. P. Lee, and D. Levy. A PDE model for imatinib-treated chronic myelogenous leukemia. *Bull. Math. Biol.*, 70:1994–2016, Oct 2008.

[69] P. S. Kim, P. P. Lee, and D. Levy. Modeling imatinib-treated chronic myelogenous leukemia: reducing the complexity of agent-based models. *Bull. Math. Biol.*, 70:728–744, Apr 2008.

[70] M. T. Klann, A. Lapin, and M. Reuss. Agent-based simulation of reactions in the crowded and structured intracellular environment: Influence of mobility and location of the reactants. *BMC Syst Biol*, 5:71, 2011.

[71] L. Kohidai and G. Csaba. Chemotaxis and chemotactic selection induced with cytokines (IL-8, RANTES and TNF-alpha) in the unicellular Tetrahymena pyriformis. *Cytokine*, 10(7):481–486, Jul 1998.

[72] D. P. Kontoyiannis, G. Chamilos, R. E. Lewis, S. Giralt, J. Cortes, I. I. Raad, J. T. Manning, and X. Han. Increased bone marrow iron stores is an independent risk factor for invasive aspergillosis in patients with high-risk hematologic malignancies and recipients of allogeneic hematopoietic stem cell transplantation. *Cancer*, 110(6):1303–1306, Sep 2007.

[73] K. Krippendorff. *Content Analysis: an Introduction to its Methodology.* Sage Publications, Beverly Hills, CA, 1980.

[74] J. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, Mar 1977.

[75] M. Laskowski, B. C. Demianyk, J. Witt, S. N. Mukhi, M. R. Friesen, and R. D. McLeod. Agent-based modeling of the spread of influenza-like illness in an emergency department: a simulation study. *IEEE Trans Inf Technol Biomed*, 15:877–889, Nov 2011.

[76] R. Laubenbacher, A. S. Jarrah, H. Mortveit, and S. S. Ravi. Mathematical formalism for agent-based modeling. In R.A. Meyers, editor, *Encyclopedia of Complexity and Systems Science*, pages 160–176. Springer, 2009.

[77] Reinhard Laubenbacher, Franziska Hinkelmann, and Matt Oremland. Agent-based models and optimal control in biology: A discrete approach. In Raina Robeva and Terrell L. Hodge, editors, *Mathematical Concepts and Methods in Modern Biology*, pages 143 – 178. Academic Press, Boston, 2013.

[78] S. Lenhart and J.T. Workman. *Optimal Control Applied to Biological Models.* Chapman and Hall/CRC, Boca Raton, FL, 2007.

[79] Jingpeng Li, Andrew J. Parkes, and Edmund K. Burke. Evolutionary squeaky wheel optimization: A new analysis framework. *Evolutionary Computation*, Jan 2011. published online.

[80] J. Linde, P. Hortschansky, E. Fazius, A. A. Brakhage, R. Guthke, and H. Haas. Regulatory interactions for iron homeostasis in Aspergillus fumigatus inferred by a Systems Biology approach. *BMC Syst Biol*, 6:6, 2012.

[81] P. L. Lollini, S. Motta, and F. Pappalardo. Discovery of cancer vaccination protocols with a genetic algorithm driving an agent based simulator. *BMC Bioinformatics*, 7:352, 2006.

[82] P.L. Lollini, F. Castiglione, and S. Motta. Modelling and simulation of cancer immunoprevention vaccine. *Int J Cancer*, 77:937–941, 1998.

[83] Carlos A. Lugo and Alan J. McKane. Quasicycles in a spatial predator-prey model. *Phys. Rev. E*, 78:051911, Nov 2008.

[84] Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, Keith Sullivan, and Gabriel Balan. MASON: A multi-agent simulation environment. *Simulation: Transactions of the society for Modeling and Simulation International*, 82(7):517–527, 2005.

[85] E. K. Manavathu, J. Cutright, and P. H. Chandrasekar. Comparative study of susceptibilities of germinated and ungerminated conidia of Aspergillus fumigatus to various antifungal agents. *J. Clin. Microbiol.*, 37(3):858–861, Mar 1999.

[86] L. Mao. Agent-based simulation for weekend-extension strategies to mitigate influenza outbreaks. *BMC Public Health*, 11:522, 2011.

[87] I. V. Martinez, E. J. Gomez, M. E. Hernando, R. Villares, and M. Mellado. Agent-based model of macrophage action on endocrine pancreas. *Int J Data Min Bioinform*, 6(4):355–368, 2012.

[88] Hirotsugu Matsuda, Naofumi Ogita, Akira Sasaki, and Kazunori Satō. Statistical mechanics of population. *Progress of Theoretical Physics*, 88(6):1035–1049, 1992.

[89] M. Meier-Schellersheim and G. Mack. Simmune, a tool for simulating and analyzing immune system behavior. *CoRR*, cs.MA/9903017, 1999.

[90] H.L. Moore. Cours d'économie politique. by Vilfredo Pareto, professeur à l'Université de Lausanne. Vol. i. pp. 430. i896. Vol. ii. pp. 426. i897. Lausanne: F. Rouge. *The ANNALS of the American Academy of Political and Social Science*, 9(3):128–131, 1897.

[91] Michael J. North, Nicholson T. Collier, and Jerry R. Vos. Experiences creating three implementations of the repast agent modeling toolkit. *ACM Trans. Model. Comput. Simul.*, 16(1):1–25, January 2006.

[92] L. C. Okell, C. J. Drakeley, T. Bousema, C. J. Whitty, and A. C. Ghani. Modelling the impact of artemisinin combination therapy and long-acting treatments on malaria transmission intensity. *PLoS Med.*, 5:e226; discussion e226, Nov 2008.

[93] Matthew Oremland. Optimization and optimal control of agent-based models. Master's thesis, Virginia Polytechnic Institute and State University, May 2011.

[94] O. Ovaskainen and S. J. Cornell. Space and stochasticity in population dynamics. *Proc. Natl. Acad. Sci. U.S.A.*, 103(34):12781–12786, Aug 2006.

[95] F. Pappalardo, I. Martinez Forero, M. Pennisi, A. Palazon, I. Melero, and S. Motta. SimB16: modeling induced immune system response against B16-melanoma. *PLoS ONE*, 6:e26523, 2011.

[96] F. Pappalardo, M. Pennisi, F. Castiglione, and S. Motta. Vaccine protocols optimization: in silico experiences. *Biotechnol. Adv.*, 28:82–93, 2010.

[97] S. J. Park and B. Mehrad. Innate immunity to Aspergillus species. *Clin. Microbiol. Rev.*, 22(4):535–551, Oct 2009.

[98] H. Parunak, Robert Savit, and Rick L. Riolo. Agent-based modeling vs. equation-based modeling: A case study and users guide. In Jaime Simão Sichman, Rosaria Conte, and Nigel Gilbert, editors, *Multi-Agent Systems and Agent-Based Simulation*, volume 1534 of *Lecture Notes in Computer Science*, pages 10–25. Springer Berlin Heidelberg, 1998.

[99] M Pascual and S. Levin. From individuals to population densities: searching for the intermediate scale of nontrivial determinism. *Ecology*, 80(7):2225–2236, 1999.

[100] Rajan Patel, Ira M. Longini, Jr., and M. Elizabeth Halloran. Finding optimal vaccination strategies for pandemic influenza using genetic algorithms. *J. Theoret. Biol.*, 234(2):201–212, 2005.

[101] M. Pennisi, R. Catanuto, F. Pappalardo, and S. Motta. Optimal vaccination schedules using simulated annealing. *Bioinformatics*, 24:1740–1742, Aug 2008.

[102] Marzio Pennisi, Francesco Pappalardo, and Santo Motta. Agent based modeling of lung metastasis-immune system competition. In Paul S. Andrews, Jon Timmis, Nick D.L. Owens, Uwe Aickelin, Emma Hart, Andrew Hone, and Andy M. Tyrrell, editors, *Artificial Immune Systems*, volume 5666 of *Lecture Notes in Computer Science*, pages 1–3. Springer Berlin Heidelberg, 2009.

[103] B. Philippe, O. Ibrahim-Granet, M. C. Prevost, M. A. Gougerot-Pocidalo, M. Sanchez Perez, A. Van der Meeren, and J. P. Latge. Killing of Aspergillus fumigatus by alveolar macrophages is mediated by reactive oxidant intermediates. *Infect. Immun.*, 71(6):3034–3042, Jun 2003.

[104] Nicolas Picard and Alain Franc. Aggregation of an individual-based space-dependent model of forest dynamics into distribution-based and space-independent models. *Ecological Modelling*, 145(1):69 – 84, 2001.

[105] Arash Rahman, Saeed Setayeshi, and M. Shamsaei. An analysis to wealth distribution based on sugarscape model in an artificial society. *International Journal of Engineering*, 20(3):211–224, 2007.

[106] Hazhir Rahmandad and John Sterman. Heterogeneity and network structure in the dynamics of diffusion: Comparing agent-based and differential equation models. *Management Science*, 54(5):998–1014, 2008.

[107] N. Rapin, O. Lund, M. Bernaschi, and F. Castiglione. Computational immunology meets bioinformatics: the use of prediction tools for molecular binding in the simulation of the immune system. *PLoS ONE*, 5:e9862, 2010.

[108] N. Rapin, O. Lund, and F. Castiglione. Immune system simulation online. *Bioinformatics*, 27:2013–2014, Jul 2011.

[109] J. H. Rex, J. E. Bennett, J. I. Gallin, H. L. Malech, and D. A. Melnick. Normal and deficient neutrophils can cooperate to damage Aspergillus fumigatus hyphae. *J. Infect. Dis.*, 162(2):523–528, Aug 1990.

[110] B. Roche, J. M. Drake, and P. Rohani. An agent-based model to study the epidemiological and evolutionary dynamics of Influenza viruses. *BMC Bioinformatics*, 12:87, 2011.

[111] Andre M. De Roos, Edward Mccauley, and William G. Wilson. Mobility versus density-limited predator–prey dynamics on different spatial scales. *Proceedings: Biological Sciences*, 246(1316):pp. 117–122, 1991.

[112] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

[113] M. Schrettl, E. Bignell, C. Kragl, C. Joechl, T. Rogers, H. N. Arst, K. Haynes, and H. Haas. Siderophore biosynthesis but not reductive iron assimilation is essential for Aspergillus fumigatus virulence. *J. Exp. Med.*, 200(9):1213–1219, Nov 2004.

[114] Jose L. Segovia-Juarez, Suman Ganguli, and Denise Kirschner. Identifying control mechanisms of granuloma formation during m. tuberculosis infection using an agent-based model. *Journal of Theoretical Biology*, 231(3):357 – 376, 2004.

[115] M. Seifert, M. Nairz, A. Schroll, M. Schrettl, H. Haas, and G. Weiss. Effects of the Aspergillus fumigatus siderophore systems on the regulation of macrophage immune effector pathways and iron homeostasis. *Immunobiology*, 213(9-10):767–778, 2008.

[116] J. Sim and C. C. Wright. The kappa statistic in reliability studies: use, interpretation, and sample size requirements. *Phys Ther*, 85(3):257–268, Mar 2005.

[117] A. M. Smith, J. A. McCullers, and F. R. Adler. Mathematical model of a three-stage innate immune response to a pneumococcal lung infection. *J. Theor. Biol.*, 276(1):106–116, May 2011.

[118] W. J. Steinbach, K. A. Marr, E. J. Anaissie, N. Azie, S. P. Quan, H. U. Meier-Kriesche, S. Apewokin, and D. L. Horn. Clinical epidemiology of 960 patients with invasive aspergillosis from the PATH Alliance registry. *J. Infect.*, 65(5):453–464, Nov 2012.

[119] Alan Stevens, James S. Lowe, and Barbara Young. *Wheater's Basic Histopathology: A Color Atlas and Text (Wheater's Histology and Pathology)*. Churchill Livingstone, 4 edition, October 2002.

[120] W. K. Sun, X. Lu, X. Li, Q. Y. Sun, X. Su, Y. Song, H. M. Sun, and Y. Shi. Dectin-1 is inducible and plays a crucial role in Aspergillus-induced innate immune responses in human bronchial epithelial cells. *Eur. J. Clin. Microbiol. Infect. Dis.*, 31(10):2755–2764, Oct 2012.

[121] A. Swierniak, M. Kimmel, and J. Smieja. Mathematical modeling as a tool for planning anticancer therapy. *Eur. J. Pharmacol.*, 625(1-3):108–121, Dec 2009.

[122] A. B. Thompson, R. A. Robbins, D. J. Romberger, J. H. Sisson, J. R. Spurzem, H. Teschler, and S. I. Rennard. Immunological functions of the pulmonary epithelium. *Eur. Respir. J.*, 8(1):127–149, Jan 1995.

[123] C. Tokarski, S. Hummert, F. Mech, M. T. Figge, S. Germerodt, A. Schroeter, and S. Schuster. Agent-based modeling approach of immune defense against spores of opportunistic human pathogenic fungi. *Front Microbiol*, 3:129, 2012.

[124] A. J. Valente, D. T. Graves, C. E. Vialle-Valentin, R. Delgado, and C. J. Schwartz. Purification of a monocyte chemotactic factor secreted by nonhuman primate vascular cells in culture. *Biochemistry*, 27(11):4162–4168, May 1988.

[125] Alan Veliz-Cuba, Abdul Salam Jarrah, and Reinhard Laubenbacher. Polynomial algebra of discrete models in systems biology. *Bioinformatics*, 26(13):1637–1643, 2010.

[126] J. E. Wang, A. Warris, E. A. Ellingsen, P. F. Jorgensen, T. H. Flo, T. Espevik, R. Solberg, P. E. Verweij, and A. O. Aasen. Involvement of CD14 and toll-like receptors in activation of human monocytes by Aspergillus fumigatus hyphae. *Infect. Immun.*, 69(4):2402–2406, Apr 2001.

[127] Ting Wang and Xiaolong Zhang. 3D protein structure prediction with genetic tabu search algorithm in off-lattice AB model. In *Proceedings of the 2009 Second International Symposium on Knowledge Acquisition and Modeling - Volume 01*, KAM '09, pages 43–46, Washington, DC, USA, 2009. IEEE Computer Society.

[128] Z. Wang, L. Zhang, J. Sagotsky, and T. S. Deisboeck. Simulating non-small cell lung cancer with a multiscale agent-based model. *Theor Biol Med Model*, 4:50, 2007.

[129] Zhihui Wang, Veronika Bordas, and Thomas Deisboeck. Identification of Critical Molecular Components in a Multiscale Cancer Model Based on the Integration of Monte Carlo, Resampling, and ANOVA. *Frontiers in Physiology*, 2(0), 2011.

[130] R. J. Ward, R. R. Crichton, D. L. Taylor, L. Della Corte, S. K. Srai, and D. T. Dexter. Iron and the immune system. *J Neural Transm*, 118(3):315–328, Mar 2011.

[131] J. A. Wasylnka and M. M. Moore. Uptake of Aspergillus fumigatus Conidia by phagocytic and nonphagocytic cells in vitro: quantitation using strains expressing green fluorescent protein. *Infect. Immun.*, 70(6):3156–3163, Jun 2002.

[132] J. A. Wasylnka and M. M. Moore. Aspergillus fumigatus conidia survive and germinate in acidic organelles of A549 epithelial cells. *J. Cell. Sci.*, 116(Pt 8):1579–1587, Apr 2003.

[133] U. Wilensky. Netlogo. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 2009. `http://ccl.northwestern.edu/netlogo/`.

[134] W. G. Wilson. Resolving discrepancies between deterministic population models and individual-based simulations. *Am. Nat.*, 151(2):116–134, Feb 1998.

[135] W.G. Wilson, A.M. Deroos, and E. Mccauley. Spatial instabilities within the diffusive lotka-volterra system: Individual-based simulation results. *Theoretical Population Biology*, 43(1):91 – 127, 1993.

[136] W.G. Wilson, E. McCauley, and A.M. Roos. Effect of dimensionality on lotka-volterra predator-prey dynamics: Individual based simulation results. *Bulletin of Mathematical Biology*, 57:507–526, 1995.

[137] A. L. Woelke, J. von Eichborn, M. S. Murgueitio, C. L. Worth, F. Castiglione, and R. Preissner. Development of immune-specific interaction potentials and their application in the multi-agent-system VaccImm. *PLoS ONE*, 6:e23257, 2011.

[138] Y. Yang, P. M. Atkinson, and D. Ettema. Analysis of CDC social control measures using an agent-based simulation of an influenza epidemic in a city. *BMC Infect. Dis.*, 11:199, 2011.

[139] Serhat Yeşilyurt and Anthony T. Patera. Surrogates for numerical simulations; optimization of eddy-promoter heat exchangers. *Computer Methods in Applied Mechanics and Engineering*, 121(14):231 – 257, 1995.

[140] T Yoshimura, K Matsushima, S Tanaka, E A Robinson, E Appella, J J Oppenheim, and E J Leonard. Purification of a human monocyte-derived neutrophil chemotactic factor that has peptide sequence similarity to other host defense cytokines. *Proceedings of the National Academy of Sciences*, 84(24):9233–9237, 1987.

[141] H. Zahedmanesh and C. Lally. A multiscale mechanobiological modelling framework using agent-based models and finite element analysis: application to vascular tissue engineering. *Biomech Model Mechanobiol*, May 2011.

[142] K. A. Zarember, J. A. Sugui, Y. C. Chang, K. J. Kwon-Chung, and J. I. Gallin. Human polymorphonuclear leukocytes inhibit Aspergillus fumigatus conidial growth by lactoferrin-mediated iron depletion. *J. Immunol.*, 178(10):6367–6373, May 2007.

[143] L. Zhang, B. Jiang, Y. Wu, C. Strouthos, P. Z. Sun, J. Su, and X. Zhou. Developing a multiscale, multi-resolution agent-based brain tumor model by graphics processing units. *Theor Biol Med Model*, 8:46, Dec 2011.

[144] Y. Zou, V. A. Fonoberov, M. Fonoberova, I. Mezic, and I. G. Kevrekidis. Model reduction for agent-based social simulation: coarse-graining a civil violence model. *Phys Rev E Stat Nonlin Soft Matter Phys*, 85(6 Pt 2):066106, Jun 2012.

Appendices

Appendix A

Overview, Design concepts, and Details (ODD) protocol for Rabbits and Grass

The model upon which this version is based is included in the sample library of NetLogo [133], a popular agent-based modeling platform. The description here is warranted as it includes the mechanics of an optimization problem, the details of which are not available elsewhere.

A.1 Purpose

The purpose of this model is to examine population dynamics of a simple environmental system. In particular, it is a model of rabbits eating grass in a field. On each day of the simulation, poison can be placed on the field in order to kill the rabbits. This version of the model is an attempt to answer the following optimization question: what is the best way of controlling (i.e., minimizing) the rabbit population while also minimizing the amount of poison used?

A.2 Entities, state variables, and scales

This section contains a description of the grid cells, spatial and temporal scales, and the rabbits. It also contains a description of the format of a *poison strategy*, the investigation of which is the key feature of the model.

Grid cells, spatial scale, and temporal scale. The world is a square grid of discrete cells, representing a field. The grid is toroidal: edges wrap around both in the horizontal and vertical directions. The distance from the center of a cell to a neighboring horizontal or vertical cell is 1 unit (thus the distance between two diagonal cells is $\sqrt{2}$). Units are abstract spatial measurements. Time steps are also abstract discrete units. A simulation consists of a finite number of time steps. The only state variable for each cell indicates whether or not the cell currently contains grass. When grass is eaten on a grid cell there is a certain probability that it will grow back at each time step. This growth happens spontaneously.

State variable	Name	Value
Side length of field	s	50 grid cells
Total grid cells	N	2500
Presence of grass	$grass?$	0 = no grass, 1 = grass.
Grass growth probability	γ	0.02
Simulation time	$total\_sim\_time$	100 time steps

Table A.1: Grid cell state variables.

Rabbits. Each time step, rabbits move, eat grass (or not), and reproduce (or not). Reproduction is asexual and based on energy level, which is raised when a rabbit eats. Rabbits lose energy both by moving and by spawning new rabbits. If a rabbit's energy level drops to 0 or lower the rabbit dies.

State variable	Name	Value
Movement cost	$move\_cost$	0.5
Energy from food	$food\_energy$	3
Birth threshold	$birth\_threshold$	8
Current energy level	$energy$	varies

Table A.2: Rabbit state variables.

Poison schedule. A poison schedule u is a vector of length $total\_sim\_time$, with each entry either 0 or 1. Each entry corresponds to one time step in the simulation; 0 means that poison is not used and 1 means that poison is used. Thus, there are a total of $2^{total\_sim\_time}$ possible poison schedules. The poison has a maximum efficacy which degrades over time with repeated use. If the poison is not used the efficacy increases again, up to the maximum.

State variable	Name	Value
Maximum efficacy	p_{max}	0.3
Degradation rate	p_{deg}	0.5
Current efficacy	p_{eff}	varies in $(0, p_{max}]$

Table A.3: Poison schedule details.

A.3 Process overview and scheduling

In order to minimize ambiguity details of model execution are presented as pseudo-code; see Algorithm 5.

Algorithm 5 Rabbits and Grass process pseudo-code.

```
setup world and initialize rabbits
set poison schedule $u$
while time steps < $total\_sim\_time$ do
    store population and grass levels
    rabbit routine:
        Face left by random amount up to 45 degrees
        Face right by random amount up to 45 degrees          ▷ simulates ‘wiggling’ movement
        move forward 1 unit
        $energy = energy - move\_cost$
        if $u(t-1) = 1$ then
            $p_{eff} = (1 - p_{deg})p_{eff}$                  ▷ efficacy degrades on repeated use
        else
            $p_{eff} = p_{eff} + p_{deg}(p_{max} - p_{eff})$  ▷ efficacy increases if not used
        end if
        if $u(t) = 1$ and $rand(0,1) < p_{eff}$ then die end if   ▷ probability of being poisoned
        if $grass? = 1$ here then
            $energy = energy + food\_energy$
            grid cell: $grass? = 0$
        end if
        if $energy > birth\_threshold$ then
            $energy = energy/2$                               ▷ energy is halved for reproduction
            create new rabbit here      ▷ rabbit inherits location and energy values from parent
            new rabbit moves
        end if
        if $energy \leq 0$ then die end if
    end routine
    for all grid cells with $grass? = 0$ do
        if $rand(0,1) < \gamma$ then $grass? = 1$ end if
    end for
    advance one time step
end while
write data to file
```

A.4 Design concepts

In the ODD protocol [47, 48] there are eleven design concepts. Those which do not apply have been omitted.

Basic principles. In essence, this model is a predator-prey system wherein the rabbits are predators and the grass is prey. Introduction of poison into the model, and having that poison modeled as a direct external influence on population levels, creates a natural setting for an optimization problem. One can study the effect of various poison strategies on population levels – in terms of minimizing the rabbit population it can be thought of as a harvesting problem, but in terms of minimizing poison it can be thought of as resource allocation.

Emergence. Rabbit population and grass levels tend to oscillate as the simulation progresses. The frequency and amplitude of these oscillations can be affected by parameter settings and initial values and hence may be described as emergent model dynamics.

Interaction. Agent interaction is indirect: since rabbit movement is executed serially, it is possible that other rabbits deplete all of the grass in a particular rabbit's potential field of movement, thereby reducing or eliminating the chance for that rabbit to gain energy.

Stochasticity. Rabbit movement is totally random in that they cannot sense whether neighboring grid cells contain grass or not. Whether grass grows back on an empty grid cell is also random, and a grid cell that has been empty for several time steps is no more likely to grow grass than a cell which has only just become empty.

Observation. Rabbit population and grass counts are recorded at each time step. The total number of rabbits alive during the course of a simulation serves as a measure of fitness of the poison schedule.

A.5 Initialization

At initialization, 20% of the grid cells contain grass; these are chosen at random. There are 150 rabbits placed at random locations throughout the grid. Each begins with a random amount of energy between 0 and 9 inclusive (rabbits with 0 energy may survive the first time step by eating grass). Total simulation time is 100 time steps and each simulation contains a poison schedule u, described in section A.2.

A.6 Optimization

Since the multi-objective optimization problem is the key feature of the model as presented here, a few clarifying details are in order. The objectives of the optimization problem are to determine, for the parameter values provided, a set of Pareto optimal poison schedules which minimize the number of rabbits while also minimizing the amount of poison used.

The number of rabbits refers to the total number of rabbits alive during the course of a simulation – not just those alive at the end of the final time step. Since a poison schedule is a binary vector of length $total\_sim\_time$, the amount of poison used is represented by the sum of the entries of that vector.

Appendix B

Overview, Design concepts, and Details protocol for SugarScape with taxation

B.1 Purpose

The version of SugarScape presented here is a modified version of the original SugarScape [36], a model in which abstract entities roam a landscape made of sugar. These agents are periodically taxed for their sugar stores – the tax rate is constant but the frequency differs from region to region. The purpose of this version of SugarScape is to investigate the effects of various taxation policies on tax income and agent population. In particular, the model is used to investigate the following question: what is the optimal taxation policy for maximizing collected income while minimizing deaths?

B.2 Entities, state variables, and scales

Ants. Each ant has a fixed vision and metabolism level for the duration of the simulation; these levels vary from ant to ant. Ants can see in the four principal directions up, down, left, and right, but cannot see any other grid cells. Metabolism determines how much sugar an ant loses ('burns') each time step. Movement is governed by vision: an ant moves to the nearest grid cell within its vision with the maximum amount of sugar. Only one ant may occupy a grid cell at any given time. Ants die if their sugar level reaches zero. There is no upper limit to how much sugar an ant may accumulate. Low vision is defined as 1, 2, or 3 and low metabolism is defined as 1 or 2. Thus, each ant belongs to one of the following four **categories**: low vision / low metabolism (LL), low vision / high metabolism (LH), high

State variable	Abbreviation	Value(s)
Location (current region)	reg	$\{0, 1, \ldots, 8\}$
Vision	vis	random in $\{1, 2, \ldots, 5\}$
Metabolism	met	random in $\{1, 2, 3, 4\}$
Sugar	sug	$\{1, 2, \ldots\}$

Table B.1: Ant state variables

vision / low metabolism (HL), and high vision / high metabolism (HH).

Grid cells. When ants consume the sugar from a grid cell, the sugar grows back at a fixed rate over subsequent time steps, up to a pre-determined maximum based on the layout of the landscape.

State variable	Abbreviation	Value(s)
Maximum sugar	s_{max}	one of $\{0, 1, 2, 3, 4\}$
Sugar	s_{here}	$\{0, 1, 2, 3, 4\}$
Grow back rate	α	1

Table B.2: Grid cell state variables

Spatial and temporal scales. The landscape for SugarScape is presented in Figure 2.8. The maximum sugar amounts for each region are given in Table B.3. There are five **region** types: those whose maximum sugar is 0, 1, 2, 3, or 4. Each ant occupies exactly one

Region	0	1	2	3	4	5	6	7	8
Max. sugar	0	0	1	1	2	3	3	4	4

Table B.3: Maximum sugar and grid cell counts for each region.

grid cell, and the map is toroidal – edges wrap in both the horizontal and vertical directions. The landscape is a 50×50 grid of cells. Given the fairly abstract nature of the model, time and space are unitless. A simulation consists of a finite number of time steps. Taxes are collected at regular intervals. The tax rate for a given ant depends on its category and current region (for example, an agent with high vision and low metabolism may be taxed at rate 0.75 in a high-sugar region but only at 0.25 in a low-sugar region). Tax amounts are always rounded up to the nearest integer – this ensures that any non-zero tax rate always collects at least 1 unit of sugar. Taxes are collected once every subsequent 5 time steps for a

Variable	Abbreviation	Value(s)	Units
Simulation duration	*total_sim_time*	50	time steps
Permissible tax rates	*tax*	$\{0,\ 0.25,\ 0.5,\ 0.75\}$	N/A
Tax interval	*tax_interval*	5	time steps

Table B.4: Taxation and temporal variables

total of ten tax cycles. The choice to tax every 5 ticks is motivated by the desire to not let the dynamics stabilize – with frequent taxation the dynamics are more immediately affected by previous tax rates.

For each of the four ant categories there are five possible tax rates depending on their current region and each of these rates may be different for each of the ten tax cycles. Thus, a **tax schedule** is a vector of length $5 \cdot 4 \cdot 10 = 200$ with each entry in $\{0, 0.25, 0.5, 0.75\}$ – this means there are a total of 4^{200} different tax schedules. The optimization problem is to determine the tax schedules which maximize the total tax income collected while minimizing the number of deaths.

B.3 Process overview and scheduling

The ABM process is presented as pseudo-code in Algorithm 6. The ant and tax routines are executed fully by one ant, then fully by another – i.e., serially. Hence state variables are updated asynchronously. Time steps are discrete units, as is movement: ants jump directly from the center of one grid cell to the center of another.

Algorithm 6 SugarScape process pseudo-code.

```
setup map and generate tax schedule                         ▷ layout is read in from a matrix in a .txt file
while simulation time < $total\_sim\_time$ do
    ant routine:
        let $vision\_cells$ be set of unoccupied grid cells within $vis$      ▷ includes current cell
        let $potential\_cells$ be set of $vision\_cells$ with maximum sugar
        move to nearest member of $potential\_cells$             ▷ select at random if more than one
        $sug = sug - met + s_{here}$                              ▷ eat and metabolize sugar
        grid cell here: $s_{here} = 0$
    end routine
    grid cells: $s_{here} = \min\{s_{here} + \alpha, s_{max}\}$          ▷ sugar gradually grows back, up to capacity
    if time mod $tax\_interval = 0$ then                      ▷ collect tax every $tax\_interval$ time steps
        for all ants do
            set $tax$ based on category, location, and tax cycle
            let $paid = \min\{sug, \lceil tax \cdot sug \rceil\}$     ▷ tax is rounded up; ants cannot pay more than they have
            $sug = sug - paid$
        end for
        update amount paid by all ants                        ▷ keep track of tax income
    end if
    ants with $sugar \leq 0$ die
    store death data
    advance one time step
end while
write all data to file
```

B.4 Design concepts

Basic principles. This version of SugarScape builds on the original by incorporating taxation. In general, the basic question under investigation is how spatially-dependent local inputs affect global dynamics. Specifically, the model investigates how local tax rates affect tax income and regional population distribution, a question which holds interest in a variety of real-world settings.

Emergence. Spatial population dynamics ought to be an emergent property of the model: for example, high tax rates in high-sugar regions might substantially alter regional population dynamics. The precise mechanism driving such changes is not built in to the model in any direct sense.

Objectives. The objective of each ant is to move to a cell within its vision with the maximum amount of sugar. There is no other consideration, and ants do not have knowledge of past or future tax rates at any location.

Sensing. Ants are aware of the sugar level and occupancy of each grid cell within their vision. They are not aware of any properties of any other ants, even those within their vision.

Interaction. Ants interact with one another indirectly in the sense that only one ant may occupy a grid cell at any given time. Thus if two ants have the same high-sugar grid cell within their vision, whichever ant is randomly selected to move first will occupy that cell. This may very well alter the movement of the other ant. In this way, serial execution and asynchronous update are key features of agent interaction. If ant order was not random (i.e., the same ant was allowed to move first each time step), population dynamics might be fundamentally altered.

Stochasticity. Ant movement is partially stochastic: if an ant sees four unoccupied grid cells with 2 sugar and three unoccupied grid cells with 3 sugar, the ant will choose the nearest cell with 3 sugar. This 'minimum distance' policy tends to lead to ants clustering on regional boundaries, as they have limited incentive to move to the interior of a region. This movement feature is discussed in [36].

Collectives. In a sense, ants form collectives which affect individuals inside and outside of the collective. This arises because only one ant may occupy a grid cell at any time. In high-sugar regions, ants in the middle of the region tend to become trapped because all available spaces are occupied. At the same time, an individual on the border of such a region is frequently unable to enter due to the high population density within the region. These collectives form entirely as a result of local interactions.

Observation. Each simulation consists of a finite number of time steps. At each time step, the following information is collected: the total amount of tax collected, and the number of deaths which occur. For each simulation, the tax policy is recorded as well. At the end of each simulation these data are written to a comma separated value (.csv) file, a universal format for spreadsheet applications.

B.5 Initialization

The model is initialized with 200 ants; each is placed at a random unoccupied location on the landscape. Ants begin with a random amount of sugar between 5 and 25 (inclusive); this value is different for each ant and chosen from a uniform distribution. Ants are initialized with vision chosen at random between 1 and 5 (inclusive) and metabolism between 1 and 4 (inclusive). Vision and metabolism of a given ant do not change over the course of a simulation.

B.6 Input data

The landscape is read in from a .txt file; this helps with implementation and makes it easier to make changes. A tax policy can either be chosen directly via code manipulation or chosen at random. The policy must be chosen prior to simulation. As such, the tax policy may be thought of as input to the model.

Appendix C

Overview, Design concepts, and Details protocol for Rabbits and Grass

The model upon which this version is based is included in the sample library of NetLogo [133], a popular agent-based modeling platform. The description here is warranted as it includes the mechanics of an optimization problem, the details of which are not available elsewhere (including the NetLogo version).

C.1 Purpose

The purpose of this model is to examine population dynamics of a simple environmental system. In particular, it is a model of rabbits eating grass in a field. On each day of the simulation, poison can be placed on the field in order to kill the rabbits. The poison kills the rabbits with a certain efficacy but has no effect on the grass. The poison costs money, so there is some interest in minimizing the number of days on which poison is used. From this scenario, a natural multi-objective optimization problem arises: what poison schedule should be used in order to minimize the total number of rabbits alive over the entire simulation while also minimizing the amount of poison? This version of the model is an attempt to answer this question.

C.2 Entities, state variables, and scales

This section contains a description of the grid cells, spatial and temporal scales, and the rabbits. It also contains a description of the format of a *poison strategy*, the investigation of which is the key feature of the model.

Grid cells, spatial scale, and temporal scale. The world is a square grid of discrete cells, representing a field. The grid is toroidal, meaning that edges wrap around both in the horizontal and vertical directions. The distance from the center of a cell to a neighboring horizontal or vertical cell is 1 unit (thus the distance between two diagonal cells is $\sqrt{2}$). Units are abstract spatial measurements. Time steps are also abstract discrete units. A simulation consists of a finite number of time steps. The only state variable for each cell indicates whether or not the cell currently contains grass. When grass is eaten on a grid cell, there is a certain probability that it will grow back at each time step. This growth happens spontaneously.

State variable	Name	Value
Side length of field	s	40 grid cells
Total grid cells	N	1600
Presence of grass	$grass?$	0 = no grass, 1 = grass.
Grass growth probability	γ	0.02
Simulation time	$total\_sim\_time$	100 time steps

Table C.1: Grid cell state variables.

Rabbits. Each time step, rabbits move, eat grass (or not), and reproduce (or not). Reproduction is asexual and based on energy level, which is raised when a rabbit eats. Rabbits lose energy both by moving and by spawning new rabbits. If a rabbit's energy level drops to 0 or lower, the rabbit dies.

State variable	Name	Value
Movement cost	$move\_cost$	1
Energy from food	$food\_energy$	3
Birth cost	$birth\_cost$	5
Birth threshold	$birth\_threshold$	8
Current energy level	$energy$	varies

Table C.2: Rabbit state variables.

Poison schedule. A poison schedule u is a vector of length $total\_sim\_time$, with each entry either 0 or 1. Each entry corresponds to one time step in the simulation; 0 means that poison is not used and 1 means that poison is used. Thus, there are a total of $2^{total\_sim\_time}$ possible poison schedules. The poison has a maximum efficacy which degrades over time with repeated use. If the poison is not used, the efficacy increases again, up to the maximum.

State variable	Name	Value
Maximum efficacy	p_{max}	0.3
Degradation rate	p_{deg}	0.5
Current efficacy	p_{eff}	varies in $(0, p_{max}]$

Table C.3: Poison schedule details.

C.3 Process overview and scheduling

In order to minimize ambiguity, the model process is presented in figure 7 as pseudo-code.

Algorithm 7 Process pseudo-code.

```
setup world and initialize rabbits
set poison schedule $u$
**while** time steps $< total\_sim\_time$ **do**
    store population and grass levels
    **rabbit routine:**
        move                                                    ▷ see section C.6
        $energy = energy - move\_cost$
        **if** $u(t-1) = 1$ **then**
            $p_{eff} = (1 - p_{deg})p_{eff}$                    ▷ efficacy degrades on repeated use
        **else**
            $p_{eff} = p_{eff} + p_{deg}(p_{max} - p_{eff})$    ▷ efficacy increases if not used
        **end if**
        **if** $u(t) = 1$ and $rand(0,1) < p_{eff}$ **then** die **end if**    ▷ probability of being poisoned
        **if** $grass? = 1$ on this grid cell **then**
            $energy = energy + food\_energy$
            grid cell: $grass? = 0$
        **end if**
        **if** $energy > birth\_threshold$ **then**
            $energy = energy/2$                                 ▷ energy is halved for reproduction
            create new rabbit here                              ▷ rabbit inherits location and energy values from parent
            new rabbit moves
        **end if**
        **if** $energy \leq 0$ **then** die **end if**
    **end routine**
    **for all** grid cells with $grass? = 0$ **do**
        **if** $rand(0,1) < \gamma$ **then** $grass? = 1$ **end if**
    **end for**
    advance one time step
**end while**
write data to file
```

C.4 Design concepts

In the proposed ODD protocol [48] there are eleven design concepts. Those which do not apply have been omitted.

Basic principles. In essence, this model is a predator-prey system, wherein the rabbits are predators and the grass is prey. Introduction of poison into the model, and having that poison modeled as a direct external influence on population levels, creates a natural setting for an optimization problem. One can study the effect of various poison strategies on population levels – in terms of minimizing the rabbit population it can be thought of as a harvesting problem, but in terms of minimizing poison it can be thought of as resource allocation.

Emergence. Rabbit population and grass levels tend to oscillate as the simulation progresses. The frequency and amplitude of these oscillations can be affected by parameter settings and initial values and hence may be described as emergent model dynamics.

Interaction. Agent interaction is indirect: since rabbit movement is executed serially, it is possible that other rabbits deplete all of the grass in a particular rabbit's potential field of movement, thereby reducing or eliminating the chance for that rabbit to gain energy.

Stochasticity. Rabbit movement is totally random in that they cannot sense whether neighboring grid cells contain grass or not. Whether grass grows back on an empty grid cell is also random, and a grid cell that has been empty for several time steps is no more likely to grow grass than a cell which has only just become empty.

Observation. Rabbit population and grass counts are recorded at each time step. The total number of rabbits alive during the course of a simulation serves as a measure of fitness of the poison schedule.

C.5 Initialization

At initialization, 20% of the grid cells contain grass; these are chosen at random. There are 120 rabbits placed at random locations throughout the grid; each begins with a random amount of energy between 1 and 8 inclusive. Total simulation time is 100 time steps, and each simulation contains a poison schedule u, described in section C.2.

C.6 Submodels

There are three types of rabbit movement investigated in this study. In the first, referred to as *random jump*, rabbits jump to a grid cell selected at random from the entire space. The second type of movement is *wiggle*, wherein rabbits execute the following commands: face left by a random amount up to 45 degrees, face right by a random amount up to 45 degrees, move forward 1 unit. The latter is similar to rabbits selecting a direction from a uniformly random distribution centered at their current heading, and thus is meant to represent a more realistic movement scheme. The final movement scheme is referred to as *neighbor 8* movement: under this rule, rabbits move to the center of one of the 8 neighboring grid cells. In all movement schemes, the energy lost by moving is the same regardless of the distance moved. The body of the text makes clear which movement scheme is being considered at any given time.

C.7 Optimization

Since the multi-objective optimization problem is the key feature of the model as presented here, a few clarifying details are in order. The objectives of the optimization problem are to determine, for the parameter values provided, a set of Pareto optimal poison schedules which minimize the number of rabbits while also minimizing the amount of poison used. The number of rabbits refers to the total number of rabbits alive during the course of a simulation, not just those alive at the end of the final time step. Since a poison schedule is a binary vector of length *total_sim_time*, the amount of poison used is represented by the sum of the entries of that vector.

Appendix D

Difference equation models

Figures D.1 and D.2 contain the full equation models discussed in the body of the text, including initial conditions. See Table 3.1 for a description of the terms used in the equations and section C.2 for state variable values.

$$
\begin{aligned}
p_0(t+1) &= (1-g(t))p_1(t)\\
p_1(t+1) &= (1-g(t))p_2(t)\\
p_2(t+1) &= (1-g(t))p_3(t)\\
p_3(t+1) &= (1-g(t))p_4(t) + g(t)p_1(t)\\
p_4(t+1) &= (1-g(t))p_5(t) + g(t)p_2(t) + 2g(t)p_7(t)\\
p_5(t+1) &= (1-g(t))p_6(t) + g(t)p_3(t) + 2g(t)p_8(t)\\
p_6(t+1) &= (1-g(t))p_7(t) + g(t)p_4(t)\\
p_7(t+1) &= (1-g(t))p_8(t) + g(t)p_5(t)\\
p_8(t+1) &= g(t)p_6(t)\\
g(t+1) &= \left(0.98 - \frac{r(t)}{1600}\right) g(t) + 0.02\\
r(t+1) &= \sum_{i=1}^{8} p_i(t)\\
p_0(0) &= 0,\\
p_i(0) &= 15, \qquad 1 \le i \le 8,\\
r(0) &= 120,\\
g(0) &= 0.20
\end{aligned}
$$

Figure D.1: The discrete model (using *random jump* movement), tracking rabbits at different energy levels.

$$
\begin{aligned}
p_0(t+1) &= \boldsymbol{m_0}(1-g(t))p_1(t) \\
p_1(t+1) &= \boldsymbol{m_1}(1-g(t))p_2(t) \\
p_2(t+1) &= \boldsymbol{m_2}(1-g(t))p_3(t) \\
p_3(t+1) &= \boldsymbol{m_3}(1-g(t))p_4(t) + \boldsymbol{m_0}g(t)p_1(t) \\
p_4(t+1) &= \boldsymbol{m_4}(1-g(t))p_5(t) + \boldsymbol{m_1}g(t)p_2(t) + \boldsymbol{m_5}2g(t)p_7(t) \\
p_5(t+1) &= \boldsymbol{m_6}(1-g(t))p_6(t) + \boldsymbol{m_2}g(t)p_3(t) + \boldsymbol{m_7}2g(t)p_8(t) \\
p_6(t+1) &= \boldsymbol{m_5}(1-g(t))p_7(t) + \boldsymbol{m_3}g(t)p_4(t) \\
p_7(t+1) &= \boldsymbol{m_7}(1-g(t))p_8(t) + \boldsymbol{m_4}g(t)p_5(t) \\
p_8(t+1) &= \boldsymbol{m_6}g(t)p_6(t) \\
g(t+1) &= \boldsymbol{m_8}\left(0.98 - \frac{r(t)}{1600}\right)g(t) + 0.02 \\
r(t+1) &= \sum_{i=1}^{8} p_i(t)
\end{aligned}
$$

Figure D.2: The updated model with movement parameters $m_0, \ldots, m_8$.

Appendix E

Squeaky wheel optimization for parameter estimation

Parameter estimation for this model is performed using a version of ‘squeaky wheel’ optimization [65], wherein parameters are updated according to their effect on the sum-of-squares error (SSE) between ABM and equation data. The process is so named because the ‘squeaky wheel’ – the equation contributing most to the error – is more likely to be fixed first. Twenty values were chosen at random between 1 and 50; these represent the *frequency values* – for each frequency value n, five control schedules were generated randomly containing n 1’s and $100-n$ 0’s; this provides a stratified random sample of controls. These 100 control schedules were used as training data: the SSE was calculated over all of the training data.

As seen in D.2, each equation in the model contains up to 3 parameters; each equation contributes a different amount to the total error between datasets. The algorithm is initialized with all parameters equal to 1. Each iteration, an equation is chosen based on its contribution to the total error: if one equation contributes twice as much error as another, it is twice as likely to be selected for update. Once selected, exactly one parameter is chosen from that equation. The parameter (with value, say, v) is altered by selecting a random number from a normal distribution with mean v and standard deviation 0.075. There is then a small probability of the value mutating, wherein it is replaced with a random number selected from a normal distribution with mean 1 and standard deviation 0.075 (this mutation feature encourages exploration of the state space and reduces risk of the algorithm getting stuck at a local minimum). If the new parameter set reduces the SSE, the new parameters replace the old. If not, the new parameters still have the possibility of replacing the old with a certain probability $w(t)$. The algorithm terminates after a fixed number of steps. Note that w is a function of time – in particular, this value decreases linearly over time, allowing the algorithm to converge. Since the algorithm uses SSE as the fitness measure, the algorithm can be used to fit parameters for various movement types by simply using the appropriate ABM data values in the error calculation. Since the SSE is calculated over

100 control schedules, the algorithm is more likely to determine parameters that are optimal with respect to control.

Appendix F

Optimal poison schedules

Figure F.1 contains four schedules from the final generation of the Pareto optimization algorithm.

$$\begin{bmatrix}
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

(a) 10 poison days.

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

(b) 20 poison days.

$$\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

(c) 30 poison days.

$$\begin{bmatrix}
1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1
\end{bmatrix}$$

(d) 46 poison days.

Figure F.1: Several poison schedules (to be read from left to right).

Appendix G

Overview, Design concepts, and Details (ODD) protocol for the SugarScape Gradient Model

This appendix contains the Overview, Design Concepts, and Details (ODD) protocol for the SugarScape Gradient model. This protocol was introduced by Grimm et al [47, 48] as a template for describing an agent-based model in detail sufficient enough that the model (and results) can be replicated by an independent researcher. The protocol also includes a list of modeling features that the ABM contains. The description in this appendix is meant to stand alone; as such, overlap may occur between the description here and that contained in the body of the text.

G.1 Purpose

The SugarScape Gradient model is a simplified version of SugarScape [36], a model in which abstract entities (ants) roam a landscape made of sugar. Ants all have the same vision strength, but they have varying metabolisms, affecting the amount of sugar burnt over time. Ants that are able to make it to the sugar-rich regions on the right side of the map tend to survive, while those who do not are more likely to run out of sugar and die. Additionally, ants are periodically taxed for their sugar stores – tax rates may be 0%, 25%, 50%, 75%, or 100%, and the rates are set regionally (hence regions with more sugar may have higher taxes, for example). Each tax cycle, tax rates may change. The purpose of this model is to investigate the effects of various taxation policies on tax income and ant population. In particular, the model is used to investigate the following question: what is the optimal taxation policy for maximizing collected income while minimizing deaths?

G.2 Entities, state variables, and scales

Grid cells, spatial, and temporal scales. The landscape is a 48×48 grid of cells containing various levels of sugar. Figure G.1 presents a snapshot of the landscape; labels refer to the amount of sugar contained on each grid cell in that region. The landscape wraps vertically but **not** horizontally; thus, the only way to reach a region with more sugar is by traveling to the right. When ants consume the sugar from a grid cell, the sugar is not depleted. Hence sugar levels are preserved throughout each simulation. Given the fairly abstract nature of the model, time and space are unitless. A simulation consists of a finite number of time steps. At regular intervals, taxes are either collected or not, depending on the tax policy being simulated.

Variable	Abbreviation	Value(s)	Units
Simulation duration	*total_sim_time*	51	time steps
Tax rate	*tax*	{0, 0.25, 0.5, 0.75, 1}	N/A
Tax interval	*tax_interval*	5	time steps

Table G.1: Taxation and temporal variables

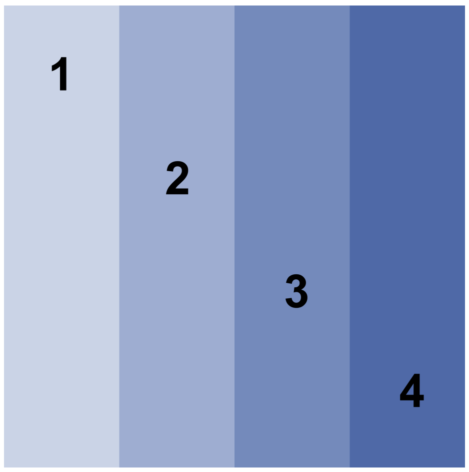


Figure G.1: The SugarScape Gradient is a 48×48 grid of cells, so named because sugar levels increase from left to right.

Taxation. Taxes are collected every 5 time steps for a total of 10 tax cycles, and tax rates may be different in each region and at each tax cycle. Hence, using the values in table G.1, a tax policy is a 4×10 matrix with entries in $\{0, 0.25, 0.5, 0.75, 1\}$. Entry (i, j) of this matrix indicates the tax rate in region i during the j^{th} tax cycle. Thus there are a total of $5^{40} \approx 9 \times 10^{27}$ possible tax policies. For each policy, the resultant number of deaths and the amount of tax collected are stored.

Ants. Each ant has a fixed vision level of 1 grid cell for the duration of the simulation: this means ants can see exactly one cell in each of the four principal directions up, down, left, and right (unless they are on the horizontal edge of the map, in which case vision does not wrap). Additionally, each ant has a fixed metabolism which determines how much sugar it loses each time step. Movement is governed by vision: an ant moves to the grid cell within its vision with the maximum amount of sugar. Multiple ants may occupy the same grid cell; each ant obtains the full amount of sugar at that location (i.e., sugar is not depleted and there is no exclusion principle in effect). Ants die if their sugar level reaches zero. Ants may accumulate up to 50 sugar.

State variable	Abbreviation	Value(s)
Location (current region)	*reg*	$\{1, \ldots, 4\}$
Vision	*vis*	1
Metabolism	*met*	random in $\{1, 2\}$
Sugar	*sug*	$\{1, 2, \ldots\ 50\}$

Table G.2: Ant state variables

G.3 Process overview and scheduling

Per the suggestion in [48], the ABM process is presented in Algorithm 8 as pseudo-code in order to give a detailed account of the order of events. The ant and tax routines are executed fully by one ant, then fully by another – i.e., serially. This means that state variables are updated asynchronously. Time steps are discrete units, as is movement: ants jump directly from the center of one grid cell to the center of another.

G.4 Design concepts

Basic principles. The SugarScape Gradient model builds on SugarScape, which can be studied in a myriad of contexts, including sociology, ecology, economics, and culture. The primary motivation for this version is to investigate the effects of various tax policies in an abstract setting. It is hoped that the model will provide insight into the relationship between

Algorithm 8 SugarScape Gradient process pseudo-code.

```
setup map and constants                                   ▷ layout is read in from a matrix in a .txt file
while simulation time $< total\_sim\_time$ do
    store population and wealth data
    ant routine:
        let $vision\_cells$ be set of grid cells within 1 grid cell of location    ▷ includes current cell
        let $potential\_cells$ be set of $vision\_cells$ with maximum sugar    ▷ maximum sugar cell may not be unique
        move to randomly selected member of $potential\_cells$
        $sug = sug - met + s_{here}$                      ▷ eat and metabolize sugar
    end routine
    if time $\bmod tax\_cycle = 0$ then                   ▷ collect tax every $tax\_interval$ time steps
        for all ants do
            $sug = sug - \lceil tax \cdot sug \rceil$     ▷ taxes are always rounded up
        end for
        update amount paid by all ants                    ▷ keep track of tax income
    end if
    ants with $sugar \leq 0$ die
    store death data
end while
write all data to file
```

tax policies and population dynamics, and that these insights might be extrapolated to some real-world system.

Emergence. Spatial population dynamics ought to be an emergent property of SugarTax: for example, frequent high taxation in high-sugar regions might substantially alter regional population counts. The precise mechanism driving such changes is not built in to the model in any direct sense.

Objectives. The objective of each ant is to move to a cell within its vision with the maximum amount of sugar. There is no other consideration, and ants do not have knowledge of past or future taxation in any location.

Sensing. Ants are aware of the sugar level and occupancy of each grid cell within their vision. They are not aware of any properties of any other ants, including those within their vision.

Stochasticity. Ant movement is partially stochastic: if there are multiple grid cells within an ants vision with the same (maximum) sugar level, the ant will move to one of these grid cells at random. This stochasticity is vital to population distributions – if the ants chose the nearest such cell, for example, populations tend to cluster on the boundary of each region. Clustering severely affects migrational patterns, as ants are not encouraged to explore the landscape. This movement feature is discussed in [36].

Observation. Each simulation occurs over a finite number of time steps. At each time step, the following information is collected: the number of ants in each region, the wealth distribution of the ants in each region (separated into disjoint bins), the total amount of tax collected, and the number of deaths occurring in each region. In addition, the tax policy implemented over the course of the simulation is recorded as well. At the end of each simulation, these data are written to a comma separated value (.csv) file, a universal format for spreadsheet applications.

G.5 Initialization

The model is initialized with an average of 62.5 ants; each is placed at a random location on the landscape. Ants begin with a random amount of sugar between 5 and 25 (inclusive); this value is different for each ant and chosen from a uniform distribution. Ants are initialized with vision 1 and metabolism either 1 or 2. Vision and metabolism of a given ant do not change over the course of a simulation.

G.6 Input data

The landscape is read in from a .txt file; this helps with implementation and makes it easier to make changes. A tax policy can either be chosen directly via code manipulation or chosen at random. The policy must be chosen prior to simulation. As such, the tax policy may be thought of as input to the model.

Appendix H

SugarScape Gradient: Difference equations

Figure H.1 contains the population equations for the 48 columns in the ABM; figure H.2 contains the wealth (i.e., sugar level) equations. Note that in order to match ABM behavior, all equations are updated **synchronously**. See figure 4.3 for the order in which equations are updated. Note too that since there are multiple updates of the equations in order to simulate one time step in the ABM, the time scale in the equation system is not quite the same. In particular, time steps in the equations are more closely aligned with *processes* in the ABM (for example, agent movement or taxation). Note that the wealth equations track

$$
\begin{aligned}
p_1(t+1) &= 0.75p_1(t) + 0.2p_2(t), && \\
p_a(t+1) &= p_{a-1}(t) + 0.75p_a(t) + 0.2p_{a+1}(t), && a \in \{13, 25, 37\}, \\
p_b(t+1) &= 0.25p_{b-1}(t) + 0.6p_b(t) + 0.2p_{b+1}(t), && b \in \{2, 14, 26, 38\}, \\
p_c(t+1) &= 0.2p_{c-1}(t) + 0.6p_c(t) + 0.2p_{c+1}(t), && c \in \{3, \ldots, 10, 15, \ldots, 22, \\
& && \quad 27, \ldots, 34, 39, \ldots, 46\}, \\
p_m(t+1) &= 0.2p_{m-1}(t) + 0.6p_m(t), && m \in \{11, 23, 35\}, \\
p_n(t+1) &= 0.2p_{n-1}(t), && n \in \{12, 24, 36\}, \\
p_{47}(t+1) &= 0.2p_{46}(t) + 0.6p_{47}(t) + 0.25p_{48}(t), && \\
p_{48}(t+1) &= 0.2p_{47}(t) + 0.75p_{48}(t). &&
\end{aligned}
$$

Figure H.1: Population equations.

sugar levels based on sugar eaten and sugar metabolized, but do not account for migration or taxation. Here, the number of ants with 0 sugar is always set to 0; this is done to simulate death. Note that according to figure 4.3, wealth values are updated after migration and

$$
\begin{aligned}
w_{1,1,0}(t+1) &= 0, \\
w_{1,1,a}(t+1) &= w_{1,1,a}(t) && a \in \{1,\dots,50\}, \\
w_{1,2,b}(t+1) &= w_{1,2,b+1}(t) && b \in \{0,\dots,49\}, \\
w_{1,2,50}(t+1) &= 0, \\
w_{2,1,0}(t+1) &= 0, \\
w_{2,1,a}(t+1) &= w_{2,1,a-1}(t) && a \in \{1,\dots 49\}, \\
w_{2,1,50}(t+1) &= w_{2,1,49}(t) + w_{2,1,50}(t), \\
w_{2,2,0}(t+1) &= 0, \\
w_{2,2,b}(t+1) &= w_{2,2,b}(t) && b \in \{1,\dots,50\}, \\
w_{3,1,a}(t+1) &= 0, && a \in \{0,1\}, \\
w_{3,1,b}(t+1) &= w_{3,1,b-2}(t) && b \in \{2,\dots,49\}, \\
w_{3,1,50}(t+1) &= w_{3,1,48}(t) + w_{3,1,49}(t) + w_{3,1,50}(t), \\
w_{3,2,0}(t+1) &= 0, \\
w_{3,2,c}(t+1) &= w_{3,2,c-1}(t) && c \in \{1,\dots,49\}, \\
w_{3,2,50}(t+1) &= w_{3,2,49}(t) + w_{3,2,50}(t), \\
w_{4,1,a}(t+1) &= 0 && a \in \{0,1,2\}, \\
w_{4,1,b}(t+1) &= w_{4,1,b-3}(t) && b \in \{3,\dots,49\}, \\
w_{4,1,50}(t+1) &= w_{4,1,47}(t) + w_{4,1,48}(t) + w_{4,1,49}(t) + w_{4,1,50}(t), \\
w_{4,2,m}(t+1) &= 0 && m \in \{0,1\}, \\
w_{4,2,n}(t+1) &= w_{4,2,n-2}(t) && n \in \{2,\dots,49\}, \\
w_{4,2,50}(t+1) &= w_{4,2,48}(t) + w_{4,2,49}(t) + w_{4,2,50}(t).
\end{aligned}
$$

Figure H.2: Wealth equations.

population values are updated after taxation. This ensures that at the end of each time step, the sum of the population equations is always the same as the sum of the wealth equations. Tax rates are percentages, but a ceiling function is applied in order to ensure that an integer amount of tax is paid.

Initial values. Initial values for each of the equations are provided in figure H.3. These values are determined by settings from the ABM. Note that in this study we focus only on agents with low vision (i.e., vision 1) and low metabolism (metabolism 1 or 2). In the original ABM, there were three other categories of ants, total of 250 inital agents. Hence, since we are focusing on one category, the initial population values must sum to $250/4 = 62.5$. These are placed at random throughout the grid. Likewise, ants begin the simulation with an amount of sugar chosen uniformly randomly between 5 and 25 (inclusive, for a total of 21 possible values); hence the number of ants in a region determines the expected number at each sugar level. There are 48 population equations tracking the number of ants in each column. The 62.5 initial agents are assumed to be distributed among the regional sugar levels uniformly at the start of each simulation.

$$\begin{aligned}
p_n(0) &= 62.5/48 & n &\in \{1,\ldots,48\},\\
w_{r,m,a}(0) &= 0 & a &\in \{0,\ldots,4\},\\
w_{r,m,b}(0) &= 62.5/(4\cdot 21) & b &\in \{5,\ldots,25\},\\
w_{r,m,c}(0) &= 0 & c &\in \{26,\ldots,50\}.
\end{aligned}$$

Figure H.3: Initial values. Wealth equations hold for $r \in \{1,\ldots,4\}$ and $m \in \{1,\ 2\}$

Appendix I

Overview, Design concepts, and Details (ODD) protocol for SugarTax

I.1 Purpose

SugarTax is a modified version of SugarScape [36], a model in which abstract entities roam a landscape made of sugar. These agents are periodically taxed for their sugar stores – the tax rate is constant, but the frequency differs from region to region. The purpose of SugarTax is to investigate the effects of various taxation policies on tax income and agent population. In particular, the model is used to investigate the following question: what is the optimal taxation policy for maximizing collected income while minimizing deaths?

I.2 Entities, state variables, and scales

Ants. Each ant has a fixed vision level for the duration of the simulation; these levels vary from ant to ant. Ants can see in the four principal directions up, down, left, and right, but cannot see any other grid cells. Additionally, each ant has a fixed metabolism which determines how much sugar it loses each time step. Movement is governed by vision: an ant moves to the grid cell within its vision with the maximum amount of sugar. Only one ant may occupy a grid cell at any given time. Ants die if their sugar level reaches zero. There is no upper limit to how much sugar an ant may accumulate.

Grid cells. When ants consume the sugar from a grid cell, the sugar grows back at a fixed rate over subsequent time steps, up to a pre-determined maximum based on the layout of the landscape.

State variable	Abbreviation	Value(s)
Location (current region)	*reg*	$\{0, 1, \ldots, 8\}$
Vision	*vis*	random in $\{1, 2, \ldots, 5\}$
Metabolism	*met*	2
Sugar	*sug*	$\{1, 2, \ldots\}$

Table I.1: Ant state variables

State variable	Abbreviation	Value(s)
Maximum sugar	s_{max}	one of $\{0, 1, 2, 3, 4\}$
Sugar	s_{here}	$\{0, 1, 2, 3, 4\}$
Grow back rate	α	1

Table I.2: Grid cell state variables

Spatial and temporal scales. The landscape for SugarTax is presented in Figure 5.2; the regions are labeled $0 - 8$. Note that the label of a region serves only as its *name*, and does not refer to the amount of sugar in that region. The maximum sugar amounts for each region (as well as the size of the region by grid cell count) are given in Table I.3. Each ant

Region	0	1	2	3	4	5	6	7	8
Max. sugar	0	0	1	1	2	3	3	4	4

Table I.3: Maximum sugar and grid cell counts for each region.

occupies exactly one grid cell, and the map is bounded on all four sides. The landscape is a 50×50 grid of cells. Given the fairly abstract nature of the model, time and space are unitless. A simulation consists of a finite number of time steps. At regular intervals, taxes are either collected or not, depending on the tax policy being simulated. Using the values in table I.4, a tax policy is a 9×10 matrix with binary entries. Entry (i, j) of this matrix indicates whether or not tax was collected in region i during the j^{th} tax cycle. An entry of 1 implies tax and an entry of 0 implies no tax. Thus, according to these values there are a total of 2^{90} possible tax policies.

I.3 Process overview and scheduling

Per the suggestion in [48], the ABM process is presented in Algorithm 9 as pseudo-code in order to give a detailed account of the order of events. The ant and tax routines are executed fully by one ant, then fully by another – i.e., serially. This means that state variables are

Variable	Abbreviation	Value(s)	Units
Simulation duration	*total_sim_time*	51	time steps
Tax amount	*tax*	10	sugar
Tax interval	*tax_interval*	5	time steps

Table I.4: Taxation and temporal variables

updated asynchronously. Time steps are discrete units, as is movement: ants jump directly from the center of one grid cell to the center of another.

Algorithm 9 SugarTax process pseudo-code.

```
setup map and constants                                   ▷ layout is read in from a matrix in a .txt file
while simulation time < $total\_sim\_time$ do
    store population and wealth data
    grid cells: $s_{here} = \min\{s_{here} + \alpha, s_{max}\}$   ▷ sugar gradually grows back, up to capacity
    ant routine:
        let $vision\_cells$ be set of unoccupied grid cells within $vis$   ▷ includes current cell
        let $potential\_cells$ be set of $vision\_cells$ with maximum sugar   ▷ maximum sugar cell may not be unique
        move to randomly selected member of $potential\_cells$
        $sug = sug - met + s_{here}$                          ▷ eat and metabolize sugar
        grid cell here: $s_{here} = 0$
        update ants' current region
    end routine
    store migration data
    if time  mod $tax\_cycle = 0$ then                    ▷ collect tax every $tax\_interval$ time steps
        for all ants do
            if $reg$ is being taxed on this cycle then
                let $paid = \min\{sug, tax\}$               ▷ ants pay as much as they have, up to tax amount
                $sug = sug - paid$
            end if
        end for
        update amount paid by all ants                    ▷ keep track of tax income
    end if
    ants with $sugar \leq 0$ die
    store death data
end while
write all data to file
```

I.4 Design concepts

Basic principles. SugarTax builds on SugarScape, which can be studied in a myriad of contexts, including sociology, ecology, economics, and culture. The primary motivation for SugarTax is to investigate the effects of various tax policies in an abstract setting. It is hoped that the model will provide insight into the relationship between tax policies and population dynamics, and that these insights might be extrapolated to some real-world system.

Emergence. Spatial population dynamics ought to be an emergent property of SugarTax: for example, frequent taxation in high-sugar regions might substantially alter regional population counts. The precise mechanism driving such changes is not built in to the model in any direct sense.

Objectives. The objective of each ant is to move to a cell within its vision with the maximum amount of sugar. There is no other consideration, and ants do not have knowledge of past or future taxation in any location.

Sensing. Ants are aware of the sugar level and occupancy of each grid cell within their vision. They are not aware of any properties of any other ants, including those within their vision.

Interaction. Ants interact with one another indirectly, in the sense that only one ant may occupy a grid cell at any given time. Thus if two ants have the same high-sugar grid cell within their vision, whichever ant is randomly selected to move first will occupy that cell. This may very well alter the movement of the other ant. In this way, serial execution and asynchronous update are key features of agent interaction. If ant order was not random (i.e., the same ant was allowed to move first each time step), population dynamics might be fundamentally altered.

Stochasticity. Ant movement is partially stochastic: if ant sees four unoccupied grid cells with 2 sugar and three unoccupied grid cells with 3 sugar, the ant will choose one of the 3-sugar cells at random. This stochasticity is vital to population distributions – if the ants chose the nearest such cell, for example, populations tend to cluster on the boundary of each region. This clustering severely affects migrational patterns, as ants are not encouraged to explore the landscape. This movement feature is discussed in [36].

Collectives. In a sense, ants form collectives which affect individuals inside and outside of the collective. This arises because only one ant may occupy a grid cell at any time. In high-sugar regions, ants in the middle of the region tend to become trapped because all available spaces are occupied. At the same time, an individual on the border of such a region is frequently unable to enter due to the high population density within the region. These collectives form entirely as a result of local interactions.

Observation. Each simulation occurs over a finite number of time steps. At each time step, the following information is collected: the number of ants in each region, the wealth distribution of the ants in each region (separated into disjoint bins), the total amount of tax collected, and the number of deaths occurring in each region. In addition, the tax policy implemented over the course of the simulation is recorded as well. At the end of each simulation, these data are written to a comma separated value (.csv) file, a universal format for spreadsheet applications.

I.5 Initialization

The model is initialized with 240 ants; each is placed at a random location on the landscape. Ants begin with a random amount of sugar between 1 and 30 (inclusive); this value is different for each ant and chosen from a uniform distribution. Ants are initialized with vision chosen at random between 1 and 5 (inclusive) and metabolism 2. Vision and metabolism of a given ant do not change over the course of a simulation.

I.6 Input data

The landscape is read in from a .txt file; this helps with implementation and makes it easier to make changes. A tax policy can either be chosen directly via code manipulation or chosen at random. The policy must be chosen prior to simulation. As such, the tax policy may be thought of as input to the model.

Appendix J

SugarTax difference equation model

The full system of difference equations is presented here, post-parameter optimization – these are the equations that were used to perform the Pareto optimization outlined in section 5.4. Table J.1 provides descriptions of the terms used in the equations. A tax policy U is a 9×10 binary matrix, with entry (i, j) indicating whether tax is applied in region i during tax cycle j. This matrix U is determined prior to equation evaluation.

The equations are iteratively evaluated 51 times, in order to replicate the time step duration of the ABM. Recall that the tax cycle is 5 time steps, which means that $tax_r(t) = 0$ for $t = 0$ and when $t \mod 5 \neq 0$. For $t > 0$ and $t \mod 5 = 0$, $tax_r(t)$ is determined by the tax policy U. Wealth bin 0 tracks the number of agents with 0 sugar; in other words, deaths. Thus the total number of deaths is $\sum_{r,t} w_{r,0}(t)$. The tax rate is exactly 10 sugar, so total tax income is $10 \cdot \sum_{i,j} U_{i,j} \cdot p_i(5(j + 1))$. Note that tax income is an approximation since some ants have less than 10 sugar when taxed.

Term	Meaning
$w_{r,b}(t)$	Number of ants in region r and wealth bin b at time t
$p_r(t)$	Population in region r at time t
$m_{x,y}(t)$	Proportion of ants migrating from region x to region y at time t
$tax_r(t)$	Binary entry indicating taxation (0 corresponds to no taxation)

Table J.1: Terms used in the equation model.

J.1 Population equations

$$
\begin{aligned}
p_0(t+1) &= (1-m_{0,2}(t))p_0(t) - w_{0,0}(t) \\
p_1(t+1) &= (1-m_{1,3}(t))p_1(t) - w_{1,0}(t) \\
p_2(t+1) &= (1-m_{2,4}(t))p_2(t) - w_{2,0}(t) + m_{0,2}(t)p_0(t) \\
p_3(t+1) &= (1-m_{3,4}(t))p_3(t) - w_{3,0}(t) + m_{1,3}(t)p_1(t) \\
p_4(t+1) &= (1-m_{4,5}(t)-m_{4,6}(t))p_4(t) - w_{4,0}(t) + m_{2,4}(t)p_2(t) + m_{3,4}(t)p_3(t) \\
&\qquad + m_{5,4}(t)p_5(t) + m_{6,4}(t)p_6(t) \\
p_5(t+1) &= (1-m_{5,4}(t)-m_{5,7}(t))p_5(t) - w_{5,0}(t) + m_{4,5}(t)p_4(t) + m_{7,5}(t)p_7(t) \\
p_6(t+1) &= (1-m_{6,4}(t)-m_{6,8}(t))p_6(t) - w_{6,0}(t) + m_{4,6}(t)p_4(t) + m_{8,6}(t)p_8(t) \\
p_7(t+1) &= (1-m_{7,5}(t))p_7(t) - w_{7,0}(t) + m_{5,7}(t)p_5(t) \\
p_8(t+1) &= (1-m_{8,6}(t))p_8(t) - w_{8,0}(t) + m_{6,8}(t)p_6(t)
\end{aligned}
$$

J.2 Migration equations

Note that migration occurs based on population values *at the same time step*, so the time index is t rather than $t+1$.

$$
\begin{aligned}
m_{0,2}(t) &= 1.993\times10^{-10}p_0(t)^5p_2(t)^2 - 7.969\times10^{-7}p_0(t)^4 \\
m_{1,3}(t) &= 3.339\times10^{-13}p_1(t)^8p_3(t) \\
m_{2,4}(t) &= 6.392\times10^{-18}p_2(t)^4p_4(t)^6 \\
m_{3,4}(t) &= 1.708\times10^{-16}p_3(t)^7p_4(t)^3 \\
m_{4,5}(t) &= \frac{0.03382p_4(t)^2 + 0.00583p_5(t)^2}{15.78 + p_4(t)p_5(t)} \\
m_{4,6}(t) &= \frac{p_6(t) + 0.1209p_4(t)^2 - 5.015}{85.71 + 214p_4(t)} \\
m_{5,7}(t) &= 3.152 + 0.01343p_7(t)^2 + 0.001075p_5(t)p_7(t) \\
&\quad - 0.0329p_5(t) - 0.3258p_7(t) - 0.0001975p_7(t)^3 \\
m_{5,4}(t) &= 7.541\times10^{-6}p_4(t) + 1.996\times10^{-6}p_5(t)^2 \\
m_{6,8}(t) &= 3.749 + 0.01551p_8(t)^2 + 0.001394p_6(t)p_8(t) \\
&\quad - 0.04202p_6(t) - 0.3874p_8(t) - 0.0002211p_8(t)^3 \\
m_{6,4}(t) &= 2.169\times10^{-6}p_6(t)^2 + 3.771\times10^{-8}p_4(t)^3 - 5.387\times10^{-10}p_4(t)^4 \\
m_{7,5}(t) &= 0.1165 + 0.0008416p_5(t) + 0.0006451p_7(t)^2 - 0.01891p_7(t) - 1.471\times10^{-5}p_5(t)^2 \\
m_{8,6}(t) &= 0.1109 + 0.0003374p_6(t) + 0.0006092p_8(t)^2 - 0.01771p_8(t)
\end{aligned}
$$

J.3 Wealth equations

All ants have metabolism 2 and the maximum sugar region has 4 sugar, so the most they can gain each time step is 2 sugar. Ants are initialized with at most 30 sugar, so accumulating 2 sugar over 51 times steps leads to a maximum of $30 + 2 \cdot 51 = 132$ sugar per ant over the course of a simulation. Ants with $131 - 132$ sugar in region r are tracked by equation $w_{r,14}(t)$; hence, this is the last wealth bin in each region.

$$\begin{aligned}
w_{0,0}(t+1) &= 0.2((1-m_{0,2}(t))w_{0,1}(t)) + tax_0(t)(0.8(1-m_{0,2}(t))w_{0,1}(t)\\
&\quad + 0.2(1-m_{0,2}(t))w_{0,2}(t))\\
&\text{For } n \in \{1, \ldots, 12\}:\\
w_{0,n}(t+1) &= (1-tax_0(t))(0.8(1-m_{0,2}(t))w_{0,n}(t) + 0.2(1-m_{0,2}(t))w_{0,n+1}(t))\\
&\quad + tax_0(t)(0.8(1-m_{0,2}(t))w_{0,n+1}(t) + 0.2(1-m_{0,2}(t))w_{0,n+2}(t))\\
w_{0,13}(t+1) &= (1-tax_0(t))(0.8(1-m_{0,2}(t))w_{0,13}(t) + 0.2(1-m_{0,2}(t))w_{0,14}(t))\\
&\quad + tax_0(t)(0.8(1-m_{0,2}(t))w_{0,14}(t))\\
w_{0,14}(t+1) &= (1-tax_0(t))(0.8(1-m_{0,2}(t))w_{0,14}(t))\\
w_{1,0}(t+1) &= 0.2((1-m_{1,3}(t))w_{1,1}(t)) + tax_1(t)(0.8(1-m_{1,3}(t))w_{1,1}(t)\\
&\quad + 0.2(1-m_{1,3}(t))w_{1,2}(t))\\
&\text{For } n \in \{1, \ldots, 12\}:\\
w_{1,n}(t+1) &= (1-tax_1(t))(0.8(1-m_{1,3}(t))w_{1,n}(t) + 0.2(1-m_{1,3}(t))w_{1,n+1}(t))\\
&\quad + tax_1(t)(0.8(1-m_{1,3}(t))w_{1,n+1}(t) + 0.2(1-m_{1,3}(t))w_{1,n+2}(t))\\
w_{1,13}(t+1) &= (1-tax_1(t))(0.8(1-m_{1,3}(t))w_{1,13}(t) + 0.2(1-m_{1,3}(t))w_{1,14}(t))\\
&\quad + tax_1(t)(0.8(1-m_{1,3}(t))w_{1,14}(t))\\
w_{1,14}(t+1) &= (1-tax_1(t))(0.8(1-m_{1,3}(t))w_{1,14}(t))\\
w_{2,0}(t+1) &= 0.1((1-m_{2,4}(t))w_{2,1}(t) + m_{0,2}(t)w_{0,1}(t)) + tax_2(t)(0.9((1-m_{2,4}(t))w_{2,1}(t)\\
&\quad + m_{0,2}(t)w_{0,1}(t)) + 0.1((1-m_{2,4}(t))w_{2,2}(t) + m_{0,2}(t)w_{0,2}(t)))\\
&\text{For } n \in \{1, \ldots, 12\}:\\
w_{2,n}(t+1) &= (1-tax_2(t))(0.9((1-m_{2,4}(t))w_{2,n}(t) + m_{0,2}(t)w_{0,n}(t))\\
&\quad + 0.1((1-m_{2,4}(t))w_{2,n+1}(t) + m_{0,2}(t)w_{0,n+1}(t)))\\
&\quad + tax_2(t)(0.9((1-m_{2,4}(t))w_{2,n+1}(t) + m_{0,2}(t)w_{0,n+1}(t))\\
&\quad + 0.1((1-m_{2,4}(t))w_{2,n+2}(t) + m_{0,2}(t)w_{0,n+2}(t)))\\
w_{2,13}(t+1) &= (1-tax_2(t))(0.9((1-m_{2,4}(t))w_{2,13}(t) + m_{0,2}(t)w_{0,13}(t))\\
&\quad + 0.1((1-m_{2,4}(t))w_{2,14}(t) + m_{0,2}(t)w_{0,14}(t)))\\
&\quad + tax_2(t)(0.9((1-m_{2,4}(t))w_{2,14}(t) + m_{0,2}(t)w_{0,14}(t)))\\
w_{2,14}(t+1) &= (1-tax_2(t))(0.9((1-m_{2,4}(t))w_{2,14}(t) + m_{0,2}(t)w_{0,14}(t)))
\end{aligned}$$

$$\begin{aligned}
w_{3,0}(t+1) &= 0.1((1-m_{3,4}(t))w_{3,1}(t)+m_{1,3}(t)w_{1,1}(t))\\
&\quad + tax_3(t)(0.9((1-m_{3,4}(t))w_{3,1}(t)+m_{1,3}(t)w_{1,1}(t))\\
&\quad + 0.1((1-m_{3,4}(t))w_{3,2}(t)+m_{1,3}(t)w_{1,2}(t)))
\end{aligned}$$

For $n \in \{1, \ldots, 12\}$:

$$\begin{aligned}
w_{3,n}(t+1) &= (1-tax_3(t))(0.9((1-m_{3,4}(t))w_{3,n}(t)+m_{1,3}(t)w_{1,n}(t))\\
&\quad + 0.1((1-m_{3,4}(t))w_{3,n+1}(t)+m_{1,3}(t)w_{1,n+1}(t)))\\
&\quad + tax_3(t)(0.9((1-m_{3,4}(t))w_{3,n+1}(t)+m_{1,3}(t)w_{1,n+1}(t))\\
&\quad + 0.1((1-m_{3,4}(t))w_{3,n+2}(t)+m_{1,3}(t)w_{1,n+2}(t)))\\
w_{3,13}(t+1) &= (1-tax_3(t))(0.9((1-m_{3,4}(t))w_{3,13}(t)+m_{1,3}(t)w_{1,13}(t))\\
&\quad + 0.1((1-m_{3,4}(t))w_{3,14}(t)+m_{1,3}(t)w_{1,14}(t)))\\
&\quad + tax_3(t)(0.9((1-m_{3,4}(t))w_{3,14}(t)+m_{1,3}(t)w_{1,14}(t)))\\
w_{3,14}(t+1) &= (1-tax_3(t))(0.9((1-m_{3,4}(t))w_{3,14}(t)+m_{1,3}(t)w_{1,14}(t)))\\
w_{4,0}(t+1) &= tax_4(t)((1-m_{4,5}(t)-m_{4,6}(t))w_{4,1}(t)+m_{2,4}(t)w_{2,1}(t)+m_{3,4}(t)w_{3,1}(t)\\
&\quad + m_{5,4}(t)w_{5,1}(t)+m_{6,4}(t)w_{6,1}(t))
\end{aligned}$$

For $n \in \{1, \ldots, 13\}$:

$$\begin{aligned}
w_{4,n}(t+1) &= (1-tax_4(t))((1-m_{4,5}(t)-m_{4,6}(t))w_{4,n}(t)+m_{2,4}(t)w_{2,n}(t)\\
&\quad + m_{3,4}(t)w_{3,n}(t)+m_{5,4}(t)w_{5,n}(t)+m_{6,4}(t)w_{6,n}(t))\\
&\quad + tax_4(t)((1-m_{4,5}(t)-m_{4,6}(t))w_{4,n+1}(t)+m_{2,4}(t)w_{2,n+1}(t)\\
&\quad + m_{3,4}(t)w_{3,n+1}(t)+m_{5,4}(t)w_{5,n+1}(t)+m_{6,4}(t)w_{6,n+1}(t))\\
w_{4,14}(t+1) &= (1-tax_4(t))((1-m_{4,5}(t)-m_{4,6}(t))w_{4,14}(t)+m_{2,4}(t)w_{2,14}(t)\\
&\quad + m_{3,4}(t)w_{3,14}(t)+m_{5,4}(t)w_{5,14}(t)+m_{6,4}(t)w_{6,14}(t))\\
w_{5,0}(t+1) &= tax_5(t)(0.9((1-m_{5,4}(t)-m_{5,7}(t))w_{5,1}(t)+m_{4,5}(t)w_{4,1}(t)+m_{7,5}(t)w_{7,1}(t)))\\
w_{5,1}(t+1) &= (1-tax_5(t))(0.9(((1-m_{5,4}(t)-m_{5,7}(t))w_{5,1}(t)+m_{4,5}(t)w_{4,1}(t)\\
&\quad + m_{7,5}(t)w_{7,1}(t))))+tax_5(t)(0.9((1-m_{5,4}(t)-m_{5,7}(t))w_{5,2}(t)\\
&\quad + m_{4,5}(t)w_{4,2}(t)+m_{7,5}(t)w_{7,2}(t))+0.1((1-m_{5,4}(t)-m_{5,7}(t))w_{5,1}(t)\\
&\quad + m_{4,5}(t)w_{4,1}(t)+m_{7,5}(t)w_{7,1}(t)))
\end{aligned}$$

For $n \in \{2, \ldots, 13\}$:

$$\begin{aligned}
w_{5,n}(t+1) &= (1-tax_5(t))(0.9(((1-m_{5,4}(t)-m_{5,7}(t))w_{5,n}(t)+m_{4,5}(t)w_{4,n}(t)\\
&\quad + m_{7,5}(t)w_{7,n}(t)))+0.1((1-m_{5,4}(t)-m_{5,7}(t))w_{5,n-1}(t)\\
&\quad + m_{4,5}(t)w_{4,n-1}(t)+m_{7,5}(t)w_{7,n-1}(t)))\\
&\quad + tax_5(t)(0.9((1-m_{5,4}(t)-m_{5,7}(t))w_{5,n+1}(t)+m_{4,5}(t)w_{4,n+1}(t)\\
&\quad + m_{7,5}(t)w_{7,n+1}(t))+0.1((1-m_{5,4}(t)-m_{5,7}(t))w_{5,n}(t)\\
&\quad + m_{4,5}(t)w_{4,n}(t)+m_{7,5}(t)w_{7,n}(t)))
\end{aligned}$$

$$
\begin{aligned}
w_{5,14}(t+1) &= (1-tax_5(t))(0.9(((1-m_{5,4}(t)-m_{5,7}(t))w_{5,14}(t)+m_{4,5}(t)w_{4,14}(t)\\
&\quad+m_{7,5}(t)w_{7,14}(t)))+0.1((1-m_{5,4}(t)-m_{5,7}(t))w_{5,13}(t)\\
&\quad+m_{4,5}(t)w_{4,13}(t)+m_{7,5}(t)w_{7,13}(t)))\\
w_{6,0}(t+1) &= tax_6(t)(0.9((1-m_{6,4}(t)-m_{6,8}(t))w_{6,1}(t)+m_{4,6}(t)w_{4,1}(t)\\
&\quad+m_{8,6}(t)w_{8,1}(t)))\\
w_{6,1}(t+1) &= (1-tax_6(t))(0.9(((1-m_{6,4}(t)-m_{6,8}(t))w_{6,1}(t)+m_{4,6}(t)w_{4,1}(t)\\
&\quad+m_{8,6}(t)w_{8,1}(t))))+tax_6(t)(0.9((1-m_{6,4}(t)-m_{6,8}(t))w_{6,2}(t)\\
&\quad+m_{4,6}(t)w_{4,2}(t)+m_{8,6}(t)w_{8,2}(t))+0.1((1-m_{6,4}(t)-m_{6,8}(t))w_{6,1}(t)\\
&\quad+m_{4,6}(t)w_{4,1}(t)+m_{8,6}(t)w_{8,1}(t)))\\
&\text{For } n\in\{2,\,\ldots,\,13\}:\\
w_{6,n}(t+1) &= (1-tax_6(t))(0.9(((1-m_{6,4}(t)-m_{6,8}(t))w_{6,n}(t)+m_{4,6}(t)w_{4,n}(t)\\
&\quad+m_{8,6}(t)w_{8,n}(t)))+0.1((1-m_{6,4}(t)-m_{6,8}(t))w_{6,n-1}(t)+m_{4,6}(t)w_{4,n-1}(t)\\
&\quad+m_{8,6}(t)w_{8,n-1}(t)))+tax_6(t)(0.9((1-m_{6,4}(t)-m_{6,8}(t))w_{6,n+1}(t)\\
&\quad+m_{4,6}(t)w_{4,n+1}(t)+m_{8,6}(t)w_{8,n+1}(t))+0.1((1-m_{6,4}(t)-m_{6,8}(t))w_{6,n}(t)\\
&\quad+m_{4,6}(t)w_{4,n}(t)+m_{8,6}(t)w_{8,n}(t)))\\
w_{6,14}(t+1) &= (1-tax_6(t))(0.9(((1-m_{6,4}(t)-m_{6,8}(t))w_{6,14}(t)+m_{4,6}(t)w_{4,14}(t)\\
&\quad+m_{8,6}(t)w_{8,14}(t)))+0.1((1-m_{6,4}(t)-m_{6,8}(t))w_{6,13}(t)+m_{4,6}(t)w_{4,13}(t)\\
&\quad+m_{8,6}(t)w_{8,13}(t)))\\
w_{7,0}(t+1) &= tax_7(t)(.8((1-m_{7,5}(t))w_{7,1}(t)+m_{5,7}(t)w_{5,1}(t)))\\
w_{7,1}(t+1) &= (1-tax_7(t))(.8((1-m_{7,5}(t))w_{7,1}(t)+m_{5,7}(t)w_{5,1}(t)))\\
&\quad+tax_7(t)(.8((1-m_{7,5}(t))w_{7,2}(t)+m_{5,7}(t)w_{5,2}(t))\\
&\quad+0.2((1-m_{7,5}(t))w_{7,1}(t)+m_{5,7}(t)w_{5,1}(t)))\\
&\text{For } n\in\{2,\,\ldots,\,13\}:\\
w_{7,n}(t+1) &= (1-tax_7(t))(.8((1-m_{7,5}(t))w_{7,n}(t)+m_{5,7}(t)w_{5,n}(t))\\
&\quad+0.2((1-m_{7,5}(t))w_{7,n-1}(t)+m_{5,7}(t)w_{5,n-1}(t)))\\
&\quad+tax_7(t)(.8((1-m_{7,5}(t))w_{7,n+1}(t)+m_{5,7}(t)w_{5,n+1}(t))\\
&\quad+0.2((1-m_{7,5}(t))w_{7,n}(t)+m_{5,7}(t)w_{5,n}(t)))\\
w_{7,14}(t+1) &= (1-tax_7(t))(.8((1-m_{7,5}(t))w_{7,14}(t)+m_{5,7}(t)w_{5,14}(t))\\
&\quad+0.2((1-m_{7,5}(t))w_{7,13}(t)+m_{5,7}(t)w_{5,13}(t)))\\
w_{8,0}(t+1) &= tax_8(t)(.8((1-m_{8,6}(t))w_{8,1}(t)+m_{6,8}(t)w_{6,1}(t)))\\
w_{8,1}(t+1) &= (1-tax_8(t))(.8((1-m_{8,6}(t))w_{8,1}(t)+m_{6,8}(t)w_{6,1}(t)))\\
&\quad+tax_8(t)(.8((1-m_{8,6}(t))w_{8,2}(t)+m_{6,8}(t)w_{6,2}(t))\\
&\quad+0.2((1-m_{8,6}(t))w_{8,1}(t)+m_{6,8}(t)w_{6,1}(t)))
\end{aligned}
$$

$$\begin{aligned}
&\text{For } n \in \{2, \ldots, 13\} : \\
w_{8,n}(t+1) &= (1 - tax_8(t))(.8((1 - m_{8,6}(t))w_{8,n}(t) + m_{6,8}(t)w_{6,n}(t)) \\
&\quad + 0.2((1 - m_{8,6}(t))w_{8,n-1}(t) + m_{6,8}(t)w_{6,n-1}(t))) \\
&\quad + tax_8(t)(.8((1 - m_{8,6}(t))w_{8,n+1}(t) + m_{6,8}(t)w_{6,n+1}(t)) \\
&\quad + 0.2((1 - m_{8,6}(t))w_{8,n}(t) + m_{6,8}(t)w_{6,n}(t))) \\
w_{8,14}(t+1) &= (1 - tax_8(t))(.8((1 - m_{8,6}(t))w_{8,14}(t) + m_{6,8}(t)w_{6,14}(t)) \\
&\quad + 0.2((1 - m_{8,6}(t))w_{8,13}(t) + m_{6,8}(t)w_{6,13}(t)))
\end{aligned}$$

J.4 Initial values

The ABM is initialized with 240 ants placed at random throughout the landscape. Hence initial population values for each region are based off of the size of each region. Table J.2 summarizes the initial values for the population equations. Ants are initialized with a uniformly random amount of sugar between 1 and 30 (inclusive), which means that bins 1, 2, and 3 for each region contain one-third of the initial population in that region each and the rest of the bins have initial value 0. Thus $w_{r,0}(0) = 0$, $w_{r,1}(0) = w_{r,2}(0) = w_{r,3}(0) = p_r(0)/3$, and $w_{r,n}(0) = 0$ for $n > 3$. There is no time lag on migration equations, so initial migration values are calculated using the initial population values in the equations given in section J.2.

Region	0	1	2	3	4	5	6	7	8
No. cells	214	217	246	251	790	279	279	112	112
Proportion of map	0.086	0.087	0.098	0.100	0.316	0.112	0.112	0.045	0.045
$p_r(0)$	20.54	20.83	23.62	24.10	75.84	26.78	26.78	10.75	10.75

Table J.2: Initial population values are based on 240 ants distributed randomly among the regions.

Appendix K

Overview, Design Concepts, and Details (ODD) protocol for an agent-based model of *A. fumigatus* in the lung

The Overview, Design Concepts, and Details (ODD) protocol for describing agent-based models was first introduced by Grimm in 2006 [47] and subsequently updated in 2010 [48]. The purpose of the protocol is to serve as a standard template for ABM description, including detail sufficient enough that models can be replicated independently. This document provides the ODD protocol for the model introduced in the body of the paper. The model is described in full detail, including citations indicating how certain state variable values were chosen.

K.1 Purpose

The purpose of this model is to simulate the human immune response to *Aspergillus fumigatus* in the lung. Specifically, the purpose is to investigate the following questions:

- What is the role of iron in both immune response and fungal growth?
- How does invasive Aspergillosis develop in immunocompromised patients?

K.2 Entities, state variables, and scales

In this section, global state variables, model parameters, and spatial and temporal scales are described first, followed by behavioral descriptions and state variables for grid cells and entities. Each description concludes with a table of state variables. The notation $N(\mu, \sigma)$ indicates a value chosen at random from the normal distribution with mean μ and standard deviation σ. The notation $rand(a, b)$ indicates a uniformly distributed random number chosen from interval (a, b). A length of one grid cell indicates the distance from the center of one cell to the center of a neighboring horizontal or vertical cell. For values that vary, initial values are given in Section K.5.

Global state variables, model parameters, and scales. This model is a simulation of the effects of an inoculation of *A. fumigatus* spores on a cross-section of lung tissue. As such, the inoculum refers to that given to the entire patient; only a fraction of those spores appear in any cross-section. The spatial scale was chosen for computational and visual considerations. As patient fate is typically determined in the first four days after inoculation [58], 96 hours was chosen as the virtual duration of a simulation.

Variable	State variable	Value	Units
Grid cell length	*grid_size*	10	microns
World size	*world_size*	$20 \times 40 \times 20$	grid cells
Duration of one time step	*time_step*	20	minutes
Total simulation time	*total_sim_time*	96	hours
Inoculation size	*init_inoc*	$6,000,000$	spores
Fraction of inoculum appearing in model	*inoc_frac*	0.00005	N/A
Maximum grid cell iron	$iron_{max}(p)$	100	none
Macrophage cytokine evaporation rate	*cyto_evap_m*	0.05	N/A
Neutrophil cytokine evaporation rate	*cyto_evap_n*	0.05	N/A

Table K.1: Global state variables and scales

Grid cells. Figure K.1 provides a labeled snapshot of the model. The **airway** is a tube of grid cells which branches about one-third of the way across the horizontal span of the model. In total, approximately 16% of the grid cells make up the airway. Airway grid cells have no properties other than the number of spores at that location. Non-specific **interstitial** tissue cells comprise approximately 80% of the grid cells. The state variables for these cells are iron, macrophage-specific cytokines, and neutrophil-specific cytokines. Four blood vessels run the length of the model, made up of **blood cells** occupying one grid cell each; they comprise approximately 4% of the grid cells. These cells serve as the recruitment sites for macrophages and neutrophils as well as the source of iron for the interstitial space. Iron is introduced in the blood cells and diffuses into surrounding tissue. Cytokine levels are first affected by the epithelial cells and diffuse from the epithelium throughout the interstitial and blood cells. These cytokines serve as aggregations of macrophage- and neutrophil-specific cytokines, such as TNF-α, IL-8, IFN$_\gamma$, and others. There are two separate values for macrophages and neutrophils because certain cytokines attract neutrophils but not macrophages [140] while others (such as MCAF) do just the reverse [124].

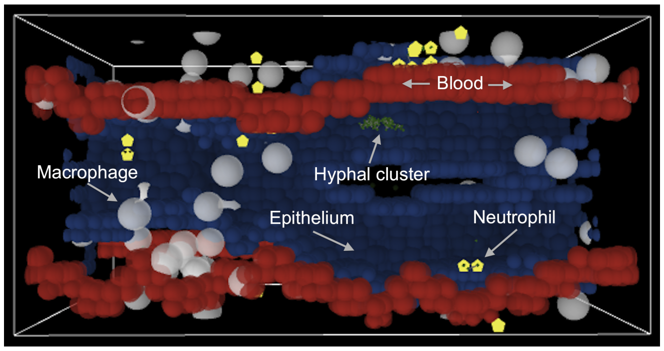


Figure K.1: Cutaway snapshot of the three-dimensional model. Spores travel from left to right through the airway.

Grid cell type	State variable	Abbreviation	Value	Units
Airway	Spore count	$fung$	varies	spores
Blood cells	Iron per hour	iph	8	none
Blood cells	Iron	$iron_p$	varies	none
Blood cells	Macrophage cytokine level	$cyto_m$	varies	none
Blood cells	Neutrophil cytokine level	$cyto_n$	varies	none
Interstitial space	Iron	$iron_p$	varies	none
Interstitial space	Macrophage cytokine level	$cyto_m$	varies	none
Interstitial space	Neutrophil cytokine level	$cyto_n$	varies	none
Blood and inter.	Iron diffusion rate	$iron_{dif}$	0.8	N/A

Table K.2: Grid cell state variables

Fungal spores. *A. fumigatus* spores drift through the airway, and if a spore meets the epithelial cell wall it lodges there with a fixed probability. This probability is small, as ciliary beating serves as the primary mechanism for elimination of the fungus [122]. Once lodged, a spore cannot be swept away. Spores that are swept away are placed on the nearest airway grid cell and continue to drift. Once a spore has traversed the entire airway, it is removed from the simulation.

Approximately 30% of spores are internalized by an epithelial cell [131]. Since approximately 3% of internalized spores survive and 34% of these germinate [132], internalized spores germinate with approximate probability 0.01. Internalized or not, all lodged spores enter a resting stage. Upon completion of the resting stage, all non-internalized spores become swollen. After remaining in the swollen stage for some time, spores germinate. Since the resting stage lasts roughly 2 hours and germination occurs between $6-8$ hours [85], the swelling process lasts approximately 5 hours. A germinated spore spawns a fungal hyphal cell, which is connected to the spore and grows into neighboring tissue in a random direction. Since behavior of hyphae are different from that of fungal spores, these agents are described separately.

State variable	Abbreviation	Value	Units
Drift speed	$speed_f$	$N(3, 0.5)$	grid cells
Probability of lodging	p_{lodge}	0.05	N/A
Probability of internalization	p_{int}	0.3	N/A
Probability of internalized spore swelling	p_{swell}	0.01	N/A
Duration of resting stage	t_{rest}	$N(120, 10)$	minutes
Duration of swollen stage	$t_{swollen}$	$N(300, 30)$	minutes
Initial health	$init\_health(f)$	100	none
Health	$health_f$	varies	none

Table K.3: Fungal spore state variables

Epithelial cells. Epithelial cells form the boundary between the airway and the interstitial tissue. Each time step, these cells first determine the number of fungal spores and hyphae within their detection radius. These counts indicate the amount of macrophage-specific and neutrophil-specific cytokines to produce, respectively. While the cells produce the cytokine levels based on fungal presence, these levels are in fact grid cell state variables, not attributes of the epithelial cells. After adjusting local cytokine levels accordingly, each epithelial cell damages any internalized spores. This continues until either the spore germinates and kills the epithelial cell, or the spore dies. It is observed in [7] that only 3% of conidia survive at least 36 hours – hence 30 hours is chosen as the approximate time it takes an epithelial cell to kill an internalized conidium.

Macrophages. Macrophages are recruited to the site of infection via the bloodstream. Each time step, if there is at least one blood cell whose macrophage-specific cytokine level is above the macrophage recruitment threshold, there is a possibility of a macrophage spawning at exactly one such location. All macrophages absorb some of the macrophage cytokines on

State variable	Abbreviation	Value	Units
Spore detection radius	s_{det}	1	grid cells
Hyphae detection radius	h_{det}	1.5	grid cells
Cytokine production factor	*cyto_rate*	100	none
Time taken to kill internal spore	e_{kill}	30	hours

Table K.4: Epithelial cell state variables

their current location, simulating uptake by receptors. Macrophages have a detection radius for conidial spores and hyphae; if a macrophage has fewer than two internalized spores, it will internalize a spore within this radius. Internalization prevents spores from growing hyphae. The internalization process takes approximately two hours [103]; since engulfed conidia are assumed unable to escape, this is used as the time it takes for a macrophage to kill an internalized spore. Macrophages do not phagocytose hyphae, but they are known to produce IL-8 to aid in neutrophil recruitment [71, 126] – this is simulated by having macrophages increase the neutrophil-specific cytokine level at their current location based on the amount of nearby hyphae. Macrophages produce ferroportin which binds to hepcidin, upon which time the complex is degraded, thereby preventing iron export [130]. Hence in the simulation, macrophages do not alter iron levels in any way.

State variable	Abbreviation	Value	Units
Recruitment threshold	m_{recr}	25	none
Probability of recruitment	$p_{recr}(m)$	0.5	N/A
Cytokine absorption rate	m_{abs}	0.05	none
Fungus detection radius	m_{det}	2	grid cells
Time taken to kill internal spore	m_{kill}	120	minutes
Neutrophil cytokine factor	m_n	5	none

Table K.5: Macrophage state variables

Fungal hyphae. Hyphae grow out of germinated fungal spores. They are attached to the spores and thus do not move. Once a hyphal cell appears, it spends time in a growth stage, during which no new hyphae can grow out of it. During the growth phase, if the hyphae has not been internalized, it attempts to absorb iron from surrounding tissue. Once the cell obtains enough iron, a new hyphal cell (or two new cells, if branching occurs) grows from the parent cell, with the new cells inheriting iron from the parent. Once it has spawned a new cell, a parent cell is no longer eligible for spawning future cells, though it does continue to take up iron from the environment. Many hyphae state variable values are unitless, abstract measures (e.g., iron levels), while others are chosen empirically.

State variable	Abbreviation	Value	Units
Duration of growth stage	t_{grow}	$N(105, 10)$	minutes
Iron level	$iron_f$	varies	none
Maximum iron level	$iron_{max}(f)$	20	none
Grid cell iron threshold	$cell\_iron_{min}$	2	none
Iron absorption rate	$iron\_abs(f)$	1	none
Iron needed for growth	$iron_{min}(f)$	2	none
Probability of branching	p_{branch}	0.15	N/A
Spacing of hyphal cells	$spacing$	0.3	grid cells
Initial health	$init\_health(f)$	100	none
Health	$health_f$	varies	none

Table K.6: Hyphae state variables

Neutrophils. Neutrophils are recruited from the bloodstream in much the same way as macrophages – if there is at least one blood cell with neutrophil-specific cytokines above a certain threshold, then there is the possibility of a neutrophil spawning at exactly one such location. To indicate absorption of cytokines by receptors, every neutrophil reduces the neutrophil-specific cytokine level at its current location. As long as there are grid cells nearby with cytokine levels above the recruitment threshold, neutrophils move towards the grid cell with highest cytokine level – this simulates the chemotactic movement of recruited neutrophils [25]. Otherwise, they move to the center of a randomly chosen neighboring non-airway grid cell. Neutrophils raise cytokine levels in order to boost recruitment of additional neutrophils by an amount equal to the number of hyphae within their detection radius. Neutrophils are initialized with a fixed number of granules, which are deposited every time step in order to degrade and kill hyphae. Various studies indicate that the approximate killing time of hyphae by neutrophils is 2 hours [29, 109]. All hyphae within the neutrophil detection radius are damaged as long as the neutrophil has not run out of granules; during this process iron is removed from the neutrophil's location in order to simulate sequestration. Additionally, neutrophils do not move while they are degrading nearby hyphae. Recruited neutrophils have a lifespan of $1-2$ days [119]; thus in the model, the average lifespan is set at 36 hours.

K.3 Process overview and scheduling

The process overview is described here as pseudocode. Bold faced italicized processes are described in detail in section K.7. In the simulation, time is discrete. Entity routines are executed serially – that is, one entity executes every command in the routine, then the next entity does the same, and so on. This means that entity state variables are updated

State variable	Abbreviation	Value	Units
Recruitment threshold	n_{recr}	15	none
Probability of recruitment	$p_{recr}(n)$	0.5	N/A
Cytokine absorption rate	n_{abs}	0.02	none
Hyphae detection radius	n_{det}	2	grid cells
Time taken to kill hyphal cell	n_{kill}	120	minutes
Initial number of granules	$init\_gran$	$N(100, 10)$	granules
Granules	$gran$	varies	granules
Life span	$life\_span(n)$	$N(2160, 240)$	minutes
Current age	age	varies	minutes

Table K.7: Neutrophil state variables

asynchronously; pertinent state variables are updated as each command is executed. The order in which entities perform the routine is randomly chosen at each time step. The code here is meant as an overview; as such, the descriptions are as non-technical as possible. Submodel pseudocode in section K.7 provides detailed instructions for each step of the process.

Algorithm 10 Lung model process pseudo-code.

```
setup map and constants
while simulation time < $total\_sim\_time$ do
    conidia routine:
        if $health_f \leq 0$ then die end if
        if mobile then
            if at edge of model then die else conidia move end if
            check for contact and lodging with epithelium
            if lodged then
                check for internalization
                begin growth countdown
            end if
        end if
        growth countdown
        if growth time reached then
            switch stage of spore do
                case swollen: germinate
                case resting and not internalized: become swollen
                case internalized: swell with probability $p_{swell}$
            if swollen then begin swelling countdown end if
        end if
    end routine
    iron uptake
    grow hyphae
    iron diffusion
    epithelial routine:
        epithelial cytokine update
        epithelial damage internal spores
    end routine
    cytokine evaporation and diffusion
    check for new macrophages
    macrophage routine:
        macrophage absorb cytokines
        if any free spores nearby then internalize spore end if
        macrophage produce neutrophil cytokine
        macrophage move
```

Algorithm 10 Lung model process pseudo-code (cont.)

```
36:         macrophage damage conidia
37:     end routine
38:     check for new neutrophils
39:     neutrophil routine:
40:         neutrophil absorb cytokines
41:         neutrophil produce neutrophil cytokine
42:         neutrophil move
43:         neutrophil damage hyphae
44:         increase age according to time scale
45:         if age limit reached then die end if
46:     end routine
47:     increase time by time_step
48: end while
```

K.4 Design concepts

Basic principles. The biological mechanics of cell-cell interactions are the basic principles underlying the design of this model. These are described in literature and derived from experimentation, both *in vivo* and *in vitro*. Some behavioral processes, such as phagocytosis, occur on a local scale between two cells, while others, such as recruitment of macrophages and neutrophils, are tissue-level processes. The model aims to tie together what is known about individual cell behavior to create an *in silico* model of a larger-scale process.

Emergence. The ability of the fungus to survive under various conditions is an emergent property of individual fungal cells competing for iron. At the same time, the tendency for survival of infection is an emergent property of the entire immune response.

Adaptation. Macrophages and neutrophils both generally travel in the direction of highest cytokine concentration; thus, they adapt their movement in response to developing infection.

Objectives. The objective of immune cells is to remove all fungal cells, which is measured simply as a count of those cells. Thus whether a patient lives or dies depends on the amount of fungus present in the system.

Learning. Agents do not change adaptive traits over time in this simulation.

Prediction. Agent prediction is similar to adaptation in this case: macrophages and neutrophils inherently predict that following the path of highest cytokine concentration will lead to the area where immune response is most needed.

Sensing. Fungal spores sense the type of grid cell they are near, be it airway or interstitial space. In addition, both spores and hyphae sense the amount of iron present at their current location, and use this information to 'decide' if they will grow. Epithelial cells, macrophages, and neutrophils can all sense nearby spores and hyphae as well as the cytokine levels of all neighboring cells. All sensing is local.

Interaction. While section K.2 provides a fairly comprehensive overview of entity interactions, it is perhaps helpful to mention several interactions which do **not** occur. Specifically, macrophages and neutrophils do not directly interact with other immune cells, either of their type or another. For example, the presence of many macrophages at one location has no bearing on the likelihood of another macrophage targeting that location. Fungal hyphae interact with other hyphae as long as they are connected – in particular, iron uptake is shared among all hyphae at a given location, and when new hyphae are spawned, they inherit iron from the parent cell.

Stochasticity. Agent movement, when not dictated by concerns such as cytokine levels, is stochastic: agents face in the direction of a randomly chosen legal space and move in that direction. Hyphae grow in random directions, leading to clumps of fungus rather than long strands; hyphal branching is stochastic as well. In general, stochasticity is employed as a substitute for unknown mechanisms of the biological entities being represented.

Collectives. Fungal hyphae form a collective of sorts, as they maintain information about every other hyphae they are connected to. Iron stores are shared among local neighbors, which has an indirect effect on the collective group. These groups arise naturally and from the local rules of individual entities; they are not entities themselves and have no distinct state variables.

Observation. Many data can be collected from the model for analysis and understanding. These include fungal cell counts, iron levels, cytokine levels, macrophage and neutrophil counts, and collateral tissue damage. Any or all of these may be collected during each simulation, depending on the needs and interests of the researcher. The data are collated in a universal spreadsheet format for ease of statistical interpretation and analysis.

K.5 Initialization

Many initialization values are given in table K.1; as such, they are not repeated here. In addition to these values, the following information may be necessary to reproduce results: all grid cells begin with cytokine and iron levels at 0. The initial inoculum is placed at one end of the airway. There are no macrophages or neutrophils until recruitment occurs via cytokine triggering. Macrophages and neutrophils are created at the blood cell where recruitment occurs. Given that the model is meant to function as an *in silico* laboratory, it is not meant to be entirely robust with respect to initial conditions. Indeed, a key feature of the model is the ability to investigate the effect of initial conditions on outcome.

K.6 Input data

The model requires a map file (in .txt format) in order to set up grid cell types and locations. The map file is a three-dimensional matrix. Grid cell layout is determined aesthetically to serve the visual representation of model dynamics. No other input data is used.

K.7 Submodels

There are seventeen submodels listed in the pseudocode in section K.3. These are described here using pseudocode. Note that while the code in section K.3 is intended to read naturally, the submodels presented here are more technical.

Submodel 1 ***conidia move***

```
face random direction down airway
move forward $speed_f$
if at edge of map then die end if                                  ▷ simulates inoculum passing through airway
```

Submodel 2 ***iron uptake***

```
for all non-airway grid cells with $iron > cell\_iron_{min}$ do
    for all non-captive hyphal or hyphal-tip fungus here with $iron_f < iron_{max}(f)$ do
        $iron_f = iron_f + (iron\_abs(f) \cdot iron_p/\text{no. fungus here})$      ▷ iron is split among all eligible fungus
        $iron_f = \min\{iron_f, iron_{max}(f)\}$                       ▷ iron capped at $iron_{max}(f)$
    end for
    $iron_p = (1 - iron\_abs(f)) \cdot iron_p$                          ▷ grid cell iron reduced
end for
```

Submodel 3 ***grow hyphae***

```
for all germinated non-captive spores which haven't spawned hyphae do
    spawn hyphal cell
    hyphal cell routine:
        move forward $spacing$ in a random non-airway direction
        set stage to 'hyphal tip'
        set growth time to $t_{grow}$                                  ▷ begin swelling phase
        $iron_f = iron_f$ of parent cell
    end routine
end for
for all hyphal tip cells with $iron_f > iron_{min}(f)$ and growth time reached do
    $num\_to\_spawn = 1$
    if $rand(0, 1) < p_{branch}$ then                                  ▷ probability of branching
        $num\_to\_spawn = 2$
        $iron_f = iron_f/3$
    else
        otherwise $iron_f = iron_f/2$
    end if
    spawn $num\_to\_spawn$ hyphal cells
    set stage to 'hyphal'                                              ▷ only 'hyphal tip' cells can spawn
    each new hyphal cell performs hyphal cell routine
end for
```

Submodel 4 *iron diffusion*

```
1: for all blood grid cells do
2:     $iron_p = \min\{(iron_p + iph \cdot time\_step/60), iron_{max}(p)\}$     ▷ iron is capped at $iron_{max}(p)$
3: end for
4: for all non-airway grid cells do
5:     give each neighbor cell $(iron_{dif} \cdot iron_p/26)$ iron     ▷ 26 neighbors in 3D
6:     airway cells: $iron_p = 0$     ▷ no iron in airway
7: end for
```

Submodel 5 *epithelial cytokine update*

```
1: for all epithelial cells do
2:     $spore\_count$ = swollen or germinated spores within $s_{det}$
3:     $hyphae\_count$ = hypal or hyphal-tip cells within $h_{det}$
4:     $cyto_m = cyto_m + cyto\_rate \cdot spore\_count$     ▷ increase cytokine level based on counts
5:     $cyto_n = cyto_n + cyto\_rate \cdot (hyphae\_count + spore\_count)$
6: end for
```

Submodel 6 *epithelial damage internal spores*

```
1: for all internalized spores do
2:     $health_f = health_f - (init\_health(f) \cdot time\_step/e_{kill})$
3: end for
```

Submodel 7 *cytokine evaporation and diffusion*

```
1: for all grid cells do
2:     $cyto_m = (1 - cyto\_evap\_m) \cdot cyto_m$     ▷ cytokine evaporation
3:     $cyto_n = (1 - cyto\_evap\_n) \cdot cyto_n$
4:     add $cyto_m/26$ to each neighboring cell     ▷ $cyto_m$ refers to cytokine level of contributing cell
5:     add $cyto_n/26$ to each neighboring cell     ▷ diffusion to 26 neighbors in 3D
6: end for
7: airway cells: $cyto_m, cyto_n = 0$     ▷ no cytokines in airway
```

Submodel 8 *check for new macrophages*

```
1: if any blood cells with $cyto_m \geq m_{recr}$ then
2:     let $L$ be one such location, selected at random
3:     if $rand(0,1) < p_{recr}(m)$ then     ▷ probability of recruitment
4:         spawn new macrophage at location $L$
5:     end if
6: end if
```

Submodel 9 *macrophage absorb cytokines*

```
1: $cyto_m = (1 - m_{abs}) \cdot cyto_m$
```

Submodel 10 *macrophage produce neutrophil cytokine*

```
1: let $h$ be number of fungus in stage ‘hyphal tip’ or ‘hyphal’ within $m_{det}$
2: grid cell here: $cyto_n = cyto_n + m_n \cdot h$
```

Submodel 11 *macrophage move*

```
1: if any neighboring grid cells with $cyto_m \geq m_{recr}$ then
2:     move to neighbor with highest $cyto_m$     ▷ follow highest cytokine concentration
3: else
4:     move to random neighboring grid cell     ▷ may wrap around map
5: end if
```

Submodel 12 *macrophage damage conidia*

```
1: for all internalized conidia do     ▷ macrophages internalize spores only, not hyphae
2:     $health_f = health_f - (init\_health(f) \cdot time\_step/m_{kill})$     ▷ damage depends on time scale
3:     ensure location is same as macrophage
4: end for
```

Submodel 13 *check for new neutrophils*

```
1: if any blood cells with $cyto_n \geq n_{recr}$ then
2:     let $L$ be one such location, selected at random
3:     if $rand(0,1) < p_{recr}(n)$ then                    ▷ probability of recruitment
4:         spawn new neutrophil at location $L$
5:     end if
6: end if
```

Submodel 14 *neutrophil absorb cytokines*

```
1: $cyto_n = (1 - n_{abs}) \cdot cyto_n$
```

Submodel 15 *neutrophil produce neutrophil cytokine*

```
1: let $h$ be number of non-internalized fungus in stage ‘hyphal tip’ or ‘hyphal’ within $n_{det}$
2: grid cell here: $cyto_n = cyto_n + h$
```

Submodel 16 *neutrophil move*

```
1: if degranulating nearby hyphae then
2:     do not move
3: else if any neighboring grid cells with $cyto_n \geq n_{recr}$ then
4:     move to neighbor with highest $cyto_n$                ▷ follow highest cytokine concentration
5: else
6:     move to random neighboring grid cell                  ▷ may wrap around map
7: end if
```

Submodel 17 *neutrophil damage hyphae*

```
1: if any hyphae within $n_{det}$ and $gran > 0$ then
2:     hyphae: $health_f = health_f - (init\_health(f) \cdot time\_step / n_{kill})$
3:     $gran = gran - 1$
4:     all grid cells within $n_{det}$: set $iron_p = 0$    ▷ neutrophils produce lactoferrin to sequester iron
5: end if
```