DATA, MACHINE LEARNING, AND POLICY INFORMED AGENT-BASED MODELING

By

David J. Butts

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computational Mathematics, Science, and Engineering – Doctor of Philosophy

2024

ABSTRACT

Agent-based models (ABMs) examine emergent phenomena that arise from individual agent rules. This work extends the basic ABM paradigm in three key areas: data integration, evaluation of policies, and incorporating machine learning techniques. The dissertation investigates how data-driven approaches can enhance the accuracy of ABMs, explores the practical applications of ABMs in developing policies for real-world issues, and examines the fusion of machine learning with ABMs to optimize model design and functionality.

The dissertation begins by establishing the background and fundamental principles of agent-based models, highlighting their evolution from simple cellular automata to intricate systems encapsulating decision-making and adaptive behavior. It examines the role of these models in simulating dynamic interactions within systems, especially in scenarios where traditional methods may fall short of capturing the complexities of agent interactions. Following the introduction of key concepts, a series of projects that function in pairs demonstrate the versatility of ABMs and address the three key areas discussed above.

The first pair addresses data integration of GPS deer movement data into a generalized Langevin model and its use in uncertainty quantification of disease spread. Exploratory data analysis revealed a discernible non-parametric trend in the GPS data with non-Gaussian statistics. This analysis led to a model that is consistent with the observed data. Subsequent incorporation of chronic wasting disease (CWD) and population dynamics were used to forecast the prevalence of CWD. This extended model was analyzed with a global sensitivity analysis that tied variance in disease prevalence to variance in the parameters of the model, providing predictions of future prevalence of the disease.

The second pair examines policy evaluation, specifically strategies for mitigating disinformation in social networks. Multiple strategies were evaluated on various topologically diverse networks that led to policy recommendations. Simulations on these graphs revealed challenges associated with large network simulations, particularly in computational cost and influence of network topologies. These challenges led to a method to miniaturize real social networks while preserving key attributes, enabling more efficient and realistic simulations to run on artificial social networks.

The final pair investigates the possibility of inverting the ABM paradigm to instead have agents learn their own rules through environmental interactions. Reinforcement learning was applied to a model of conflict based on capture the flag, where an agent learned in progressively difficult competitions. The emergence of deterrence was explored through adding asymmetries between competing teams, and differential equation-based models were created to help interpret results.

To Guido van Rossum, Leslie Lamport, and Linus Torvalds.
Without you, creating this dissertation would have literally been impossible.

ACKNOWLEDGEMENTS

Throughout my time spent in graduate school, I have received a large amount of support from many people. I am sure I have not listed everyone who has helped me along the way, but that does not mean I do not appreciate your support.

First and foremost, I want to thank to my advisor Michael Murillo. Your guidance, mentorship, and friendship has shaped me into a better scientist and a better person. I will greatly miss the many conversations had over wine and old fashions, and time spend kayaking and walking around campus. I also want to thank the remaining members of my PhD committee. Arika Ligmann-Zielinska, I really enjoyed your class in agent-based modeling, and your continued assistance with the sensitivity analyses in my projects. John Luginsland, it is always fun to have conversation with you, and I hope to be as humorous as you when I have a "real job". More seriously, thank you for your support when I was searching for jobs. Yuying Xie, you taught what I can only describe as the hardest class I have ever taken. However, I appreciate it because not only did I learn a lot during your class, but I gained confidence in my abilities to solve difficult problems. To each of you on my committee, thank you again for your guidance and support throughout my time in graduate school.

I also want to thank the Murillo Group's current and past members. Specifically, Zach Johnson, Jorge Martinez-Ortiz, Luciano Silvestri, Jannik Eisenlohr, Chris Gerlach, Thomas Chuna, Luke Stanek, and close collaborators Liam Stanton and Jeff Haack. You have all helped me improve my research and presentation skills through countless meetings and practices. I also really enjoyed our many happy hours, including the ones that occurred virtually throughout the pandemic. It was great being part of a research group that effortlessly doubled as a group of friends. Outside of the Murillo group, I want to thank Cole Stewart who volunteered to read and provide feedback for many of the chapters in this dissertation.

I would like to thank the the many collaborators here at Michigan State and at Los Alamos National Lab who have been a part of the research presented in this dissertation and research I have done outside what is discussed here. I would definitely not be where I am now without you, and it has been great learning about your areas of expertise.

One of the hardest parts of leaving Michigan will be leaving Mid-Michigan Runners. For the past few years you have all helped me keep my sanity, and I will really miss Tuesday night runs. It amazes me how many memories we have made through training runs, the sweater run, races, vacations, parties, and general get togethers. Thank you, Seth, Simon, Robert, Lewis, Jenn, and the rest!

Ben, Sam, Joe, Terrence, and Will, you guys are some of my oldest friends. Thank you for making an effort just about every year for the past 10 years to come see me on my birthday.

I hope we can keep at least a yearly get together going in the future. I want to thank you guys, and the rest of the Watertown crew, for the much needed breaks throughout graduate school.

Finally, I want to thank my family. In particular, for the emotional and moral support of my mother, Sarah Rabin, brother, Jake Butts, and sister, Rachel Butts, that kept me moving forward throughout my PhD. Thank for being available, especially through the hardest times in graduate school. And thank you for putting up with my stress and complaining for at least the past six years! To my girlfriend Fran, thank you for your support throughout graduate school. You listened to endless rants when deadlines were closing in, but you helped stay focused on getting things done. Most importantly, thank you for being willing to start a new job in a different state to avoid living across the country from me. I am looking forward to starting a new chapter in life.

TABLE OF CONTENTS

CHAPTER 1:
INTRODUCTION

1.1 Background on agent-based models

Models are indispensable tools for researchers, offering invaluable insights into various phenomena when real-world experiments are impractical or unfeasible. There are several reasons that might sway a researcher toward modeling over direct experimentation. For instance, the sheer cost of certain experiments can be prohibitively high, as exemplified by the expenses tied to launching payloads into space, which can cost up to tens of thousands of dollars per kilogram [1]. In such scenarios, modeling acts as a precursor, allowing researchers to assess the feasibility of their endeavors before actual implementation. Equally, time-critical situations, such as the planning of evacuation routes amidst looming weather threats, necessitate immediate action; waiting for real-time data, in such cases, is not only impractical but also perilous. Beyond these tangible constraints, there also exist ethical barriers. For example, deliberately releasing pathogens to study the propagation of diseases is not an option. While these examples are far from exhaustive, they underscore the myriad challenges that models help researchers navigate.

Just as there are numerous reasons for utilizing models, the realm of modeling is vast. There are many tools and methods tailored to diverse phenomena. These range from mathematical constructs such as partial differential equations, to tangible replicas like solar system models, or even surrogate organisms like rats in drug trials. Among these, computational models – those that rely on computers to be evaluated – are widely applied across various fields, and can significantly mitigate the financial, temporal, and ethical limitations associated with conducting experiments. One subtype, the agent-based model, stands out, particularly for simulating complex, multi-agent systems. This dissertation aims to delve deeper into this computational paradigm.

What is commonly referred to as agent-based modeling encompasses a trio of model types: cellular automata (CA), individual-based models (IBM), and agent-based models (ABM). Of these, CAs, with their simple arrays and iterative rule-based updates, are the most straightforward. IBMs and ABM, in contrast, encapsulate more complexity. They can comprise arrays, but their defining feature is the incorporation of agents, entities capable of mobility and decision-making. Historically, the distinction lay in their focus: IBMs emphasized the variability among individuals, whereas ABMs prioritized decision-making and adaptive behavior. Now, the terms “individual-based” and “agent-based” are often used interchangeably [2].

Fundamentally, agent-based modeling is a simulation technique. It characterizes the evolution of a system's state through a rule-driven process. Here, the "system" encapsulates the entirety of the model, specifically both the agents and the environment. A "state" captures a specific configuration of these agents and the environment, and "rules" guide the transitions between these states. The complexity of rules isn't strictly tied to mathematical formulations [3, 4]. To illustrate, consider binary CA models such as: John von Neumann's self-replicating device [5], James Conway's game of life [6], or Stephan Wolfram's Wolfram models [7]. In these models, their systems consists of single arrays with cells that can have a value of either 0 or 1. With each iteration, each cell either maintains or switches its based on neighboring cell values. Even such simple systems can generate many states. For an array with n cells, there are a total of 2^n possible. This complexity is only amplified in intricate models. Take, for instance, the forest management ABM described in [8]. This model contains three types of agents: government, conservationist, and logging-company agents, each of which has a distinct forestry goal. The model represents the forest has two two-dimensional array that encodes the availability and monetary value of trees. Each agent type uses these availability and value maps to construct conceptual maps that encode plans to fulfill their forestry goals. Capturing the vast plethora of potential states in such models reveals the depth and adaptability of agent-based modeling.

By describing models as transitions between states due to actions being taken, agent-based modeling employs a bottom-up approach. Such an approach allows for the observation of emergent collective behaviors among agents in the model. Moreover, this approach ensures modularity, making complexity additions to the model seamless. As an example, consider modeling an influenza-like illness. A common modeling approach is to use susceptible-infected-recovered (SIR) compartmental models [9]; however, agent-based models offer richer dynamics. Figure 1.1 contrasts an agent-based SIR model with its ordinary differential equation (ODE) counterpart. The right panel plots the number of susceptible (green), infected (red), and recovered (yellow) individuals versus time. The thicker lines are the solutions to the ODEs and each thinner line is from a single execution of the ABM; in total the ABM was executed 100 times.

In the ABM version, individual agents move randomly in space. Infected agents, represented with red points, can transmit the disease to susceptible individuals, represented with green points, when they come into close contact. Eventually, infected individuals recover and become immune; these recovered individuals are shown as yellow points. This agent-based methodology allows for a direct exploration of potential correlations in the model. For instance, it might provide insights into how the initial spatial distribution of infected individuals can influence the rate or extent of disease spread. However, when the goal is to

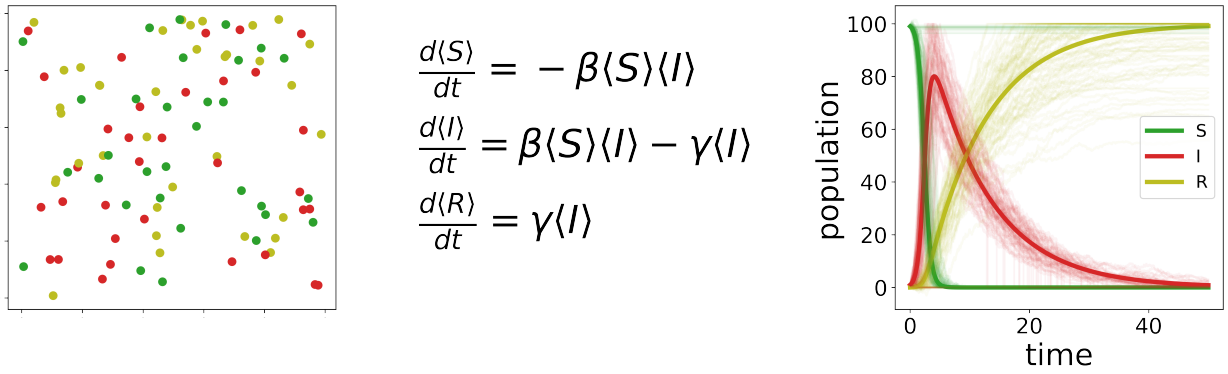


Figure 1.1: On the left is a snapshot of an SIR model implemented as an ABM. Here, red points represent infected individuals who can infect the susceptible green points. Red points eventually recover and become immune yellow points. The center shows the mean-field limit of this model, which is the first order ODE implementation of an SIR model. On the right individual runs of the ABM are shown with translucent lines, while bold lines are the solutions of the ODEs. Notably, while the ODE implementation effectively describes average behavior, it fails to capture certain dynamics like stochastic extinction, where the disease ceases to spread before infecting the entire population.

summarize the larger-scale dynamics without getting into the specifics of individual interactions, we often turn to the mean-field approximation. For diseases, this might manifest as ordinary differential equations (ODEs) that take into account the average behavior of agents. In the case of our SIR example, the ODEs are a first-order approximation of the ABM and arise in the mean-field limit [10] of the model. This limit assumes that pairwise correlations among agents factor into a product of averages ($\langle SI\rangle = \langle S\rangle\,\langle I\rangle$); however, other approximations could be used to derive higher order sets of equations [11, 10]. For instance, a second order approximation would require nine total equations to describe each single and pairwise correlation of S, I, and R. Although ODE approximations simplify the dynamics, they can sometimes omit essential details present in the full agent-based model. Such ODE models condense the complex interactions into parameters, like β for disease transmission, that average out the diverse individual interactions.

Taking the idea further, to account for agent attributes like age or gender, one would expand the agent-based model rules to factor in these variables. The differential equation-based model would then require a new set of equations and parameters for each added attribute. This could quickly lead to more complex models like partial differential equations (PDEs), which are inherently more challenging to solve. Despite this, the strength of agent-based models lies in their inherent flexibility to integrate added complexities. However, the trade-off is computational efficiency; ABMs are generally more computationally demanding than solving ODE models, especially when one needs multiple model runs, such as in training

or uncertainty quantification scenarios.

Beyond the cases presented, agent-based modeling has applications in a wide range of fields including: finance [12], optimization [13], human behaviors [14], sociology [15], healthcare [16, 17], and many more [18].

1.2 Areas of improvement for agent-based models

As the field of agent-based modeling evolves, it grapples with an inherent tension regarding the appropriate level of model detail. This tension anchors itself in the age-old principles of parsimony versus accuracy. Some ABM practitioners posit that models should be as streamlined as possible, capturing only essential features for clarity, generalizability, and to prevent overfitting. This perspective aligns with Grimm et al.'s discussion the Medawar zone, which represents the optimal range of model complexity for maximum benefit [19, 20]. Epstein [21] further elucidates this viewpoint, highlighting how even a potentially "wrong" yet simple model can underpin the foundation of fields, e.g. Hooke's Law and the Lotka-Volterra model. Conversely, Edmonds and Moss [22] introduce the "Keep It Descriptive Stupid" (KIDS) approach, contrasting it against the traditional "Keep It Simple Stupid" adage. They argue for comprehensive models, and emphasize simplification only when it's justifiable. Central to this debate is the overarching question of how one strikingly captures the complexity of reality in computational representations—a challenge that has been echoed in disciplines beyond ABM, reflecting broader epistemological concerns about representation, comprehension, and forecasting [23, 24].

This competition between parsimony and realism illuminates three facets of agent-based modeling warranting deeper exploration. 1) Data integration: exploring how data can be harnessed to determine the necessary complexity of a model and, moreover, once a model's structure is established, how data can further guide the determination of its functional form and parameterization. Rand [25] discusses the development and challenges of agent-based models in an era of big-data. Additionally, many authors have applied a data-driven approach to a variety of applications including forecasting diseases [26, 27], family formation [28], and adoption of solar power [29]. 2) Policy evaluation: delving into how agent-based models can be effectively applied to real-world challenges, thereby bridging the gap between theoretical modeling and practical problem-solving. Macal provides an extensive overview of ABM's various real-world applications, focusing on both its benefits and the associated challenges [18]. 3) machine learning incorporation: investigating the fusion of machine learning (ML) techniques with agent-based modeling to aid in defining the model.

In pursuit of exploring, refining, and pushing the boundaries of agent-based modeling, this dissertation delves into the previously discussed facets of this paradigm. Each subsequent

pair of chapters represents unique projects I undertook aimed at advancing the design and utility of agent-based models across diverse applications. Chapter 2 incorporates previously published work [30] that showcases how data can be used to develop an agent-based model of the movement of deer. Through exploratory data analysis, pivotal features from multiple datasets were gleaned, culminating in a data-consistent random walk model that underpinned the agent rules. Chapter 3 builds upon this work incorporating disease and population dynamics into the model. Using data from multiple sources, the model is parameterized, and a global sensitivity analysis is performed. This analysis provides methods for quantitatively exploring variance in the models output to variance in the data used as input to the model. Chapter 4 pivots to creating agent-based models that can be used to make decisions in the real world, and is based on previously published work [31]. In particular, this chapter focuses on strategies to curtail the spread of disinformation in social networks. This chapter presents an ABM simulating the spread of disinformation and evaluates different mitigation strategies, with the goal of informing users of the model of which strategies are viable. In Chapter 5, I discuss methods for generating simulated social network. These generated networks are smaller versions of larger social networks, created such that they preserve attributes of the original network. The capability to generate networks in this fashion will allow for more experiments to be run while not compromising the results of the experiments from over-simplifications. Transitioning to the intersection of ML and ABM in Chapter 6, we start by proposing new models of deterrence to use as a baseline comparison to ML-infused ABMs. Chapter 7, the focuses on the inclusion of reinforcement learning (RL) within ABMs, with the aim of inferring policies directly from an ABM. We explore deterrence in an agent-based model of capture the flag where players in the game are controlled by RL agents.

To ensure clarity and ease of navigation, each chapter commences with a succinct summary, spotlighting the principal contributions and discoveries. Through discussing diverse applications, from deer movement patterns to the spread of disinformation on social networks, this dissertation presents a comprehensive investigation into the future of agent-based modeling.

CHAPTER 2:
DATA-DRIVEN AGENT-BASED MODEL BUILDING FOR ANIMAL MOVEMENT THROUGH EXPLORATORY DATA ANALYSIS

2.1 Summary

This chapter delves into the initial facet of ABMs: data integration. The work discussed in this chapter is based on my previously published work in [30]. The primary goal was to develop a data-driven ABM to model the movements of deer. To achieve this, two distinct datasets were analyzed: one of deer relocation data and the other consisting of resource selection functions. Exploratory data analysis revealed an absence of correlations between these datasets and a discernible non-parametric trend in the relocation data. To address this trend, the Fused-lasso machine learning technique was employed, enabling the trend's detection and subsequent removal.

My major contribution in this work was the development of a random walk model that utilized non-Gaussian noise. The random walk model was able to accurately produce simulated data that was consistent with the data used to develop it, and captured movement features that are absent in more traditional modeling techniques. Such features are important when considering the management of diseases and populations of animals, which is discussed in the following chapter.

2.2 Introduction

Understanding how wildlife move and use the landscape has interested wildlife professionals since the inception of wildlife ecology and management [32]. Researchers have explored behaviors such as migration [33, 34, 35], resource use [36, 37, 38], space use [39, 40], and risk avoidance [41, 42]. Knowledge obtained from such research improves our ability to manage and conserve wildlife [43]. Quantifying movement behavior is aided by incorporating observations into mathematical models, such as hidden Markov models [44, 45, 46] aimed at detecting behavior switching, state-space models [47, 48, 49] used to correct measurement errors and identify behaviors, and models of random walks on potential surfaces [50, 51] that attempt to tie movements to resource distributions.

As technology for collecting telemetry data advances, higher-quality data are becoming available. Such additional data provide insights into complex behavior that can be used to create more realistic models. One way to bridge the gap between data generation and analysis is through exploratory data analysis (EDA) [52, 53, 54]. EDA aids the development

of more realistic models by enabling features of the data to be identified and allowing model assumptions to be checked. EDA methods are broad and vary with the application, but many approaches, including visualizing data, exploring correlations, and generating statistics, can be applied to any domain. The importance of checking model assumptions was highlighted in a series of papers that revealed that 16 out of 17 models claiming to provide evidence for Levy flights were incorrect [55]. Problems with the analyses included misinterpretation of data [56], the use of inaccurate fitting methods [57], and the assumption of a heavy tail without testing alternative hypotheses. EDA can mitigate these problems by allowing assumptions to be verified and by exposing model weaknesses.

We provide insight about the importance of EDA and present a specific example of applying EDA to model development using movement data and resource-selection functions (RSFs). The rest of the chapter is summarized in Fig. 2.1 and organized as follows. In the next section, we give an overview of EDA and present our data and EDA approach. Our analysis begins with visualizing and identifying features that we later either corrected for or incorporated into our models. We then test for correlations between the movement data and RSFs. Next, we present a machine-learning technique to remove trends from our movement data. The section is concluded by introducing copulas and kernel-density estimates (KDEs) that are used to construct bivariate distributions from the marginals. In the following section, we develop a Langevin random-walk model with non-Gaussian noise that models the movement of a single deer. This model is then discretized to create a multiple-deer agent-based model (ABM) of deer movement with parameters obtained from our movement data. We compare our Langevin model to three other random-walk models to illustrate the importance of EDA and its utility for model development. Finally, we show the results of ABM simulations of 15 deer in three groups to illustrate how parameters can be sampled from the data and to illustrate the distributions of areas covered. We discuss group overlap patterns exhibited in our simulations; studying such group overlap patterns could lead to a better understanding of disease spread.

2.3 Exploratory Data Analysis

In this chapter, we use EDA to suggest a model's rules and to provide empirical parameters for it. As emphasized by Tukey [52], developing models in this order mitigates the introduction of biases.

In general, we group EDA into two steps. The first step is to explore and adjust data quality [58, 59]. This step can include data transformations (e.g., logarithmic compression), imputation, smoothing noisy signals, resampling on more convenient grids and data fitting. In the second step, patterns are sought in the cleaned data. These patterns may appear as

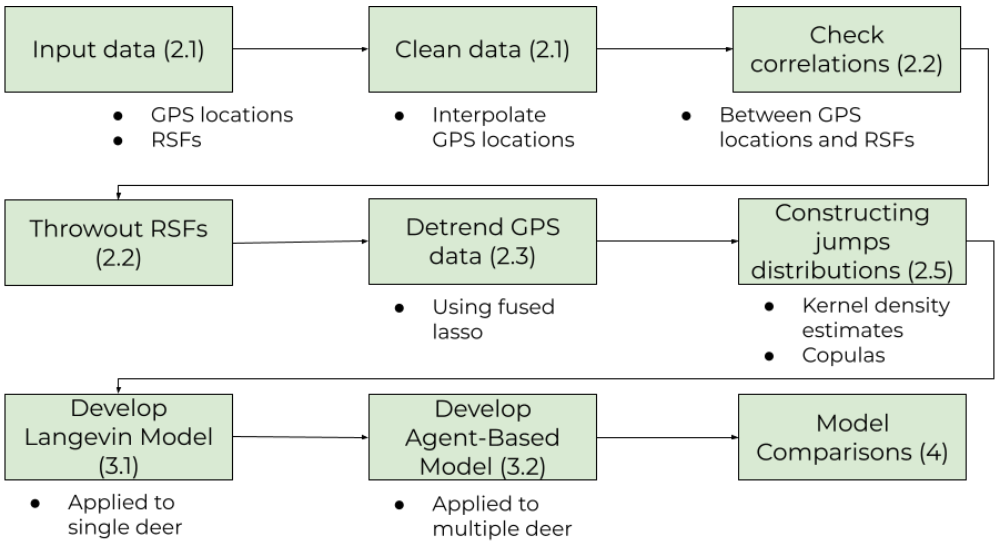


Figure 2.1: Summary of chapter organization. Each green box represents a major step taken in the corresponding section listed in parentheses.

shapes of distributions, summary statistics, correlations or causal relations. More complex patterns can be discovered using machine-learning techniques that are capable of finding patterns that are difficult for humans to find. Central to all of these EDA efforts is visualization. Visualization very clearly reveals patterns [60] such as trends (with line plots) and correlations (with scatter plots); the functional forms of distributions are readily revealed and quantified with, for example, histograms, violin plots and box-and-whisker plots. High-dimensional data can be viewed using parallel plots [61, 62] and/or using dimensionality-reduction techniques [63, 64]. All of these analysis and visualization techniques are readily available in most programming languages, such as in Python's matplotlib [65], scikit-learn [66] seaborn [67] and yellowbrick [68] libraries. In what follows, we will employ these EDA steps to analyze a real dataset. As we will see, some EDA results directly impact the choice of rules for an ABM, whereas other results provide insight and error detection without directly impacting model formulation.

2.3.1 Initial Data Analysis

The first step in EDA is initial data analysis (IDA), which serves to clean and explore data without impacting model formation or attempting to answer questions. The goals of IDA are to remove bad data, identify outliers, expose inconsistencies, perform transformations, assess the relevance of data and begin visualizations.

To illustrate the IDA process, we begin with a specific dataset relevant to our goal of characterizing deer movement patterns. We examine data from GPS-collared deer that were captured, collared, and monitored according to and with approval by State University of

New York College of Environmental Science and Forestry Institutional Animal Care and Use Protocol no. 2005-1. These data tracked the movements of 71 white-tailed deer inhabiting central New York, USA from 2008 to 2009 and have been published previously [69, 70].

The data collected on each deer consisted of location data, in the form of GPS locations and turn angles recorded for each deer in 5-hour intervals, together with the age and sex of each deer. Locations were recorded for between 8 and 600 days for each deer in this study; the duration of data collection for each deer varied with the life span of the deer and of the GPS-collar battery, with how long each collar remained on each deer, and with mechanical error.

RSFs are also included in the dataset (see Fig. 2.5); RSFs are models that yield values proportional to the probability that a unit of resource will be used [71]. RSFs can be constructed using many methods [71], including methods that account for observed animal movements [51]. The RSFs that we examined were created using a step-selection method to assess resource selection by the collared deer in this study; these RSFs have been described previously [69] (link). Six RSFs are included in the dataset, corresponding to resource selection in three seasons for both males and females.

We began our EDA of this dataset by visualizing the data. Fig. 2.2 depicts a single deer's recorded GPS locations in universal transverse mercator (UTM) coordinates in a scatter plot showing spatial locations and in other plots showing UTM coordinates over time. Note that, although the collar is attached to a single deer, it reflects that deer's individual behaviors and its interactions with other deer and the environment. This trajectory shows that the deer spent time in two regions. We refer to these regions as "basins," and we refer to jumps between basins as "basin hops." In the time-series data for UTM coordinates, basins can be seen as relatively constant coordinate values at which the animal remained over a substantial period of time. A basin hop is indicated by a sudden shift in at least one coordinate. The number of basins visited varied for each deer in the dataset. Movement within basins and basin-hopping behavior may reflect different behavioral states of an animal; for example, slow motion in a basin may indicate foraging, while basin hops may indicate migration or dispersal [72].

Positions were recorded with non-uniform time intervals, and there were instances of missing entries due to GPS collar malfunctions. For later convenience, the dataset was mapped onto a uniform grid with no missing entries. A uniform grid with N_g grid points was chosen based on the number of measurements in the dataset. The value at each grid point was found by linearly interpolating between the closest points in the original dataset on either side of the chosen grid point.

After interpolating the movement data, we examined the jumps j, calculated as the

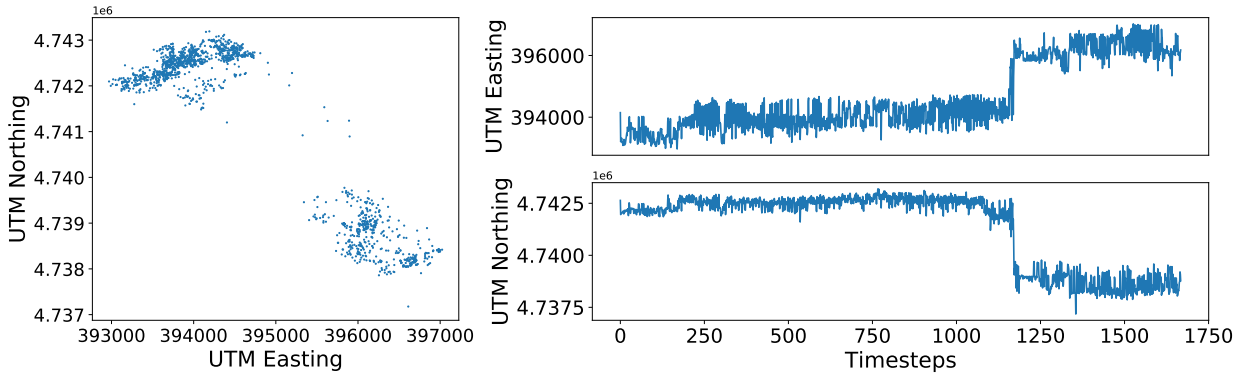


Figure 2.2: Identifying basins and basin hops. On the left, we show a single deer's recorded GPS locations in UTM coordinates. There are two discernible regions, which we refer to as basins. On the right, the two UTM coordinates are plotted separately vs. the five-hour timestep. Basins are indicated by relatively constant locations, and a jump between basins, which we term a basin hop, is seen as a sudden shift in at least one coordinate.

distances between two adjacent recorded locations. In Fig. 2.3, we show an example of a jump distribution, which, like the others, closely resembles the zero-mean Laplace distribution

$$f(j, b) = \frac{1}{2b} e^{-|j|/b}. \tag{2.1}$$

The parameter b characterizes the scale of the distribution and, for n jumps, has the maximum likelihood estimate

$$\hat{b} = \frac{1}{n} \sum_{i=0}^{n} |j_i|. \tag{2.2}$$

Jump distributions are typically assumed to be Gaussian or power-law distributions [44, 45, 46, 47, 48, 49, 50, 51, 56, 57, 55], resulting in normal random-walk or Levy-flight models, respectively. A Laplace distribution decays more slowly than a Gaussian distribution, allowing for larger jump sizes; however, Laplace jumps are not as extreme as those of a Levy flight. Thus, jump sizes allowed by Laplace noise range between those predicted by a normal random-walk model and those of a Levy-flight model. Comparisons of maximum likelihood estimates for each type of distribution to the jump data are made in Fig. 2.3, which shows that a Laplace distribution is superior to a Gaussian, without over-predicting the tail as would a power law.

We used Akaike's information criterion (AIC) as a metric to compare Gaussian and Laplace models for all of our movement data, as shown in Fig. 2.4. For each deer, we treat UTM Easting and Northing movement data separately, and we compute AIC scores for each using both the Gaussian and Laplace models. Each blue point corresponds to a single deer's

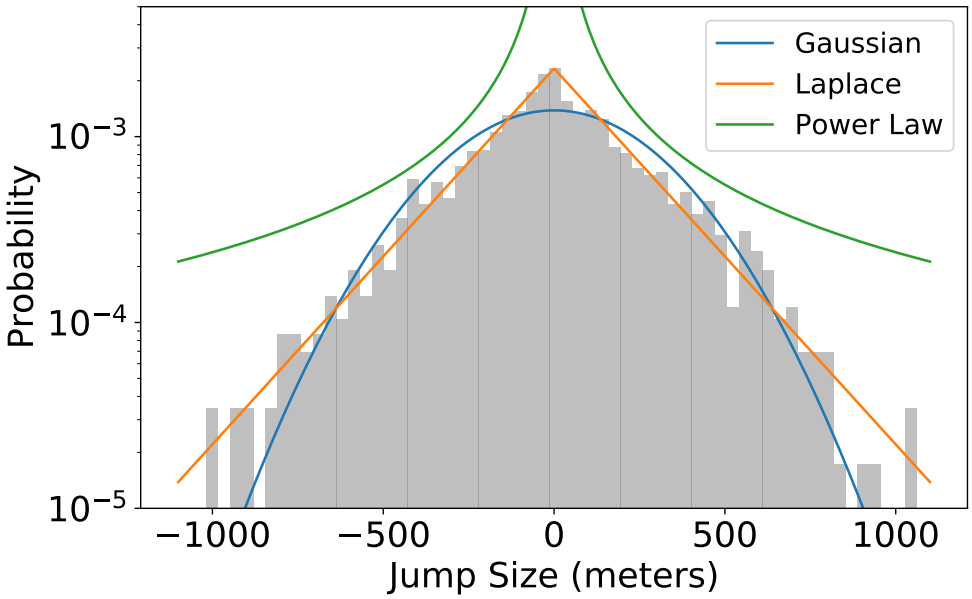


Figure 2.3: Comparison of jump distributions. Shown are the data (grey) and maximum likelihood fits for power-law (green), Gaussian (blue) and Laplace (orange) distributions. The data and the Laplace distribution have tail jump sizes in between those of the Gaussian and power-law models.

interpolated UTM Easting AIC scores, and each orange point to its UTM Northing AIC scores. The difference between the AIC for the Gaussian model and the AIC for the Laplace model for each point is shown on the vertical axis, and the Gaussian AIC for each point is shown on the horizontal axis. The dashed grey line indicates where the vertical axis value is zero, i.e., where both AIC measurements are equal. Points lying above the gray line represent movement data for which a lower AIC is obtained using a Laplace-distribution fit. For all but two of these points, the Laplace fit had a lower AIC. Both of our proposed fits had a single parameter, which indicates that for all of our data points lying above the grey line, the Laplace distribution fit the corresponding movement data more accurately.

After visualizing and interpolating our data, we treated the basin-hopping trends we observed. Many EDA methods and time-series models assume that data are stationary and account for trends separately. Before further analysis, the basin-hopping trend needed to be removed. One method for removing trends from animal-movement data is to use a potential function [50, 73, 74] whose gradient guides the motion of an animal.

These potentials can be used to construct potential surfaces, which can be linked to the distribution of resources [51]. The RSFs provided to us could possibly be used to generate a potential surface, but they were created at a discrete spatial resolution, which resulted in flat regions with sharp boundaries. These discontinuities make it difficult to approximate

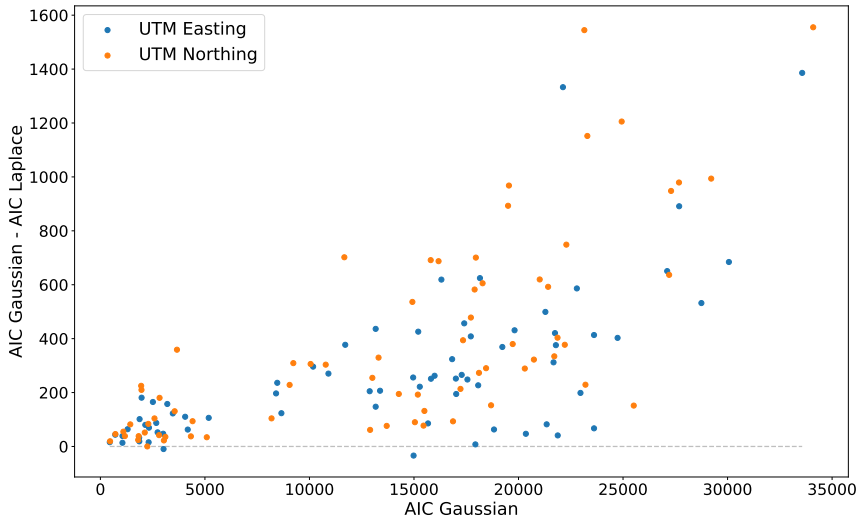


Figure 2.4: AIC comparison for model selection. Each point corresponds the UTM Easting or Northing movement data of a single deer. Each blue point corresponds to a single deer's interpolated UTM Easting AIC scores, and each orange point to its UTM Northing AIC scores. The difference between the AIC for the Gaussian model and the AIC for the Laplace model for each point is shown on the vertical axis, and the Gaussian AIC for each point is shown on the horizontal axis. The dashed grey line indicates where the vertical axis value is zero, i.e., where both AIC measurements are equal. Both models had one parameter; therefore, points above the gray line correspond to data that are fit more accurately with a Laplace distribution. All but two of our data points corresponded to movement data that are fit more accurately with Laplace distributions.

the gradient of an RSF; a smoothing transformation is necessary to accurately incorporate these RSFs into a model. Moreover, such a discontinuous gradient can mask correlations between an animal's positions and an RSF. Using a Gaussian filter, we created smoothed maps; Fig. 2.5 shows a comparison of the original and smoothed RSFs. The large regions of constant values in the original map transition between each other much more gradually, resulting in a smoother gradient.

2.3.2 Examining Movement Relative to the RSFs

Although our RSFs were not intended to inform animal movement, but rather to inform seasonal habitat use and selection of resources given availability in the surrounding landscape [75, 69, 76, 37], we investigated whether correlations between RSF values and deer locations and preferences for moving to relatively higher RSF values exist coincidentally. The results for deer locations are shown in Fig. 2.6 and for deer movement are given in Table 2.1. Note that while there appear to be correlations between the RSFs and deer locations and

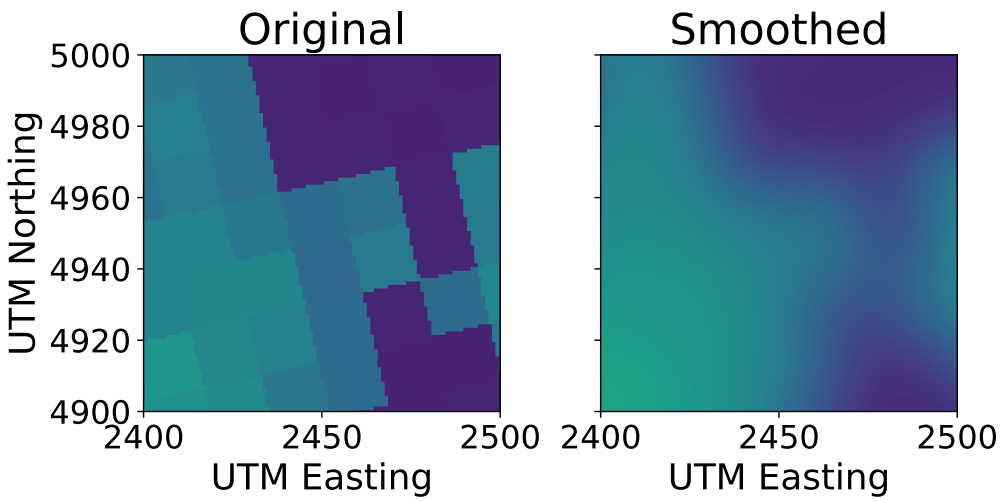


Figure 2.5: Smoothing resource-selection functions (RSFs). On the left, we show a section of the original RSF with its sharp boundaries. On the right, we show the same section after smoothing using a Gaussian filter. This operation created a smoother function. In these plots, lighter colors represent higher RSF values, and darker colors represent lower values. We measured whether animals spent more time at positions with higher RSF values and whether they tended to move towards positions with higher RSF values and compared the results for the original and smoothed maps.

movements, they are not strong but are very sex dependent; the reasons for this are unknown. However, there appear to be no correlations between our RSFs and movement steps, as Table 2.1 shows. Thus, while aspects of the RSFs have potentially useful information, the lack of a clear picture of the role of this data in describing deer movements led us not to incorporate this data into our model development.

Change in value using original map	male	female	Change in value using smoothed map	male	female
higher	27.7%	26.2%	higher	47.6%	48.3%
lower	27.2%	26.4%	lower	47.7%	48.4%
equal	45.1%	47.4%	equal	4.6%	3.4%

Table 2.1: Correlations between deer movements and RSFs. This table shows how often deer moved to locations with higher, lower or equal resource values in the original (left) and smoothed (right) resource maps. These values are calculated using the six RSFs and all deer movement data. Each deer uses the map which corresponds to its gender and the time of year it was tracked. A single deer can use all three seasonal maps for a single gender. The data reveal a lack of preference for movement relative to these RSFs.

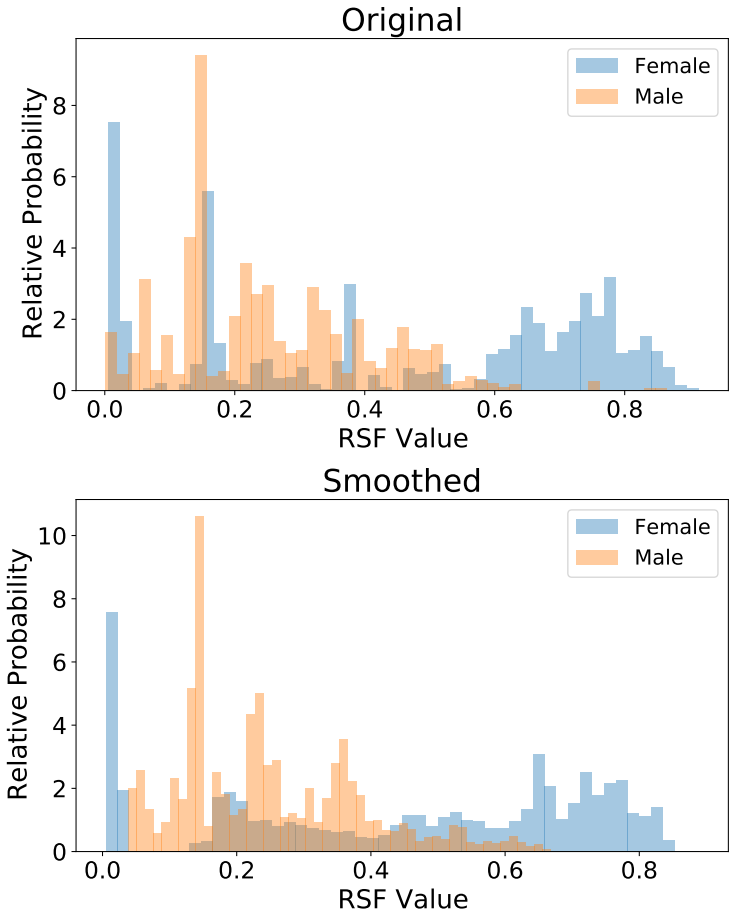


Figure 2.6: Exploration of correlations between deer locations and RSFs. The top histogram shows the probabilities of males and females occupying a location with a given value on the original set of resource maps, and the bottom histogram shows the same probabilities for the smoothed set of resource maps. The results are very similar for the two sets of maps and do not provide evidence that deer locations are correlated with higher values on these resource maps.

2.3.3 Removing Trends From Non-Stationary Time-Series Data

Because the basin-hopping trend was surprisingly uncorrelated with our RSF values, we were unable to remove the basin-hopping trend using these RSFs. We then turned to regression analysis. When a trend has a known functional form (e.g., exponential growth or seasonality), it can be fit and subtracted from the data. In Fig. 2.2, we see that movement trends in the time-series data for deer locations are functionally flat regions connected by discontinuities. The fused-lasso (least absolute shrinkage and selection operator) machine-learning technique [77] allows us to automatically find such a function.

The fused-lasso method aims to find a non-parametric function θ that fits uniformly sampled time-series data y, where y is either of the UTM coordinates with n values (see Fig.

2.2). Because θ can take on any value at any point in its domain, the fused-lasso method first constrains the fit to be close to the data through minimizing $||\theta - y||_2^2 = \sum(\theta_i - y_i)^2$. Letting

$$\mathcal{D} = \begin{bmatrix} 1 & -1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 & -1 \end{bmatrix} \in \mathbb{R}^{n-1 \times n}, \tag{2.3}$$

the regularizer $\lambda||\mathcal{D}\theta||_1 = \lambda \sum |\theta_i - \theta_{i+1}|$ penalizes jumps in the fit θ by pushing the difference in adjacent points in θ to 0. Notice that this regularizer has a different purpose than it would in cross-validation. The strength of the penalty is controlled by the hyperparameter $\lambda \geq 0$. When $\lambda = 0$, there is no penalty for jumps, and the resulting fit θ is equal to the data. As $\lambda \to \infty$, no jumps are allowed, and θ will be the average of the data. Written mathematically, the fused-lasso method aims to minimize the relation

$$\min_{\theta \in \mathbb{R}^n} ||y - \theta||_2^2 - \lambda||\mathcal{D}\theta||_1. \tag{2.4}$$

For a given λ, the quantity in equation 2.4 can be minimized using optimization software; here, we used CVXPY [78, 79]. Next, we consider how to choose a value for λ.

To choose the optimal λ, we need to define the critical jump size, defined as

$$\mathcal{J} = \bar{j} + 3\sigma_j, \tag{2.5}$$

that differentiates between a basin hop and random motion. Here, $\bar{j}$ is the average jump size, and σ_j is the standard deviation of the jump sizes in a trajectory. When optimizing λ, we will compare the jumps in the data, j^y, to jumps produced by the fits, j^θ. With these definitions, we define another loss function to be optimized. This is done by finding the jumps in the data y, and setting to zero the sizes of any jumps that are less than $\mathcal{J}$. This process results in jumps given by

$$j_i^y = \begin{cases} |y_{i+1} - y_i|, & |y_{i+1} - y_i| \geq \mathcal{J} \\ 0, & |y_{i+1} - y_i| < \mathcal{J} \end{cases}. \tag{2.6}$$

Here, the superscript indicates that the jump is calculated from data, and the subscript enumerates the total number of jumps. Our objective is to find the value of λ_{best} corresponding to a fit that best reproduces j_i^y. The optimization in Equation 2.4 is solved for n values of

λ, $(\lambda_1, \ldots, \lambda_n)$ and the jumps it produces,

$$j_i^\theta = |\theta_{i+1} - \theta_i|, \tag{2.7}$$

are compared to j_i^y. We select λ_k and corresponding θ such that

$$||j^y - j^\theta||_2^2 = \sum_i (j_i^y - j_i^\theta)^2 \tag{2.8}$$

is minimized.

Using synthetic data, we illustrate how the fused-lasso approach can be used to detect basin hops in Fig. 2.7. The left panel illustrates the threshold for basin hops, which is set at the mean jump size plus three standard deviations. The middle panel shows that the model, with this choice of threshold, correctly characterizes the largest jumps in the data as basin hops. Finally, the right panel shows the fit (orange line) to the synthetic data (blue dots) and shows that this fit successfully captures the functional form of the trend. To test the robustness of our fitting procedure, we varied the number and sizes of basin hops and the amount of noise. Fig. 2.8 shows this comparison; in each row, one parameter is varied, while the other two are held constant. We found that the fused-lasso method reliably fits basin-hopping in different regimes, and we thus used this method to remove basin-hopping trends from our data.

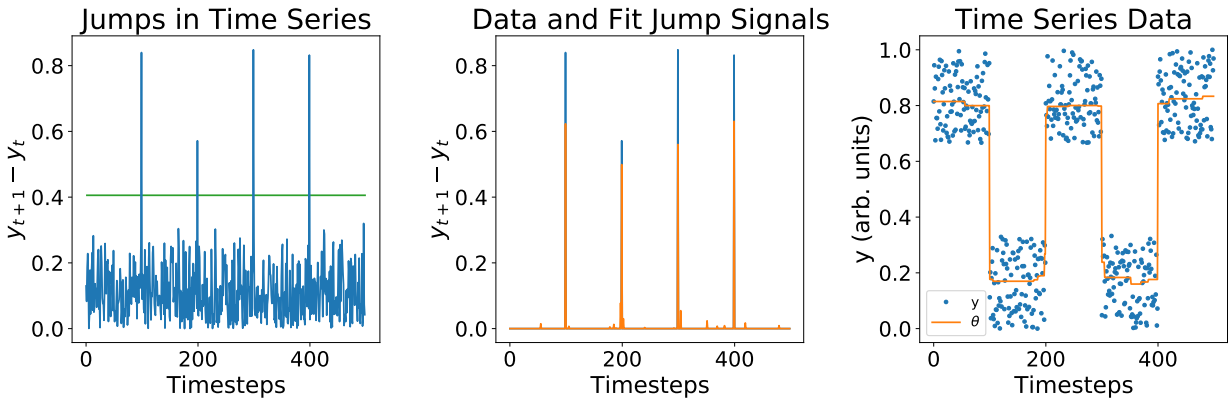


Figure 2.7: Example of fused-lasso fitting. Simulated data is used to illustrate our fused-lasso fitting procedure. In the left panel, we show all jump sizes from the simulated data in blue, and the critical jump size from Equation 2.5 is shown in green. In the middle panel, we show the critical jumps from data in blue, and those from the fused-lasso fit in orange. In the right panel, the simulated location data are shown in blue, with the fused-lasso fit overlaid in orange.

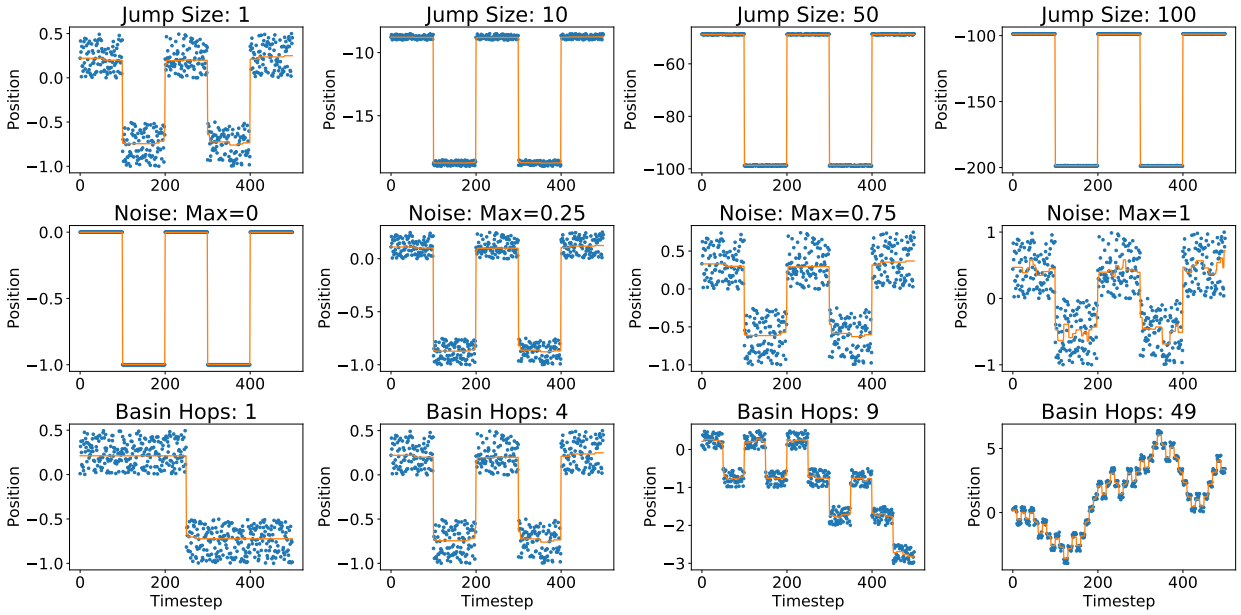


Figure 2.8: Testing the robustness of the fused-lasso method using well-controlled simulated data in arbitrary units. We varied characteristics of simulated time-series data, including the number and sizes of basin hops and the amount of noise, to assess the robustness of our lasso-fitting procedure. In each row, one parameter is varied, while the other two are held constant. In the top row, the basin-hop jump size is varied. In the middle row, the maximum noise value is varied; the noise added in these plots is sampled from a uniform distribution between zero and the maximum value listed in the title above each panel. In the bottom row, the number of basin hops is varied. When not varying, the basin-hop jump size is set to 1, the maximum value for the noise is set to 0.5, and the number of basin hops is set to 4. Our procedure is able to produce realistic fits to the data in many regimes.

2.3.4 Statistical Inference From Stationary Time-Series Data

Beyond visualizing and detrending data, many techniques can be used to mine useful information from time-series data [80]. In this section, we introduce two metrics that we used to analyze time-series data: autocorrelation functions and the mean-squared displacement. Calculating autocorrelation functions generated insights about repetitive motion in our data, though such motion was not a feature we had aimed to reproduce in our model. An examination of mean-squared displacements provided evidence for basins in our data and thus a rationale for including basins in our models.

Time Autocorrelation Function

The first metric we explored was the time autocorrelation function (ACF). ACFs can be used in combination with Fourier transforms for pattern discovery in time-series data [81] to reveal insights and test a model's assumptions [82].

The ACF R measures how correlated a time series $r(t)$ is with itself over a time lag. For discrete location measurements, the ACF can be calculated for a lag $k = t' - t$ as

$$R(k) = \frac{\sum_{i=1}^{N-k}(r_i - \bar{r})(r_{i+k} - \bar{r})}{\sum_{i=1}^{N}(r_i - \bar{r})^2}, \tag{2.9}$$

where $\bar{r}$ is the average position. Using the Fourier transform

$$\hat{R}(\omega) = \int_{-\infty}^{\infty} R(k)e^{-2\pi ik\omega}dk, \tag{2.10}$$

the ACF can be decomposed into its component frequencies. These frequencies measure the time scale of repetitive behavior in the time series. The power $|\hat{R}(\omega)|$ of each frequency ω, can be calculated to determine the contribution of each frequency to the total ACF. In practice, we evaluated a discrete Fourier transform using the fast Fourier transform algorithm. For many deer, we found that the ACF of their positions primarily took the form of exponential decay, with high- and low-frequency periodic signals superimposed. Fig. 2.9 shows an example of the ACF and corresponding power spectra, which highlight the high and low frequencies in the movement data. The high-frequency peak corresponds to daily repetition in activity, which is likely circadian in origin. The low-frequency peak consists of fewer power spectral data points and hence is not as statistically significant.

The method presented above assumed that the data were uniformly sampled. Alternative approaches, such as those employing the Lomb-Scargle periodogram [83, 84], allow irregular data to be treated directly without interpolation. However, because interpolation was required for other methods used in this work, we implemented the fast Fourier transform method in our analysis.

Mean-Squared Displacement

The second metric we examined was the mean-squared displacement (MSD). The MSD is defined as $E[|r(t) - r(0)|^2]$. The MSD measures how far an animal moves on average and can be used to classify diffusion by the power by which the MSD increases versus time. In Fig. 2.10, we examined the MSD of our stationary (within-basin movement) and non-stationary (basin-hopping) data. In the non-stationary data, there was no clear relationship between time and the MSD. In addition, basin hops caused large shifts in the MSD. After removing the basin-hopping trend, the MSD was constant in time, as expected. A constant MSD results when an animal remains in one region; for example, the MSD is approximately constant if an animal remains in a basin.

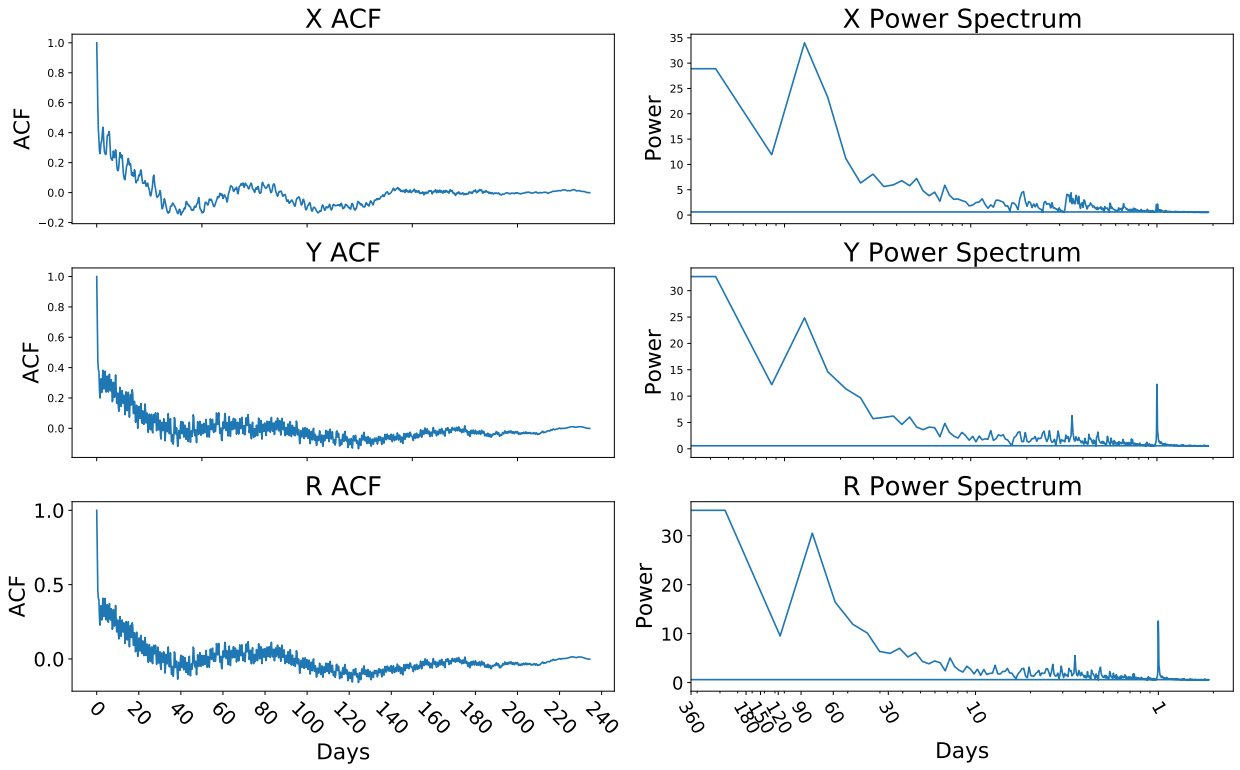


Figure 2.9: Time autocorrelation functions and their Fourier transforms. ACFs of a deer's motion in the x, y, and $r = \sqrt{x^2 + y^2}$ directions are shown in the left column. In the right column, we show the corresponding power spectra of these ACFs. The high-frequency oscillation in the ACFs is seen as a daily peak in the power spectra, and the lower-frequency oscillation is captured by a wider peak at around one to three months.

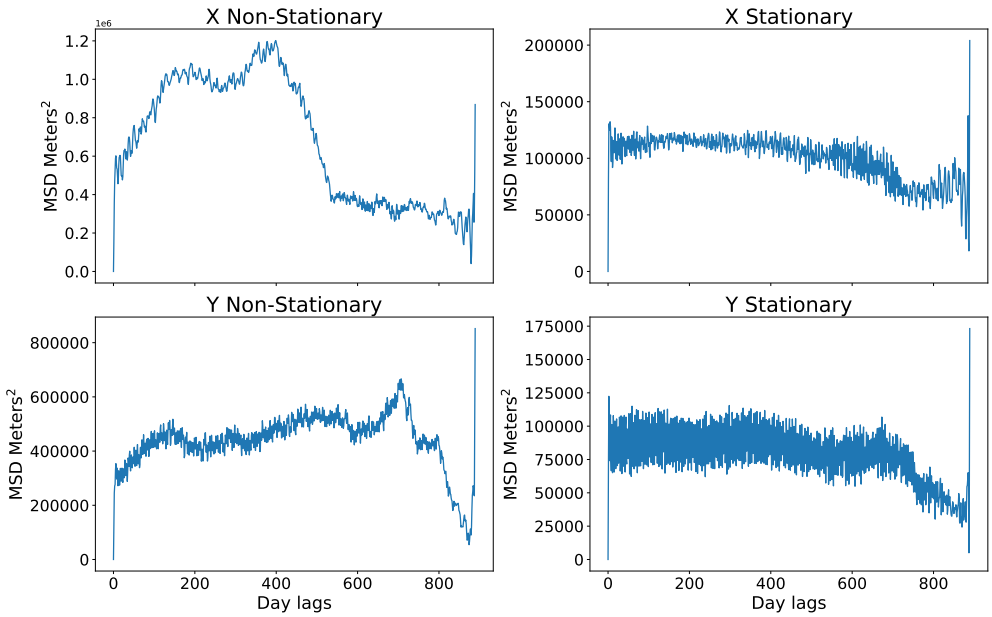


Figure 2.10: Mean-squared displacements. We show the mean-squared displacements for our non-stationary data in the left column and for our stationary data in the right column. In neither case is there an obvious t^a dependence; moreover, the stationary data reflect a "non-diffusing" deer, as evidenced by the mean-squared displacement being constant over all but the longest lags.

2.3.5 Recovering Correlations in Position Data

The above EDA explored our data in each coordinate separately. In doing so, we assumed that the data in each coordinate were independent and imposed an importance on the coordinate system that our data happened to be measured in. For example, by rotating the coordinates such that a basin hop is perpendicular to one axis, a trend that was observed in both coordinates before rotation would be observed in only one of the new, rotated coordinates. Changing coordinates would affect our fused-lasso regression analyses, jump distributions, and other metrics examined in our EDA. Therefore, it is important to ensure one coordinate system is used consistently.

One way to remove the assumption of independence between coordinates is through correlated random walks [85]; however, these models assume that the resulting bivariate jump distribution is normally distributed, something we did not observe in our data. To preserve correlations in the data and to avoid imposing a form on the jump distributions that we pick, we used two methods to estimate bivariate distributions from the data directly: copulas and KDEs. Copulas allow us to generate a bivariate distribution from the marginal distributions, which we know well from our stationary data. Jumps in the trends do not follow an obvious distribution, so we use KDEs to approximate these instead.

Copulas

Animal locations are often recorded in arbitrary coordinates that are useful for modelers. There is no reason to assume that motion in these directions should be independent. Therefore, any distributions stemming from data in these coordinates should be bivariate to account for correlations. When the marginal distributions are well known, copulas provide a way to estimate a bivariate distribution while preserving the observed marginals. A copula $C(\cdot)$ is a function that joins a multivariate distribution function $F_{UV}(u, v)$ to its one-dimensional marginals $F_U(u), F_V(v)$ [86] through the relation

$$F_{UV}(u, v) = C(F_U(u), F_V(v)). \tag{2.11}$$

Using copulas, one can construct a joint distribution of two variables while knowing only the individual marginal distributions [87]. Thus, we can estimate the bivariate jump distribution for our data from the one-dimensional distributions we observed. Copulas have been used in ecological studies to accommodate under-reporting in wildlife-vehicle crash data [88], to approximate joint space use among animals [89], and to describe distributions of multiple species of animals [90].

Mappings from the marginals to the joint distribution function are not unique; a C must be chosen for each application. While there are many options for creating copulas [86], for convenience, we employed the Gaussian copula by allowing C to be a multivariate Gaussian, to model jumps in our stationary data (data with trends removed).

In Fig. 2.11, we graphically show an example of the procedure we employed to generate a bivariate jump distribution using a copula. First, for each individual deer, we identified jumps in their separate UTM coordinates. We considered only jumps that happened simultaneously in both coordinates, and we treated jumps that happened simultaneously in both coordinates as single, two-dimensional data points. The resulting two-dimensional distribution of within-basin jump data was fit with a zero-mean bivariate Gaussian distribution, as shown in the upper-left panel of Fig. 2.11. The marginals of these data are also shown in the upper-left panel of Fig. 2.11. This bivariate Gaussian distribution was then sampled for illustration purposes; the resulting data, together with their associated marginals, are shown in the upper-right panel of Fig. 2.11. The correlations from our data in the upper-left panel are preserved here, but the marginals which were Laplace distributions in the upper-left are Gaussian distributions in the upper-right. Using the probability integral transformation, the data from the bivariate Gaussian distribution were transformed to have uniform distributions, while retaining correlations; the resulting uniform, bivariate distribution, shown in lower-left panel of Fig. 2.11, is the copula of the two jump-coordinate variables of the

fitted bivariate distribution shown in the upper-right panel of Fig. 2.11. Finally, the copula was used together with the observed Laplace-distributed marginals to create a non-Gaussian bivariate jump distribution with marginal distributions that were constructed from our data. This bivariate jump distribution is correlated, has marginals that match our data, and can be sampled efficiently.

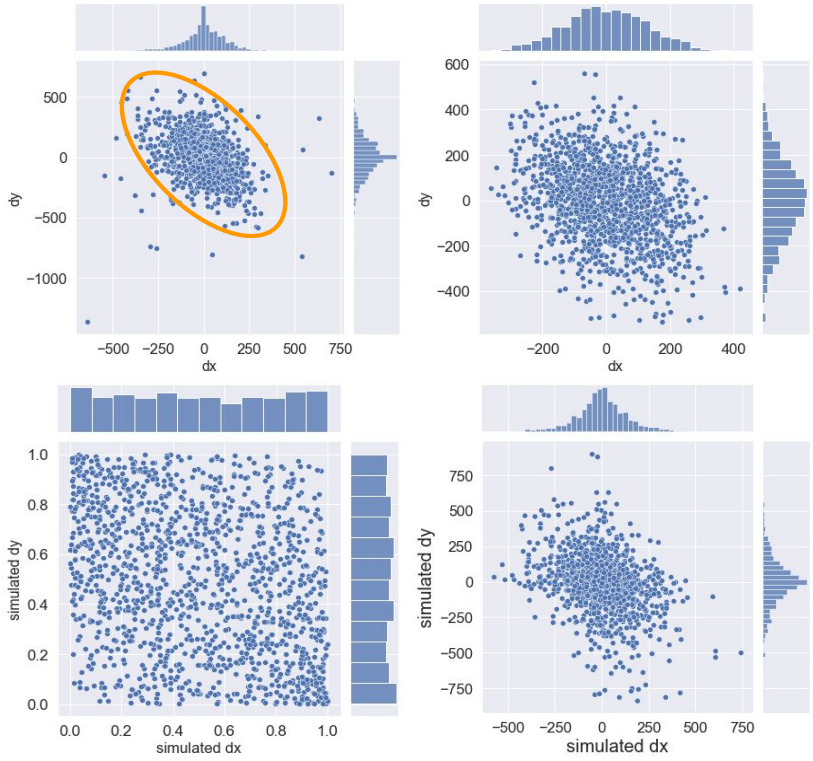


Figure 2.11: Illustration of a procedure for generating a bivariate probability distribution from one-dimensional marginal distributions using a copula. In the upper-left panel, jump data for an individual deer are shown, and the marginals from these data are shown on the top and right sides of the main figure in the panel. A bivariate Gaussian was fit to the marginals data, and this fit is indicated by the orange oval. In the upper-right panel, points sampled from the the bivariate Gaussian fit are shown, together with their marginals. Using the probability integral transformation, the bivariate Gaussian data and marginals were transformed to uniform data and marginals, while retaining correlations; the resulting uniform, bivariate distribution is the copula and is shown in the lower-left panel. Finally, the copula was used with the observed Laplace-distributed marginals to create a non-Gaussian bivariate jump distribution whose marginal distributions are constructed from our real data.

Kernel-Density Estimates

The power of copulas comes from their ability to constrain the marginal distributions of an approximate bivariate distribution by exploiting known marginal distributions. When

known marginal distributions are unavailable, a KDE can be used to estimate a multivariate distribution $\hat{f}$. KDEs have been used in animal ecology studies in methods for estimating home ranges for animals [91, 92].

To construct a KDE, we consider a probability density kernel K over each observed point x and create a weighted average of the distances between the current observed point and all other points in the dataset X_i, through

$$\hat{f}(x) = \frac{1}{nh^2} \sum_{i=1}^{n} K \left\{ \frac{(x - X_i)}{h} \right\}. \tag{2.12}$$

Here, h is a smoothing parameter that controls how closely the output resembles the kernel used. In our application, each x is a two-dimensional point that describes the size of a basin hop in each coordinate.

We used KDEs to estimate the bivariate jump distribution of the trends removed from our movement data. First, we identified the jumps in the trend. In some cases, the jumps were not aligned in the separate coordinates, and a smoothing procedure was therefore used to correct for slight misalignments within the data. We approximated the bivariate jump distribution of the trend using a Gaussian KDE with $h = \bar{\Delta r}/4$ meters, where $\bar{\Delta r}$ is the average distance between points. This value of h was chosen to yield a smooth distribution function without losing details present in the original dataset.

In summary, throughout our EDA, we explored many aspects of our data to illuminate features that then guided the selection of the form of our model. Visualizing our data informed us of the basin-hopping features in our data and demonstrated that jumps in the data are not normally distributed. We found that correlations between our movement data and our RSFs are weak, and we have thus excluded RSF data from our model.

When examining the MSDs of our data, we found additional evidence of basins in which deer locations were constrained. Using these findings, in the next section, we propose a model that can reproduce the features of the data revealed in our EDA.

2.4 Modeling

As part of the process of constructing an ABM, we first developed a mathematical model of a single deer's movement that was consistent with our EDA; this model was then discretized to form the basis of our ABM. This intermediate step, of generating a model for an individual deer's movements before formulating an ABM, allows us to exploit the EDA performed above.

Our EDA results suggest that a stochastic drift (i.e., basin-hopping), non-Gaussian random-walk model would capture important features of our data well; hence, we chose

to develop a model of this form as our single-animal movement model. Our starting point was the usual Langevin equation, which is usually written in terms of particle velocities as

$$\frac{dv}{dt} = -\gamma v(t) + \psi(t). \tag{2.13}$$

Our modification of the Langevin model constructs a random process directly from the data, using a copula to approximate a bivariate distribution of jumps. This results in a correlated random-walk model with non-Gaussian noise. Moreover, we include basins in our model through the use of a harmonic potential to constrain deer within a region. To simulate the basin-hopping trend, basins move according to a Poisson process; this choice will be discussed in the subsection below.

While our Langevin model is applied to a single animal and misses the effects of heterogeneity, these effects can be incorporated into an ABM extension of the Langevin model. An ABM can model multiple individuals with different ages and sexes and can integrate other data streams, such as knowledge of the locations of bodies of water and impenetrable objects. Additionally, other features, such as a disease model, are readily incorporated into an ABM to study disease transmission, without the need to rederive the movement model. Our goal is for our ABM to serve as a foundational framework into which additional wildlife models can be incorporated; such a model will facilitate the study of wildlife phenomena in which movement plays an important, underlying role. In the next two subsections, we describe the formulation of our Langevin model and then its extension to an ABM.

2.4.1 Single-Animal Langevin Dynamics on a Potential-Energy Surface

We develop the stationary and non-stationary components of our Langevin model of a single animal's movements separately. We will first discuss the stationary portion of our Langevin model. Consider a single tagged animal at a position (x, y) and moving in a two-dimensional space. The lack of a trend in the MSD data suggests that deer remain localized in a basin with center (c_x, c_y) that acts as a potential energy surface, which we model as

$$U = \frac{(x - c_x)^2}{a_x} + \frac{(y - c_y)^2}{a_y}, \tag{2.14}$$

where a_x and a_y describe the widths of the basin. The stationary movement within the basin is modeled by

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = 2 \begin{bmatrix} \gamma_x a_x & 0 \\ 0 & \gamma_y a_y \end{bmatrix}^{-1} \begin{bmatrix} x - c_x \\ y - c_y \end{bmatrix} dt + \begin{bmatrix} \frac{\lambda_{xx}}{\gamma_x} & \frac{\lambda_{xy}}{\gamma_x} \\ \frac{\lambda_{yx}}{\gamma_y} & \frac{\lambda_{yy}}{\gamma_y} \end{bmatrix} \begin{bmatrix} d\Psi_x \\ d\Psi_y \end{bmatrix}, \tag{2.15}$$

which is analogous to a harmonically trapped particle obeying the Langevin equation [93]. The first term is the gradient of equation 2.14 and models the effects of the basin, and the second term contains all of the correlations and describes the random motion of an animal.

Next, we model the animal's non-stationary basin-hopping. We assumed that basin hops were independent events that occurred at distinct, random times, and we calculated the average rate of basin hops from the data. Our assumptions are consistent with a Poisson process; thus, we used a Poisson process to model basin-hopping behavior. In a Poisson process, the inter-arrival times t between k basin hops are described by an exponential distribution

$$P(\mathbf{c} \text{ moves at t}) \sim \frac{k}{T} e^{-(k/T)t}, \tag{2.16}$$

where T is the total number of timesteps at which the movement of an animal is observed, and k/T is the average rate of basin hops. The cumulative sum of the inter-arrival times provides the times at which a basin moves instantaneously. The sizes of the basin hops are sampled from a bivariate distribution constructed from the removed trends using a KDE.

The stationary and non-stationary components of our model are fit separately. In order to fit the stationary component described by Equation 2.15, we first need to remove the effects of the basin from our stationary data. We do not know the basin's parameters a priori, so we choose many possible sets of basin parameters that are then used to subtract the basin from our stationary data. In practice, we let $C_{1x} = 2/\lambda_x a_x$ and $C_{1y} = 2/\lambda_y a_y$ and perform a grid search over values for C_{1x} and C_{1y}. Once the basin is removed, we fit a Gaussian copula, using the method described in Section 2.3.5, to the resulting data to capture the parameters and random processes in Equation 2.15. To determine the best set of C_{1x} and C_{1y}, we run our model 10 times for each set of parameters and corresponding random process fit, and we select the parameters that minimize the least-squares differences in the variances and covariances of locations between the simulated location data and real location data.

To fit the parameters of the non-stationary Poisson process, we set k to the number of critical jumps observed in a trajectory; then T/k is the ratio of the total number of timesteps to the number of critical jumps. Each of these parameters, as well as the removed trends, were stored for each deer. Using the method described in Section 2.3.5, we construct a bivariate jump distribution from the removed trend with a Gaussian KDE, from which we sample basin hops.

After fitting the parameters of our Langevin model to all of our deer, we simulated a trajectory by selecting a set of parameters for a single deer and the corresponding trend and then running our model. An animal was given an initial position $(x(0), y(0))$, and the basin center (c_x, c_y) was chosen. The entire stationary component of the trajectory was simulated

using

$$\begin{bmatrix} x(t+1) \\ y(t+1) \end{bmatrix} = \begin{bmatrix} x(t) - C_{1x}(x(t) - c_x) + K_x \\ y(t) - C_{1y}(y(t) - c_y) + K_y \end{bmatrix}. \tag{2.17}$$

This equation is a discretized version of equation 2.15. Here, K represents a random number sampled from the bivariate jump distribution constructed using a Gaussian copula. After simulating the stationary component, we simulated a trend. This was done by sampling the times at which basin hops occur, using Equation 2.16.

We then sampled the basin-hop sizes from the KDE constructed from the removed trend jumps (i.e., the basin hops) and added the simulated trend to the simulated stationary trajectory. With this model, we are able to simulate the trajectory of a deer with movements that follow patterns seen in the original dataset. In the next subsection, we extend this single-animal model to a multiple-animal ABM.

2.4.2 Multiple-Group Agent-Based Model

We began with a Langevin model that treated a single deer at a time because the data that we have tracks individual deer. However, in practical applications, we require models of multiple groups with population diversity. Moreover, many environmental and social effects are implicitly contained in the harmonic potential used to constrain deer within a basin. While we would prefer to incorporate these effects explicitly in our models, our ability to do so is limited without additional data. Nonetheless, to move toward a multiple-animal simulation, we assumed that a number of non-tracked animals were together with the tracked animals in a basin and that these non-tracked and tracked animals formed a group that underwent basin hops together.

Here we provide a simplified description of our model. For a full description following description following the Overview, Design concepts and Details (ODD) protocol [94], see Appendix A. Our ABM consists of group designations (collectives within the population) and deer. The simulation contains a number of groups, each of which is initialized with a shared basin center, a number of deer, and a set of parameters and trends trained from a tracked deer. Based on their parameters, groups determine when they will undergo a basin hop and how many basin hops they will undergo. At every step of the simulation, the deer in each group update their positions (x, y) individually using equation 2.17. Each group then determines whether the current timestep requires a basin hop. If so, the group samples a jump size from the KDE fit to the trend data and moves the center of the basin along with its deer according to the sampled jump. Our ABM can be used as a foundation for models that involve deer movement. For example, a disease or deer-management model could be added to the ABM to study additional phenomena while allowing for deer movement that is consistent with real movement data; this will be the subject of future work.

2.5 Results

One of our goals was to demonstrate the importance of EDA in developing models. We compare our Langevin single-deer model (data and basin-hopping; DBH) with three other models that use different assumptions. The first two single-deer models are random-walk models without basin hopping: one with Gaussian noise (Gaussian random-walk; GRW), and the other with non-Gaussian noise constructed from the location data via a copula (data and random-walk; DRW). These models ignore trends in the data; such models might be selected if the data were not visualized and the basin hops were not observed. The third model incorporates basin hopping and employed Gaussian noise (Gaussian and basin hopping; GBH). These models are not an exhaustive list of alternatives, but they illuminate how assumptions made about trends and types of noise can affect the accuracy of movement models. We additionally present example results from our multiple-deer ABM that display how groups of deer can come into contact.

The models compared to our Langevin model were fit using methods similar to those used for our Langevin model. For models without basin hopping, a zero-mean Gaussian distribution was fit to the bivariate jump distribution constructed from the non-stationary data. The GRW model sampled this fit directly, while the DRW model passed the same jumps through a Gaussian copula to simulate a deer's jumps. Models with basin hops fit a Gaussian distribution to the bivariate jump distribution of the stationary data. The GBH model sampled this fit directly, but the DBH model passed the jumps through a Gaussian copula, so that its jumps were sampled from a Gaussian copula-based distribution. Both the GBH and DBH models sampled a Gaussian KDE, as described in section 2.4, to simulate a trend that was used for both models.

We compared the four individual-deer models by training them on a single deer in our dataset and performing 100 simulations using each model. Both the GRW and DRW models appear to capture correlations in the location data, as evidenced in Fig. 2.12 by the similarity of the orientations of the distributions to those of the data. The basin-hopping models produce very similar distributions in Fig. 2.13 because they share the same basin-hopping trend in each of the 100 trials, and this trend represents larger jumps than are seen with the underlying random motion.

Next, we further examined the differences between models that included jumps sampled from a Gaussian distribution and those that included jumps sampled from a copula-based distribution generated using a Gaussian copula. In Fig. 2.14, we show the counts of positions of a deer moving with Gaussian-distributed jumps within a basin (reds) superimposed on the counts of positions moving with copula-based-distributed jumps (blues). Blues in Fig. 2.14, corresponding to positions moving with copula-based-distributed jumps, are

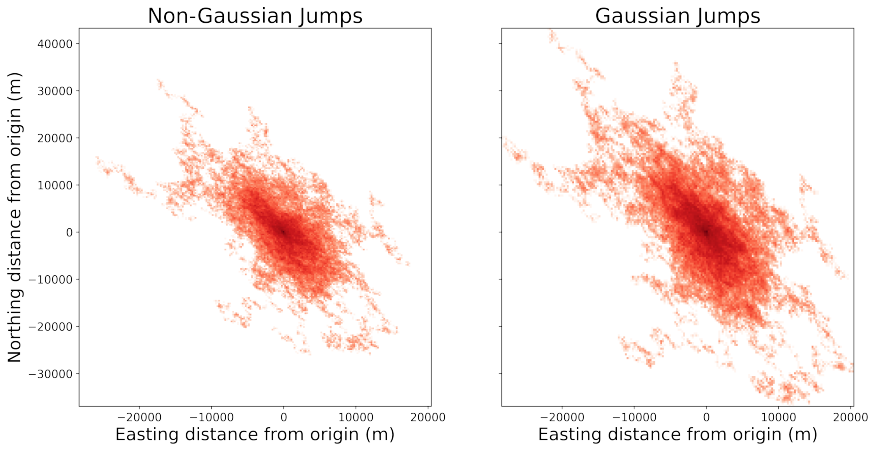


Figure 2.12: A comparison of trajectories for two random-walk models that do not include basin hops. We compare how often locations were visited using a random-walk model that uses jumps sampled from a bivariate non-Gaussian distribution constructed using a Gaussian copula trained on our data, shown in the left panel, with a random-walk model that samples jumps from a Gaussian distribution fit to our data, shown in the right panel. Each model was used to generate 100 trajectories, and the densities of the locations visited are shown.

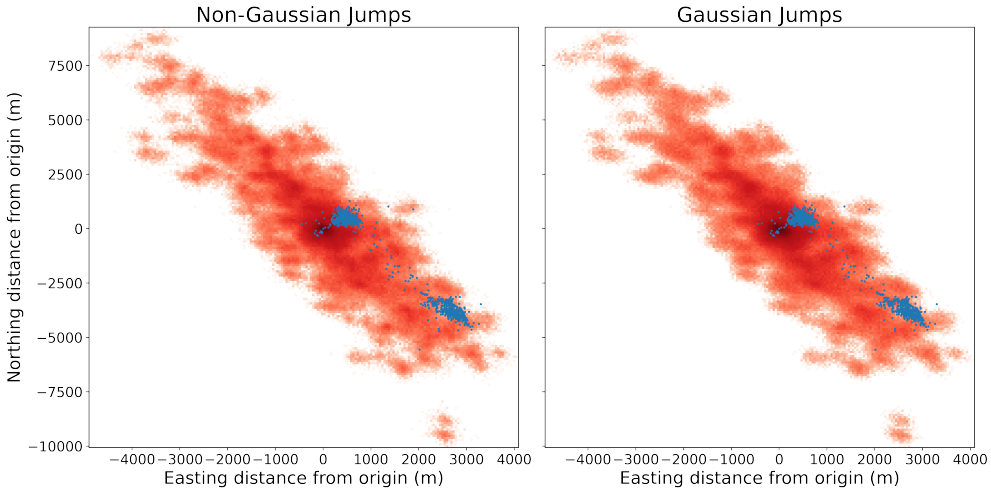


Figure 2.13: A comparison of trajectories for two random-walk models that include basin hops. We compare a random-walk model using jumps sampled from a bivariate non-Gaussian distribution constructed using a Gaussian copula trained on our data, shown in the left panels, with a random-walk model sampling jumps from a Gaussian distribution fit to our data, shown in the right panels. Both models include the same basin-hopping trend produced by sampling a KDE trained on our data. Each model was used to generate 100 trajectories, and the densities of the locations visited are shown. The blue dots show the data on which the models were trained.

spread out over a larger area than reds in the figure, corresponding to positions moving with Gaussian-distributed jumps; i.e., the deer movements modeled using a copula-based distribution covered a larger area than those modeled using Gaussian-distributed jumps.

To further illustrate how these two models differ from each other, in Fig. 2.15, we show the logarithms of the magnitudes of the differences between these two sets of position counts; the position counts generated from Gaussian jumps (reds in Fig. 2.14) are subtracted from the position counts generated from copula-based jumps (blues in Fig. 2.14). Short-distance movements are reflected in the center of Fig. 2.15, and the green color at the center of the figure indicates that far more short-distance movements occur with copula-based-distributed jumps than with Gaussian-distributed jumps. The yellow points away from the center of the figure indicate that occasional larger jumps are also more common among the copula-based-distributed jumps than among the Gaussian-distributed jumps. Intermediate distances, shown as red points, were less common among the copula-based-distributed jumps than among Gaussian-distributed ones.

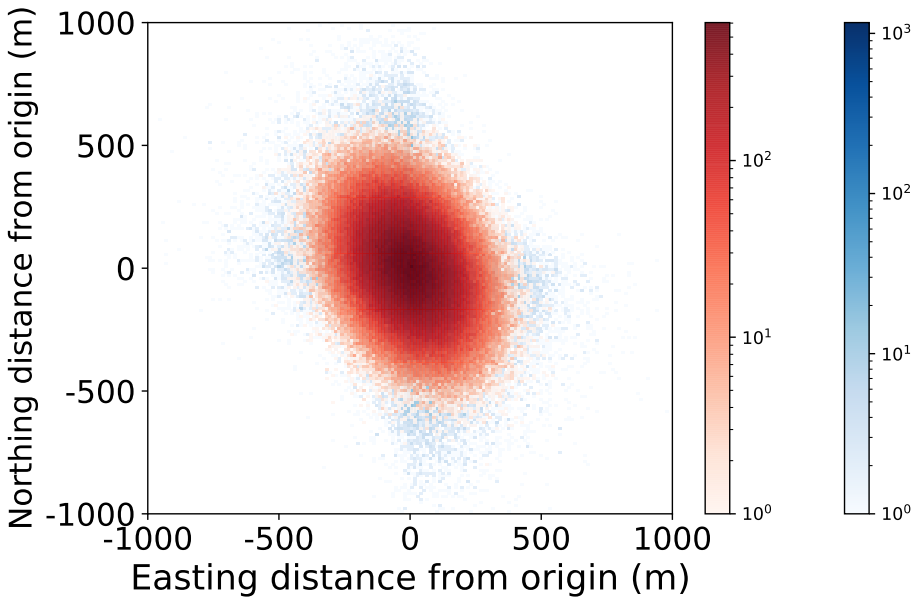


Figure 2.14: Densities (counts) predicted by both copula-based- and Gaussian-distributed noise in a basin centered at the origin ($c_x = c_y = 0$). Density predictions from our Langevin model for both Gaussian and copula-based noise, using parameters from a deer in our dataset, are shown. The deer density for the copula-based model is shown in blues; over that density is the density from the Gaussian-based model in reds. It is evident that more distant locations can be reached using a copula-based noise than with a Gaussian noise.

After examining how different sets of assumptions affected the movement patterns of our Langevin-based model, we ran our ABM to explore the behaviors of multiple deer. First, to

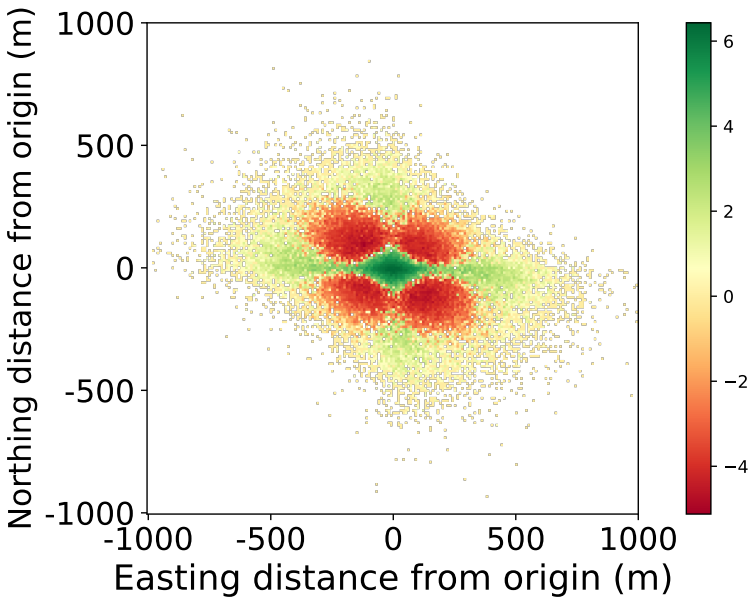


Figure 2.15: Difference in deer densities from Fig. 2.14. The log of the magnitude of the differences is reported. When using copula-based-distributed movements instead of Gaussian, there is a large correction for smaller-sized jumps that allow for the occurrence of rare, very large movements. Further, when using copula-based-distributed jumps, there are fewer movements to intermediate-range locations compared to Gaussian-distributed jumps.

construct our multiple-deer model, we fit each deer's data to our Langevin model, creating distributions of parameters. By sampling these distributions, we created an ABM of three groups, each containing five deer. In Fig. 2.16, each group's approximate density is outlined with contours in a unique color for each group, with the initial center of the group marked with an 'x' of the same color. The blue group underwent a basin hop into the region occupied by the green group. This is an example of a behavior that may have been missed using a model without basin hops. In the context of disease management, this behavior could allow for the transmission of a disease between two groups.

2.6 Conclusion

We presented an EDA approach as the starting point for model building that resulted in less-biased models. Our approach was applied to a real dataset consisting of GPS deer locations and RSFs. The use of an EDA provided a formal testing ground for exploring our data and testing model hypotheses. For example, we found weak correlations between our movement data and RSFs, leading us not to incorporate our RSFs in the construction of our models. It is possible that different RSFs, perhaps constructed in a data-driven way to account for observed animal movements [51], could aid the development of improved, future models.

Our initial EDA identified important features of our movement data, namely that there

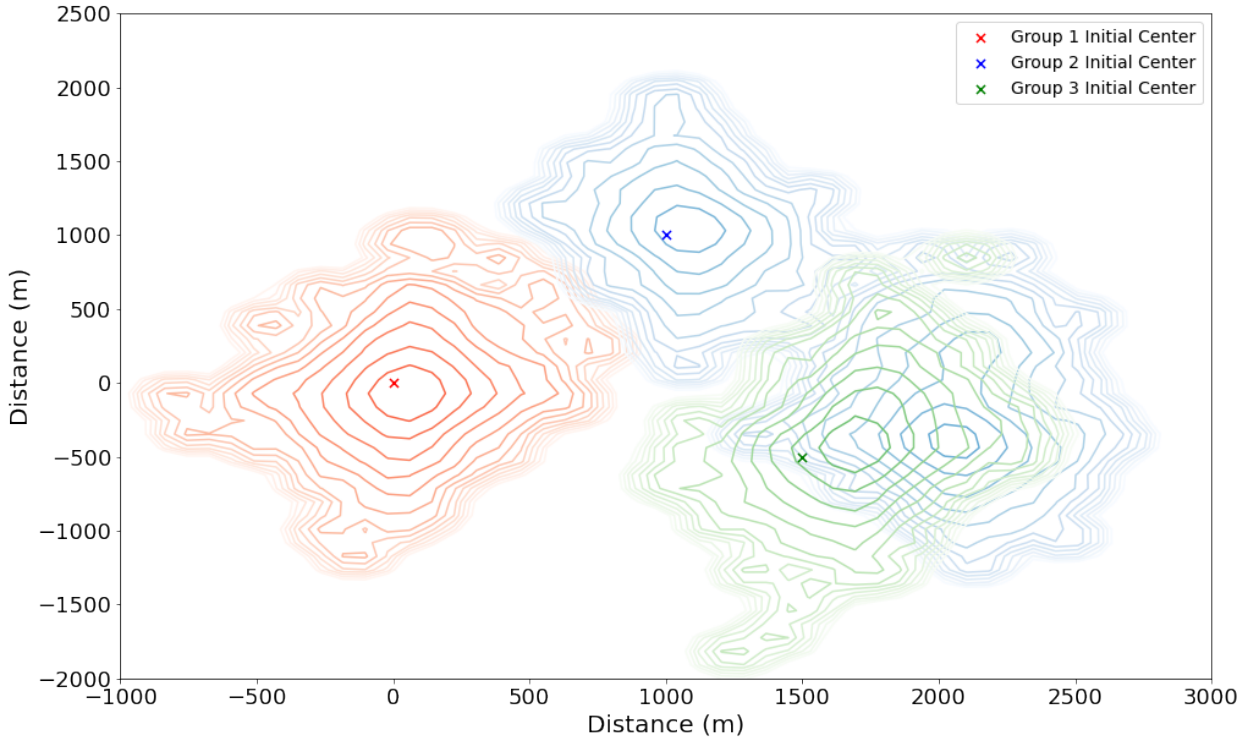


Figure 2.16: Sample multiple-deer agent-based model output. The results of a simulation of the movements of three deer groups (red, blue and green), each with five deer, are shown. The initial location of the center of each group is indicated by a colored 'x'. In this sample, the blue group underwent a basin hop into the region occupied by the green group.

were disparate geographic regions in which animals spent time, and that animals hopped between these regions. We referred to these features as basins and basin hops, respectively, and our models would need to be able to reproduce them. Further exploration of the data revealed that jumps within these basins do not follow a Gaussian distribution.

In our IDA, we identified the non-stationary basin-hopping trend that we attempted to remove from our data with the use of our RSFs. When we found that the movement data and the RSFs were only weakly correlated, we turned to the fused-lasso machine-learning method to remove trends from our data. This method successfully and automatically detected trends in our data and was used to split our data into stationary and non-stationary components.

Beyond our initial data analysis, we examined ACFs and their Fourier transforms, which highlighted behaviors on multiple time scales. We also calculated MSDs, which provided additional evidence for the presence of basins that constrained deer movements. We then used the insights gained from performing our EDA to develop a model of deer movement. We began with the intermediate step of building a model for a single animal and then extended this model to a multiple-animal ABM.

Our EDA results suggested that a random-walk model that featured a copula-based jump distribution would capture important features of our movement data. Additionally, the model would require terms that constrain deer to a basin but allow for the basin to move intermittently. We created such a model for a single deer and compared it with three other models that employed assumptions that could be made in the absence of EDA. We found large differences in the amount of area covered by deer in each model.

Our single-deer Langevin model was expanded to a multiple-deer ABM that featured multiple groups of deer. Our ABM was able to reproduce features of deer movement that our Langevin model could not. In particular, our ABM simulations occasionally revealed two deer groups moving into the same region. The ability to model such phenomena can be important for modeling the spread of disease.

In future work, we will extend our movement ABM to model the spread of chronic wasting disease in white-tailed deer. This disease requires accurate modeling of movement and space use to understand its transmission and geographic spread.

CHAPTER 3:
A GLOBAL SENSITIVITY ANALYSIS OF AN AGENT-BASED MODEL OF CHRONIC WASTING DISEASE

3.1 Summary

The following chapter builds upon the work discussed in Chapter 2, and is derived from a paper currently under review, for which I am a coauthor [95]. This work extends the animal movement model described in [30] by incorporating deer ecology and chronic wasting disease (CWD) dynamics. The refined model produces both realistic population dynamics of Midwestern white-tailed deer, and CWD dynamics that are consistent with field observations from Wisconsin. This allows for predictions on the progression of CWD and its impacts on white-tailed deer populations.

There is a large uncertainty in many of the parameters used in modeling CWD. Because of this, a comprehensive sensitivity analysis was performed. Such an analysis not only allows for uncertainty in the results to be reported, but also allows the relative importance of parameters to be ranked. My main contributions in this project were aiding in the incorporation of population and disease dynamics, and implementing the sensitivity analysis.

3.2 Background on sensitivity analysis

All models inherently have uncertainties associated with them; such uncertainties may arise for many reasons, such as unknown initial conditions, variation in parameters, or uncertainties in a model's structure [96]. Regardless of the form of these uncertainties, sensitivity analysis is a crucial tool for evaluating the influence these uncertainties have on the variability of a model's outputs [97]. In this chapter, we focus on uncertainties related to model parameters. Broadly, sensitivity analyses are categorized into local and global types [98]. Local sensitivity analysis explores sensitivity near a nominal point, while global sensitivity analysis expands this to the entire sample space, including interactions between inputs [98, 99, 100, 101, 102].

The first step of any sensitivity analysis is identifying the quantities of interest, denoted as Q. These are the metrics that evaluate the model's performance, and can be direct outputs from the model or derived from post-processed model data. Q is considered a function of the p input variables $\mathbf{x} = [x_1, \ldots, x_p] \in \mathbb{R}^p$. This input vector is a random vector, where each element is sampled from a corresponding probability distribution.

In local sensitivity analysis, Q is expanded around a nominal point $\bar{\mathbf{x}}$. While the nominal point can be any value, commonly it is chosen to be the mean value of the input parameters

[96]. Following the notation of [96],

$$Q(\mathbf{x}) = Q(\bar{\mathbf{x}}) + \sum_{i=1}^{p}(x_i - \bar{x}_i)\frac{\partial Q}{\partial x_i}\bigg|_{\bar{\mathbf{x}}} + \sum_{i=1}^{p}\sum_{j=1}^{p}\frac{(x_i - \bar{x}_i)(x_j - \bar{x}_j)}{2}\frac{\partial^2 Q}{\partial x_i \partial x_j}\bigg|_{\bar{\mathbf{x}}} + \ldots, \tag{3.1}$$

where only the first few terms of the expansion are written. The second term of the expansion consists of p derivatives of Q. Each of these derivatives describe the change in Q near $\bar{\mathbf{x}}$ due to a small change in x_i. These derivatives are referred to as first-order sensitivities. It is worth noting, however, that they only measure sensitivity near $\bar{\mathbf{x}}$, are linear approximations, and ignore interactions among inputs. To ensure comparability across different units, sensitivities must be scaled. Two scaling approaches are to multiply the i^{th} sensitivity by the mean or standard deviation of the i^{th} element of $\bar{\mathbf{x}}$, resulting in the scaled sensitivity coefficient and sensitivity index, respectively.

A global sensitivity analysis can remedy some of issues of a local one. Specifically, the full input space can be explored, the linearity assumptions are eased, and interactions between inputs can be examined. However, a global sensitivity analysis is often much more involved to perform compared to a local one [97, 101, 98, 102]. Global sensitivity analysis methods fall under broad categories, such as regression-based, variance-based, and density-based [103]. In this chapter we will focus on variance-based global sensitivity analysis [104].

Variance-based methods rely on decomposing the variance of the quantities of interest as

$$var(Q) = \sum_{i=1}^{p} D_i(Q) + \sum_{i<j}^{p} D_{ij}(Q) + \ldots + D_{12\ldots}D(Q), \tag{3.2}$$

where $D_i = var\left[\mathbb{E}(Q|x_i)\right]$, $D_{ij} = var\left[\mathbb{E}(Q|x_i, x_j)\right] - D_i(Q) - D_j(Q)$, and so on [101]. From this decomposition, the sensitivity indices

$$S_i = \frac{D_i(Q)}{var(Q)} \text{ and} \tag{3.3}$$

$$S_{T_i} = \sum_{l \in \mathcal{I}} S_l \tag{3.4}$$

are calculated, where $\mathcal{I}$ is the set of all subsets that include i (e.g. i, ij with $j \neq i$, and so on). S_i is referred to as the first-order sensitivity coefficient, and S_{T_i} is the total effect index [105, 104, 106]. The first-order sensitivity coefficient describes the amount of variance in Q due to the input x_i individually, while the total effect index describes the variance in the output that can be accounted for by x_i, including interactions among other inputs. An overview of performing a variance-based global sensitivity analysis is discussed in [104].

This method relies on generating sequences of quasi-random numbers, referred to as Sobol sampling. Each sequence consists of $p+2$ points in $\mathbb{R}^p$, where p is the number of inputs to the model. The first two points, A and B, are generated by independently sampling each of the p parameters. Then, an additional p points are generated by combining components from A and B, these points are denoted as $Ab_1, \ldots, Ab_p$. A point Ab_i is generated by replacing the i^{th} component of A with the i^{th} component of B.

As discussed, it is typically more expensive to perform a global sensitivity analysis compared to a local one, but global analyses are more informative. Mixed methods can alleviate the issue of computational expense [97] by ranking the importance of inputs using a computationally inexpensive screening method. This ranking is then used to select a subset of parameters to include in the global sensitivity analysis.

In this chapter, we focus on the Morris method [107] for screening parameters. This method estimates the first-order sensitivities at many random points in the sample space using a series of trajectories. Each trajectory consists of $p+1$ points in $\mathbb{R}^p$. The first point is sampled randomly, then each of the following p points differs from the previous point in a single element by Δx_i. This process is carried out such that each element is changed once. Q is calculated at each point in the trajectory ($Q(\mathbf{x})$ and $Q(\mathbf{x} + \tilde{\boldsymbol{x}}_i)$), where the vector $\tilde{\boldsymbol{x}}_i$ represents a vector of zeros except for the i^{th} that has a value Δx_i. Using these measurements, the sensitivity of Q with respect to x_i can be estimated using the forward difference approximation

$$\frac{dQ}{dx_i}(\mathbf{x}) \approx \frac{Q(\mathbf{x} + \tilde{\boldsymbol{x}}_i) - Q(\mathbf{x})}{\Delta x_i}. \tag{3.5}$$

This process is repeated for R trajectories that start at random points in the sample space. The average, μ, average of the absolute value, μ^*, and standard deviation, σ, of the R estimates are then used to rank the inputs. The average effect of an input is measured by μ; however, sign cancellations may lead to misleading conclusions. The total effect of an input estimate is measured by μ^* and is unaffected by sign cancellations. Higher values of μ^* indicate parameters of which Q is more sensitive. Non-linear effects are estimated by σ, where larger values correspond to more interactions between parameters. Creating a scatter plot of σ versus μ^* is a useful tool for choosing important parameters. Parameters that are close to the origin are less important. Figure 3.1 shows an example with four points. The parameter x_1 would be excluded from further analysis because Q is not sensitive to x_1, and x_1 does not interact with other parameters. The parameter x_2 does not interact with other parameters, however Q is sensitive to it. The point x_3 interacts with other parameters, but Q is not very sensitive to it. Lastly, x_4 interacts with other parameters, and Q is sensitive to it. Therefore x_2, x_3, and x_4 would be included in further analysis.

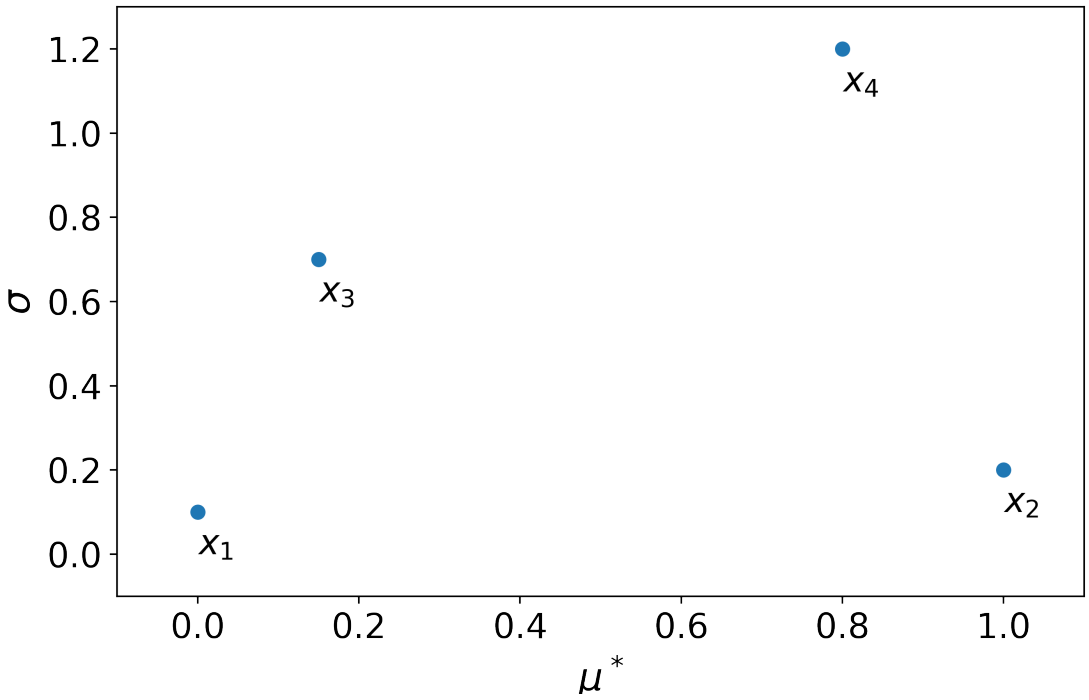


Figure 3.1: Example scatter plot of σ versus μ^*. This plot can be used to choose important features from the Morris method. On the vertical axis, σ approximates how much a parameter interacts with other parameters. On the horizontal axis, μ^* estimates how sensitive Q is to the parameter. In this example, Q is sensitive to the parameters x_2 and x_4. The parameters x_3 and x_4 interact with other parameters. The parameter x_1 would be excluded from further analysis.

3.3 Sensitivity analysis results

In this section, I overview the results of the sensitivity analysis performed on our model of CWD in Midwestern white-tailed deer. We performed a mixed method sensitivity analysis that utilized the Morris method to screen 15 parameters, and a variance-based global sensitivity analysis to examine a subset of 11 parameters that the Morris method identified as important.

Two modules were added to the movement model, population and disease dynamics. With the addition of these modules, many parameters were added as well. There is a fair amount of uncertainty in these parameters. However, they can be tuned such that our model reproduced realistic short- and long-term population dynamics of Midwestern white-tailed deer. Additionally, minimum and maximum values for each parameter can be estimated. Each of these parameters is treated as a random variable. In an effort to not bias these parameters, we assume that they were distributed following triangular distributions. Such

distributions only require the minimum, maximum, and most likely value of a parameter. These values are listed in Table 3.1 for each parameter.

name	min	mode	max	μ^*	σ
0) group number	50	251	630	.0074	.01437
1) direct transmission rate	.0001	.00052	.002	2321.6	4224.1
2) indirect transmission rate	.0001	.00052	.002	2589.3	4976.8
3) prion shedding rate	.001	20	1	0.5952	2.13531
4) prion half-life	3	48	120	0.05414	0.15123
5) disease mortality rate	.00005	.00015	.002	3871.2	10699.0
6) adult male harvest mortality rate	.0002	.0031	.007	698.5	1401.7
7) adult female harvest mortality rate	.0002	.0013	.007	1518.5	2870.5
8) yearling male harvest mortality rate	.0002	.003	.007	654.6	1360.9
9) yearling female harvest mortality rate	.0002	.00144	.007	1131.0	2579.4
10) fawn morality	.0002	.0012	.003	1856.5	3891.8
11) spring immigration/emigration rate	0	.00267	.03	336.8	724.9
12) fall immigration/emigration rate	0	.00267	.03	468.9	1042.7
13) spring dispersal rate	0	.01267	.03	181.8	329.6
14) fall dispersal rate	0	.00433	.03	239.8	569.4

Table 3.1: Parameter distributions and sensitivity estimates. We list the minimum, mode, and maximum value for each of the 15 parameters. These values are used to define a triangular distribution that are sampled to generate inputs to our model for sensitivity analyses. The total effect, μ^*, and estimate for non-linear effects σ, are listed in the last two rows. Using μ^* and σ as guides, we removed parameters from future analysis (see Fig. 3.2).

Using the distributions for our input parameters, we evaluated 79 trajectories for the Morris method. Equation 3.5 was calculated 79 times for each input parameter, using a spacing Δx_i that perturbed the previous point by 10% percent. In total, our model was run 1264 times to screen the parameters. We approximated μ^* and σ for each of the input parameters; they are listed in Table 3.1. In Fig. 3.2, σ and μ^* for each parameter is shown as a labeled point. Based on results of the screening, we decided to remove the number of groups, prion shedding rate, prion half life, and spring dispersal rate from future analysis. The remaining parameters were used in a variance-based global sensitivity analysis. We generated 95 Sobol samples from the remaining 11 parameters for a total of 1235 sets of inputs. In Table 3.2 ,we show the total effect index of our parameters after five, ten, fifteen, and twenty years of simulated time.

3.4 Conclusion

Based on our results in Table 3.2, we can see that prevalence of CWD is most sensitive to harvesting female deer throughout the simulation. Likely, this is due to the fact that when fawns are born in the simulation, they are placed in a close proximity of their female parent.

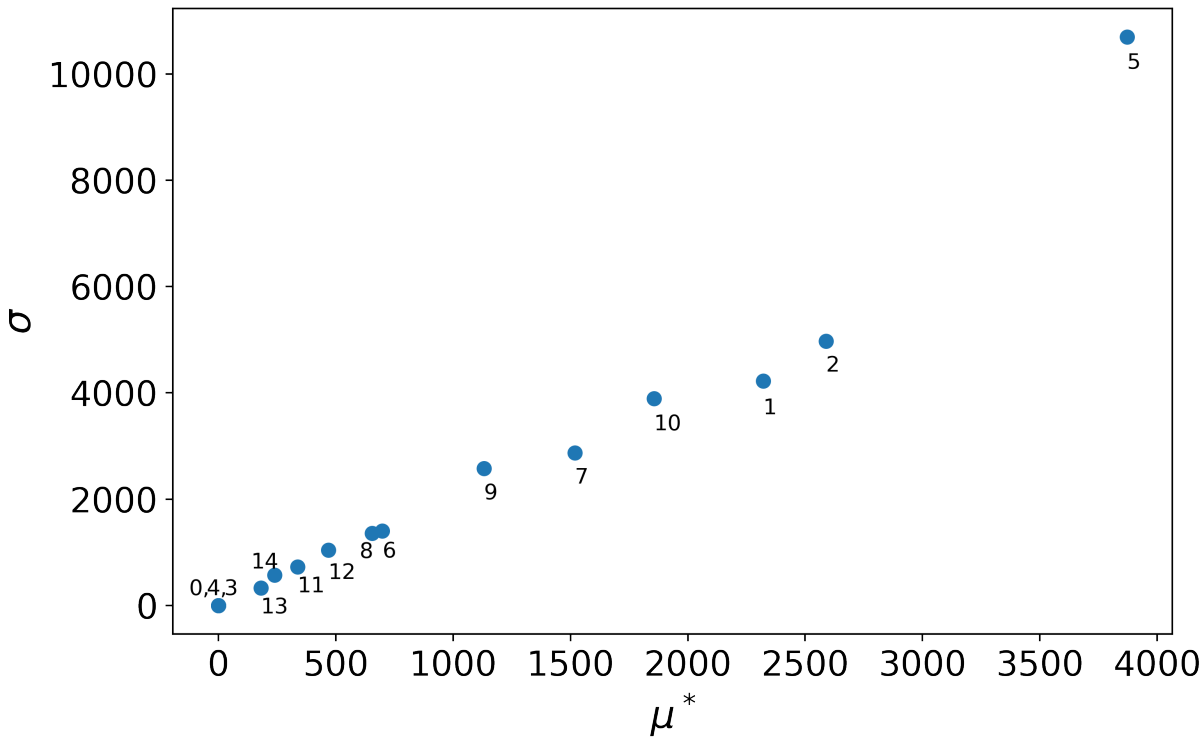


Figure 3.2: Scatter plot of σ versus μ^* for the model inputs listed in Table 3.1. The points 0, 4, and 3 are approximately on top of each other. Based on this observation, group number (0), prion shedding rate (3), prion half life (4), and spring dispersal (13) we removed from our global sensitivity analysis. We decided to keep fall dispersal rate (14) in our future analysis as we did not want to full remove dispersal behaviors from the model.

Therefore if that female is infected, or if they are in a contaminated part of the environment, there is a high probability that the fawn will become infected. These ideas are reinforced by the low sensitivity to male harvest. However, over time prevalence becomes more sensitive to harvesting all deer. These observations suggests that culling should be done over longer timescales, focusing on female deer early on. Early in the simulation, prevalence of CWD is more sensitive to direct transmission of the disease. As time goes on, prevalence becomes more sensitive to indirect transmission compared to direct transmission. This is likely due to the prions needing to accumulate in the environment to have a larger effect. Regardless, the early importance of direct transmission to late importance of indirect transmission suggest focusing on culling efforts soon after the detection of the disease. While the results discussed in this chapter have been specific to one scenario, our model can be applied to other regions to model CWD.

name	S_{T_i}(5 yr)	S_{T_i}(10 yr)	S_{T_i} (15 yr)	S_{T_i}(20 yr)
1) direct transmission rate	0.073	0.124	0.11	0.084
2) indirect transmission rate	0.068	0.065	0.076	0.089
5) disease mortality rate	0.082	0.082	0.073	0.076
adult male harvest mortality rate	0.056	0.042	0.054	0.078
adult female harvest mortality rate	0.194	0.186	0.16	0.15
yearling male harvest mortality rate	0.048	0.04	0.056	0.065
yearling female harvest mortality rate	0.102	0.128	0.115	0.123
fawn morality	0.082	0.091	0.11	0.101
spring immigration/emigration rate	0.063	0.061	0.062	0.069
fall immigration/emigration rate	0.126	0.098	0.095	0.093
fall dispersal rate	0.107	0.084	0.087	0.072

Table 3.2: Results from global sensitivity analysis. Here we list the total effect index after 5, 10, 15, and 20 years of simulated time. This index measures the amount of variance in disease prevalence that can be accounted for by each input, including interactions amount the inputs. Throughout the simulation, prevalence is most sensitive to adult female harvest rate and least sensitive to yearling male harvest rate. Early in the simulation, prevalence is more sensitive to direct transmissions than indirect; however, prevalence becomes more equally sensitive to these inputs as the simulation goes on.

CHAPTER 4:
MATHEMATICAL MODELING OF DISINFORMATION AND EFFECTIVENESS OF MITIGATION POLICIES

4.1 Summary

The following chapter explores the second facet of ABMs: real-world applications. This chapter is based on my previously published work in [31]. Here, the focus is on decision-making processes in the context of disinformation spread on online social networks. A graph-based ABM was developed and employed to evaluate six strategies for combating the spread of disinformation. These strategies fall under the broad categories of content moderation, education, and counter-campaigns, with two strategies from each of these categories being implemented. This approach involved extensive simulations carried out on thousands of graph structures, including those modeled after real social networks, to test the efficacy of these strategies.

My major contributions in this work were implementing the models, and thoroughly exploring the effectiveness of disinformation combating strategies on various types of graphs. Care was taken to tie the simulated strategies to real world analogs, ensuring that the findings offer valuable insights for practical decision-making in tackling disinformation. This research not only aids in studying disinformation, but also serves as a guide for creating ABMs that can aid in decision-making.

4.2 Introduction

The spread of disinformation has brought numerous adverse consequences, such as the manipulation of the 2016 US presidential election [108, 109], COVID vaccine hesitancy [110, 111, 112], and the growth of QAnon [113]. Advances in the development of chatbots are creating new concerns [114, 115, 116, 117]. In response, considerable efforts have been devoted to detecting [118, 119, 120, 121] and combating [122, 123, 124, 125] disinformation. Disinformation tracking, bot detection, and credibility scoring tools [126] have been developed, but the spread of malicious information remains a major challenge. Disinformation has escalated to a degree that the US Congress is examining intervention policies [127]. These policies can be divided into two categories: individual-empowering and structure-changing policies [128].

Individual-empowering policies help individuals to evaluate information they are exposed to and include policies such as fact-checking social-media platforms [125]. Ideally, when social-media users interact with disinformation, they would be warned and discouraged from believing or spreading it further [128]. The reliability of information sources can also be rated

[129]. This rating can be performed by experts or users who either rate many articles from a single source, generating an aggregate score, or rate sources directly. Individuals are more skeptical of sources with low ratings and are less likely to interact with information from such sources [129]. In addition to fact-checking and rating sources, one of the most powerful ways to empower individuals is through education. For example, instructional materials for teaching critical thinking can be created, such as guides for librarians to teach students to be aware of fake news [124], or for teachers to teach young students to think critically about news they come across on social media [122].

In contrast to individual-empowering policies, structure-changing policies prevent individuals from being exposed to certain disinformation entirely. Policies that fall in this category are primarily implemented by social-network operators who monitor the content on their sites and remove users or content they deem unacceptable [127]. Additionally, groups can run counter-campaigns on social media to drown out disinformation with facts [130, 131, 125].

Mathematical models are often used to evaluate, choose and optimize strategies for implementing such policies for combating disinformation, because it is usually impractical to test such strategies in the real world. Many models can be applied to study the spread of disinformation [132, 133], including the voter model [134, 135, 136], the Axelrod model [137], epidemiological models [138, 139, 140, 141, 142], the attraction-repulsion model [143], the naming-game model [144, 145], and the binary agreement model [146, 147, 148, 149, 150].

In this work, we adopt a policy-driven approach to understanding and combating disinformation, diverging from the prevalent trend in the literature that often emphasizes the mathematical or statistical mechanics of models. By both offering the theoretical underpinnings of our modeling approach and linking our modeling directly to real-world policies, we hope to provide insights that can aid in effective policymaking.

Towards this goal, we employ the binary agreement model with committed minorities, an agent-based model developed by Xie et al. This model was chosen not merely because it is well studied but because it encapsulates critical properties of disinformation spread. Furthermore, it highlights a pivotal moment in majority opinion, influenced by a committed minority that permits minority rule. In the binary agreement model, a tipping point occurs when a critical fraction of a network population, p_c, which is only approximately one tenth of the population on complete graphs, advocates a particular opinion strongly [146]. On heterogeneous graphs, p_c is even lower, as the average connectivity of the graph decreases [151]; such low values of p_c make it very challenging to mitigate disinformation. The fact that this model exhibits a tipping point is important because tipping points have been observed in real human interactions. For example, in groups assigned to identify an item in an image, a 25% minority can sway the majority's answer [148]. Moreover, a goal of

disinformation campaigns is often to sway public opinion towards a tipping point. The ability of a committed minority to overtake majority opinions can have large effects in the real world, as has been seen with social-media influence campaigns leading up to the 2016 US presidential election [108, 109], influence campaigns that have affected societal responses to the COVID-19 pandemic [110, 111, 112], and the rise of QAnon [113].

The basic binary agreement model of Xie et al. [146] can be extended to incorporate more realistic features. Examples of extensions that have been developed include modifying the propensity of an agent in the mixed state to share one of its opinions [152], varying the number of interactions needed for an agent to change opinions [152, 153], adding a competing committed minority [147], expanding the number of possible opinions [154], varying the level of commitment of the minority [145], and including heterogeneous [155] and dynamic [156] graphs.

While many studies have examined the effects of a single change to the binary agreement model on the dynamics of the model, to our knowledge, no study has examined the effects of incorporating one or more disinformation-management strategies. Here, we explore strategies that have previously been identified as potential real-world strategies for combating the spread of disinformation [127]. We implement individual-empowering and structure-changing policies – in the form of content-moderation, education, and counter-campaign strategies – in a modified version of the binary agreement model, and we examine their ability to move, smooth, or remove the tipping point exhibited by the model when implemented on several weighted, heterogeneous networks. We incorporate these strategies into our model by removing or adding agents or altering agents' susceptibility to other opinions. Properties of the tipping point provide us with metrics that can be used to quantify the effectiveness of various strategies for countering disinformation. We apply our methods to several types of synthetic networks, including small-world and scale-free networks that capture many features of social networks. Additionally, we apply our methods to real social-network data, from Asian users of LastFM [157].

4.3 Methods

4.3.1 Binary Agreement Model and Its Computational Implementation

We construct agent-based models in which agents are connected by a graph and follow the binary agreement model proposed by Xie et al. [146] Each agent can hold one of the single opinions A or B, or the mixed opinion AB, and can be either committed or uncommitted to that opinion. Pairs of agents update their opinions through one of 12 interactions, using the update rules given by Xie et al. [146]; for convenience, these update rules are summarized

in Table 4.1. We refer to these basic update rules as the opinion update rules.

before interaction (speaker $\xrightarrow{\text{opinion}}$ listener)	after interaction (speaker – listener)
$A \xrightarrow{A} A$	$A - A$
$A \xrightarrow{A} B$	$A - AB$
$A \xrightarrow{A} AB$	$A - A$
$B \xrightarrow{B} A$	$B - AB$
$B \xrightarrow{B} B$	$B - B$
$B \xrightarrow{B} AB$	$B - B$
$AB \xrightarrow{A} A$	$A - A$
$AB \xrightarrow{A} B$	$AB - AB$
$AB \xrightarrow{A} AB$	$A - A$
$AB \xrightarrow{B} A$	$AB - AB$
$AB \xrightarrow{B} B$	$B - B$
$AB \xrightarrow{B} AB$	$B - B$

Table 4.1: Binary agreement model update rules for the three opinions A, B and AB. Here, our convention is that opinion A is disinformation. Each row of the table shows one possible interaction. In the left column, the speaker's state is listed at the tail of the arrow, the opinion that speaker shares is above the arrow, and the listener's state is at the head of the arrow. The right column shows the outcome of the interaction; the speaker's state is listed first followed by the listener's state.

Committed agents are introduced by choosing a fraction, p_a, of agents who always hold the opinion A, regardless of their interactions with others. In our model, this committed minority spreads disinformation; i.e., the opinion A is disinformation, and the opinion B is the truth. In general, p_a is varied over a wide range to find the critical value that defines the model's tipping point; disinformation mitigation strategies aim to move this tipping point to a more favorable value.

We applied the opinion update rules to agents who were connected on simulated social networks, which will be discussed in the subsection below. At each time step, for each agent in the network, one of that agent's neighbors was selected randomly and uniformly. For each such pair of agents, either the agent or the neighbor was randomly assigned to be the speaker, and the other was assigned to be the listener. Probabilistically, the speakers shared their opinions with the listeners; such interactions determined the new opinions of both the speakers and the listeners. We show these rules, which govern our computational implementation of our model, in Fig. 4.1. We refer to these rules as the simulation update rules.

Unless otherwise specified, all agents not committed to A, i.e., not in the committed minority, were initialized as uncommitted to B, i.e., as holding the opinion B but not committed to that opinion. The simulation was updated repeatedly, over many time steps, until either no agent changed state and none were in a mixed state, or 5,000 time steps had passed. The fractional densities n_A and n_B of the nodes was then computed.

We examined the tipping point in our simulations by comparing the fraction of agents with opinion B (the truth), denoted by n_B, with the fraction of agents committed to the opinion A (disinformation), denoted by p_a. Many simulations on each graph type were performed for each anti-disinformation strategy and the basic binary agreement model; see the Simulations subsection below for further details. At the end of each simulation, we recorded the values of n_B and p_a. We compared these results across simulations; for each strategy and the basic model, and for each graph type, we plotted n_B vs. p_a on a single plot. For the basic binary agreement model without any intervention, a tipping point is evident on the plot; at the tipping point, n_B falls from approximately one to zero once $p_a \approx .1$, as is shown by the black dot-dash line in Fig. 4.4 and Fig. 4.5. We compare the locations of the tipping point, i.e., the value of p_a at which n_B begins to fall steeply toward zero, for each strategy to the location of the tipping point in the basic model and to the locations of the tipping points seen with other strategies.

The goal of our work was to measure the effects of several disinformation-mitigation strategies on the spread of disinformation using our model. To measure the spread of disinformation in our simulations, we examined to what extent each strategy moved, smoothed, or removed the tipping point exhibited by our model. Concretely, to measure how much a strategy moved the tipping point relative to another condition, we measured relative changes in the value of p_a at the tipping point between conditions. Smoothing and removal of a tipping point were noted as qualitative changes. These are the outputs of our simulations that we examine in the Results section below.

4.3.2 Simulated Social Networks

Artificial social networks were modeled using weighted, directed graphs $G = (V, E)$. Agents $v_i \in V$ were represented by the graph's vertices, and an edge between agents, $(v_i, v_j, w_{ij}) \in E$, represented a connection that allowed v_i to interact with v_j with probability w_{ij}. We generated several graph types that were initially undirected and unweighted, all of which were created using NetworkX [158]. The graphs we generated, and their required arguments, are listed in Table 4.2. We created random and small-world networks by varying the probability of creating an edge in Erdős-Rényi graphs, and by varying the initial number of edges and the probability of rewiring an edge in Watts-Strogatz graphs. Scale-free graphs were created by tuning the number of edges to attach from a new node to existing nodes in Barabási-Albert

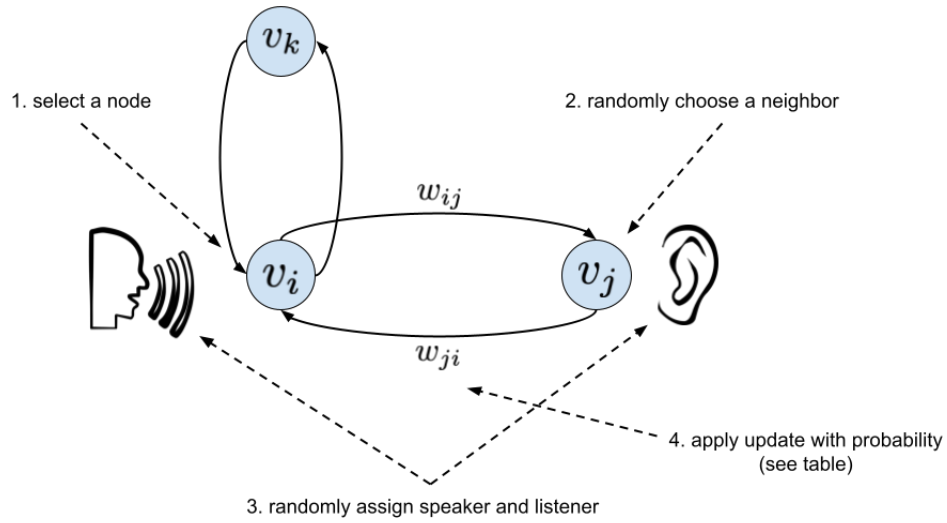


Figure 4.1: Simulation update rules governing computational implementation of the binary agreement model. At each time step, for each node v_i (i.e., each individual) in the graph (step 1), one of its neighbors v_j was selected randomly (step 2). In each pair of nodes, one was randomly assigned to be the speaker, and the other was assigned to be the listener (step 3). With probability w_{ij} (the weight of the edge between the speaker and the listener), the nodes interacted and updated their opinions (step 4). See Table 4.1 for a table of opinion update rules.

graphs. Additionally, we examined barbell, lattice, and complete graphs.

Real social networks can be very large. Here, our numerical results are based on relatively small graphs with 400 nodes, a value chosen to minimize computational cost while retaining accuracy. This value was chosen by executing the binary agreement model on networks of progressively larger size and comparing the steady state of the simulations to the mean-field (large-node) limit of the model. Consistent with Xie et al. (see Fig. 1.a) [146], we found that results obtained with a complete graph with 400 nodes agree well with the mean-field approximation. However, as a final study, we performed simulations on the real social network of Asian users of LastFM [157], which has 7,624 nodes and 27,806 edges; this result will be discussed in the next section. A graph of the Asian users of LastFM is shown in Fig. 4.2.

The basic topology of G contains the presence or absence of an edge. Each edge was converted into two weighted, directed edges that formed a loop between the two connected nodes. These edges can be can be assigned real-valued edge weights that reflect the relative strength of a social interaction. We used a graph's structure to determine its weights; nodes were more likely to interact with neighbors that they shared more connections in common with. Similar to graph transitivity, we define social transitivity as the ratio of the number of neighbors shared between the successor and predecessor nodes of an edge to the total number

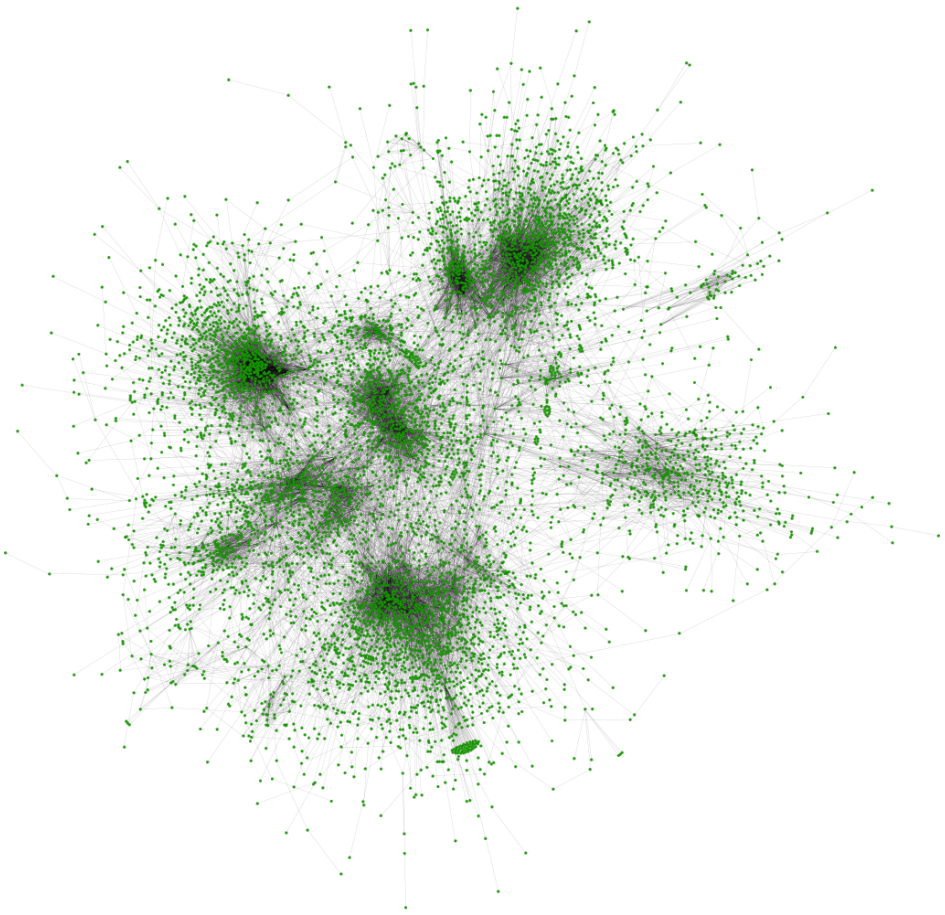


Figure 4.2: Asian users of the LastFM social network. Users are represented as green nodes, and edges represent a mutual follower relationship between users. This social network consists of 7,624 nodes and 27,806 edges. The placements of the nodes were determined using the Yifan Hu algorithm [159].

of neighbors of the predecessor node. We then weighted each edge by its social transitivity. Mathematically, the weights in the graph are given by

$$w_{ij} = \frac{1 + \text{number of shared neighbors between } v_i \text{ and } v_j}{\text{number of neighbors of } v_i}. \tag{4.1}$$

Here, the additional 1 in the numerator accounts for the fact that v_i and v_j share a connection. We used these weights to encode the probability that two nodes will interact in our model. A listener that shares many neighbors with a speaker has a higher probability of switching state than one that shares few neighbors with the speaker.

graph type	N	k	p	m	n	m_1	m_2
complete_graph	400						
watts_strogatz_graph (small world)	400	$.02N$	.5				
watts_strogatz_graph (small world)	400	$.12N$	.5				
watts_strogatz_graph (small world)	400	$.25N$	.5				
watts_strogatz_graph (random)	400	$.02N$	1				
watts_strogatz_graph (random)	400	$.12N$	1				
watts_strogatz_graph (random)	400	$.25N$	1				
erdos_renyi_graph (random)	400		.02				
erdos_renyi_graph (random)	400		.12				
erdos_renyi_graph (random)	400		.25				
barabasi_albert_graph	400			4			
barabasi_albert_graph	400			24			
barabasi_albert_graph	400			50			
grid_2d_graph				200	200		
barbell_graph						199	2

Table 4.2: Graphs that were explored and their parameters. We list the graph types and their required arguments. For the Watts-Strogatz graphs, the argument k is the initial number of nearest neighbors that are connected, and p is the probability of rewiring an edge. For the Erdős-Rényi graphs, p is the probability of creating an edge between a pair of nodes. For the grid graph, the arguments m and n are the dimensions of the grid. For the barbell graph, m_1 is the number of nodes in each of two fully connected subgraphs that are attached by m_2 intermediate nodes. Every graph had 400 nodes, but graphs had different numbers of edges. We also explored a real-world graph of Asian users of LastFM [157], not listed here.

4.3.3 Mitigation Strategies

Using our modified version of the binary agreement model, we evaluated three mitigation policies – content moderation, education, and counter-campaigns – that are discussed in the following subsubsections. In this subsection, we first provide an overview of the strategies we consider for implementing those policies, and the implementation of these strategies in our model. In the subsubsections below, we discuss details of their implementation in our model.

We first discuss content moderation, in the form of banning some users who spread disinformation from social media platforms. We implemented two content-moderation strategies in our model by removing agents from our graphs in two ways: removing influential agents in the committed minority that spreads disinformation, or removing randomly chosen agents in the committed minority.

Next, we consider two educational strategies. These strategies aim to educate individuals broadly, and target those who are interacting with disinformation. Broad education consists of teaching individuals how to identify disinformation to reduce the chances that they will

believe disinformation or hold immutable opinions. We refer to this strategy as a skepticism strategy, and we implemented it in our model by reducing committed agents' level of commitment to ideas. In contrast, a targeted educational strategy includes fact-checking information and labeling sources, for example by providing labels on online videos or warnings on cigarettes, to bias individuals towards the truth. We refer to this strategy as an attentive strategy, and implemented it in our model by giving all non-committed agents a bias towards sharing and believing the truth.

Finally, we discuss counter-campaign strategies that counter disinformation with facts. A counter-campaign strategy can be driven by large agencies; for example, the US Centers for Disease Control and Prevention provides guidance about sharing vaccine information [160]. Alternatively, counter-campaigns can be conducted by groups of people who share information to combat disinformation directly. We modeled counter-campaign strategies by introducing a second committed minority that is committed to the truth and competes with the original committed minority that is spreading disinformation.

Using our model, we examined the effects of each of these strategies separately. The strategies we implemented in our model are shown in Fig. 4.3, and the policies, specific strategies, and our implementations of them in our model are given in Table 4.3.

Content Moderation

We examined the strength of content moderation by removing different proportions of committed agents from a graph over an order-of-magnitude range; we removed 0.25%, 0.5%, 1%, 2%, or 2.5% of the total number of nodes in the graph, for values of p_a that varied from 0.03 to 0.13. Committed nodes were removed before we executed each simulation examining content-moderation strategies.

One content-moderation strategy we considered required removing influential agents in the committed minority that were spreading disinformation. Thus, we needed to identify influential nodes in the committed minority in a graph. We define influence using two key metrics: degree centrality and betweenness centrality. Degree centrality measures the number of connections an agent has, while betweenness centrality measures how centrally located an agent is in the graph. Both measures of centrality are normalized to fall between zero and one. The degree centrality of a node is how many connections that node has divided by the total number of possible connections, while its betweenness centrality is calculated as how many shortest paths pass through that node divided by the total number of shortest paths in the graph. An agent with both high degree centrality and high betweenness centrality has the potential to spread disinformation directly to many agents and to spread disinformation to disparate parts of a graph that might not interact without that agent. Using these metrics

policy	**strategy**	**implementation in model**	**result**
content moderation	• temporary or permanent bans of highly connected users spreading disinformation • temporary or permanent bans of randomly chosen users spreading disinformation	• removing influential agents in the committed minority, i.e., influential committed-minority nodes in a graph • removing randomly chosen agents in the committed minority, i.e., randomly chosen committed-minority nodes in a graph	• slightly increases required size of committed minority for tipping point • random and targeted approach perform similarly, except on graphs with unique topology, e.g., barbell
education	• skepticism: teaching critical thinking • attention: providing individuals with ratings for sources and fact-checking of information	• reducing the level of commitment to ideas of agents, i.e., nodes, in the committed minority • giving all agents, i.e., nodes, not in the committed minority a bias towards towards the truth	• skepticism greatly increases the required size of a committed minority • attention has little to no effect on the tipping point
counter-campaigns	• spread of facts by groups to counter disinformation	• introducing an opposing minority of agents, i.e., nodes, committed to the truth	• larger counter-campaigns increase the tipping value of p_a

Table 4.3: Policies and strategies for combating disinformation and their implementations in our model. The policies we considered are given in the first column. In the middle column, we list the specific strategies for implementing these policies that we examined in this work. In the third column, we list our implementations of these strategies in our model. In the final column, we summarize the effect each policy has on the tipping point in the binary agreement model.

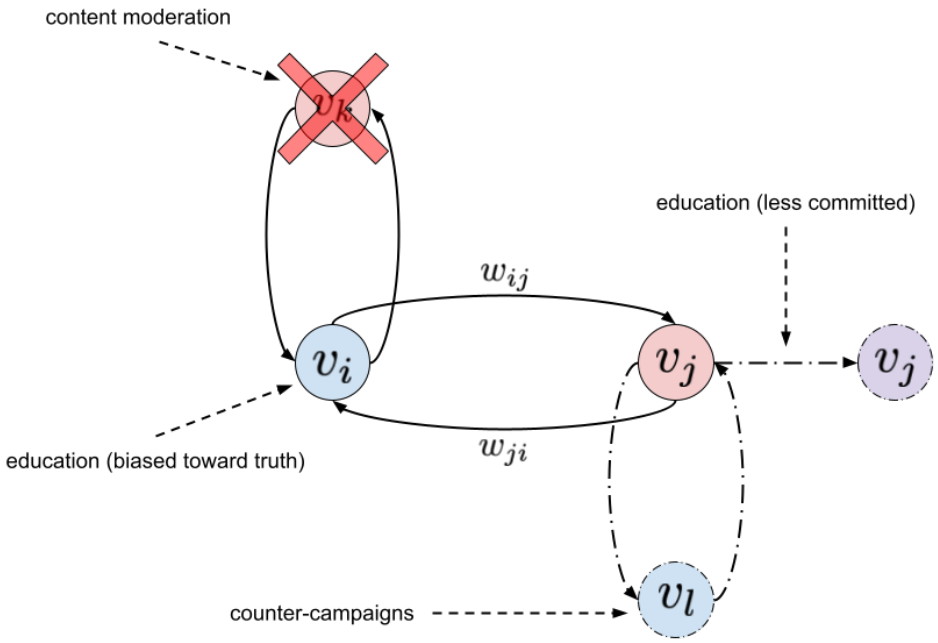


Figure 4.3: Mitigation strategies examined using our model. Solid arrows represent edges in the graph, dot-dashed arrows represent a change made to the graph structure, and dashed arrows represent comments. Blue nodes represent agents with the true opinion, red nodes represent agents in the committed minority who believe disinformation, and purple nodes represent agents with variable commitment to an opinion. Two strategies for content moderation, both involving banning users from a network, were implemented in our model by removing disinformation-spreading, committed-minority nodes from a graph. An education strategy aimed at increasing skepticism was implemented by reducing the level of commitment to ideas of committed-minority nodes. Another education strategy that brought attention to disinformation was implemented by biasing all non-committed nodes towards the truth. We also introduced counter-campaigns into a graph by adding a second committed minority that was committed to the truth. Each strategy was investigated separately.

and a normalization method, we define the influence of a node as:

$$\mathcal{I}(v) = \frac{1}{2}\Big(C_D(v) + C_B(v)\Big), \tag{4.2}$$

where C_D is the degree centrality and C_B is the betweenness centrality. This metric allowed us to identify influential nodes in the committed minority. In each simulation in which influential committed-minority nodes were removed, the most influential committed-minority nodes were removed, up to the desired percentage of all nodes in the graph.

Clearly, if enough nodes were removed, the size of a committed minority would be too small for it to overtake the majority. Moreover, the size of the committed minority could be decreased by removing nodes at random as well as by removing influential nodes. Therefore,

we compared the effects of our strategy of removing the most influential committed-minority nodes against the effects of a second strategy of removing randomly selected committed-minority nodes.

Education

Another mitigation policy we examined was education. In particular, we examined two strategies based on education: skepticism and attention. We implement a skepticism strategy in our model by relaxing the strength of commitment, c, held by the committed minority and endowing committed agents with a probability of changing their opinion after an interaction. This probability is defined as

$$p(c) = \begin{cases} 2(c - c_{min})/(1 - c_{min})^2 & c_{min} \leq c \leq 1, \\ 0 & \text{else.} \end{cases} \tag{4.3}$$

Here, c_{min} is the lowest level of commitment that a committed individual can have; we assume $c_{min} = 1/2$. Equation 4.3 is a triangular distribution that linearly increases between zero and four when $1/2 \leq c \leq 1$. Note that as c_{min} tends to one, $p(c)$ tends towards a delta function centered at $c = 1$. This implies that all agents are fully committed, and thus we recover the original binary agreement model. In this strategy, a partially committed agent becomes fully uncommitted if it switches its opinion to the mixed state.

We implemented an attention strategy consisting of fact checking and source rating into our model by biasing non-committed agents towards the truth. In our model, the true opinion is B, and we assume that our attention strategy leads agents to be more likely to believe B. Therefore, agents in the mixed state are more likely to switch their opinion to B than to A. Additionally, those who already believe the truth have a preference toward keeping that opinion. To implement these two effects, we introduce a truth bias, $\beta = .1$, that is the difference between the probability that an agent with the mixed opinion will share B in an interaction with a neighbor and the probability that the agent will share A in an interaction. The truth bias β is also used to increase the probability that an agent with opinion B will change its opinion after an interaction. Mathematically, we have $P(AB \text{ shares } B) - P(AB \text{ shares } A) = \beta$, and $P(B \text{ switches to } AB | \text{neighbor shared } A) = 1 - \beta$.

Counter-campaigns

Counter-campaigns were implemented by initializing a group of agents to be committed to the opinion B, which is the truth, to combat disinformation from those who are committed to

A, which is disinformation. To examine the effects of different sizes of counter-campaigns, we considered competing minorities consisting of either 5% or 15% of the total population, and we initialized these minorities to be committed to B. We assumed that counter-campaigns would begin in a local region of the graph, similar to an echo-chamber. This effect was incorporated by initializing the graphs in a non-random way, without changing the topology of the graph. Using NetworkX's "`spring_layout`" routine that implements the Fruchterman-Reingold force-directed algorithm, we assigned a position to each node in the graph that was contained in the box $[-1, -1] \times [1, 1]$. We then ensured that all nodes that were committed to A were initialized on the left side of the graph (their first coordinate was negative), and that those that were committed to B were on the right side (their first coordinate was positive). All remaining individuals were initialized as uncommitted to B, i.e., holding the opinion B but not committed to that opinion.

4.3.4 Simulations

For each of the six anti-disinformation strategies, we conducted 30 simulations on a new instantiation of each of the 15 graph topologies listed in Table 4.2 and 30 simulations on the real social network. Each of these 2880 simulations executed our model either 11 (content moderation) or 15 (education and counter campaigns) times, corresponding to the values of p_a we examined, and had a new initial configuration of committed agents. In total, our model was executed 118,080 times, and required over 18,000 core hours on a distributed memory supercomputer. Each simulation was ran until either no agent changed state and none were in a mixed state, or 5,000 time steps had passed. The results of these simulations were averaged for each graph topology; this allowed us to examine the variance in outcomes and outlier results.

4.4 Results

We investigated the effects of three anti-disinformation policies, implemented in the form of six specific strategies for combating disinformation, using a binary agreement model we modified to incorporate these strategies. We conducted a total of 2880 simulations on 15 different graph topologies to explore the effects of each of these strategies, and an additional 30 simulations on a real social network. At the end of each simulation, we recorded the fraction of agents with opinion B at steady state and the fraction of agents committed to the opinion A. Following the notation in Xie. et al. [147], these values are denoted by n_B and p_a, respectively. For each anti-disinformation strategy, we plotted n_B vs. p_a and examined how the value of p_a at the tipping point varied among the strategies employed. We also examined whether a strategy affected the shape of the tipping point or removed the tipping point altogether. Below, we discuss our results for each strategy. Because of space

limitations, we show results from complete graphs and the real social network in the main manuscript; the results for all graph types are shown in Appendix B.

4.4.1 Content Moderation

Content moderation was implemented by removing committed nodes from a graph before we executed our simulations. Committed nodes were removed either based on their influence or randomly. In Fig. 4.4a, we show an example of the effects of content moderation on a complete graph, and in Fig. 4.5a we show the results on a real social network. Due to computational limitations, only up to 1% of the committed agents were removed from the real social network. Results for removing highly influential committed nodes are shown with solid lines, and results for removing committed nodes randomly are shown with dashed lines. Results for the basic binary agreement model in the absence of intervention are indicated with a black dot-dash line. Different line colors indicate different percentages of the total nodes that were removed. Each opaque line shows the average result of 30 simulations, and each partially transparent line shows the result of a single simulation. Similar plots for content moderation implemented on all of our graphs are shown in Appendix B.

As expected, as more nodes were removed from a graph, the initial size of the committed minority needed to be larger to overtake the majority; this feature was shared among all graph types. For most graphs, removing nodes randomly performed similarly, if not identically, to removing based on influence. However, on the real network, removing based on our influence metric outperformed removing randomly. As shown in Fig. 4.2, the LastFM social network has many clusters of nodes. These clusters are interconnected by a few nodes, and our metric does well at identifying these nodes. For barbell, grid, and the small world graphs with a low average degree, removing nodes randomly outperformed using our influence metric. The unique topology of the grid and barbell network ranks many of the nodes as equally important (e.g., nodes in the fully connected components of a barbell graph) and continuously ranks certain nodes as most important (e.g., the center node of the grid graph), while removing randomly we might find a better set of nodes. Once a large number of nodes have been removed, either method for removing nodes work equally well, except for our real network. These results suggest using the influence metric to remove a small number of nodes from a graph, unless the graph has special topology that may severely constrain how nodes are selected. If a large number of nodes (relative to the size of the graph) are being removed, a random approach is better due to the extra computational cost for little to no gain in performance.

4.4.2 Education

We also examined the use of education that aimed to increase individual's skepticism and attention. Individuals who are more skeptical of information are less likely to be fully committed to an opinion. Whereas attentive individuals are biased towards the truth. Here, we implemented skepticism by relaxing the commitment of individuals and attention by introducing a bias towards sharing and holding the opinion B. Using a complete graph in Fig. 4.4b and a real social network in Fig. 4.5b, we show the effects of skepticism with an orange dashed line, attention with a green dotted line, a combination of both with a solid blue line, and no education with a black dot-dashed line. Each opaque line is the average of 30 runs, and transparent lines are singular runs. We show similar plots for applying education policies to all graph types in Appendix B.

Attention had a small effect for all graph types. In random graphs, when an attention strategy was used the size of the committed minority needed to overtake the majority grew slightly with the average connectivity. Attention had no effect for on the real social network nor complete, barbell, and grid graphs. Contrary to attention, skepticism required the size of the committed minority to be between 30% and 35% in order to overtake the majority. Using both attention and skepticism together produces similar results as just using a strategy based on skepticism. These results suggest that educating people to be more skeptical can greatly reduce the dangers of a committed minority.

4.4.3 Counter-Campaigns

Additionally, we examined counter-campaigns that were implemented by introducing an additional group committed to the opinion B, denoted by p_b. For a small counter-campaign, we let $p_b = .05$, and $p_b = 0.15$ for a large one. In Fig. 4.4c, we examine the effects of different sized counter-campaigns on a complete graph, and a real social network in Fig. 4.5c. As before, we show the result of no intervention with a black dot-dashed line. We show the effects of the small counter-campaign with a solid blue line and the large counter-campaign with a dashed orange line. Each opaque line is the average of 30 simulations, and transparent ones are individual runs. Again, similar plots for counter-campaigns applied to all graph types are shown in Appendix B.

As expected, when a larger counter-campaign is used there is a larger portion of individuals with the opinion B at steady-state. The only exceptions are the barbell, grid, and real social network graphs, where small and large counter-campaigns have the same effect; this is likely due to their unique topologies. On most graphs when a small counter-campaign is introduced, the tipping point is located when p_a is between $.1 - .12$, and past the tipping point $n_B \approx 0.05$: the size of the counter-campaign. This behavior is not observed on the grid graph, instead n_B appears to decay exponentially to .4.

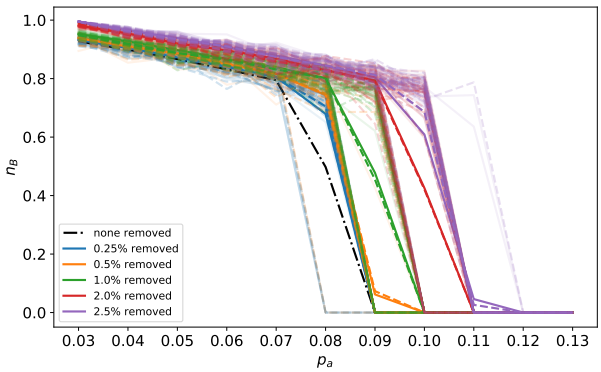
For most graphs, when a large counter-campaign is introduced the tipping point is located when p_a is between $.15 - .2$, and past the tipping point $n_B \approx .18 - .3$. Again, the exceptions are the barbell, grid and real social network graphs, where large and small counter-campaigns act similarly and n_B decays exponentially to approximately .5. For the remaining graphs, the final proportion of individuals with the true opinion is larger than the initial counter-campaign, showing the counter-campaign convinced non-committed individuals of their viewpoint. While these results suggest that the larger the counter-campaigns is, the better, our large counter-campaign was unable to keep the majority of individuals from holding the disinformation.

4.5 Discussion

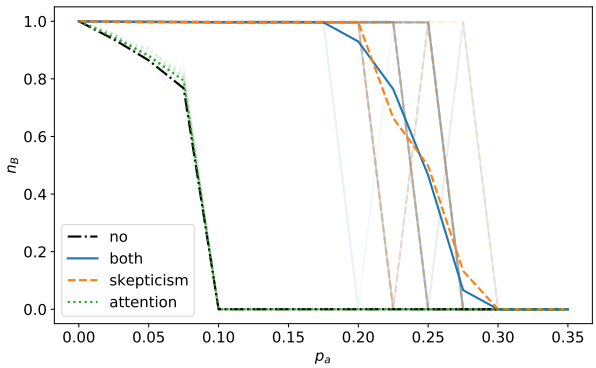
In this work, we investigated the spread of disinformation on various weighted, directed, heterogeneous graphs with committed minorities using the binary agreement model. We evaluated the effectiveness of three types of policies, namely content moderation, education, and counter-campaigns. For each policy we implemented two strategies to combat the spread of disinformation. These strategies were tested on hundreds of graphs based on 15 graph topologies, and a real social network generated from data of Asian users of LastFM [157]. Over 18,000 core hours were used to explore the effectiveness of the mitigation strategies on these networks.

Regarding content moderation, we found that removing nodes based on our influence metric did not significantly outperform removing nodes randomly for most graphs. As more nodes were removed, the two methods converged towards each other. This is encouraging, because our simple influence metric scales as $O(|V||E| + |V^2|)$, which can quickly become burdensome to calculate for large graphs. However, on the real social network graph a more targeted approach outperformed a random one. Therefore, our model suggests that removing sources of disinformation as they are identified is a viable method for implementing content moderation, however a more targeted approach might prevail on graphs with unique topology.

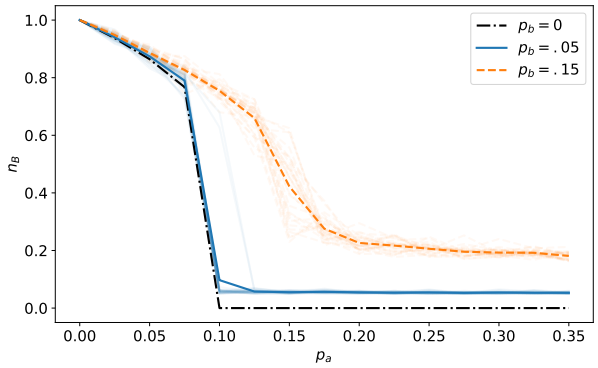
We also explored education-based policies and found that a strategy that increased people's skepticism had a notably stronger positive impact in our model than a strategy that focused on people's attention to disinformation. In fact, our model suggested that strategies such as fact-checking, had little effect, and only showed a small positive effect when any change occurred. However, it is worth noting that our definitions for skepticism and attention are crude due to the simplicity of our model. Therefore, while our model suggests that strategies that bring attention to disinformation may not be very effective, we caution against blindly following this advice. We recommend implementing any method that can in-



(a) The results of removing the most influential nodes are shown with solid lines, and the results of removing nodes randomly are shown with dashed lines. Different colors indicate different fractions of the total number of nodes that were removed. Removing nodes randomly performs similarly to removing the most influential nodes, as more nodes are removed.

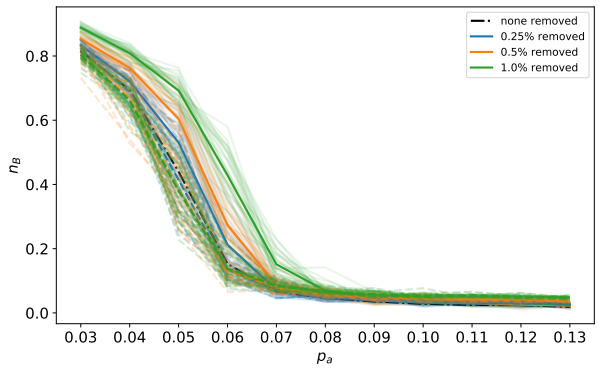


(b) The results of increased skepticism are shown with an orange dashed line, the results of increased attention are shown with a dotted green line, and the results of both are shown with a solid blue line. Attention has little to no effect, while skepticism greatly reduced the reach of the committed minority.

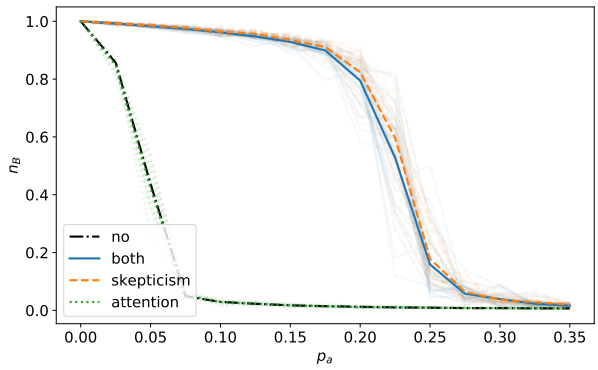


(c) The results of a small counter-campaign are shown with blue lines, and the results of a large counter-campaign are shown with a dashed orange line. The large counter-campaign was able to convince more uncommitted individuals of their opinion, but overall were unable to stop the minority from taking over.

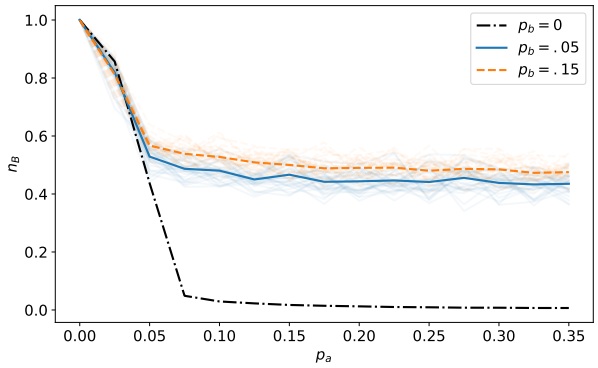
Figure 4.4: Strategies based on a) content moderation, b) education, and c) counter-campaigns applied to a complete graph. In each subfigure, the fraction of nodes with opinion B at steady-state, n_B, is plotted versus the fraction of nodes committed to A, p_a. For reference, the black dot-dash line shows how n_B varies with p_A when no nodes are removed, i.e., for the basic binary agreement model in the absence of a mitigation strategy. Each opaque line shows an average of 30 simulations, while the partially transparent lines represent individual simulations.



(a) The results of removing the most influential nodes are shown with solid lines, and the results of removing nodes randomly are shown with dashed lines. Different colors indicate different fractions of the total number of nodes that were removed. Removing nodes based on influence outperforms removing nodes randomly.



(b) The results of increased skepticism are shown with an orange dashed line, the results of increased attention are shown with a dotted green line, and the results of both are shown with a solid blue line. Attention has little to no effect, while skepticism greatly reduced the reach of the committed minority.



(c) The results of a small counter-campaign are shown with blue lines, and the results of a large counter-campaign are shown with a dashed orange line. Both sizes of counter-campaigns resulted in the graph being approximately equally split in opinions. The larger counter-campaign resulted in slightly more individuals believing the truth.

Figure 4.5: Strategies based on a) content moderation, b) education, and c) counter-campaigns applied to the Asian users of the LastFM network [157]. In each subfigure, the fraction of nodes with opinion B at steady-state, n_B, is plotted versus the fraction of nodes committed to A, p_a. For reference, the black dot-dash line shows how n_B varies with p_A when no nodes are removed, i.e., for the basic binary agreement model in the absence of a mitigation strategy. Each opaque line shows the average of 30 simulations, while the partially transparent lines represent individual simulations.

crease skepticism, while noting that strategies that focus on immediately biasing individuals towards the truth may not be as effective as those aimed at more broadly educating people to be critical thinkers.

After exploring the effectiveness of counter-campaigns, we found that small counter-campaigns were ineffective at combating the spread of disinformation for most graphs. When employed, they left only the original counter-group holding the true opinion. Contrarily, larger counter-campaigns were able to sway some uncommitted individuals, suggesting those committed to the disinformation had less reach. However, in the real social network, grid, and barbell graphs both large and small counter-campaigns resulted in an approximately equal split between individuals holding the true or disinformed opinion. We believe this is due to the topology of these graphs and our initialization of the committed agents. We initialized counter-campaigns and the original committed minority on opposite sides of graphs to account for them starting locally on the graph. However, in the real social network, grid, and barbell graph these two sides are more strongly intraconnected than interconnected, resembling echo-chambers. Thus each competing group of committed agents overtake their respective side of the graph, resulting in a fairly equal split of opinions. In contrast, previous work has explored initializing the competing committed group randomly on graphs [147].

For each strategy we implemented based on content moderation, education, and counter-campaigns, we found that the fraction of individuals with the true opinion versus the fraction of individuals committed to the disinformation were quantitatively similar. The vast majority of individuals held the true opinion until a critical number of individuals were committed to the disinformation. Then there was a tipping point in opinions where the disinformation group quickly dominated the popular opinion. Past this critical value the number of people believing each opinion stays constant. The different implementations of our polices were able to quantitatively vary these features. Most notably, education and content moderation increased the required size of the minority needed to sway the majority and counter-campaigns preventing individuals from being swayed by disinformation. The different implementations of our polices were able to quantitatively vary these features through various mechanisms. The quantitative changes produced by these strategies are summarized in Table 4.3. Content moderation and education address two sides of the disinformation issue: the source and the receiver. Content moderation attempts to curtail the spread at its origin, while education, especially fostering skepticism, equips individuals to resist misleading narratives. On the other hand, counter-campaigns introduce a competitive narrative to challenge and diminish the influence of the primary disinformation source. Each strategy has its merits, but their effectiveness can vary based on the structure and nature of the network in question. Most notably, education and content moderation had stronger effects on the tipping point than

content moderation. Specifically, skepticism moved the tipping point forward, while larger counter campaigns increased the number of individuals that remained with the true opinion after the tipping point. As the ability of generative models to produce convincing material grows they might be used to generate disinformation that is difficult to detect [117]. In such scenarios, educating individuals will become evermore important.

Based on the results of our simulations, education-based policies that increase skepticism and counter-campaign policies were the most promising. Examples of real world policies that increase skepticism could include creating media literacy programs. Such programs teach individuals to recognize trustworthy sources and discern fact from opinion. Additionally, incorporating critical thinking curriculum into the education curriculum at an early age can build a foundation of skepticism. Another potential strategy would be creating feedback loops on social media platform that notify users when they engage with potential disinformation. Example of real world strategies that focus on counter-campaigns could range from launching corrective advertising campaigns that directly address and correct false narratives, to establishing rapid response teams. These teams would monitor social media for emerging disinformation so that they could quickly launch counter-efforts.

In summary, our goal in this work was to create a model that captures realistic mitigation strategies within the confines of a binary agreement model so that our result may be applicable to both modelers and policymakers. Additionally, there is a vast number of other policies, strategy implementations, and combinations of the two that could be explored. This leaves many avenues for future studies to extend our work. One promising direction is to incorporate a more realistic model of the spread of opinions, such as the Attraction-Repulsion Model [143]. This model allows for multiple, continuous-valued opinions to be held by each individual. This more nuanced representation of opinions allow individuals to become more or less closely aligned with their neighbors. Additionally, we could enhance the realism of the model by incorporating multiple online and offline social networks, dynamic networks, and news sources. Beyond improving the model, future studies can explore policies we did not test, optimal parameters for existing policies, or learning entirely new policies.

CHAPTER 5:
AN ATTRIBUTE-PRESERVING METHOD TO MINIATURIZE LARGE SOCIAL NETWORKS

5.1 Summary

This chapter builds upon the work discussed in Chapter 4. Specifically addressing the difficulty in running simulations on real social networks, as well as the non-representative nature of simulated social networks. In this chapter, a method for miniaturizing social networks is presented; though, this method can be applied to any type of network. There are many ways to generate graphs, such as Erdős-Rényi and Watts-Strogatz models; however, this proposed method differs in that it creates graphs that have multiple similar metrics to a given graph but are reduced in size. Many more simulations can be evaluated on these miniaturized graphs compared to large network, without sacrificing accuracy.

My main contribution in this chapter is the implementation of a parallelized version of the parallel tempering algorithm. This algorithm generates miniaturized versions of social networks, with the constraint that the error in a set of metrics is minimized. I present an example of miniaturizing a real social network while maintaining the density, degree assortativity coefficient, and clustering coefficient. This example provides all the necessary steps to initialize and execute the algorithm.

5.2 Introduction

Networks serve as a fundamental tool for modeling systems composed of interconnected entities, offering insights across various disciplines [161]. Biologists, for example, utilize networks to describe genetic dependencies impacting protein expression [162, 163, 164]. In the realm of social sciences, networks represent social interactions, aiding in understanding how community structures influence behavioral dynamics [165, 166, 167, 168]. Similarly, networks are fundamental in modeling contagion-like phenomena such as the spread of diseases [169, 170, 171], ideas [29, 172], and misinformation [31, 173]. Networks also shed light on the intricacies of broader social-ecological systems, guiding scientists, stakeholders, and policymakers. The proliferation of large network datasets, increasingly available through public data repositories [174, 175, 176], open new avenues for simulating real-world systems on realistic graphs. These datasets present notable opportunities to simulate the dynamics of many real-world systems on realistic graphs.

However, the large size and complexity of existing network datasets pose significant scalability challenges, particularly for researchers relying on graphs as a tool for modeling and analysis. As networks grow in size and complexity, running simulations incur prohibitive

computational costs, often restricting our ability to accurately model associated real-world systems [177]. While new scalable algorithms have been proposed to mitigate these issues [178, 179], they are often developed for specific modeling tasks and may not be generally applicable outside of that task. Additionally, computing solutions that have sought to capitalize on the increased capabilities of distributed memory computing clusters require expertise in graph-based parallel computing paradigms, imposing additional constraints on network science practitioners [180, 181].

One approach to addressing the scalability challenges of large networks is the generation of structurally similar yet miniaturized synthetic graphs, enabling more manageable analysis and simulations. Existing graph generation methods excel at matching a narrow set of characteristics of the graphs that they represent at the expense of control over a broader network metrics. For instance, Erdős-Rényi [182] and Barabási-Albert [183] models are adept at replicating specific graph characteristics, such as degree distributions, but they fall short in controlling broader network features, such as community structure. Our proposed graph generator is designed to provide more generalized control over a range of properties, aligning closely with the structural intricacies of large-scale real-world networks. This approach not only facilitates efficient model testing but also ensures that the models retain greater relevance and applicability to real-world scenarios.

We propose a graph generator that employs the optimization technique parallel tempering [184, 185]. This algorithm consists of multiple instances of the Metropolis Monte Carlo algorithm, called replicas, each at a unique temperature level. Periodic swaps of temperatures between these replicas allow the algorithm to overcome shortcomings traditionally associated with optimization, such as convergence to sub-optimal solutions. The process begins with a seed network – a smaller, preliminary version of the network – and a set of graph metrics to match from a larger network. The algorithm then iteratively refines this seed network, ensuring that it evolves towards resembling the target graph in terms of the chosen metrics. This method offers an effective means of generating miniaturized networks that retain the structural essence of their larger counterparts.

5.3 Methods

Our objective is to generate a miniaturized undirected graph $G = (V, E)$ with vertices $v \in V$, and edges $(v_i, v_j) \in E$, such that it closely resembles a set of graph properties of a given target graph $G_{\text{target}} = (V_{\text{target}}, E_{\text{target}})$. Importantly, we aim to maintain structural and statistical similarity while ensuring $|V| < |V_{\text{target}}|$ and $|E| < |E_{\text{target}}|$, thereby achieving a reduction in both the number of nodes and edges compared to G_{target}. This reduction is crucial for simplifying the computational complexity while preserving the essential characteristics of the

original network. We employ the Monte Carlo optimization algorithm, parallel tempering, known for its efficacy in sampling complex optimization landscapes by avoiding local optima to achieve this goal. The following subsections detail the algorithm and how it is applied to optimize miniaturized graphs.

5.3.1 Metropolis Monte Carlo

As discussed, our objective is to create a graph that minimizes a sum of metric errors, χ, for various graph properties like assortativity, density, and clustering coefficient. The error function is defined as:

$$\chi = \sum_i \chi_i(G, G_{\text{target}}), \tag{5.1}$$

where each χ_i represents the error in a specific graph property, G is the graph under consideration, and G_{target} is the target graph with desired properties. The Metropolis Monte Carlo (MC) algorithm forms the foundation of our approach to minimizing χ. The MC algorithm on its own can be used to explore the space of possible graph configurations, and is particularly useful for optimizing graph properties due to its ability to sample from complex probability distributions.

The algorithm is defined as follows. Initially, a seed graph G_0 is initialized. In each iteration of the algorithm, a uniformly random number of modifications between 1 and 10, such as adding, removing, or rewiring edges, are proposed to the current graph G to obtain a new graph G'. In this implementation, moving an edge has a 50% chance of being selected while adding and removing each have a 25% chance of being selected; however these probabilities can be adjusted. The change in error, $\Delta\chi = \chi(G') - \chi(G)$, is calculated, and the proposed change is accepted with a probability $p = \min\left(1, e^{-\beta\Delta\chi}\right)$, where β represents the inverse temperature. The process is repeated with a new proposed change until a specified number of iterations is reached. While the algorithm converges to an optimal solution given enough iterations and a suitable choice of β, setting a value for β itself is challenging. When β is low, many configurations are explored without converging towards a specific solution. Conversely, when β is high, the algorithm will quickly converge to the closest minimum of χ, regardless if it is optimal. This balancing of β is known as the exploration-exploitation trade off. Practitioners suggest tuning β such that the acceptance probability is about 23% [186, 187] to balance the trade off.

This balancing can be achieved by assuming

$$\chi_i = \sum_i \omega_i |m_i(G) - m_i(G_{\text{target}})|, \tag{5.2}$$

where each $m_i \in [0, 1]$ is a min-max scaled metric that is weighed by ω_i. This scaling ensures

that metrics are on a comparable scale, preventing any one metric from disproportionately influencing the optimization process and makes differences in weights more directly comparable. The weights ω_i are set such that each metric is equally weighted based on how much a they change on average when a graph is perturbed. This balancing is achieved by executing a MC simulation with $\beta = 0$ and $\omega_i = 1$ for T_{est} iterations. After each iteration j, the value $|m_i(G) - m_i(G_{\text{target}})|_j$ is recorded for each metric. Based on these values, the weights are adjusted as:

$$\omega_i = \frac{T_{est}}{\sum_j |m_i(G) - m_i(G_{\text{target}})|_j}, \tag{5.3}$$

ensuring a balanced consideration of all metrics. Subsequently, another MC simulation is executed with these determined weights for another T_{est} iterations. After each iteration, the absolute change in error $(|\Delta\chi|)_j = (|\chi(G) - \chi(G')|)_j$ is calculated. From these measurements, the optimal β is defined as

$$\bar{\beta} = -\frac{T_{est} \ln(.23)}{\sum_j (|\Delta\chi|)_j}, \tag{5.4}$$

which inversely relates to the acceptance probability. With this optimal value for β, we move to extending the MC algorithm in the next section.

5.3.2 Parallel Tempering

Even with an optimal β to balance exploration-exploitation trade off, a single instantiation of the MC algorithm may still struggle to find the global optimal solution in complex error landscapes. Parallel tempering addresses this challenge by running multiple Metropolis MC simulations (replicas) simultaneously, each assigned a different β. This multi-replica approach allows hotter (lower β) replicas to explore the surface broadly, while cooler (higher β) replicas focus on exploiting promising regions. Periodic exchanges of β values between adjacent replicas prevents replicas from getting trapped in local optima, while maintaining their ability to converge towards optimal solutions.

The parallel tempering algorithm is defined as follows. Initially, N replicas of the MC algorithm are initialized, each with a unique β_i and initial graph. Each replica independently runs the MC model as described in Sec. 5.3.1. However, at every T_{swap} iterations, adjacent replicas propose to swap their β values. This swap is accepted with probability $\alpha_{\text{swap}} = \min\left(1, e^{(\beta_i - \beta_j)(\chi(G_j) - \chi(G_i))}\right)$, facilitating dynamic adjustment to the exploration process.

We implement a parallelized version of the parallel tempering algorithm using OpenMPI. In this implementation, each replica runs independently on a separate MPI rank, except during the swap proposal phase. This approach allows for efficient execution of multiple replicas with minimal additional computational cost. Communication among the ranks involves a structured multi-step process. We assume an even number of replicas, with ranks

sequentially numbered starting from 0. In the first step, even-numbered ranks send their inverse temperature β_i and current error $\chi(G_i)$ to the next odd-numbered rank $i+1$. The rank $i+1$ evaluates α_{swap} to determine if a swap should occur. If a swap is to occur, rank $i+1$ sends rank i its own β_{i+1} and then both ranks exchange there own value of β for their neighbor's value; if not, no exchange happens. In the subsequent step, even-numbered ranks greater than zero send their β_i to the preceding rank $i-1$ that then decide whether to swap β values in a similar manner. A visual schematic of this communication process is illustrated in Fig. 5.1, providing a clearer understanding of the algorithm's structure.

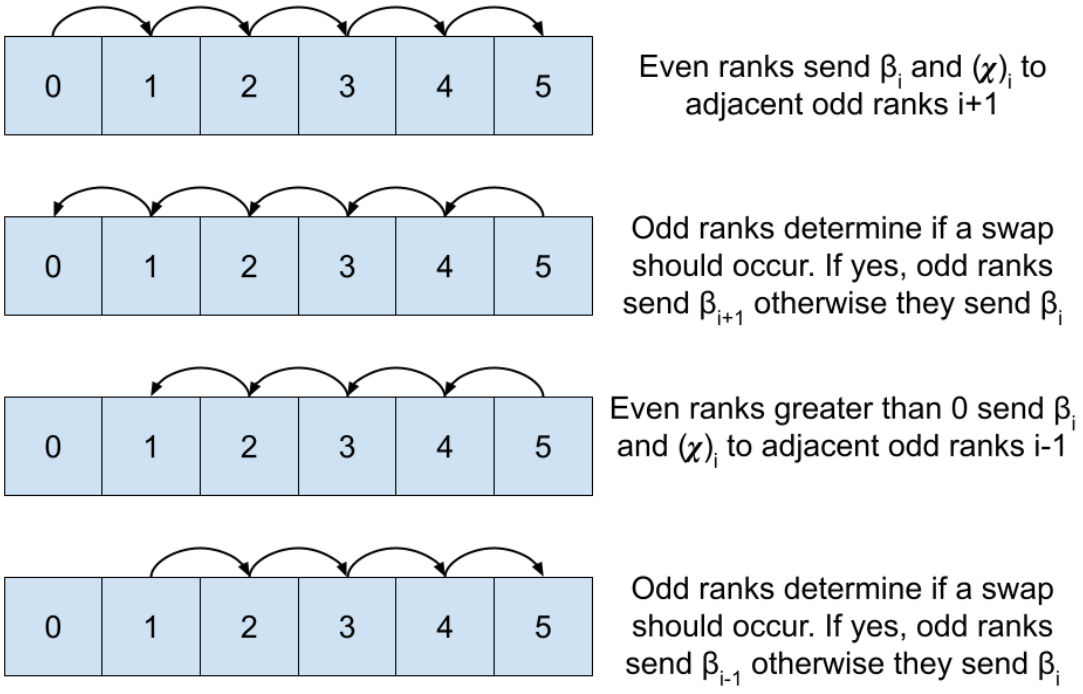


Figure 5.1: Schematic of communication between ranks. In the first step, even-numbered ranks send their current β_i and $\chi(G_i)$ forward. If the corresponding odd rank determines a switch should occur, it sends β_{i+1} backward and updates its own β_{i+1} to β_i, otherwise it sends β_i backward. Next, even-numbered ranks greater than 0 send their current β_i and $\chi(G_i)$ backward and a similar process is repeated.

In addition to the optimal $\bar{\beta}$ calculated in Sec. 5.3.1, we incorporate a range of β values to achieve different average acceptance probabilities. We introduce both higher (more exploitative) and lower (more exploratory) β values than $\bar{\beta}$. Here, we set acceptance probabilities for adjacent replicas to vary by a factor of two (e.g. $\bar{\beta}/2, \bar{\beta}, 2\bar{\beta}$). This strategy is designed to ensure a broad exploration of the solution space, while still effectively converging to op-

timal solutions. Employing the Metropolis Monte Carlo and parallel tempering algorithms as outlined, we shift to applying these techniques to the miniaturization of a social network, as detailed in the next section.

5.4 Results

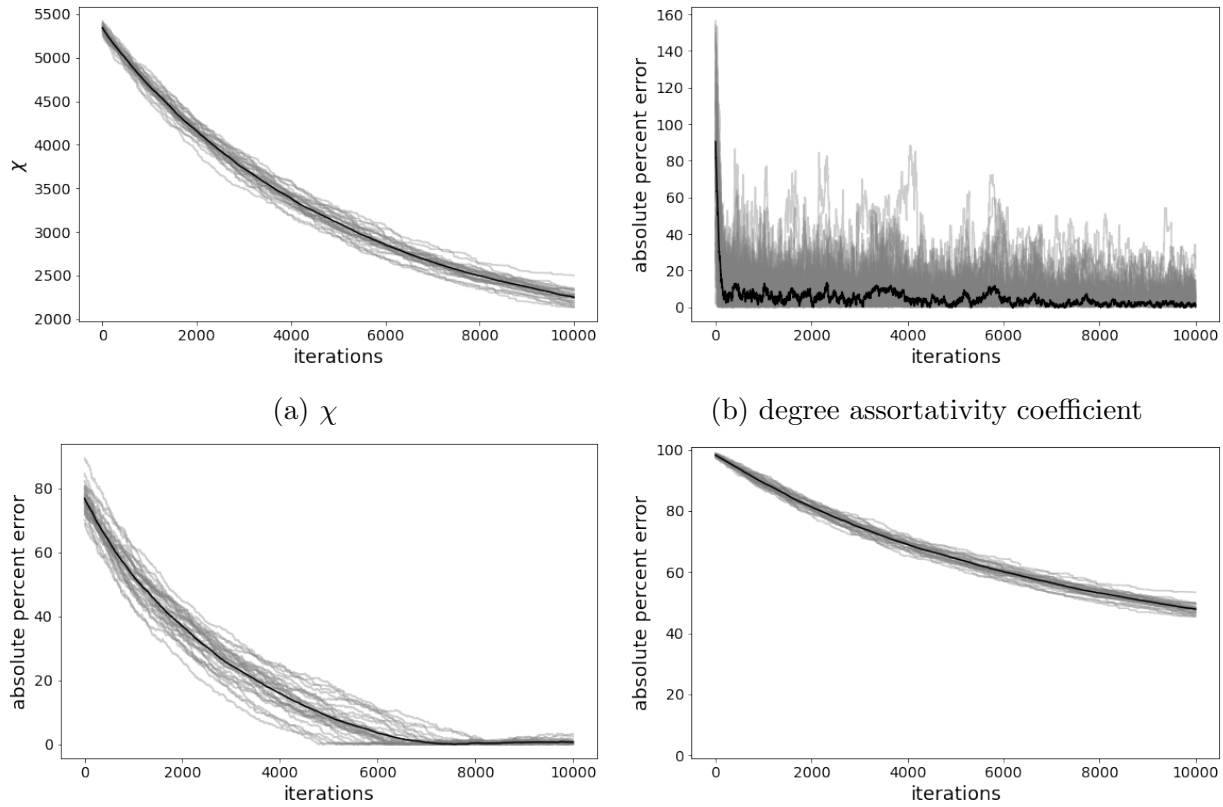
In this section, we present the results of applying our graph miniaturization methods to the Hamsterster online social network [188, 174]. This original network consists of 2426 nodes and 16,630 edges, representing individual users and their friendship connections, respectively (see Fig. 5.4). We miniaturized this network to 600 nodes, achieving an approximately fourfold reduction (see Fig. 5.5). The primary goal of this miniaturization was to closely match the original network's density, degree assortativity coefficient, and clustering coefficient.

For the miniaturization process, our seed graph was an Erdős-Rényi graph with 600 nodes and edge inclusion probability of .01. To determine the weights for the metrics, we conducted 10 independent MC simulations, each running for 100 iterations. These simulations were used to evaluate Eq. 5.3 for each metric. The computed average weights from these 10 simulations are listed in Table 5.1. Using these weights, we performed another set of 30 independent MC simulations, each running for 100 iterations, to calculate $\bar{\beta} = .78873$ per Eq. 5.4.

metric	weight
density	166081
degree assortativity coefficient	1146
clustering coefficient	8709

Table 5.1: Weights (ω) for metrics. These weights were determined by averaging the results from 10 independent MC simulations and evaluating Eq. 5.3. Using these values will weight each metric equally.

Using the calculated weights and $\bar{\beta}$, we executed the MC algorithm for 10,000 iterations. The outcomes of these simulations are shown in Fig. 5.2, where panel (a) illustrates the trajectory of χ, and panels (b), (c), and (d) display the absolute percent error in the degree assortativity coefficient, density, and clustering coefficient, respectively. In these figures, individual results from the 10 simulations are shown in grey lines, while the black line is the average trend. Notably, χ decreases smoothly across the simulations, indicating an overall successful miniaturization, with most metrics trending towards minimal error. However, matching the clustering coefficient proved to be more challenging, as evidenced by its persistent error rate of over 40% even after 10,000 iterations, in contrast to the near-zero error in the other metrics. The algorithm more readily captured the target network's density, showing a smoother and more steady convergence towards zero error, compared to the degree assortativity coefficient, which also showed improvement but with more variation.



(a) χ (b) degree assortativity coefficient

(c) density (d) clustering coefficient

Figure 5.2: Loss (χ) and target metrics versus iterations for 30 MC runs. The grey lines are the results from individual runs, and the black line is the average of the result. Overall, χ is smoothly decreasing for each of the runs. The degree assortativity coefficient and density are being matched well on average, though the degree assortativity coefficient has much more variance. While the clustering coefficient is decreasing, it struggles to get below 50% error. Even with the challenge of matching clustering, a single MC run is trending towards the target metrics.

Following from this example we turn to the parallel tempering algorithm. Here we chose to use 6 replicas. This includes $\bar{\beta}$, three lower and two higher betas. Here the set of $\beta_i = \{1.90472, 1.53273, 1.16073, 0.78873, 0.41674, 0.04474\}$. In Fig. 5.3, we show the trajectory of χ of each replica and their values for each metric, in addition to the β value each replica has throughout the simulation. With the exception of the trajectory with the lowest β, χ is trending downward for all replicas. As with the single replica run, the degree assorativity coefficient and density are fairly easy to match though with more variance in the degree assortativity coefficient. Given enough iterations, replicas with higher β values correctly capture the clustering coefficient as well. Exchanges of β between replicas mostly happen during isolated parts of the simulation and mostly occur between the central values. The

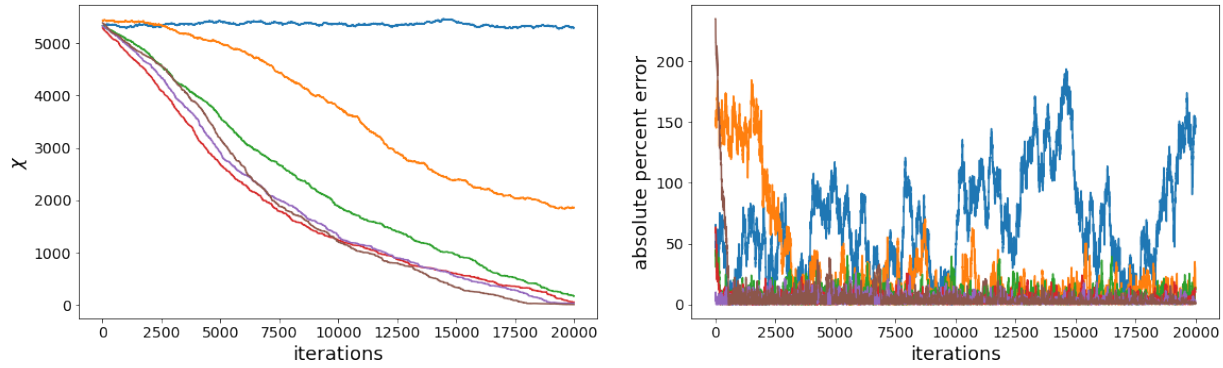
steady decrease in error among the metrics, coupled with the the observation that replicas with higher β values perform better suggest that the error surface has a dominating minimum that is fairly easy to find. This is further illuminated by the replicas with low β values struggling to find a region where the metrics significantly deviate enough to outperform other replicas.

To highlight the similarity in the miniaturized and target networks structures, we display the target graph and best miniaturized graph in Fig. 5.4 and Fig. 5.5, respectively. Both of these graphs were drawn using the force atlas algorithm in Gephi [189]. The target network consists of 2426 nodes and 16,630 edges, while the miniaturized one consists of 600 nodes and 1002 edges. Many of the features displayed in these figures are driven by the method used to draw them; however, both appear similar in that they have a centralized connected component surrounded by smaller sub-graphs. In both graphs, the central component also appears to have long strings of connected nodes.

To further examine how similar the generated graph's metrics are to the target graph's metrics, we list the value of the metrics after 20,000 iterations, as well as their initial values and the target values in Table 5.2. In agreement with Fig. 5.2, we see that all of the metrics converge to values near their targets.

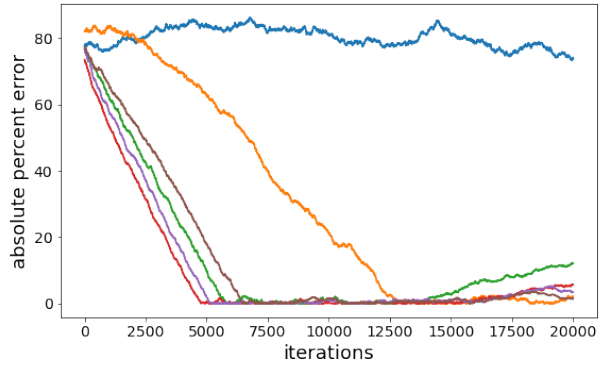
metric	initial	final	target
degree assortativity coefficient (scaled)	0.46803	0.52333	0.52370
degree assortativity coefficient (unscaled)	−0.06393	0.04667	0.04740
density	0.01002	0.00557	0.00565
clustering coefficient	0.01003	0.53757	0.53753

Table 5.2: The initial and final metrics of a miniaturized network versus the target metrics. As displayed in Fig 5.2, each of the metrics eventually converge to their target values.

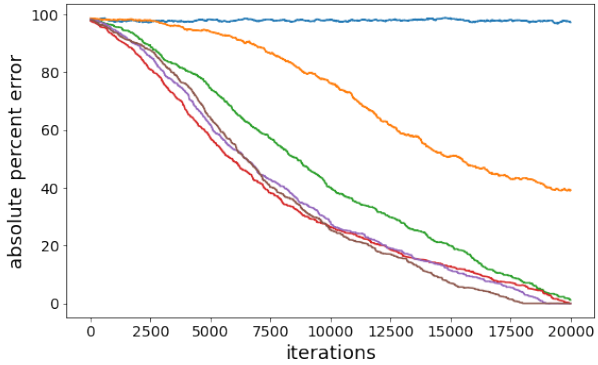


(a) χ

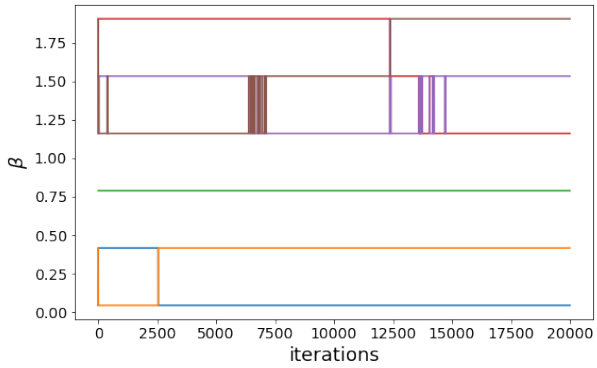
(b) degree assortativity coefficient



(c) density



(d) clustering coefficient



(e) β

Figure 5.3: Loss (χ), target metrics, and β versus iterations for a parallel tempering run, with each color representing a replica. As with the single replica run, the degree assortativity coefficient and density are matched quickly, with more variation in the degree assortativity coefficient. Given enough iterations the cluster coefficient is also matched while preserving the other metrics. The replica with the smallest β explored many configurations but did not share information outside of one exchange.

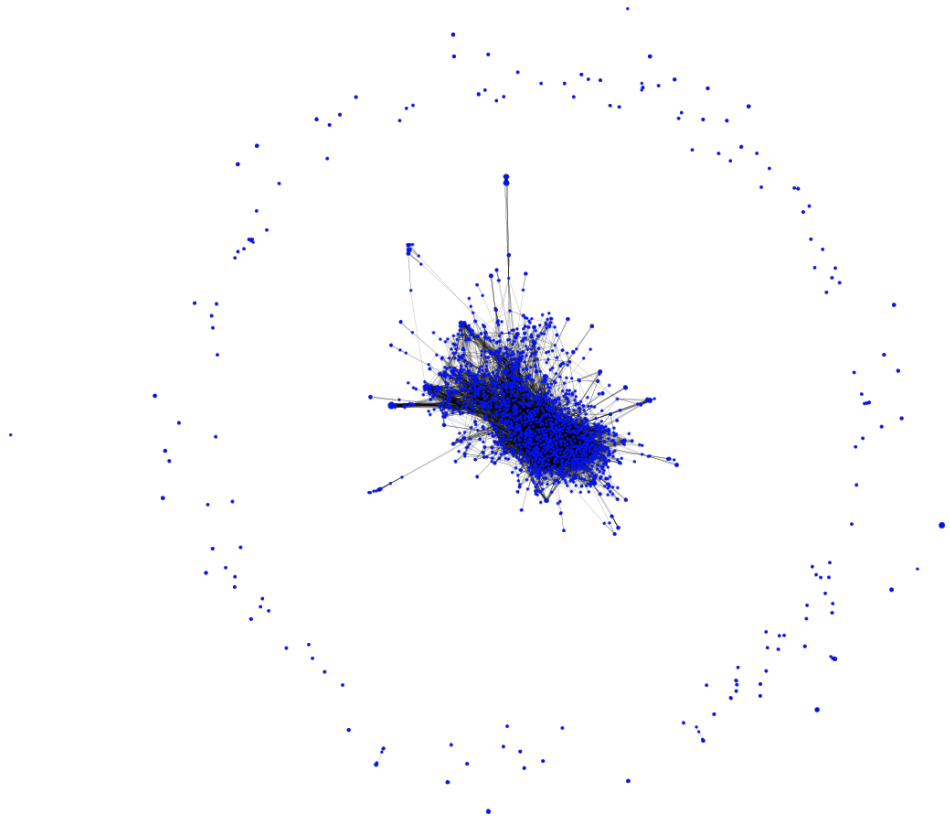


Figure 5.4: Hamsterster social network. Blue nodes represent users of the online social network, and black edges represent a friendship between users. This graph has 2426 nodes and 16,630 edges. The attributes we aim to match for this graph are listed in Table 5.2.

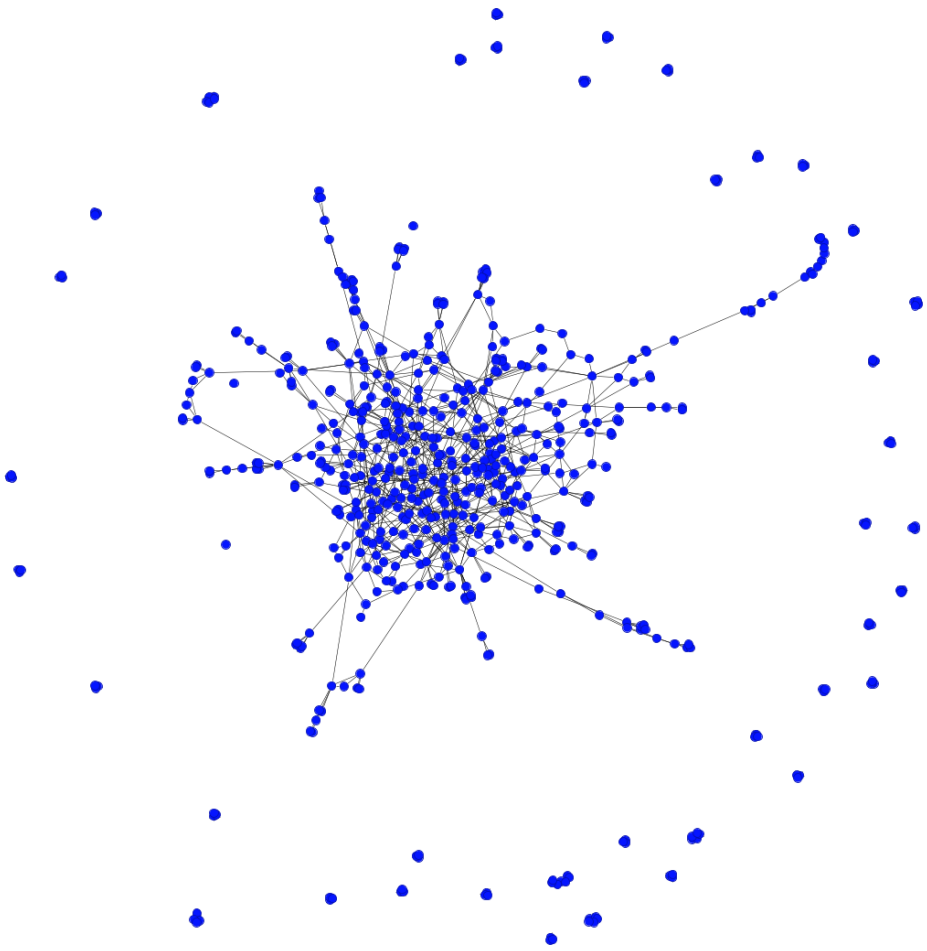


Figure 5.5: Example of a miniaturized graph. This graph is a miniaturized version of the Hamsterster social network. Similar to Fig. 5.4, the blue nodes represent users of the social network and the black edge represent friendships between users.

5.5 Discussion

In this work we proposed a method for miniaturizing a target network such that the resulting network has attributes that resemble the target one. To achieve this, we implemented a version of the parallel tempering algorithm. We present a method for selecting the optimal β for a single replica, and then produce β values for additional replicas from the optimal one. Our implementation of the parallel tempering algorithm is parallelized in such a way that allows for additional replicas to be added with minimal computational cost. Using our algorithm, we presented an example of successfully miniaturizing a graph by approximately fourfold, while preserving the degree assortativity coefficient, density, and clustering coefficient.

In our example, we were careful to chose a target network that was large enough to be miniaturized by a fair amount but not so large we could not display it. Another advantage to our target network, was having the ability to exactly calculate our metrics of interest in a short time. Nonetheless, our methods can be applied to any network, so long as the target metrics can be calculated.

In our exploration of miniaturizing graphs a single replica performed fairly well. We tuned β by running preliminary simulations and adjusting β until it accepted around 23% of the suggested changes. While mildly tedious, it produced decent results by matching two of the three metrics and tending towards matching the last. By shifting to parallel tempering we were able to achieve much better results. However, the main benefit of gaining exploration did not aid our search. While there were exchanges between the replicas, they were not among the best performing replicas. We believe this is mostly driven by the metrics we chose to match causing the error surface to not be very complex. However, different sets of parameters very well could complicate the error surface providing a better use case for parallel tempering.

While the need to balance the exploration-exploitation trade off was not prevalent in this work, we still prefer the multi-replica algorithm over the single replica one because our algorithm scales with negligible computational cost as replicas are added. It is worth noting that care should be taken in the number and spacing of replicas [190]. Additionally, our choice for the optimal β was not the best performing replica as it intentionally allows for missteps to avoid local optima. However, without running multiple long simulations it is not possible to know the complexity of the error surface. Therefore, we suggest using our method for determining an optimal β and ensuring replicas with higher and lower values are added to the simulation.

This work shows that it is possible to miniaturize a network, while preserving multiple attributes. Future studies should investigate how the reduction in the graph relates to the error. For example, if the graph is miniaturized by too large of an amount, it will no

longer be possible to match metrics, and conversely a small reduction should prove easy to match metrics. However, the exact nature of this relation is unknown. We examined a set of metrics we thought were important, but there is a quasi-infinite number of metric combinations that could be studied. At a minimum, exploring various combinations of metrics to determine the difficult in matching them would be useful for future studies. In a similar vein, the initialization of graphs can be done such that they start closer to the target metrics. Perhaps the use of greedy algorithms before the parallel tempering could aid in finding optimal solutions. Most importantly, future work should study the amount of error incurred from the miniaturization process and tie these errors to the metrics that are matched. Specifically, investigating how much a result should be expected to change as a function of reduction amount and metrics preserved.

The work presented here is an important first step to miniaturizing social network for use in simulations. Our parallel tempering algorithm with its excellent scaling and ability to match metrics is a valuable contribution to optimizing graphs.

CHAPTER 6:
STOCHASTIC DIFFERENTIAL EQUATION BASED MODELING OF DETERRENCE

6.1 Summary

This chapter begins the shift to the final facet: machine learning. The main objective through this chapter the development of a differential equation-based model of deterrence, the results of which will inform decisions in future work on applying ABMs to modeling conflict. The development of this model is my main contribution in the chapter. The analysis of this model is in its early stages. Nonetheless, two types of global sensitivity analyses are performed on a version of the model geared towards modeling short-term conflicts. Utilizing the Python package SALib a variance-based Sobol and a PAWN global sensitivity analysis are performed to examine the sensitivity of conflict outcomes to various parameter of the model. Two profiles for competing actors are defined and the sensitivity analyses are performed on the 3 possible combinations of actors.

6.2 Introduction

Conflict and deterrence have been central themes in the study of international relations and military strategy for centuries [191, 192, 193]. The relevance of these themes has evolved with the changing nature of warfare, prompting the need for sophisticated analytical tools to understand and predict conflict dynamics. The use of mathematical and computational methods in this domain can be traced back to pioneers like Thomas Schelling, whose work in game theory and strategic analysis laid the groundwork for quantitative approaches to conflict studies [194]. The importance of such studies lies not only in their academic value but also in their practical applications. Governments and military strategists have long sought effective means to deter potential adversaries, and understanding the underlying mechanics of conflict can inform more effective policies and strategies. The mathematical modeling of conflict allows for the analysis of complex systems and interactions, providing insights that are often not apparent through qualitative methods alone.

Over the years, various models have been developed to analyze conflict, each with its unique focus and limitations. For instance, the Lanchester model [195], is renowned for its application in conventional warfare scenarios, providing insights into force attrition and the outcomes of direct engagements. Many real world battle have been studied through the lens of Lanchester [196, 197, 198, 199], and various extensions of Lanchester models have been proposed, such three-way combat [200], irregular warfare [201], and adding spatial components [202]. Another example, the Richardson model [203], delves into the dynamics of arms

races, highlighting the feedback loops inherent in competitive armament policies. This model has also applied to Vietnam war [204] and middle east [205]. Guerrilla warfare models, such as the one developed by Deitchman [206], offer perspectives on asymmetric conflicts where conventional military metrics may not apply. While these models have significantly contributed to our understanding of conflict dynamics, they primarily address tactical aspects of warfare. Strategic, long-term considerations, particularly in the realm of deterrence, have been less emphasized. Additionally, these models often do not account for the psychological elements of conflict, such as the mindset of actors regarding deterrence, nor do they adequately incorporate the stochastic nature of long-term conflict dynamics.

This chapter introduces a mathematical model that extends the traditional framework of conflict analysis to address these gaps. Our model integrates the foundational principles of the Lanchester model with new components that reflect the strategic, long-term aspects of conflict and deterrence. Key innovations of our model include:

1. A system of ordinary differential equations (ODEs) that goes beyond fight-to-the-finish scenarios, reflecting long-term strategic considerations.

2. Explicit accounting for the role of deterrence, not just as a consequence of initial conditions but as an integral component of the conflict dynamics.

3. Inclusion of psychological state equations, modeled as stochastic differential equations (SDEs), to represent the deterred or undeterred mindset of actors, incorporating a pitchfork bifurcation mechanism.

4. The introduction of noise terms in the psychological state equations to capture random fluctuations and events over extended periods. Coupling terms that link the psychological state of actors to their resource levels, aligning with the deterrence model.

5. Additional terms to enhance realism, including a surrender term that allows actors to withdraw under certain conditions and a refresh term that simulates rebuilding capabilities between conflicts.

By incorporating these elements, our model provides a more comprehensive and realistic framework for analyzing conflict and deterrence over extended periods. This approach not only fills existing gaps in the literature but also offers practical insights for policymakers and strategists seeking to understand and navigate the complex landscape of modern conflicts.

6.3 Methods

Our model is detailed in this section. Because the model describes several processes, we develop the model progressively in several subsections beginning with a review of the Lanchester model in the next subsection. We extend this model to include a resource refresh term. In the following subsection we introduce a psychology model that reflects the mindset of the actors; this model is stochastic to reflect long-term random events that trigger conflicts and couples to the extended Lanchester model.

6.3.1 Lanchester's models of conflict: Review

Lanchester models are a widely used class of conflict models used to study combat dynamics. Typically consisting of systems of coupled ordinary differential equations, these models predict the outcome of fight-to-the-finish conflicts. The models revolve around key parameters, α_x and α_y, representing the combat strengths of two opposing actors with resource levels x and y. For simplicity, we label the actors by the resources they control, i.e. actor x and actor y. While Lanchester models often look similar, they model different quantities, e.g., personnel or armaments. Here, we treat x and y as general resources that can be adapted to other quantities through post-processing. This flexibility allows for broader applications, such as equating resource depletion to specific outcomes like causalities in a battle. Minguel-Castro et al. [193] overview various adaptations of Lanchester models tailored to different conflict scenarios. However, we build upon Lanchester's foundational models [195], aiming to integrate stochastic events and psychological aspects of deterrence, thereby extending the models' applicability to modern, multifaceted conflict environments.

Lanchester proposed two fundamental models to predict conflict dynamics in one-on-one and all-on-all scenarios. Both of these models describe the time evolution of the number of combatants available to each actor. The one-on-one model, formulated as

$$\frac{dx}{dt} = -\alpha_y y x, \tag{6.1}$$

$$\frac{dy}{dt} = -\alpha_x x y, \tag{6.2}$$

assumes the loss rate of each actor is proportional to the size of the opposing force and their own force. Such a model resembles scenarios akin to series of individual duels among the combatants. In contrast, the all-on-all model represented by

$$\frac{dx}{dt} = -\alpha_y y, \tag{6.3}$$

$$\frac{dy}{dt} = -\alpha_x x, \tag{6.4}$$

assumes losses are proportional to the size of the opposing force resembling a scenario where combatants are engaged in collective battle. This fundamental distinction in the models provides different insights into conflict dynamics, with the one-on-one model emphasizing the impact of mutual engagement, while the all-on-all model focuses on the sheer force size.

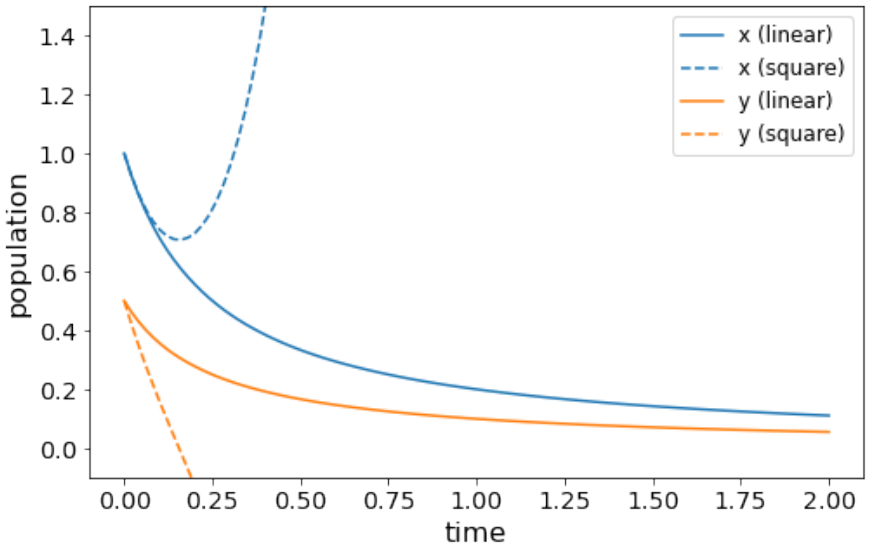


Figure 6.1: Example of Lanchester's laws. In this example each actor's available resources are displayed with either a solid or dashed lines. The two colors represent which of Lancheser's laws describe the dynamics of the actors' resources.

A comparison of these models is illustrated in Fig. 6.1, with parameters set at $\alpha_x = 4$ and $\alpha_y = 8$. The initial resources available to actor x are depicted in blue, and those of actor y in orange. Solid lines represent the one-on-one model, while dashed lines correspond to the all-on-all model. In both scenarios, actor x emerges as the victor. Notably, the all-on-all enters a nonphysical regime once an actor's resources are depleted; traditionally, the results past this point are disregarded. Conversely, the one-on-one is unaffected by this issue due to its inherent non-linear dynamics. This example, among many possible parameter sets and initial conditions, illustrates the foundational principles of these models. However, from these models general laws can be derived. From Eq. 6.1 and Eq. 6.2 Lanchester's linear law can be derived. This law states that team x wins if $\alpha_x x_0 > \alpha_y y_0$. That is, there is an equal exchange for the number of fighters and their strength in determining the outcome. Similarly from Eq. 6.3 and Eq. 6.4 Lanchester's square law can be derived. This law states that team x wins if $\alpha_x x_0^2 > \alpha_y y_0^2$. Here the initial number of fighters have a disproportional importance compared to their strength.

Due to the non-physical behavior of the all-on-all model, we will use only the one-on-one

variant of Eqns. 6.3 and 6.4 in what follows. Additionally, when fitting Lanchester's models to real battle data, researcher have found similar performance among the all-on-all and one-one-one models [197]. In the next subsection we add a term needed to extend the Lanchester model to longer time scales over which actors can regroup and replenish their resources. We then turn to adding the psychological state of the actors x and y.

6.3.2 Lanchester's models of conflict: Replenish

We are interested in long-term strategic modeling, and thus we require a term that allows actors to replenish their resources. We introduce a resource equation of the form

$$dx/dt = g(1 - x/C), \tag{6.5}$$

which includes a growth rate g of resources and a term $(1 - x/C)$ that limits the growth to a capacity C. With this term alone, the resources obey

$$x = C - (C - x(t_0))e^{g(t_0-t)/C}, \tag{6.6}$$

which steadily tends toward C on time scale g^{-1}. Combined with the Lanchester model, we obtain

$$\frac{dx}{dt} = -\alpha_y yx + g_x \left(1 - x/C_x\right), \tag{6.7}$$

$$\frac{dy}{dt} = -\alpha_x xy + g_y \left(1 - y/C_y\right). \tag{6.8}$$

6.3.3 Psychology model

One of the main modifications of Lanchester models we considered was the addition of a psychological state for the actors. We incorporated such a state as a random walk on a potential energy surface. The stochastic differential equations that dictate the psychological state consist of multiple terms that capture physical phenomenon. In Section 6.3.3 we discuss the first of such terms that constructs a quasi-binary state resembling the willingness of an actor to engage in conflict. Then in Section 6.3.3 we discuss the threat-response term that tries to match the psychological states of the actors. Next in Section 6.3.3 we discuss a term that captures an actor surrendering due to excessive losses. Lastly, we address our deterrence in Section 6.3.3 that utilizes Lanchester's square law to inform an actor's psychological state.

Quasi-binary state

To introduce a quasi-binary state into our model, we utilized a pitchfork bifurcation, commonly characterized by two stable and one unstable fixed points under certain conditions. We define the psychological potential function as

$$U(S) = S^4 - 2S^2, \tag{6.9}$$

where S represents the psychological state of an actor. This potential, visualized in Fig. 6.2, has minima at $S = \pm 1$ and a local maximum at $S = 0$. These minima correspond to distinct psychological states: near $S = -1$ is the soft war state, characterizing a period of non-engagement, and near $S = 1$ is the total war state, representing full combat engagement. The stability of these minima and instability of the local maximum at $S = 0$ imply that an actor's state will settle near one of these two extremes. The variable γ controls the randomness of transitions between the soft and total war states. Larger values of γ allow for more frequent shifts between soft and total war, reflecting the unpredictable nature of an actor's decision-making process under varying external pressures

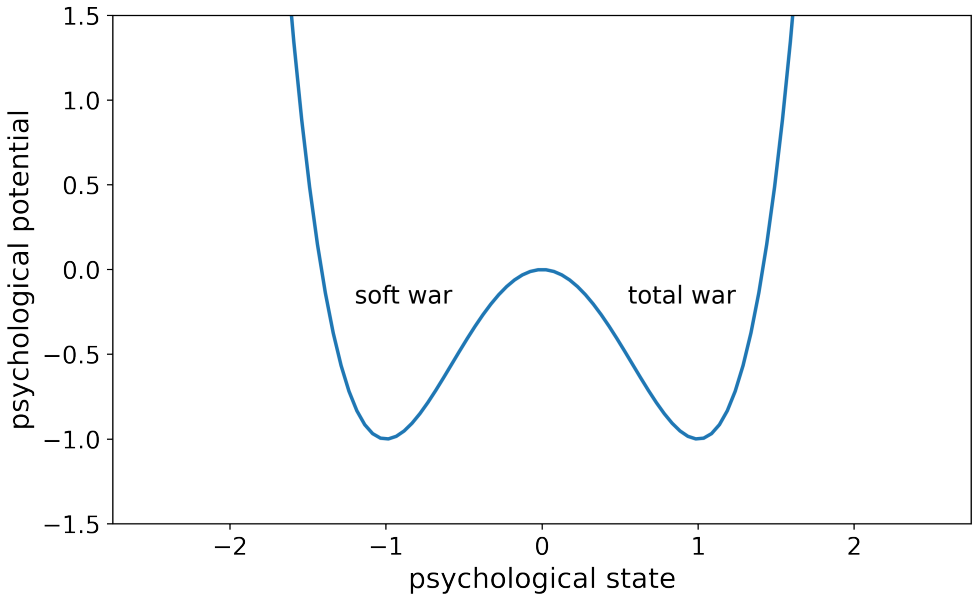


Figure 6.2: Psychological state potential energy surface. When an agent is in the left well, they are in a peacetime state, and utilize few resources. If an actor is in the right state, they will use resources to their fullest extent. Using this potential to describe the psychological state of the actors creates a quasi-binary state system.

An actor's psychological state is described by the stochastic differential equation

$$\frac{dS}{dt} = -4S^3 + 4S + \gamma\eta, \tag{6.10}$$

where η represents non-correlated Gaussian white noise, characterized by $\langle\eta\rangle = 0$ and $\langle\eta(t)\eta(t')\rangle = \delta(t - t')$. This equation captures the dynamics of the psychological state (S), where the term $-4S^3 + 4S$ reflects the tendency of the state towards soft and total war. The addition of $\gamma\eta$ simulates the unpredictable nature of psychological responses in conflict situations.

The evolution of this psychological state is illuminated through the Fokker-Planck equation, which describes the evolution of the probability density function $P(S, t)$:

$$\frac{\partial P}{\partial t} = \frac{\partial}{\partial S}\left(\frac{dU}{dS}P\right) + \gamma\frac{\partial^2 P}{\partial S^2}; \tag{6.11}$$

(see [207] for a review). This equation predicts how the distribution of psychological states evolves, providing insights into the likelihood of different states over time. The steady-state probability density function is given by

$$P(S) = \frac{N}{\gamma}\exp\left(-\frac{U(S)}{\gamma}\right), \tag{6.12}$$

where N is a normalization constant. This equation describes a long-term view of the psychological states, indicating the probabilities of an actor being in various states under stable conditions.

In Fig. 6.3 we visualize the psychological potential (shown in blue) with the steady-state probability density function $P(S)$ for different values of γ – .25 in orange, 1 in green, and 4 in red. This illustration highlights how the size of random steps that is controlled through γ influences the likelihood of an actor's psychological state transitioning between states. When the size of random steps is lower, indicated by lower values of γ, the psychological state is most likely to be near one of the minima, as indicated by the pronounced peaks in $P(S)$. Conversely, when γ is higher, the size of random steps is larger and an actor's psychological state is less influenced by the potential. This is indicated by $P(S)$ tending towards a uniform distribution. Looking ahead, we will further develop our model by assigning individual psychological states S_x and S_y to each actor, laying the groundwork for additional, more complex terms.

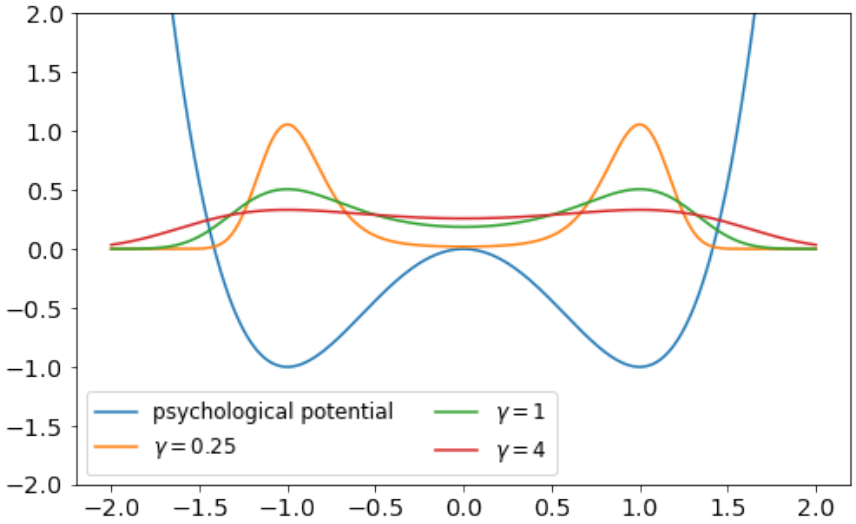


Figure 6.3: Solutions of Eq. 6.12 for various values of γ. The psychological potential is displayed in blue, with three values of the size of random steps in Eq. 6.10. The orange, green, and red lines display the probability of a random walker being found at a certain location. When the step sizes are small, the random walker is most likely to be found in one of the wells. As the temperature increases, the distribution approaches a uniform distribution, meaning the random walker is likely to be found in or between the wells.

Threat-Response

Building upon our model's framework of psychological states, we now introduce a new component: the threat-response term. This term is designed to dynamically adjust an actor's psychological state in response to perceived changes in their enemy's stance. When an actor perceives that their enemy is in a higher psychological state, this term acts to elevate the actor's own state, potentially increasing the rate resources are utilized. Conversely, if an actor's psychological state is higher than their enemy's, the term works to moderate their state, aligning closer to the enemy's level. This mechanism allows for modelling strategic adjustments made by actors in response to their enemy's actions. The relative strength of the threat-response term is controlled by a constant β; higher values of β imply more pronounced adjustments.

For actor x, the threat-response term is expressed as $\beta(S_y - S_x)$, and for actor y it is

$\beta(S_x - S_y)$. We integrate the threat-response term with the quasi-binary state as

$$\frac{dS_x}{dt} = -4S_x^3 + 4S_x + \frac{\beta}{2}(S_y - S_x), \tag{6.13}$$

$$\frac{dS_y}{dt} = -4S_y^3 + 4S_y + \frac{\beta}{2}(S_x - S_y), \tag{6.14}$$

where each actor has their own psychological state denoted by a subscript. To gain insight on how this term alters the dynamics, we examine the location and stability of fixed points of the system, listed in Table 6.1 and visualized in Fig. 6.4.

S_x	S_y
0	0
1	1
-1	-1
$\sqrt{4-\beta}/2$	$(4-\beta)(\sqrt{4-\beta}-4+\beta/2)/\beta$
$-\sqrt{4-\beta}/2$	$-(4-\beta)(\sqrt{4-\beta}-4+\beta/2)/\beta$
$\sqrt{-Z-\beta+8}/4$	$\sqrt{8-Z-\beta}(\beta-Z-8)/8$
$-\sqrt{-Z-\beta+8}/4$	$-\sqrt{8-Z-\beta}(\beta-Z-8)/8$
$\sqrt{Z-\beta+8}/4$	$\sqrt{Z-\beta+8}(\beta+Z-8)/8$
$-\sqrt{Z-\beta+8}/4$	$-\sqrt{Z-\beta+8}(\beta+Z-8)/8$

Table 6.1: Fixed points and stability of fixed points for threat-response term. Here $Z = \sqrt{-3\beta^2 - 16\beta + 64}$.

In our analysis, nine fixed points are identified, though their presence and positions depend on the values of β. Particularly noteworthy are $\beta = 0$, $\beta = 8/3$, and $\beta = 4$, where the fixed points tend to align or overlap.

These these alignments and convergences, are demonstrated in Fig. 6.4 through quiver plots for $\beta = 0.01$ (Fig. 6.4a), $\beta = 1$ (Fig. 6.4b), $\beta = 3$ (Fig. 6.4c), and $\beta = 5$ (Fig. 6.4d). In these figures, stable nodes, saddle points, and unstable nodes are represented as blue squares, purple circles, are red diamonds, respectively. The arrows indicate the derivatives of S_x and S_y with respect to time, illustrating the evolution of the actor's psychological states and the speed of change.

When β is near zero, the system has nine fixed points: four stable nodes at $(\pm 1, \pm 1)$ and $(\pm 1, \mp 1)$, four saddles points at $(0, \pm 1)$ and $(\pm 1, 0)$, and an unstable node at $(0, 0)$. In this regime, each actor acts independently, influenced predominately by the quasi-binary state. This independence results in actors gravitating towards the nearest minima unless starting at a psychological state of zero. As β increases, the saddle-points at $(-1, 0)$ and $(0, 1)$ approach the stable node at $(-1, 1)$, while the saddle points $(0, -1)$ and $(1, 0)$ approach the stable node at $(1, -1)$. Notably these groups of fixed points are approaching $(0, 0)$ along the line

$S_x = -S_y$. At $\beta = 8/3$, these points collide, transforming into two saddle points. With further increases to β, these saddle points continue to move towards the central unstable node, until at $\beta = 4$, they merge into a single saddle point at $(0, 0)$. Throughout these changes, the stable nodes at $(\pm 1, \pm 1)$ remain fixed. When β exceeds 4, actors are more inclined to align their psychological states with their enemies. If positioned symmetrically on either side of zero, (on the line $S_x = -S_y$), actors are drawn to the $(0, 0)$ state. Otherwise, their states are attracted to a soft war for negative dominate psychological states, or total war for positive ones.

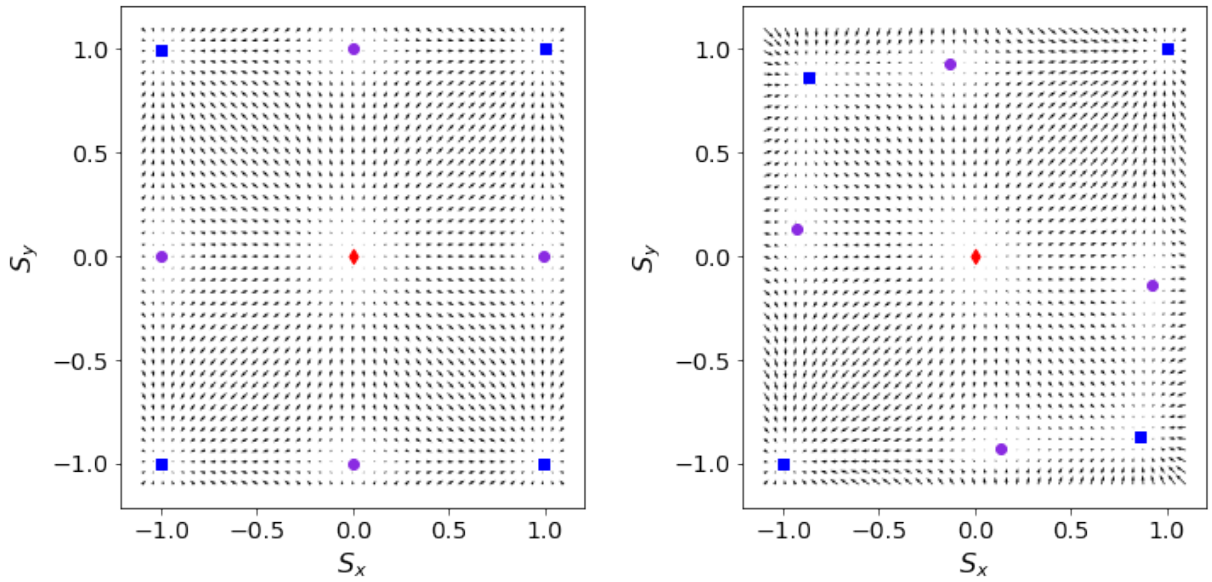
The dynamics of these fixed points align with the anticipated behavior of a defensive term in our model. Actors who assign a lower weight to their enemy's psychological state remain largely unaffected by their enemy's state, indicating independent decision making. In contrast, as the weight increases, actors tend to mirror their enemy's psychological state, reflecting adaptive strategies in response to changing conflict dynamics. Crucially, all fixed points are bounded within the region $[-1, -1] \times [1, 1]$, ensuring that the values of S_x and S_y remain in this realistic range, regardless of the value of β. This boundedness prevents the psychological state of the actors from reaching unrealistic extremes such as ∞ or $-\infty$, thereby maintaining the model's applicability to real-world scenarios.

Surrender

Having established the dynamics of the threat-response term, we now turn our attention to another aspect of our model: the surrender term. This term plays a role in simulating the conditions and consequences of surrender, and further enhancing our understanding of decision-making in conflict scenarios. Comprising of two components, the surrender term firstly acts as a switch that turns of the term once an actor's psychological state reaches $S = -1$. The second component models the reduction in an actor's psychological state proportional to a losses in resources, characterized by R. This captures the diminishing morale and willingness to continue conflict as losses mount. The influence of this term is controlled by the constant, ϵ. For actor x, the term is $\epsilon\,(S_x + 1)\left(x - \frac{1}{R}\right)$, and $\epsilon\,(S_y + 1)\left(y - \frac{1}{R}\right)$, for actor y.

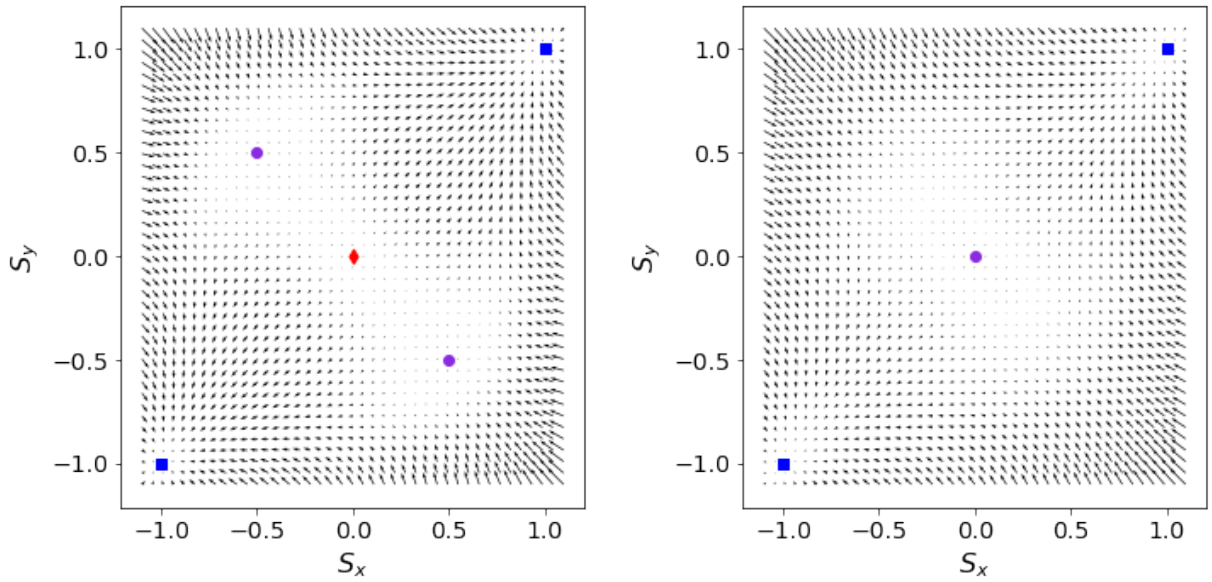
Considering the coupling of these terms to the available resources, identifying and visualizing fixed points for this system becomes challenging. Nonetheless, we can examine the influence of the surrender term on the psychological state of an individual actor. For simplicity, let's consider the psychological state of actor x describe by the equation:

$$\frac{dS_x}{dt} = -\epsilon\,(S_x + 1)\,(x - R)\,. \tag{6.15}$$



(a) When β approaches 0, the saddle-points align with the stable points. This alignment in combination with the unstable node push the psychological state towards the nearest corner.

(b) Here we let $\beta = 1$. Once β hits 8/3 these saddle-points combine with the stable nodes, resulting in two saddle-points.



(c) Here we let $\beta = 3$. Once β is greater than 8/3, the resulting saddle-points tend towards the center.

(d) Here we let $\beta = 5$. Once β passes 4 there is a saddle-point at the center with stable nodes at ($\pm$1,$\pm$1). Actors will either be in a total war or peace state.

Figure 6.4: Stable points as a function of β, where blue squares are stable nodes, purple circles are saddle points, and red diamonds are unstable nodes.

We simplify our analysis by assuming that losses, $l = (x - R)$, remain constant. Using this assumption, the solution to Eq. 6.15 is

$$s(t) = (s_0 + 1)e^{-\epsilon l t} - 1, \tag{6.16}$$

where s_0 is the initial psychological state of the actor. This result shows that the psychological state of the actor exponentially decays to -1, indicating an increasing inclination to surrender as losses continue. The rate of this decay is controlled by the weight ϵ and losses, l. An actor with a lower value of ϵ will have a slower reaction to accumulating losses, reflecting a more resilient or less responsive attitude towards surrender, compared to an actor with a higher ϵ, who will react more swiftly. This variation can be interpreted as different tolerances for surrendering among actors in conflict scenarios.

Confidence-Deterrence term

The final component we introduce to our model is the integration of confidence and deterrence factors. Traditional one-shot fight-to-the-finish models implicitly incorporate deterrence within their logical structure, but often do not explicitly model it. An actor's decision to engage in battle is based on their perceived performance, e.g. evaluating one of Lanchester's laws. Only if an actor decides to engage is the model evaluated. In contrast, our model seeks to explicitly incorporate deterrence into the psychological states of the actors. This includes not only a weaker actor being deterred, but also the strong actor gaining confidence based on strategic assessments. Specifically, each actor evaluates their likelihood of success by applying Lanchester's square law, comparing the combat strengths $\alpha_y y^2$ and $\alpha_x x^2$ weighted by δ. This approach allows us to simulate not only the physical aspects of conflict but also the psychological warfare that often precedes and dictates the course of engagements.

In our model, each actor evaluates their combat strength by subtracting it from their enemy's strength. For actor x this has the form $\alpha_x x^2 - \alpha_y y^2$ and for actor y its $\alpha_y y^2 - \alpha_x x^2$. An actor who perceives a higher combat strength gains confidence, leading to an increase in their psychological state. Conversely, an actor who determines that they are at a combat disadvantage is deterred, and their psychological state is decreased correspondingly.

6.3.4 Lanchester with deterrence

Using the discussed terms, we move to couple the equations for resources available to and psychological states of the actors. The model is further generalized to allow for each actor

to separately weight each term, denoted by subscripts. The full model is given by:

$$\frac{dx}{dt} = -\alpha_y y x \mathcal{B}(S_y) + g_x \left(1 - \frac{x}{C_x}\right), \tag{6.17}$$

$$\frac{dy}{dt} = -\alpha_x x y \mathcal{B}(S_x) + g_y \left(1 - \frac{y}{C_y}\right), \tag{6.18}$$

$$\frac{dS_x}{dt} = -4S_x^3 + 4S_x + \frac{\beta_x}{2}(S_y - S_x) + \frac{\epsilon_x}{2}(S_x + 1)(x - R_x) + \delta_x(\alpha_x x^2 - \alpha_y y^2) + \gamma_x \eta_x, \tag{6.19}$$

$$\frac{dS_y}{dt} = -4S_y^3 + 4S_y + \frac{\beta_y}{2}(S_x - S_y) + \frac{\epsilon_y}{2}(S_y + 1)(y - R_y) + \delta_y(\alpha_y y^2 - \alpha_x x^2) + \gamma_y \eta_y, \tag{6.20}$$

where $\mathcal{B}$ bounds S_y and S_x between -1 and 1. Each of the parameters and the range of values they can take is listed in Table 6.2.

variable	description
$0 \leq x \leq 1$	fraction of total available resources for actor x
$0 \leq y \leq 1$	fraction of total available resources for actor y
$-\infty \leq S_x \leq \infty$	psychological state of actor x
$-\infty \leq S_y \leq \infty$	psychological state of actor y
$0 \leq \beta_{x(y)} \leq 1$	strength of threat-response term
$0 \leq \epsilon_{x(y)} \leq 1$	strength of surrender term
$0 \leq R_{x(y)} \leq 1$	losses an actor is willing to take willing to take
$0 \leq \delta_{x(y)} \leq 1$	strength of confidence-deterrence term
$0 \leq \gamma_{x(y)} \leq 1$	controls size of random jumps
$0 < C_{x(y)} \leq 1$	maximum number of resources for an actor
$0 < \alpha_{x(y)} \leq 1$	fighting strength of an actor
$0 \leq g_{x(y)} \leq 1$	growth rate of an actor's resources

Table 6.2: Model parameters and acceptable values.

While these parameters can be set to any number in their range, we construct a few sets of parameters to represent types of actors. The sets are broken down into two types based on an actor's risk aversion. A conservative actor is less likely to accept losses, more likely to surrender, have medium-level of confidence, and more likely to respond to an enemy with a matched response compared to a reckless actors. These traits are captured for conservative actors by assuming $R = 1$, $\epsilon = 1$, $\delta = .5$, and $\beta = 1$, while for reckless actors we assume $R = 0$, $\epsilon = 0$, $\delta = 1$, and $\beta = 0$. In both cases, we set $C = 1$ to represent each actor attempting to build resources to the same highest level.

Using our model and actor profiles, we performed sensitivity analyses on our model for confrontations between different actor profiles. We solved out model using the RK-45 implementation in scipy [208], and examined the sensitivity of the steady-state value of x, y, S_x and S_y to the 4 remaining actor specific parameters and the initial conditions of resources

and psychological states. In practice we solved the system of equations until a simulated time $t = 100$, with a max time step of .1. Utilizing the Python package SALib [209, 210], we generated 1,024 Sobol quasi-random sequences, each sequence consists of 10 sets of values for the model's parameters and initial condition. In total, our model was executed 10,240 times. Using these sets of inputs and corresponding outputs, we performed a Sobol, variance-based global sensitivity analysis [104], and the moment-independent global sensitivity analyses PAWN [211].

A variance-based global sensitivity analysis relates the variance in a model's parameters to variance its outputs. Typically this relation is presented with the first-order (S_1) and total effect (S_T) Sobol indices. The first-order effect measures how much of the model's output variance can be described by a single input on its own, while the total effect captures interactions among parameters as well. While often informative, variance-based methods are not well suited for highly-skewed or bimodal output distributions. In these cases moment-independent methods better capture the importance of inputs on outputs. In this study, we utilized the PAWN method that measures the change in the cumulative distribution function of the output when all inputs are allows to vary versus when inputs are held constant. This change is measured using the Kolmogorov–Smirnov statistic.

6.4 Results

In this section we discuss the results of our sensitivity analyses conflicts between two conservative actors (Table 6.3), two reckless actors (Table 6.4), and a conservative and reckless actor (Table 6.5). These results are presented in three tables, one for each pair of actors. Each table is divided into four subtables where the final states of the resources, denoted $x(\infty)$ and $y(\infty)$, and psychological states, denoted $S_x(\infty)$ and $S_y(\infty)$, are presented. In each subtable, we bold the sensitivity index that corresponds to the parameter that each model output is most sensitive.

In all scenarios, the distributions of values for $x(\infty)$ and $y(\infty)$ are left skewed with a peak near 1, while the distributions of values for $S_x(\infty)$ and $S_y(\infty)$ are both bimodal, with peaks at ± 1 for the 10,240 simulations we examined. We performed a variance-based sensitivity analysis on these simulations, but since the distributions of the final states are not well described by variance alone, we also present a moment-free sensitivity analysis.

In all of our analyses we see that the variance-based and moment-free analyses produce similar rankings for which parameters $x(\infty)$, $y(\infty)$, $S_x(\infty)$ and $S_y(\infty)$ are sensitive. While the ranking of parameters are similar, the difference in sensitivity indices is smaller in the moment-free analysis. As expected, this is more evident in the sensitivities of the psychological states that have bimodal distributions, and thus are not well described by variance

alone. Nonetheless, all model outputs are most sensitive to the initial conditions of the psychological states, $S_x(0)$ and $S_y(0)$, with $x(\infty)$ and $S_y(\infty)$ most sensitive to $S_y(0)$, while $y(\infty)$ and $S_x(\infty)$ most sensitive to $S_x(0)$. The observation that the final values of the psychological state are most sensitive to their initial conditions agrees with the behavior we see in Fig. 6.4. This figure examines the behavior of the psychological state when only the quasi-binary state and threat-response term are at play. Specifically, in Fig. 6.4a when the threat-response term is small there are four stable nodes that determine the possible final values of the psychological states that are completely controlled by the initial conditions of the psychological states. Even as the threat-response term becomes larger there are consistently at least two stable nodes that determine the final values of the psychological state, and which value the model tends towards is fully determined by the initial value of the psychological states. While there are other terms at play, they must not be strong enough to overcome the effects of the quasi-binary state term.

A reckless actor's final psychological state has a low sensitivity to their opponent's initial psychological state this is partly expected as when $\epsilon = 0$, some influence of the opponent's psychological state is removed, but nonlinear effects can still enter through the confidence-deterrence term. However, a conservative actor is influenced by their opponent's initial psychological state. When examining the PAWN indices, conservative actors are about twice as sensitive to their own initial psychological state as their opponent's initial psychological state, while reckless actors are over 6 times more sensitive to their own initial psychological state compared to their opponents.

The final value of an actor's resources are second most sensitive to growth rate of their resources and their enemy's fighting strength. A reckless actor is about slightly more sensitive to their own fighting strength compared to a conservative actor. Additionally reckless actors are more sensitive to their own initial conditions compared to conservative actors.

parameter	S_T	S_1	T
$x(0)$	0.010	0.000	0.030
$y(0)$	0.043	0.002	0.036
$S_x(0)$	0.105	0.015	0.160
$S_y(0)$	**0.658**	**0.387**	**0.361**
α_x	0.033	0.013	0.116
α_y	0.213	0.109	0.138
g_x	0.275	0.169	0.139
g_y	0.049	0.005	0.053

(a) $x(\infty)$

parameter	S_T	S_1	T
$x(0)$	0.061	0.011	0.039
$y(0)$	0.024	−0.005	0.031
$S_x(0)$	**0.676**	**0.404**	**0.357**
$S_y(0)$	0.113	−0.022	0.156
α_x	0.195	0.105	0.151
α_y	0.054	0.007	0.115
g_x	0.052	0.004	0.050
g_y	0.265	0.180	0.133

(b) $y(\infty)$

parameter	S_T	S_1	T
$x(0)$	0.074	0.010	0.039
$y(0)$	0.019	−0.003	0.028
$S_x(0)$	**0.976**	**0.803**	**0.454**
$S_y(0)$	0.142	0.016	0.268
α_x	0.038	0.004	0.117
α_y	0.046	0.000	0.113
g_x	0.026	−0.001	0.064
g_y	0.007	0.003	0.049

(c) $S_x(\infty)$

parameter	S_T	S_1	T
$x(0)$	0.011	0.000	0.027
$y(0)$	0.061	0.012	0.034
$S_x(0)$	0.147	0.038	0.264
$S_y(0)$	**0.962**	**0.820**	**0.453**
α_x	0.034	0.009	0.121
α_y	0.022	0.000	0.104
g_x	0.002	0.002	0.042
g_y	0.016	−0.007	0.068

(d) $S_y(\infty)$

Table 6.3: Global sensitivity analysis for conservative actor vs conservative actor. Here S_T is the total effect Sobol index, S_1 is the first-order effect Sobol index, and T is the PAWN index. For convenience, we bold the most important input as judged by each index. The final value of resources is most sensitive to their opponent's initial psychological state, while the the final value of the psychological state is most sensitive to its own initial condition.

parameter	S_T	S_1	T
$x(0)$	0.037	0.010	0.028
$y(0)$	0.046	-0.009	0.029
$S_x(0)$	0.053	0.028	0.074
$S_y(0)$	**0.643**	**0.437**	**0.413**
α_x	0.057	0.022	0.132
α_y	0.247	0.137	0.148
g_x	0.252	0.164	0.208
g_y	0.029	0.003	0.042

(a) $x(\infty)$

parameter	S_T	S_1	T
$x(0)$	0.041	0.008	0.032
$y(0)$	0.033	0.002	0.035
$S_x(0)$	**0.675**	**0.453**	**0.407**
$S_y(0)$	0.041	0.002	0.074
α_x	0.236	0.136	0.168
α_y	0.070	0.027	0.140
g_x	0.032	0.004	0.041
g_y	0.250	0.150	0.216

(b) $y(\infty)$

parameter	S_T	S_1	T
$x(0)$	0.043	0.005	0.030
$y(0)$	0.043	0.000	0.027
$S_x(0)$	**0.988**	**0.894**	**0.500**
$S_y(0)$	0.000	0.000	0.078
α_x	0.056	0.006	0.197
α_y	0.058	0.020	0.200
g_x	0.012	0.000	0.041
g_y	0.006	-0.002	0.058

(c) $S_x(\infty)$

parameter	S_T	S_1	T
$x(0)$	0.035	0.016	0.029
$y(0)$	0.054	-0.011	0.031
$S_x(0)$	0.000	0.000	0.076
$S_y(0)$	**0.990**	**0.892**	**0.500**
α_x	0.056	0.011	0.203
α_y	0.042	0.004	0.201
g_x	0.002	-0.002	0.053
g_y	0.000	0.000	0.049

(d) $S_y(\infty)$

Table 6.4: Global sensitivity analysis for reckless actor vs reckless actor. Here S_T is the total effect Sobol index, S_1 is the first-order effect Sobol index, and T is the PAWN index. For convenience, we bold the most important input as judged by each index. The final value of resources is most sensitive to their opponent's initial psychological state, while the the final value of the psychological state is most sensitive to its own initial condition.

parameter	S_T	S_1	T
$x(0)$	0.012	0.002	0.034
$y(0)$	0.051	0.002	0.044
$S_x(0)$	0.121	0.023	0.170
$S_y(0)$	**0.638**	**0.378**	**0.354**
α_x	0.032	0.010	0.109
α_y	0.218	0.111	0.142
g_x	0.284	0.180	0.140
g_y	0.050	0.009	0.054

(a) $x(\infty)$

parameter	S_T	S_1	T
$x(0)$	0.040	0.007	0.033
$y(0)$	0.032	0.000	0.036
$S_x(0)$	**0.651**	**0.429**	**0.403**
$S_y(0)$	0.028	−0.003	0.066
α_x	0.235	0.135	0.167
α_y	0.071	0.031	0.142
g_x	0.040	0.005	0.046
g_y	0.265	0.165	0.208

(b) $y(\infty)$

parameter	S_T	S_1	T
$x(0)$	0.043	0.005	0.031
$y(0)$	0.042	0.000	0.029
$S_x(0)$	**0.987**	**0.894**	**0.500**
$S_y(0)$	0.000	0.000	0.073
α_x	0.056	0.007	0.190
α_y	0.059	0.020	0.186
g_x	0.012	0.000	0.043
g_y	0.006	−0.002	0.059

(c) $S_x(\infty)$

parameter	S_T	S_1	T
$x(0)$	0.011	0.000	0.026
$y(0)$	0.073	0.007	0.034
$S_x(0)$	0.170	0.047	0.304
$S_y(0)$	**0.962**	**0.811**	**0.455**
α_x	0.027	−0.005	0.121
α_y	0.030	−0.007	0.113
g_x	0.007	0.000	0.039
g_y	0.029	−0.005	0.067

(d) $S_y(\infty)$

Table 6.5: Global sensitivity analysis for reckless actor (x) vs conservative actor (y). Here S_T is the total effect Sobol index, S_1 is the first-order effect Sobol index, and T is the PAWN index. For convenience, we bold the most important input as judged by each index. The final value of resources is most sensitive to their opponent's initial psychological state, while the the final value of the psychological state is most sensitive to its own initial condition.

6.5 Conclusions

In this work we proposed a model for long-term strategic studies of the statistical properties of conflict. This model was adapted from a Lanchester model with replenishment that was further coupled to a psychology model. Our psychology model included multiple terms representing responding to threats, surrendering, confidence, and deterrence. We defined multiple profiles for actors involved in a conflict, and performed extensive sensitivity analyses to examine the outcomes of matches between these actor profiles. These results informed the importance of fighting capabilities, resource levels, and psychological states on the outcome of the conflict.

We examined two types of sensitivity analyses based on the distributions of outputs from our model that produced similar rankings among the parameters. Overall, we saw that the initial conditions of the psychological state were the dominate parameter in determining the final state of the model in all scenarios. This result was not unexpected based on the examinations of subsets of the model terms in Sec. 6.3. Even though the quasi-binary state has values between 0 and 1, in future iterations of this model, an additional parameter should control its relative strength to other terms. An alternative could be tuning the noise term such that it allows for the psychological state to switch.

When examining the final level of resources among the actors, we found this final value was second most sensitive to how quickly the resources are replenished and how strong the opponent's fighting strength was. Such an observation implies that a stronger enemy could be successfully defended by replenishing resources faster than they can destroy, even if your ability to fight is less than your opponents. For both types of actors this behavior was present without a dependence on the profile of the opponent, suggesting this observation is robust to profile type. When observing the first-order effect, we see much of the variance in the amount of final resources is being driven by interactions among the parameters. Future work could examine higher order indicies that describe the effects of subsets of parameters.

In our analysis, we also saw that reckless agent's final level of resources were more sensitive to their initial conditions compared to conservative actors. This observation combined with the one that reckless agent's have a lower sensitivity to their opponent's initial psychological state than conservative actors suggest self-centered behaviors from the reckless agents. A conservative agent who finds themselves in a conflict with such a reckless agent would benefit from adopting strategies that effect an opponents resources before a conflict breaks out as well as attempting to adjust their psychological states to more favorable values.

In future studies we hope to continue to explore theoretical conflicts using additional actor profiles. These profiles can constrain the same parameters we have explored here, or even be less restrictive. We note that in preliminary sensitivity analyses that allowed all model

parameters to vary resulted in result that were hard to map to actionable results. However, performing such a large analysis did aid in model building by informing which parameters the model was sensitive. Comparing this information to our expectations allowed for changes to be implemented that were more aligned to our modeling goals. Additional actor profiles can be mapped to real world leaders to help inform strategies in a conflict. The choices made here were coarse, but a fair starting point.

While our analysis here focused on a deterministic version of the model, we plan to address including stochastic terms in future work. The addition of stochastic terms will require new quantities of interest to be examined. In preliminary runs of a stochastic model, we found the number of, size of, and duration of conflicts were interesting and dynamic quantities. Another topic worth exploring is the use of other base models. In this work, we extended a two player version of Lanchester's square law. However, there are many other versions, such as adding a third player, that provide an opportunity for richer conflict dynamics. A final area of interest for future work is improvements to our psychological potential energy surface. Here, we only included a static potential that captured a two state system. However, adjusting the parameters of this surface, as well as making it time dependent could lead to informative new insights. While the development of this model is in an early stage, it shows promise for expanding the capabilities of ordinary differential equation based models to explicitly account for deterrence. Additionally, insights gained from this model can lead to improved experiments in more complex models, such as agent-based models, of conflict.

CHAPTER 7:
APPLYING REINFORCEMENT LEARNING TO AGENT-BASED SIMULATIONS OF CONFLICT

7.1 Summary

In this chapter we shift fully to incorporating machine learning into agent-based models. The main objective is to create an agent-based implementation of the game capture the flag such that reinforcement learning techniques can be applied to control players in the game. The reinforcement learning packages Gymnasium [212] and stable-baselines3 [213] are utilized to frame the implementation of capture the flag as a reinforcement learning problem and subsequently train a reinforcement learning agent. The reinforcement learning agent is incrementally trained to solve increasingly complex tasks. These tasks starting with controlling a single player to capture an undefended flag and work towards controlling a team to play against another team. Additionally, aspects of deterrence are examined in the game. Namely, the willingness of a reinforcement learning agent to play as the penalty it receives for being captured increases. While this work is preliminary, there is an extensive discussion of paths to continue this project.

7.2 Introduction

Conflict is a complex phenomenon that is influenced by various factors ranging from individual differences to societal contexts. For the purposes of this thesis, we define conflict as a scenario where two or more groups have incompatible interests, and the actions of these groups interfere with one another. On a personal level, conflict arises from differences in relationships, values, facts, structures, and interests [214, 215]. While personal disputes arise from these individual differences, on a macro scale, the dynamics shift and evolve. At this broader scale, power dynamics, climate [216], inequality [217], and cultural and social contexts play significant roles [218]. Beyond these, broader material and existential reasons further incite conflict [219]. Addressing and understanding conflict demands a holistic, multidisciplinary approach, merging insights from diverse fields including psychology, sociology, political science, and computer science. This understanding can aid in anything from resolving personal disagreements to mediating national-scale disputes. A particularly intriguing dimension of conflict is its asymmetric nature, where the entities in opposition have unequal resources.

Asymmetric conflicts are studied in many contexts. For example, Clark and Konrad explore multifront wars, illustrating scenarios where a defending team is spread across multiple fronts. In contrast, the attacking team is only required to capture a single front to be

victorious [220]. In another study, Dunne et al. propose a three-stage game where a weaker challenger triumphs over a stronger incumbent, leveraging unconventional means that catch the incumbent off-guard [221]. A distinct exploration of asymmetric conflict dynamics can be seen in the works of Johnson and MacKay, who explore the implications of Lanchester's laws [222] in modern warfare [223]. These laws predict the outcome of conflicts, either one-on-one or all-against-all, by evaluating the size and combat capabilities of the engaged groups. The parameters of Lanchester's laws make them particularly well-suited for studying asymmetric conflict. Chapter 6 discusses extended Lanchester models with added replenishing terms and a model of actor's psychological states. Building on the extended Lanchester models, this chapter employs the game of capture the flag (CTF) as a representative model to delve deeper into asymmetric conflict dynamics and deterrence.

Capture the flag is a popular adversarial game that has been used to study conflict in cyber security [224, 225, 226] and robotics [227, 228, 229, 230], among other fields. In the physical world, the game is typically played in a large area, such as a field or gymnasium, and consists of two teams that compete to capture each other's flag, while simultaneously defending their own. Gameplay unfolds through a series of strategic decisions, such as whether to defend one's own flag or attack the opponent's flag, and how to coordinate with one's team members. Computational adaptations of CTF follow the general rules of the game, however the inherent flexibility of the medium allows for adaptations that may be unfeasible to study in a physical game of the CTF. For example, in a computational implementation players ability to detect their surroundings can be varied. Such discrepancies can be mapped onto real conflict scenarios, and allow CTF to be used to study asymmetric conflict.

Capture the flag lends itself well to the modeling technique reinforcement learning (RL). Reinforcement learning is often framed as a Markov decision process (MDP), and revolves around two fundamental components: an agent, which serves as the learning entity, and an environment, with which the agent interacts. The agent takes a sequence of actions within the environment over a defined number of iterations, T. At each iteration, t, the state of the environment is represented by $s_t \in \mathcal{S}$, where $\mathcal{S}$ represents the set of all possible configurations. The agent observes the current state, s_t, and selects an action, a_t, from the set of possible actions, $\mathcal{A}$. Performing the action transitions the environment from s_t to s_{t+1}, while the agent receives a reward r_{t+1} from the reward set, $\mathcal{R}$. Notably, the state may remain unchanged ($s_t = s_{t+1}$), and the reward can be positive, negative, or zero. The goal of the agent is to learn a policy, π, that represents the series of action that maximize the expected cumulative reward. Typically, rewards are discounted using a discount factor $\gamma \in [0, 1]$, to balance short-term gains against long-term goals. Thus, the total cumulative

reward at iteration t is computed as

$$R_t = \sum_{i=0}^{T-t-1} \gamma^i r_{t+i+1}, \tag{7.1}$$

where an agent with γ close to 0 exhibits shortsighted behavior, while γ closer to 1 implies a more farsighted approach.

Applied to CTF, a team is controlled by a reinforcement learning agent, similar to a coach or general directing the actions of the team. The positions of players and flags dictate the state of the system. As dictated by the reinforcement agent, players take actions to update their positions, and are correspondingly rewarded based on events in the game, such as capturing the flag or getting captured. There are many algorithms to solve RL tasks [231, 232], many of which are performed in discrete action and state spaces. Such as Q-learning [233], deep Q-learning [234], value-iteration [235], and temporal difference [236]. Some of these methods can handle continuous state spaces, but fail when both the state and action space are continuous. Other algorithms have been developed to solve such problems [237, 238, 239, 240]. Recently genetic algorithms have been applied to training reinforcement learning agents, and been found to be faster to train and produce comparable solutions as other methods in certain scenarios [241, 242]. The Python package stable-baselines3 [213] provides implementations of many standard RL algorithms. In particular, their implementation of the Proximal Policy Optimization (PPO) algorithm is utilized in this chapter.

Our overarching goal is to use reinforcement learning to train teams in our capture the flag environment that have various levels of ability. These abilities can include, how individuals perceive and move through the environment, the size of teams, and the length of time a team is trained. By competing these teams against each other, we can examine their deviations from simpler models, such the Lanchester model discussed in Chapter 6. Such an exploration can be used as a proxy to study asymmetric conflict in the real world.

7.3 Methods

7.3.1 Creating Capture the Flag Environment

There are many variations of CTF that vary in the number of players and flags, as well as the rules for capturing these entities. Traditionally, CTF is played with two teams, in a rectangular, two-dimensional space such as a field. The goal of each team is to capture the enemy's flag and return in to their side without being captured while simultaneously defending their own flag. For our implementation, we have relaxed some of these assumptions.

We implemented a version of CTF in Python using Shapely [243] that allowed us to

easily define elements of the games as geometries and test for their intersections. We refer to the area the game unfolds as the field, and it is represented as an adjustable rectangle $\Omega = [x_{min}, x_{max}] \times [y_{min}, y_{max}] \in \mathbb{R}^2$. The field is split into a left and right side, with an impenetrable boundary that keeps the players within its bounds. One team defends the right side ($(x_{max} - x_{min})/2 > 0$), while the other defends the left side ($(x_{max} - x_{min})/2 < 0$). Players and flags are represented as circles in Ω with radii R_p and R_f, respectively. A team's players and their flag initially start on the team's side of the field, but players can move throughout the field as the game progresses. The game is broken into a series of iterations, where players select a velocity vector, v, to move themselves to a new location, with a maximum magnitude of v_{max}. During each iteration there are two events that can occur, depending on entities overlapping. These events are: 1) a player getting captured, when a two opposing players overlap, 2) a flag getting captured, when a player overlaps an opposing flag, and 3) a player hits the boundary, when a player overlaps with the boundary (see Table 7.1). When players overlap, the player that is on the side they are defending captures the opposing player, removing them from the game. When a player overlaps with the opposing team's flag, the flag is captured and the game ends. If a player intersects the boundary, they are moved back in bounds, perpendicular to the intersection. The team that is able to capture the enemy's flag without having all their players captured wins. If neither team successfully captures the enemies flag within 1000 iterations, they both lose.

7.3.2 Applying Reinforcement Learning to Capture the Flag Environment

With the basics of game play defined, players require a method to determine how they move during a game. One method would be to supply each player with a series of rules to determine their next move, however we do not want to limit ourselves to strategies that we propose. Instead, we wish to learn policies that can adapt to what other players are doing. To accomplish this, we framed the game as a reinforcement learning task.

Each simulation consists of two team comprised of N_1 and N_2 players. The players on each team are supplied actions either by an agent or a supplied set of rules. The players on a team are unable to fully detect their environment, and instead rely on a set of R rays of length L that are evenly spaced around their body. These rays can detect the distance from, type of, and velocity of objects they intersect. The possible object types are teammate, enemy, team flag, enemy flag, boundary, and open space. The agent (or rule set) that controls each team can observe the positions and velocities of each team member, if the member is defending or attacking, and the information detected by each ray. In total there are $(4R+5)N_1$ and $(4R+5)N_2$ variables for the teams. As an example, a single iteration of a

simulation is shown in Fig. 7.1. Here, the bounds of the field are given by $x_{min} = y_{min} = -1$ and $x_max = y_{max} = 1$, with the boundary represented as a black rectangle. The teams are denoted with red and blue entities, where the larger circles are the teams flags, and the smaller circles are the individuals that make up the teams. The blue lines emanating from the blue player are the rays it uses to perceive the environment. Most of the rays are intersecting walls, however there are rays intersecting each flag and the red player. Therefore, the blue player can see these entities during this iteration. However, in a later iteration if they rays are no longer intersecting these entities the blue player would not detect them. In this example, only the blue players rays were shown to reduce clutter in the image.

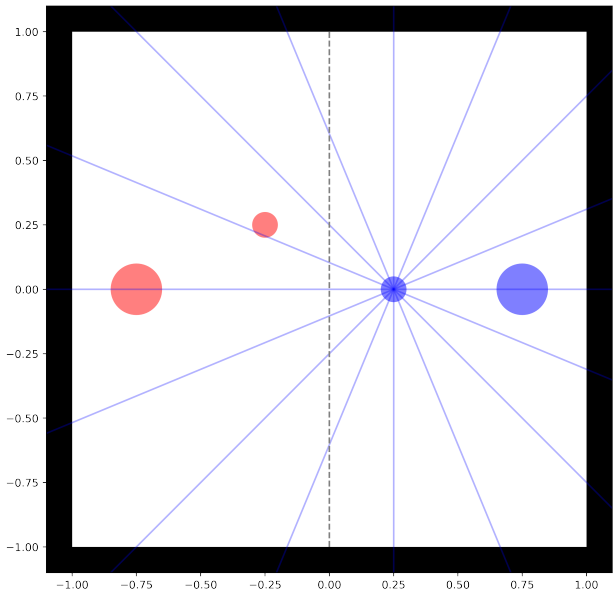


Figure 7.1: Example capture the flag environment. The boundary marking out-of-bounds is shown in black. The larger colored circles are each team's flag, while the smaller circles are the players. The rays emanating from the blue player allow the player to observe the environment. Along each ray the player can detect what type of object the ray is intersecting, the distance to the object, and the velocity the object is moving.

Using the input from the players on a team, an agent determines the action each player should take. In this model, a player's action is a velocity vector used to update its position. The RL agent uses a neural network with two hidden layers, each with 140 neurons with tanh activation functions to produce a velocity vector for each player. Each iteration after each player has updated thier position, the RL agents receive rewards based on the player-involved events that occurred. The events and their associated rewards are listed in Table 7.1. Negative rewards are given in an effort to penalize behaviors, while positive ones encourage behaviors. In this model, each step taken incurs a reward of -1 to encourage the agent to

finish the game quickly. Similarly, a reward of -2 is given when players touch the boundary to encourage them to stay in bounds. The final negative rewards are given when a player or their flag is captured. Each of these rewards are relative to the number of iterations that pass before they occur, such that they are 0 if the player or flag are never captured in the maximum number of iterations. Positive rewards are given for capturing the enemy flag or players. These rewards start at 1000 and are reduced by the number of iterations that pass to encourage a quick solution.

event	**reward**
player takes action	-1
player touches boundary	-2
player captures flag	$1000 - \text{iterations}$
player captures enemy	$1000 - \text{iterations}$
player's flag is captured	$\text{iterations} - 1000$
player is captured	$\text{iterations} - 1000$

Table 7.1: Rewards for capture the flag. Negative rewards penalize behaviors while positive ones encourage them. The first two rewards encourage the RL agent to act quickly and keep players in bounds. The second two rewards encourage the agent to capture entities, while the final two rewards encourage the agent to avoid being captured and protect their flag.

Based on our implementation of CTF and associated rewards, we used the Proximal Policy Optimization (PPO) algorithm implemented through stable-baselines3 [213] to train RL agents to control a team. Our training occurred as a series of progressively more complex tasks that are discussed below. In each of these simulations, we used $x_{min} = y_{min} = -1$ and $x_{max} = y_{max} = 1$. The flags had a radius of $R_f = .1$, while players had a radius of $R_p = .05$ and $R = 16$ rays of length of $L = 2$.

The simplest task we examined was training a single agent to capture a flag without a defender. For this scenario, the field was not partitioned into sides, so the single player was always in an attacking state. We made this choice to ensure that the agent examined the flag from all directions, and would not simply learn to move in a singular direction. Only the first three rewards in Table 7.1 are available to the agent in this scenario. We trained the RL agent controlling the player for 3 million iterations, while monitoring its average rewards over time. These iterations consisted of multiple games, called episodes. In each episode, the agent and flag are spawned at uniformly random locations in the field. An episode is completed when the player either successfully captures the flag, or 1000 iterations have passed. We selected the number of training iterations by examining when the average reward per episode leveled near the maximum of 1000. The resulting trained agent was used as a base model for proceeding teams that have a single player.

With a baseline agent trained, we added complexity to the task by introducing a second team, as well as partitioning the field into sides. In this new scenario, all of the rewards in Table 7.1 were available. Each episode the side the agent defended switched to prevent the agent from developing a preference to move the player in one direction. Training in this scenario employed self-play, allowing the RL agent to learn against a previous version of itself. That is, the new team was controlled by a static version of the baseline agent; however, the RL agent continued to train. The RL agent was trained for an additional 7 millions steps for a total of 10 million training steps. This process could be repeated, where the static team is the one that was trained for 10 million steps, and the learning team continues to train against its improved adversary. However, we only examined one iteration of self-play here.

The final scenario we examined was the emergence of deterrence in the game. This scenario is important for understanding how agents adapt to high-risk, high-reward situations, aligning with our broader objective of examining strategic decision-making in complex environments. To this end, we improved the defending ability of the static agent by having it following two rules: it would move directly towards the enemy at full speed if the enemy was on its side of the field, and otherwise, it would move randomly. To investigate the threshold at which the RL agent would refrain from aggressive play, it was trained against this improved defender, with a varying negative reward for being captured ranging from 0 to -10,000 points. The RL agent was trained for an additional 1 million iterations for each value. As in previous scenarios, we ensured randomness in the initial placement of entities, with each starting on their respective sides of the field.

7.4 Results

Recall the simplest scenario we examined, where an RL agent was trained to control a single-player team to capture a flag anywhere in the field. In Fig. 7.2, we show the progression of average rewards per episode against training iterations in blue. Initially, the agent quickly learns basic navigation, as evidenced by balancing negative and positive rewards within approximately $500,000$ iterations. Within the subsequent 1.5 million iterations, the agent reliably develops a strategy to move quickly towards the flag. Given the random initial conditions and step penalties, a perfect 1000 point reward is unattainable, leading to variation in the maximum achievable reward. Nevertheless, the agent consistently attains rewards near the theoretical maximum of 1000 points. Training was extended for an additional 1.5 million iterations to reinforce the acquired behaviors and expose the agent to rarer edge cases. This phase of training demonstrates the agent's proficiency in mastering basic strategies, setting a foundation for more complex scenarios.

Building on the understanding of the RL agent's learning process, we next explore the

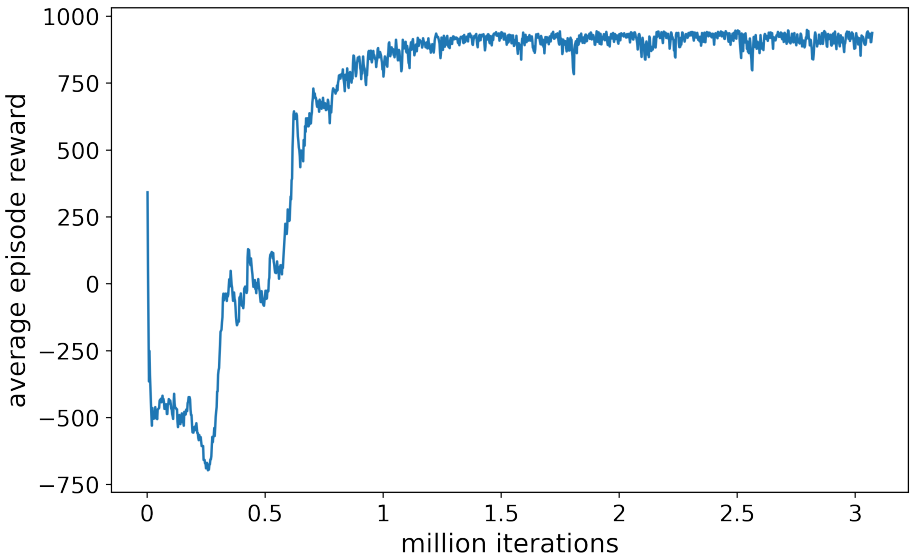
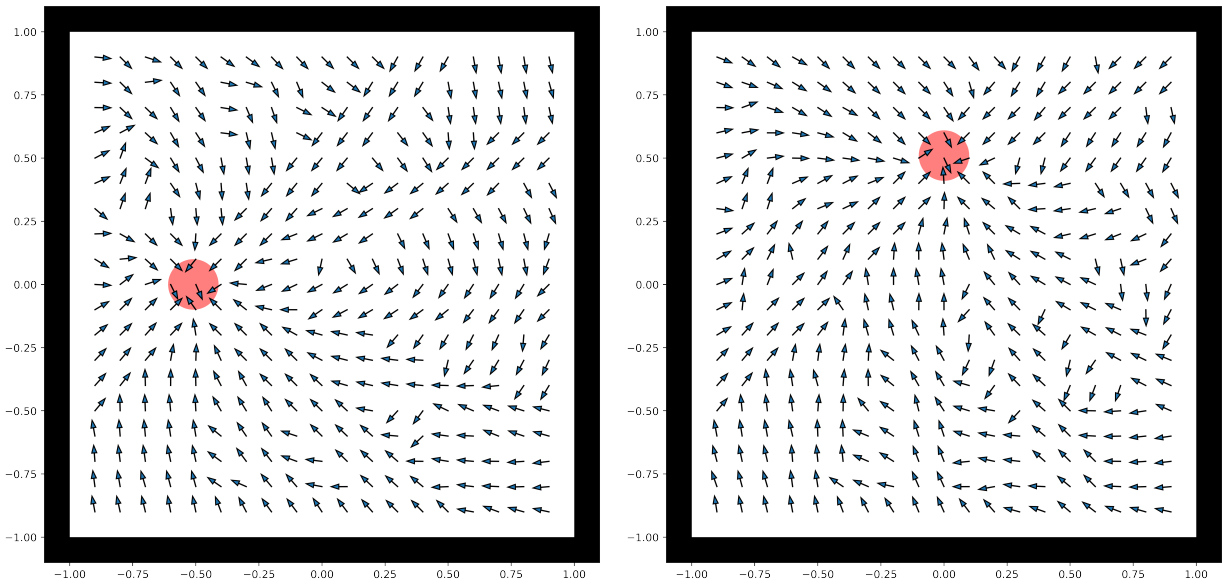


Figure 7.2: Average reward per episode versus training iteration of an RL agent learning to capture a flag with a single player. The RL agent learns basic navigation within 500,000 iterations and is able to balance negative and positive rewards. Within an additional 1 million iterations, the RL is reliably capturing the flag.

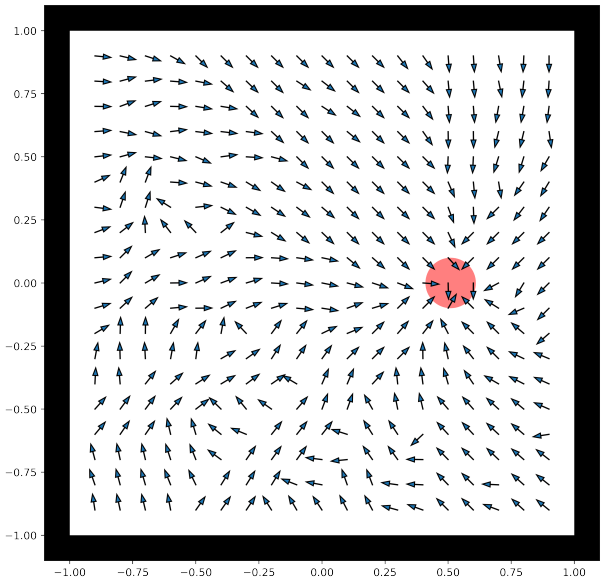
specific actions undertaken by the trained agent in varying scenarios. Figure 7.3 illustrates the agent's action in four different placements of the enemy flag. Similarly, Fig. 7.4 demonstrates the agent's behavior in the absence of an enemy flag. In these figures, the agent's actions are represented by arrows indicating the direction a player movement at each location, with the enemy flag marked as a red circle. In all cases, the agent has learned to prevent the player from hitting the black boundary, as indicated by all actions by the boundary leading the player away. In the scenario with no flag, the agent's general search strategy is evident, characterized by moving the player in a clockwise pattern around the field, spiralling around central point. These behaviors highlight the agent's adaptive strategies in different game environments.

In the next scenario, we shifted to training the agent against a static version of itself. Figure 7.5 shows the actions the agent would supply a blue player at various field positions, represented by arrows. The red player, depicted as a smaller red circle aims to defend the red flag, while the RL agent is tasked with defending the blue flag, both indicated by larger colored circles. It is important to note that in this figure, we assume that both the red and blue agents are stationary when determining these actions. However, in a live game scenario, these agents likely to be in motion. Consequently, these predicted actions occur within a parameter space the RL agent may not have previously encountered. This aspect should be considered when interpreting the actions.

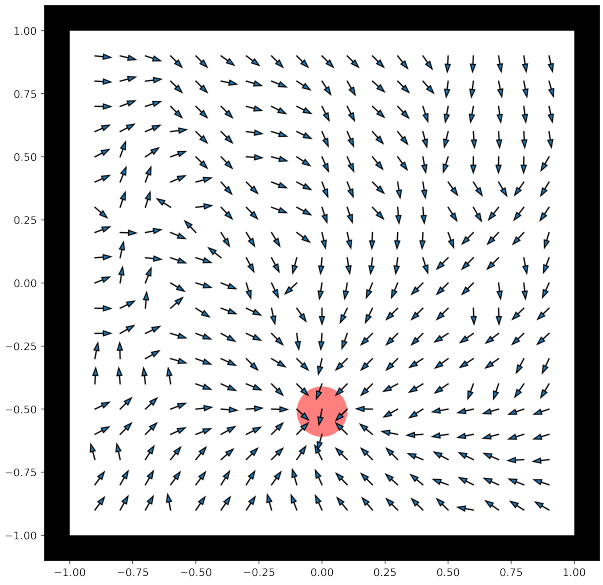


(a) Enemy flag on the left half of field.

(b) Enemy flag on the top half of field.



(c) Enemy flag on the right half of field.



(d) Enemy flag on the bottom half of field.

Figure 7.3: Actions of single player attacking a flag. In each subfigure the arrows denote the direction the attacking player would move. The black square is the boundary for the field, and the red circle is the enemy's flag. The agent has learned to avoid the boundary, as denoted by all of the arrows pointing away from the boundary. Additionally, the agent has generally learned how to move towards the flag in various locations in the field.

In Fig. 7.5a, the red player is positioned defensively. When the RL is on the offensive, it generally advances towards the enemy flag. Its actions are unaffected by the defending red

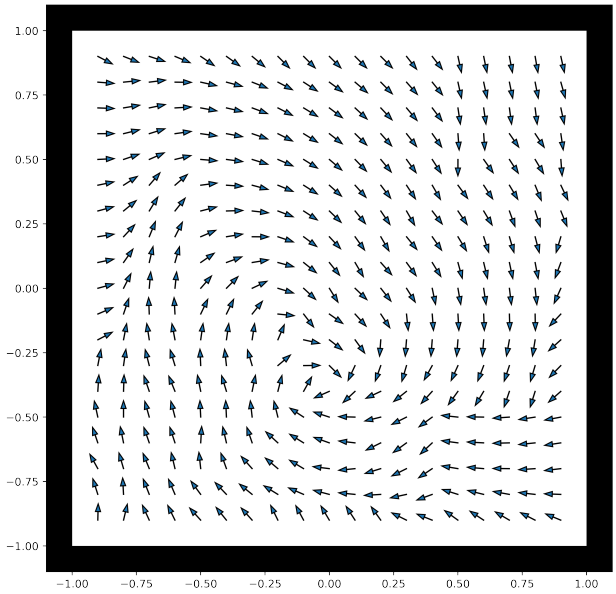
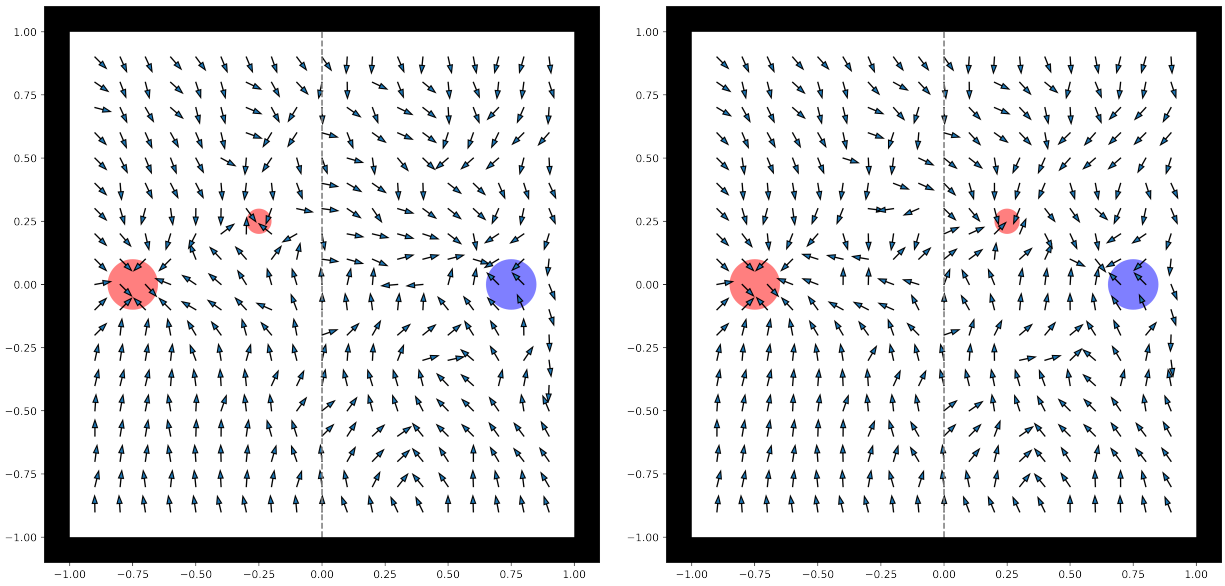


Figure 7.4: Actions of a single player without a flag. Similar to Fig. 7.3, the arrows denote the direction the attacking player would move. The black square is the boundary for the field, and the red circle is the enemy's flag. When the agent is unable to detect a flag, it searches by moving clockwise around the field around a central point.

player when distant from it. Interestingly, when the agent is near both the red player and the center of the field, it shifts towards its own flag, indicating a defensive strategy. However, in its own defensive position, the agent consistently moves towards its flag. Notably, near the bottom of the field, the agent does not shift to an offensive strategy, possibly indicating a gap in its training.

Contrastingly, Fig. 7.5b places the red player in an offensive position. In this scenario, when the RL agent is on the offensive it predominantly heads towards the enemy flag, except in areas adjacent to the red player near the center line. In a defensive position, the agent has learned to target the attacking red player. This is evidence by arrows near the red player directing towards it. These scenarios demonstrate the agent's capacity for dynamic strategy adjustments in response to different threats.

In our final analysis, we explored the emergence of deterrence by varying the negative reward associated with the agent's player being captured. Figure 7.6 presents the average reward (depicted in blue) and average number of steps taken (in orange) across 30 simulations, plotted against the varying penalties for being captured. As expected, with a minimal penalty, the agent attempts to capture the flag, indicated by the relatively low number of steps coupled with a negative reward. However, as the penalty increases, a significant shift in the agent's strategy is observed. Notably, at a penalty of approximately 8000 points, the



(a) If the red team is in a defensive position, the blue player moves to capture the flag when they are in an offensive position and defend their flag in a defensive one. There is a slight preference for moving around the defensive red player. Around the center line in front of the red player, we see the agent's actions leading to a defensive position.

(b) If the red team is in an offensive position, the blue player moves to capture the agent when they are in a defensive position, and move to capture the flag in an offensive one. The blue agent decides to move back to their side if they are close to the center and near the red agent.

Figure 7.5: Actions of single player against single player. In these scenarios, the field is partitioned into two sides by a grey dashed line, where the red team defends the left and the blue team defends the right. The large circles denote the teams flags, and the smaller circle is a red player. The arrows denote the action a blue player would make, assuming all entities are at rest.

agents average score raises to around -1000 points, with an average of 1000 steps taken. This indicates a strategic change where the agent opts to remain on its side of the field throughout the game, effectively exhibiting deterrence.

7.5 Conclusion

In this chapter, we successfully implemented a numerical version of capture the flag and applied reinforcement learning to develop strategic gameplay. Our preliminary results demonstrate the capability of training an RL agent to control a single player in progressively complex scenarios.

Initially, the agent was tasked with capturing a randomly place flag. Starting from this basic task aided the RL agent in learning robust strategies. In early attempts, an agent always started on a specified side of the field. Having never experienced the flag behind it,

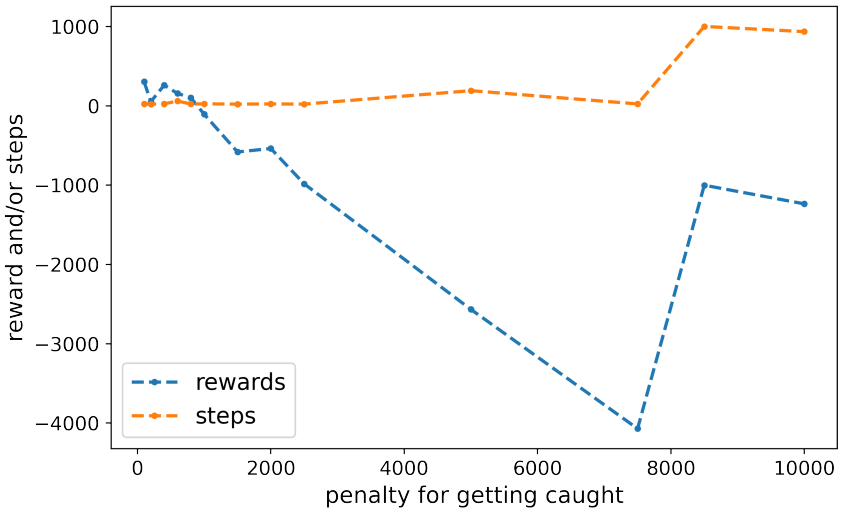


Figure 7.6: Average reward and number of steps versus penalty of being caught. The average number of steps (orange) and reward (blue) for 30 simulations is shown versus the penalty of being caught. When the penalty is low, the RL agent attempts to capture the enemy flag. However, once the penalty reaches approximately 8000, the agent is deterred from play and spends the total 1000 timesteps on their side of the field.

the agent learned to always move in a single direction regardless of its observations. This insight influenced the initialization of subsequent scearios.

Once the baseline agent was trained to stay in bounds and capture a flag, we moved to training it against another player. When the new player followed a prescribed set of rules, the RL excelled in adapting to offensive rolls, however it struggled in subsequent defensive roles. This challenge was partially mitigated through self-play that allowed the agent to experience both offensive and defensive roles during training and kept the opposing player at a similar ability level. While this approach led to improved strategic behavior, such as retreating to defend its flag, it also highlighted areas for improvement. For example, as seen in Fig. 7.5a, the agent occasionally fails to optimally avoid defenders.

The final area we examined was the emergence of deterrence. By varying the negative reward an RL agent was given for its player being captured, we were eventually able observe it no longer play the game. This threshold was reached only at a relatively high penalty, highlighting the agent's initial propensity for risk-taking or goal-directed behavior despite potential negative outcomes. This observation suggests that the agent's decision making process might require substantial negative reinforcement to override its objective-focused strategies. Interestingly, it also implies that with further training iterations, a lower penalty threshold might achieve similar deterrence effects.

While the results we have been able to produce so far are informative, there are areas that this research can be improved in the future. We sort these improvements into the three broad categories of improved training, scenarios, and algorithms. Regarding training improvements, RL agents should be trained for many more iterations to provide ample opportunities to develop and refine sophisticated strategies, particularly through increased self-play iterations. A significant limitation of our work was the lack of hyperparameter tuning, primarily driven by computational constraints. Nonetheless, future efforts should focus on exploring various neural network architectures, including adjustments in the number of hidden layers, their sizes, and activation functions. Additionally, fine-tuning player hyperparameters, such as the number and size of sensory rays, could lead to more nuanced and effective strategies.

In terms of scenario development, future research should explore crafting sculpted scenarios specifically designed to expedite the learning process of RL agents. In this work, the starting locations of entities were randomized, which is effective for broadly sample the state space of the game. However, strategically selecting initial conditions to ensure diversity can speed up the RL agent's learning. This approach would expose the agent to a wide range of situations in a more systematic manner, enhancing its ability to build strategies. Additionally, experimenting with varying team sizes and altering player attributes (e.g. player hyperparmeters) presents an exciting opportunity. With these changes, we can dive deeper into the dynamics of asymmetric conflict, examining how differences in team composition and capabilities affect strategic outcomes.

Finally, with respect to algorithms the more general multi-agent reinforcement learning (MARL) should be explored. While we utilized single-agent reinforcement learning as a starting point in this work, many complex real-world problems are not well represented in the framework. Using MARL, both teams would be controlled by RL agents that learn simultaneously. Many RL algorithms can be generalized to multiple players. For example, [244] examines extending trust region policy optimization, deep deterministic policy gradient, and deep-q networks, while Kar et al. [245] discuss extending Q-learning to the multi-agent version, QD-learning. Buşoniu et al. [246] provides a review many MARL algorithms. However, transitioning from single-agent to multi-agent RL introduces unique challenges as outlined by Nguyen el al. [247]. These areas include partial observability, non-stationarity, continuous action spaces, multi-agent training schemes, and transfer learning. Our implementation has already began to address partial partial observability and continuous spaces, providing a solid foundation for tackling the other challenges in MARL.

In summary, this chapter lays the groundwork for addressing asymmetric conflict with reinforcement learning through the use of a simulated capture the flag game. Through many

scenarios, we have demonstrated the adaptability of RL agents in dynamic environments. While are current results are promising, we have illuminated several areas for potential improvement to further extend the capabilities of RL in simulations of conflict.

CHAPTER 8:
CONCLUSION

Throughout this dissertation, we explored areas of improvement for the field of agent-based modeling, namely data integration, policy evaluation, and the incorporation of machine learning. These areas were examined through projects presented in pairs of chapters, where each chapter consisted of a unique yet interconnected project.

Chapter 2 and 3 focused on using data to develop and refine ABMs, specifically applied to the spread of chronic wasting disease in white tailed deer. These chapters serve as comprehensive examples of data-driven model development and underscore the challenges in modeling uncertainty. While our model produced simulations closely resembling real data, the complexity of model creation and computational demands highlighted the need for more efficient methodologies.

The data integration projects underscored two major takeaways: the importance of exploratory data analysis (EDA) and sensitivity analysis in agent-based modeling. EDA was instrumental in identifying subtle features present in our data that shaped the model's development. While integrating these features was challenging in our case, it may be more straightforward in other applications. Future modeling efforts should prioritize extensive data examination to extract key features, thereby enhancing model accuracy across various fidelities. Similarly, sensitivity analysis (SA) plays an important role in model building and assessment. SA aided in understanding the impact of parameters on model outcomes, guiding decision on which features to include. For example, if a SA reveals that added parameters have little to no effect on important model outputs, you can safely omit those parameters from the model. Moreover, SA can rank parameter importance, offering insights and understanding of the model that would otherwise be unattainable.

In Chapters 4 and 5, our research shifted towards employing ABMs for policy assessment. Chapter 4 focused on evaluating strategies to mitigate the spread of disinformation across social networks. We dedicated thousands of core hours to analyze and rank various strategies across different graph topologies and conducted a similar evaluation on a real social network. However, the significant computational demands posed challenges. These challenges led to the focus of the subsequent chapter, where we explored ways to reduce the size of graphs while preserving certain attributes. Such a process enhances the feasibility of applying models to real-world social networks.

These projects highlighted the necessity of substantial computing resources for an in-depth examination of policy implications using ABMs. Although our research demonstrated

techniques to lessen the demand for extensive resources, we were careful in our approach to ensure these methods still accurately reflect real-world scenarios. This balance is crucial; as we develop more efficient computational strategies, it is imperative to maintain a high degree of realism in our models. This ensures that the insights and recommendations derived from ABMs remain relevant and applicable to actual policy-making. Future research should continue to focus on optimizing computational efficiency while preserving the integrity and realism of the models, thereby enabling a comprehensive and pragmatic evaluation of policies in various domains.

Lastly, in Chapters 6 and 7, our focus shifted to the integration of machine learning within ABMs, with a particular emphasis on models of conflict and deterrence. The first chapter in this segment laid the groundwork, enhancing existing mathematical models of conflict to explicitly incorporate aspects of deterrence. Building on this, the following chapter presented preliminary efforts to apply reinforcement learning as the rules in agent-based models. This approach was then used to examine the emergence of deterrence in a model of conflict, thus illustrating how machine learning can contribute to the sophistication of ABMs in complex scenarios.

While preliminary, these chapters illuminate the power of combining machine learning with ABMs. This fusion not only enhances the analytical capabilities of ABMs but also opens up new avenues for exploring intricate systems and behaviors. This advancement points to a future where ABMs can be more effectively used to simulate and understand complex phenomena, offering insights that might be difficult to obtain through traditional modeling methods. Future research can further refine these methodologies, exploring their applicability across a broader spectrum of disciplines and deepening our understanding of the emergent properties within these systems.

The contributions of this dissertation to the field of agent-based modeling highlight potential avenues for future exploration. The dissertation demonstrates that integrating traditional modeling techniques with contemporary data and machine learning approaches can lead to the development of more robust tools, better equipped to address the complexities of today's challenges.

BIBLIOGRAPHY

[1] Harry Jones. The recent large reduction in space launch cost. *48th International Conference on Environmental Systems*, 2018.

[2] Steven F. Railsback and Volker Grimm. *Agent-Based and Individual-Based Modeling: A Practical Introduction*. Princeton University Press, 2011.

[3] C. M. Macal and M. J. North. Tutorial on agent-based modeling and simulation. In *Proceedings of the Winter Simulation Conference, 2005.*, pages 14 pp.–, 2005.

[4] C. M. Macal and M. J. North. Tutorial on agent-based modeling and simulation part 2: How to model with agents. In *Proceedings of the 2006 Winter Simulation Conference*, pages 73–83, 2006.

[5] John von Neumann and Arther Burks. *Theory of Self-Replicating Automata*. University of Illinois Press, 1966.

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

[7] Stephen Wolfram. *A New Kind of Science*. Wolfram Media Inc., Champaign, Ilinois, USA, 2002.

[8] Christopher Bone and Suzana Dragićević. Simulation and validation of a reinforcement learning agent-based model for multi-stakeholder forest management. *Computers, Environment and Urban Systems*, 34(2):162–174, 2010.

[9] William Ogilvy Kermack, A. G. McKendrick, and Gilbert Thomas Walker. A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 115(772):700–721, 1927.

[10] Thomas House, Geoffrey Davies, Leon Danon, and Matt J Keeling. A motif-based approach to network epidemics. *Bulletin of Mathematical Biology*, 71:1693–1706, 2009.

[11] Jaewook Joo and Joel L Lebowitz. Pair approximation of the stochastic susceptible-infected-recovered-susceptible epidemic model on the hypercubic lattice. *Physical Review E*, 70(3):036114, 2004.

[12] Anirban Chakraborti, Ioane Muni Toke, Marco Patriarca, and Frédéric Abergel. Econophysics review: II. Agent-based models. *Quantitative Finance*, 11(7):1013–1041, 2011.

[13] M. Barbati, G. Bruno, and A. Genovese. Applications of agent-based models for optimization problems: A literature review. *Expert Systems with Applications*, 39(5):6020 – 6028, 2012.

[14] Li An. Modeling human decisions in coupled human and natural systems: Review of agent-based models. *Ecological Modelling*, 229:25 – 36, 2012. Modeling Human Decisions.

[15] Federico Bianchi and Flaminio Squazzoni. Agent-based models in sociology. *Wiley Interdisciplinary Reviews: Computational Statistics*, 7(4):284–306, 2015.

[16] Sean Barnes, Bruce Golden, and Stuart Price. Applications of agent-based modeling and simulation to healthcare operations management. In *Handbook of healthcare operations management: methods and applications*, pages 45–74. Springer, 2013.

[17] Melissa Tracy, Magdalena Cerdá, and Katherine M Keyes. Agent-based modeling in public health: current applications and future directions. *Annual review of public health*, 39:77–94, 2018.

[18] Charles M Macal. Everything you need to know about agent-based modelling and simulation. *Journal of Simulation*, 10:144–156, 2016.

[19] Volker Grimm, Eloy Revilla, Uta Berger, Florian Jeltsch, Wolf M Mooij, Steven F Railsback, Hans-Hermann Thulke, Jacob Weiner, Thorsten Wiegand, and Donald L DeAngelis. Pattern-oriented modeling of agent-based complex systems: lessons from ecology. *science*, 310(5750):987–991, 2005.

[20] Craig Loehle. A guide to increased creativity in research: inspiration or perspiration? *Bioscience*, 40(2):123–129, 1990.

[21] Joshua M Epstein. Why model? *Journal of artificial societies and social simulation*, 11(4):12, 2008.

[22] Bruce Edmonds and Scott Moss. From kiss to kids–an 'anti-simplistic'modelling approach. In *International workshop on multi-agent systems and agent-based simulation*, pages 130–144. Springer, 2004.

[23] David O'Sullivan, Tom Evans, Steven Manson, Sara Metcalf, Arika Ligmann-Zielinska, and Chris Bone. Strategic directions for agent-based modeling: avoiding the yaawn syndrome. *Journal of land use science*, 11(2):177–187, 2016.

[24] William Rand and Roland T Rust. Agent-based modeling in marketing: Guidelines for rigor. *International Journal of research in Marketing*, 28(3):181–193, 2011.

[25] William Rand. Theory-interpretable, data-driven agent-based modeling. *Social-behavioral modeling for complex systems*, pages 337–357, 2019.

[26] Srinivasan Venkatramanan, Bryan Lewis, Jiangzhuo Chen, Dave Higdon, Anil Vullikanti, and Madhav Marathe. Using data-driven agent-based models for forecasting emerging infectious diseases. *Epidemics*, 22:43–49, 2018.

[27] Elizabeth Hunter, Brian Mac Namee, and John Kelleher. An open-data-driven agent-based model to simulate infectious disease outbreaks. *PloS one*, 13(12):e0208775, 2018.

[28] Mazhar Sajjad, Karandeep Singh, Euihyun Paik, and Chang-Won Ahn. A data-driven approach for agent-based modeling: Simulating the dynamics of family formation. *Journal of Artificial Societies and Social Simulation*, 19(1):9, 2016.

[29] Haifeng Zhang, Yevgeniy Vorobeychik, Joshua Letchford, and Kiran Lakkaraju. Data-driven agent-based modeling, with application to rooftop solar adoption. *Autonomous Agents and Multi-Agent Systems*, 30:1023–1049, 2016.

[30] David J Butts, Noelle E Thompson, Sonja A Christensen, David M Williams, and Michael S Murillo. Data-driven agent-based model building for animal movement through exploratory data analysis. *Ecological Modelling*, 470:110001, 2022.

[31] David J Butts, Sam A Bollman, and Michael S Murillo. Mathematical modeling of disinformation and effectiveness of mitigation policies. *Scientific Reports*, 13(1):18735, 2023.

[32] Aldo Leopold. The Conservation Ethic. *Journal of Forestry*, 31(6):634–643, 10 1933.

[33] Hugh Dingle and V. Alistair Drake. What Is Migration? *BioScience*, 57(2):113–121, 02 2007.

[34] Navinder J. Singh, Andrew M. Allen, and Göran Ericsson. Quantifying migration behaviour using net squared displacement approach: Clarifications and caveats. *PLOS ONE*, 11(3):1–20, 03 2016.

[35] Mevin B. Hooten, Henry R. Scharf, Trevor J. Hefley, Aaron T. Pearse, and Mitch D. Weegman. Animal movement models for migratory individuals and groups. *Methods in Ecology and Evolution*, 9(7):1692–1705, 2018.

[36] Sophie Bestley, Toby A. Patterson, Mark A. Hindell, and John S. Gunn. Feeding ecology of wild migratory tunas revealed by archival tag records of visceral warming. *Journal of Animal Ecology*, 77(6):1223–1233, 2008.

[37] B.F. Manly, L. McDonald, D.L. Thomas, T.L. McDonald, and W.P. Erickson. *Resource Selection by Animals: Statistical Design and Analysis for Field Studies*. Springer Netherlands, 2007.

[38] Douglas H. Johnson. The comparison of usage and availability measurements for evaluating resource preference. *Ecology*, 61(1):65–71, 1980.

[39] Geert Aarts, Monique MacKenzie, Bernie McConnell, Mike Fedak, and Jason Matthiopoulos. Estimating space-use and habitat preference from wildlife telemetry data. *Ecography*, 31(1):140–160, 2008.

[40] D. John Anderson. The home range: A new nonparametric estimation technique. *Ecology*, 63(1):103–112, 1982.

[41] D. B. Shepard, A. R. Kuhns, M. J. Dreslik, and C. A. Phillips. Roads as barriers to animal movement in fragmented landscapes. *Animal Conservation*, 11(4):288–296, 2008.

[42] A. David M. Latham, M. Cecilia Latham, Mark S. Boyce, and Stan Boutin. Movement responses by wolves to industrial linear features and their effect on woodland caribou in northeastern alberta. *Ecological Applications*, 21(8):2854–2865, 2011.

[43] Mevin B. Hooten, Devin S. Johnson, Brett T. McClintock, and Juan M. Morales. *Animal Movement Statistical Models For Telemetry Data*. Taylor & Francis, 2017.

[44] Kim Whoriskey, Marie Auger-Mèthè, Christoffer M. Albertsen, Frederick G. Whoriskey, Thomas R. Binder, Charles C. Krueger, and Joanna Mills Flemming. A hidden markov movement model for rapidly identifying behavioral states from animal tracks. *Ecology and Evolution*, 7(7):2112–2121, 2017.

[45] Roland Langrock, Ruth King, Jason Matthiopoulos, Len Thomas, Daniel Fortin, and Juan M. Morales. Flexible and practical modeling of animal telemetry data: hidden markov models and extensions. *Ecology*, 93(11):2336–2342, 2012.

[46] Ann E. McKellar, Roland Langrock, Jeffrey R. Walters, and Dylan C. Kesler. Using mixed hidden Markov models to examine behavioral states in a cooperatively breeding bird. *Behavioral Ecology*, 26(1):148–157, 09 2014.

[47] Ian D. Jonsen, Ransom A. Myers, and Joanna Mills Flemming. META-ANALYSIS OF ANIMAL MOVEMENT USING STATE-SPACE MODELS. *Ecology*, 84(11):3055–3063, 2003.

[48] Ian D. Jonsen, Joanna Mills Flemming, and Ransom A. Myers. ROBUST STATE-SPACE MODELING OF ANIMAL MOVEMENT DATA. *Ecology*, 86(11):2874–2880, 2005.

[49] Toby A. Patterson, Len Thomas, Chris Wilcox, Otso Ovaskainen, and Jason Matthiopoulos. State-space models of individual animal movement. *Trends in Ecology & Evolution*, 23(2):87 – 94, 2008.

[50] Haiganoush K. Preisler, Alan A. Ager, and Michael J. Wisdom. Analyzing animal movement patterns using potential functions. *Ecosphere*, 4(3):art32, 2013.

[51] Thèo Michelot, Pierre Gloaguen, Paul G. Blackwell, and Marie-Pierre Ãtienne. The langevin diffusion as a continuous-time model of animal movement and habitat selection. *Methods in Ecology and Evolution*, 10(11):1894–1907, 2019.

[52] John W Tukey et al. *Exploratory data analysis*, volume 2. Reading, Mass., 1977.

[53] Frederick Hartwig and Brian E Dearing. *Exploratory data analysis*. Sage, 1979.

[54] Matthew B Miles and A Michael Huberman. *Qualitative data analysis: An expanded sourcebook*. sage, 1994.

[55] Andrew M. Edwards. Overturning conclusions of lèvy flight movement patterns by fishing boats and foraging animals. *Ecology*, 92(6):1247–1257, 2011.

[56] Andrew M. Edwards, Richard A. Phillips, Nicholas W. Watkins, Mervyn P. Freeman, Eugene J. Murphy, Vsevolod Afanasyev, Sergey V. Buldyrev, M. G. E. da Luz, E. P. Raposo, H. Eugene Stanley, and Gandhimohan M. Viswanathan. Revisiting lévy flight search patterns of wandering albatrosses, bumblebees and deer. *Nature*, 449(7165):1044–1048, 2007.

[57] Andrew M. Edwards. Using likelihood to test for lèvy flight search patterns and for general power-law distributions in nature. *Journal of Animal Ecology*, 77(6):1212–1222, 2008.

[58] Erhard Rahm and Hong Hai Do. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4):3–13, 2000.

[59] Tamraparni Dasu and Theodore Johnson. *Exploratory data mining and data cleaning*, volume 479. John Wiley & Sons, 2003.

[60] Jonathan Schwabish. *Better Data Visualizations: A Guide for Scholars, Researchers, and Wonks*. Columbia University Press, 2021.

[61] Robert M Edsall. The parallel coordinate plot in action: design and use for geographic visualization. *Computational Statistics & Data Analysis*, 43(4):605–619, 2003.

[62] Takayuki Itoh, Ashnil Kumar, Karsten Klein, and Jinman Kim. High-dimensional data visualization by interactive construction of low-dimensional parallel coordinate plots. *Journal of Visual Languages & Computing*, 43:1–13, 2017.

[63] Andrej Gisbrecht and Barbara Hammer. Data visualization by nonlinear dimensionality reduction. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(2):51–73, 2015.

[64] Samuel Kaski and Jaakko Peltonen. Dimensionality reduction for data visualization [applications corner]. *IEEE signal processing magazine*, 28(2):100–104, 2011.

[65] John D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science Engineering*, 9(3):90–95, 2007.

[66] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[67] Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021.

[68] Benjamin Bengfort and Rebecca Bilbro. Yellowbrick: Visualizing the scikit-learn model selection process. *Journal of Open Source Software*, 4(35):1075, 2019.

[69] Amy Quinn. Influences of movement behavior and space use in evaluating disease risk amoung white-tailed deer in central new york. *Syracuse, New York: State University of New York College of Environmental Science and Forestry*, 2010.

[70] David M. Williams, Amy C. Dechen Quinn, and William F. Porter. Informing disease models with temporal and spatial contact structure among gps-collared individuals in wild populations. *PLOS ONE*, 9, 01 2014.

[71] Mark S Boyce, Pierre R Vernier, Scott E Nielsen, and Fiona K.A Schmiegelow. Evaluating resource selection functions. *Ecological Modelling*, 157(2):281–300, 2002.

[72] Toby A. Patterson, Marinelle Basson, Mark V. Bravington, and John S. Gunn. Classifying movement behaviour in relation to environmental conditions using hidden markov models. *Journal of Animal Ecology*, 78(6):1113–1123, 2009.

[73] D. R. Brillinger, H. K. Preisler, A. A. Ager, and J. G. Kie. *The Use Of Potential Functions In Modelling Animal Movement*, pages 385–409. Springer New York, New York, NY, 2012.

[74] David R. Brillinger, Haiganoush K. Preisler, Alan A. Ager, John G. Kie, and Brent S. Stewart. Employing stochastic differential equations to model wildlife motion. *Bulletin of the Brazilian Mathematical Society*, 33(3):385–408, 2002.

[75] A. Quinn Dechen, David Williams, W. Porter, M. Smith, F. DeSantis, and McNulty F. Risk assessment of a chronic wasting disease outbreak in new york: Final report. *New York State Department of Environmental Conservation*, 2009.

[76] Daniel Fortin, Hawthorne L. Beyer, Mark S. Boyce, Douglas W. Smith, Thierry Duchesne, and Julie S. Mao. Wolves influence elk movements: Behavior shapes a trophic cascade in yellowstone national park. *Ecology*, 86(5):1320–1330, 2005.

[77] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108, 2005.

[78] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[79] Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

[80] Tak chung Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164 – 181, 2011.

[81] Christos Berberidis, Walid G. Aref, Mikhail Atallah, Ioannis Vlahavas, and Ahmed K. Elmagarmid. Multiple and parital periodicity in mining in time series databases. In *Proceedings of the 15th European Conerene on Artificial Intelligence*, 2002.

[82] Stéphane Dray, Manuela Royer-Carenzi, and Clément Calenge. The exploratory analysis of autocorrelation in animal-movement studies. *Ecological Research*, 25(3):673–681, 2010.

[83] Guillaume Péron, Chris H. Fleming, Rogerio C. de Paula, and Justin M. Calabrese. Uncovering periodic patterns of space use in animal tracking data with periodograms, including a new algorithm for the lomb-scargle periodogram and improved randomization tests. *Movement Ecology*, 4(1):19, 2016.

[84] Jacob T. VanderPlas. Understanding the lomb–scargle periodogram. *The Astrophysical Journal Supplement Series*, 236(1):16, may 2018.

[85] Devin S. Johnson, Joshua M. London, Mary-Anne Lea, and John W. Durban. Continuous-time correlated random walk model for animal telemetry data. *Ecology*, 89(5):1208–1215, 2008.

[86] Tsutomu T. Takeuchi. Constructing a bivariate distribution function with given marginals and correlation: application to the galaxy luminosity function. *Monthly Notices of the Royal Astronomical Society*, 406(3):1830–1840, 08 2010.

[87] Pravin K. Trivedi and David M. Zimmer. Copula Modeling: An Introduction for Practitioners. *Foundations and Trends(R) in Econometrics*, 1(1):1–111, April 2007.

[88] Yajie Zou, Xinzhi Zhong, Jinjun Tang, Xin Ye, Lingtao Wu, Muhammad Ijaz, and Yinhai Wang. A copula-based approach for accommodating the underreporting effect in wildlife-vehicle crash analysis. *Sustainability*, 11(2), 2019.

[89] Justin T. French, Hsiao-Hsuan Wang, William E. Grant, and John M. Tomeček. Dynamics of animal joint space use: a novel application of a time series approach. *Movement Ecology*, 7(1):38, 2019.

[90] Marti J. Anderson, Perry de Valpine, Andrew Punnett, and Arden E. Miller. A pathway for multivariate analysis of ecological communities using copulas. *Ecology and Evolution*, 9(6):3276–3294, 2019.

[91] B. J. Worton. Kernel methods for estimating the utilization distribution in home-range studies. *Ecology*, 70(1):164–168, 1989.

[92] D. Erran Seaman and Roger A. Powell. An evaluation of the accuracy of kernel density estimators for home range analysis. *Ecology*, 77(7):2075–2085, 1996.

[93] G. E. Uhlenbeck and L. S. Ornstein. On the theory of the brownian motion. *Phys. Rev.*, 36:823–841, Sep 1930.

[94] Volker Grimm, Steven F Railsback, Christian E Vincenot, Uta Berger, Cara Gallagher, Donald L DeAngelis, Bruce Edmonds, Jiaqi Ge, Jarl Giske, Juergen Groeneveld, et al. The odd protocol for describing agent-based and other simulation models: A second update to improve clarity, replication, and structural realism. *Journal of Artificial Societies and Social Simulation*, 23(2), 2020.

[95] Noelle E Thompson, David J Butts, Michael S Murillo, David M Williams, Sonja A Christensen, William F Porter, and Gary J Roloff. An individual-based model for direct and indirect transmission of chronic wasting disease in free-rangin white-tailed deer. *Ecological Modelling*, (under review).

[96] Ryan G McClarren, Penrose McClarren, and RL Penrose. *Uncertainty quantification and predictive computational science*. Springer, 2018.

[97] Arika Ligmann-Zielinska, Peer-Olaf Siebers, Nicholas Magliocca, Dawn C Parker, Volker Grimm, Jing Du, Martin Cenek, Viktoriia Radchuk, Nazia N Arbab, Sheng Li, et al. 'one size does not fit all': a roadmap of purpose-driven mixed-method pathways for sensitivity analysis of agent-based models. *Journal of Artificial Societies and Social Simulation*, 23(1), 2020.

[98] Andrea Saltelli, Stefano Tarantola, and KP-S Chan. A quantitative model-independent method for global sensitivity analysis of model output. *Technometrics*, 41(1):39–56, 1999.

[99] Jérôme Morio. Global and local sensitivity analysis methods for a physical system. *European journal of physics*, 32(6):1577, 2011.

[100] Arika Ligmann-Zielinska. Spatially-explicit sensitivity analysis of an agent-based model of land use change. *International Journal of Geographical Information Science*, 27(9):1764–1781, 2013.

[101] Bertrand Iooss and Paul Lemaître. A review on global sensitivity analysis methods. *Uncertainty management in simulation-optimization of complex systems: algorithms and applications*, pages 101–122, 2015.

[102] Andrea Saltelli, Stefano Tarantola, Francesca Campolongo, Marco Ratto, et al. *Sensitivity analysis in practice: a guide to assessing scientific models*, volume 1. Wiley Online Library, 2004.

[103] Emanuele Borgonovo and Elmar Plischke. Sensitivity analysis: A review of recent advances. *European Journal of Operational Research*, 248(3):869–887, 2016.

[104] Andrea Saltelli, Paola Annoni, Ivano Azzini, Francesca Campolongo, Marco Ratto, and Stefano Tarantola. Variance based sensitivity analysis of model output. design and estimator for the total sensitivity index. *Computer physics communications*, 181(2):259–270, 2010.

[105] IM Soboĺ. Sensitivity estimates for nonlinear mathematical models. *Math. Model. Comput. Exp.*, 1:407, 1993.

[106] Toshimitsu Homma and Andrea Saltelli. Importance measures in global sensitivity analysis of nonlinear models. *Reliability Engineering & System Safety*, 52(1):1–17, 1996.

[107] Max D Morris. Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33(2):161–174, 1991.

[108] Adam Badawy, Emilio Ferrara, and Kristina Lerman. Analyzing the digital traces of political manipulation: The 2016 russian interference twitter campaign. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 258–265, 2018.

[109] Adam Fourney, Miklos Z Racz, Gireeja Ranade, Markus Mobius, and Eric Horvitz. Geographic and temporal trends in fake news consumption during the 2016 us presidential election. In *CIKM*, volume 17, pages 6–10, 2017.

[110] Sahil Loomba, Alexandre de Figueiredo, Simon J Piatek, Kristen de Graaf, and Heidi J Larson. Measuring the impact of covid-19 vaccine misinformation on vaccination intent in the uk and usa. *Nature human behaviour*, 5(3):337–348, 2021.

[111] Talha Burki. Vaccine misinformation and social media. *The Lancet Digital Health*, 1(6):e258–e259, 2019.

[112] Warren Cornwall. Officials gird for a war on vaccine misinformation, 2020.

[113] Matthew W. Hughey. The who and why of qanon's rapid rise. *New Labor Forum*, 30(3):76–87, 2021.

[114] Derek du Preez. Chatgpt has the potential to spread misinformation 'at unprecedented scale'. https://diginomica.com/chatgpt-has-potential-spread-misinformation-unprecedented-scale, 2023.

[115] Jack Brewster, Lorenzo Arvanitis, and McKenzie Sadeghi. Misinformation monitor: January 2023. https://www.newsguardtech.com/misinformation-monitor/jan-2023/, 2023.

[116] Chatbots trigger next misinformation nightmare. https://www.axios.com/2023/02/21/chatbots-misinformation-nightmare-chatgpt-ai, 2023.

[117] Josh A. Goldstein, Girish Sastry, Micah Musser, Renee DiResta, Matthew Gentzel, and Katerina Sedova. Generative language models and automated influence operations: Emerging threats and potential mitigations, 2023.

[118] Monther Aldwairi and Ali Alwahedi. Detecting fake news in social media networks. *Procedia Computer Science*, 141:215–222, 2018. The 9th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2018) / The 8th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2018) / Affiliated Workshops.

[119] Supanya Aphiwongsophon and Prabhas Chongstitvatana. Detecting fake news with machine learning method. In *2018 15th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, pages 528–531, 2018.

[120] Jun Lin, Glenna Tremblay-Taylor, Guanyi Mou, Di You, and Kyumin Lee. Detecting fake news articles. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 3021–3025, 2019.

[121] Stephanie Preston, Anthony Anderson, David J. Robertson, Mark P. Shephard, and Narisong Huhe. Detecting fake news on facebook: The role of emotional intelligence. *PLOS ONE*, 16(3):1–13, 03 2021.

[122] Kai Shu, Amrita Bhattacharjee, Faisal Alatawi, Tahora H Nazer, Kaize Ding, Mansooreh Karami, and Huan Liu. Combating disinformation in a social media age. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(6):e1385, 2020.

[123] Florian Saurwein and Charlotte Spencer-Smith. Combating disinformation on social media: Multilevel governance and distributed accountability in europe. *Digital Journalism*, 8(6):820–841, 2020.

[124] Joanna M Burkhardt. *Combating fake news in the digital age*, volume 53. American Library Association Chicago, IL, 2017.

[125] Karishma Sharma, Feng Qian, He Jiang, Natali Ruchansky, Ming Zhang, and Yan Liu. Combating fake news: A survey on identification and mitigation techniques. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(3):1–42, 2019.

[126] Tools that fight disinformation online. https://www.rand.org/research/projects/truth-decay/fighting-disinformation/search.html.

[127] JA Gallo and CY Cho. Social media: Misinformation and content moderation issues for congress. *Congressional Research Service Report*, 46662, 2021.

[128] David MJ Lazer, Matthew A Baum, Yochai Benkler, Adam J Berinsky, Kelly M Greenhill, Filippo Menczer, Miriam J Metzger, Brendan Nyhan, Gordon Pennycook, David Rothschild, et al. The science of fake news. *Science*, 359(6380):1094–1096, 2018.

[129] Antino Kim, Patricia L Moravec, and Alan R Dennis. Combating fake news on social media with source ratings: The effects of user and expert reputation ratings. *Journal of Management Information Systems*, 36(3):931–968, 2019.

[130] Ceren Budak, Divyakant Agrawal, and Amr El Abbadi. Limiting the spread of misinformation in social networks. In *Proceedings of the 20th international conference on World wide web*, pages 665–674, 2011.

[131] Xinran He, Guojie Song, Wei Chen, and Qingye Jiang. Influence blocking maximization in social networks under the competitive linear threshold model. In *Proceedings of the 2012 siam international conference on data mining*, pages 463–474. SIAM, 2012.

[132] Mei Li, Xiang Wang, Kai Gao, and Shanshan Zhang. A survey on information diffusion in online social networks: Models and methods. *Information*, 8(4), 2017.

[133] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146, 2003.

[134] Thomas M Liggett et al. *Stochastic interacting systems: contact, voter and exclusion processes*, volume 324. springer science & Business Media, 1999.

[135] Giordano De Marzo, Andrea Zaccaria, and Claudio Castellano. Emergence of polarization in a voter model with personalized information. *Physical Review Research*, 2(4):043117, 2020.

[136] Luís Carlos F Latoski, WG Dantas, and Jeferson J Arenzon. Curvature-driven growth and interfacial noise in the voter model with self-induced zealots. *Physical Review E*, 106(1):014121, 2022.

[137] Robert Axelrod. The dissemination of culture: A model with local convergence and global polarization. *Journal of conflict resolution*, 41(2):203–226, 1997.

[138] William Ogilvy Kermack and Anderson G McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character*, 115(772):700–721, 1927.

[139] Daryl J Daley and David G Kendall. Epidemics and rumours. *Nature*, 204(4963):1118–1118, 1964.

[140] Daniel P Maki and Maynard Thompson. *Mathematical models and applications: with emphasis on the social, life, and management sciences*. Prentice Hall, 1973.

[141] José RC Piqueira, Mauro Zilbovicius, and Cristiane M Batistela. Daley–kendal models in fake-news scenario. *Physica A: Statistical Mechanics and its Applications*, 548:123406, 2020.

[142] Saeed Jamalzadeh, Kash Barker, Andrés D González, and Sridhar Radhakrishnan. Protecting infrastructure performance from disinformation attacks. *Scientific Reports*, 12(1):12707, 2022.

[143] Robert Axelrod, Joshua J Daymude, and Stephanie Forrest. Preventing extreme polarization of political attitudes. *Proceedings of the National Academy of Sciences*, 118(50), 2021.

[144] Luc Steels. A self-organizing spatial vocabulary. *Artificial life*, 2(3):319–332, 1995.

[145] Xiang Niu, Casey Doyle, Gyorgy Korniss, and Boleslaw K Szymanski. The impact of variable commitment in the naming game on consensus formation. *Scientific reports*, 7(1):1–11, 2017.

[146] Jierui Xie, Sameet Sreenivasan, Gyorgy Korniss, Weituo Zhang, Chjan Lim, and Boleslaw K Szymanski. Social consensus through the influence of committed minorities. *Physical Review E*, 84(1):011130, 2011.

[147] Jierui Xie, Jeffrey Emenheiser, Matthew Kirby, Sameet Sreenivasan, Boleslaw K Szymanski, and Gyorgy Korniss. Evolution of opinions on social networks in the presence of competing committed groups. *PLoS One*, 7(3):e33215, 2012.

[148] Damon Centola, Joshua Becker, Devon Brackbill, and Andrea Baronchelli. Experimental evidence for tipping points in social convention. *Science*, 360(6393):1116–1119, jun 2018.

[149] Sreeja Nair, Kin Wai Ng, Adriana Iamnitchi, and John Skvoretz. Diffusion of social conventions across polarized communities: an empirical study. *Social Network Analysis and Mining*, 11:1–17, 2021.

[150] William W Hackborn, Tetiana Reznychenko, and Yihang Zhang. Consensus building by committed agents. *CODEE Journal*, 12(1):2, 2019.

[151] David Galehouse, Tommy Nguyen, Sameet Sreenivasan, Omar Lizardo, G Korniss, and B Szymanski. Impact of network connectivity and agent commitment on spread of opinions in social networks. In *Proceedings of the 5th International Conference on Applied Human Factors and Ergonomics*, pages 2318–2329, 2014.

[152] Andrew M Thompson, Boleslaw K Szymanski, and Chjan C Lim. Propensity and stickiness in the naming game: Tipping fractions of minorities. *Physical review E*, 90(4):042809, 2014.

[153] Casey Doyle, Sameet Sreenivasan, Boleslaw K Szymanski, and Gyorgy Korniss. Social consensus and tipping points with opinion inertia. *Physica A: Statistical Mechanics and its Applications*, 443:316–323, 2016.

[154] Mauro Mobilia. Commitment versus persuasion in the three-party constrained voter model. *Journal of Statistical Physics*, 151(1):69–91, 2013.

[155] Weituo Zhang, Chjan Lim, and Boleslaw K Szymanski. Analytic treatment of tipping points for social consensus in large random networks. *Physical Review E*, 86(6):061134, 2012.

[156] Dina Mistry, Qian Zhang, Nicola Perra, and Andrea Baronchelli. Committed activists and the reshaping of status-quo social consensus. *Physical Review E*, 92(4):042805, 2015.

[157] Benedek Rozemberczki and Rik Sarkar. Characteristic Functions on Graphs: Birds of a Feather, from Statistical Descriptors to Parametric Models. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*, page 1325–1334. ACM, 2020.

[158] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[159] Yifan Hu. Efficient, high-quality force-directed graph drawing. *Mathematica journal*, 10(1):37–71, 2005.

[160] How to tailor covid-19 vaccine information to your specific audience. https://www.cdc.gov/vaccines/covid-19/hcp/tailoring-information.html.

[161] Mark Newman. *Networks*. Oxford university press, 2018.

[162] Bert Vogelstein, David Lane, and Arnold J. Levine. Surfing the p53 network. *Nature*, 408(6810):307–310, November 2000. Number: 6810 Publisher: Nature Publishing Group.

[163] Jingchun Chen and Bo Yuan. Detecting functional modules in the yeast protein–protein interaction network. *Bioinformatics*, 22(18):2283–2290, September 2006.

[164] Albert-Laszlo Barabasi and Zoltan N Oltvai. Network biology: understanding the cell's functional organization. *Nature reviews genetics*, 5(2):101–113, 2004.

[165] Matthew O. Jackson. *Social and economic networks*. Princeton University Press, Princeton, N.J. Woodstock, 2011.

[166] Morris H DeGroot. Reaching a consensus. *Journal of the American Statistical association*, 69(345):118–121, 1974.

[167] Nicholas Economides. The economics of networks. *International Journal of Industrial Organization*, 14(6):673–699, October 1996.

[168] W Brian Arthur. *Increasing returns and path dependence in the economy*. University of michigan Press, 1994.

[169] Romualdo Pastor-Satorras and Alessandro Vespignani. Epidemic spreading in scale-free networks. *Physical review letters*, 86(14):3200, 2001.

[170] István Z Kiss, Joel C Miller, Péter L Simon, et al. Mathematics of epidemics on networks. *Cham: Springer*, 598:31, 2017.

[171] Thomas House. Modelling epidemics on networks. *Contemporary Physics*, 53(3):213–225, 2012.

[172] Sebastiano A Delre, Wander Jager, Tammo HA Bijmolt, and Marco A Janssen. Will it spread or not? the effects of social influences and network topology on innovation diffusion. *Journal of Product Innovation Management*, 27(2):267–282, 2010.

[173] Daron Acemoglu, Asuman Ozdaglar, and Ali ParandehGheibi. Spread of (mis) information in social networks. *Games and Economic Behavior*, 70(2):194–227, 2010.

[174] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015.

[175] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[176] Marinka Zitnik, Rok Sosič, Sagar Maheshwari, and Jure Leskovec. BioSNAP Datasets: Stanford biomedical network dataset collection. http://snap.stanford.edu/biodata, August 2018.

[177] Srijan Sengupta. Statistical Network Analysis: Past, Present, and Future, October 2023. arXiv:2311.00122 [stat].

[178] Grzegorz Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, SIGMOD '10, pages 135–146, New York, NY, USA, June 2010. Association for Computing Machinery.

[179] Robert Ryan McCune, Tim Weninger, and Greg Madey. Thinking Like a Vertex: A Survey of Vertex-Centric Frameworks for Large-Scale Distributed Graph Processing. *ACM Computing Surveys*, 48(2):25:1–25:39, October 2015.

[180] Nadathur Satish, Changkyu Kim, Jatin Chhugani, Hideki Saito, Rakesh Krishnaiyer, Mikhail Smelyanskiy, Milind Girkar, and Pradeep Dubey. Can traditional programming bridge the Ninja performance gap for parallel computing applications? *ACM SIGARCH Computer Architecture News*, 40(3):440–451, June 2012.

[181] Nadathur Satish, Narayanan Sundaram, Md. Mostofa Ali Patwary, Jiwon Seo, Jongsoo Park, M. Amber Hassaan, Shubho Sengupta, Zhaoming Yin, and Pradeep Dubey. Navigating the maze of graph analytics frameworks using massive graph datasets. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 979–990, New York, NY, USA, June 2014. Association for Computing Machinery.

[182] Paul Erdős, Alfréd Rényi, et al. On the evolution of random graphs. *Publ. math. inst. hung. acad. sci*, 5(1):17–60, 1960.

[183] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[184] David J Earl and Michael W Deem. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7(23):3910–3916, 2005.

[185] Robert H Swendsen and Jian-Sheng Wang. Replica monte carlo simulation of spin-glasses. *Physical review letters*, 57(21):2607, 1986.

[186] Gareth O Roberts and Jeffrey S Rosenthal. Optimal scaling for various metropolis-hastings algorithms. *Statistical science*, 16(4):351–367, 2001.

[187] Andrew Gelman, Walter R Gilks, and Gareth O Roberts. Weak convergence and optimal scaling of random walk metropolis algorithms. *The annals of applied probability*, 7(1):110–120, 1997.

[188] Hamsterster. Hamsterster social network. http://www.hamsterster.com.

[189] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: an open source software for exploring and manipulating networks. In *Third international AAAI conference on weblogs and social media*, 2009.

[190] Yves F Atchadé, Gareth O Roberts, and Jeffrey S Rosenthal. Towards optimal scaling of metropolis-coupled markov chain monte carlo. *Statistics and Computing*, 21:555–568, 2011.

[191] Paul K Huth. Deterrence and international conflict: Empirical findings and theoretical debates. *Annual Review of Political Science*, 2(1):25–48, 1999.

[192] Moshe Kress. Modeling armed conflicts. *Science*, 336(6083):865–869, 2012.

[193] Gerardo Minguela-Castro, Ruben Heradio, and Carlos Cerrada. Automated support for battle decision making. *Military Operations Research*, 27(4):5–24, 2022.

[194] Thomas C Schelling. Arms and influence. In *Strategic Studies*, pages 96–114. Routledge, 2008.

[195] Frederick William Lanchester. *Aircraft in warfare: The dawn of the fourth arm*. Constable limited, 1916.

[196] Manvi Sahni and Sumanta Kumar Das. Performance of maximum likelihood estimator for fitting lanchester equations on kursk battle data. *Journal of Battlefield Technology*, 18(2):23–30, 2015.

[197] Thomas W Lucas and Turker Turkes. Fitting lanchester equations to the battles of kursk and ardennes. *Naval Research Logistics (NRL)*, 51(1):95–116, 2004.

[198] Ian R Johnson and Niall J MacKay. Lanchester models and the battle of britain. *Naval Research Logistics (NRL)*, 58(3):210–222, 2011.

[199] Jerome Bracken. Lanchester models of the ardennes campaign. *Naval Research Logistics (NRL)*, 42(4):559–577, 1995.

[200] Moshe Kress, Jonathan P. Caulkins, Gustav Feichtinger, Dieter Grass, and Andrea Seidl. Lanchester model for three-way combat. *European Journal of Operational Research*, 264(1):46–54, 2018.

[201] Moshe Kress. Lanchester models for irregular warfare. *Mathematics*, 8(5):737, 2020.

[202] Eduardo González and Marcelo Villena. Spatial lanchester models. *European Journal of Operational Research*, 210(3):706–715, 2011.

[203] William R Caspary. Richardson's model of arms races: description, critique, and an alternative model. *International Studies Quarterly*, 11(1):63–88, 1967.

[204] Norman Z Alcock and Keith Lowe. The vietnam war as a richardson process. *Journal of Peace Research*, 6(2):105–111, 1969.

[205] Jean-Christian Lambelet. *A dynamic model of the arms race in the Middle East, 1953-1965*. Society for General Systems Research, 1971.

[206] Seymour J Deitchman. A lanchester model of guerrilla warfare. *Operations Research*, 10(6):818–827, 1962.

[207] Hannes Risken and Hannes Risken. *Fokker-planck equation*. Springer, 1996.

[208] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[209] Takuya Iwanaga, William Usher, and Jonathan Herman. Toward SALib 2.0: Advancing the accessibility and interpretability of global sensitivity analyses. *Socio-Environmental Systems Modelling*, 4:18155, May 2022.

[210] Jon Herman and Will Usher. SALib: An open-source python library for sensitivity analysis. *The Journal of Open Source Software*, 2(9), jan 2017.

[211] Francesca Pianosi and Thorsten Wagener. A simple and efficient method for global sensitivity analysis based on cumulative distribution functions. *Environmental Modelling & Software*, 67:1–11, 2015.

[212] Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023.

[213] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[214] Christopher W Moore. *The mediation process: Practical strategies for resolving conflict*. John Wiley & Sons, 2014.

[215] Claude-Hélène Mayer. *Intercultural Mediation and Conflict Management Training*. Springer, 2020.

[216] Solomon M Hsiang, Marshall Burke, and Edward Miguel. Quantifying the influence of climate on human conflict. *Science*, 341(6151):1235367, 2013.

[217] Christopher Cramer. Does inequality cause conflict? *Journal of International Development: The Journal of the Development Studies Association*, 15(4):397–412, 2003.

[218] Frances Stewart, Douglas Holdstock, and Antonio Jarquin. Root causes of violent conflict in developing countriescommentary: Conflict—from causes to prevention? *bmj*, 324(7333):342–345, 2002.

[219] Gerry O'Reilly. *Aligning geopolitics, humanitarian action and geography in times of conflict*. Springer, 2019.

[220] Derek J Clark and Kai A Konrad. Asymmetric conflict: Weakest link against best shot. *Journal of Conflict Resolution*, 51(3):457–469, 2007.

[221] J Paul Dunne, María DC García-Alonso, Paul Levine, and Ron P Smith. Managing asymmetric conflict. *Oxford Economic Papers*, 58(2):183–208, 2006.

[222] Frederick William Lanchester. *Aircraft in warfare: The dawn of the fourth arm*. Constable limited, 1916.

[223] Dominic DP Johnson and Niall J MacKay. Fight the power: Lanchester's laws of combat in human evolution. *Evolution and Human Behavior*, 36(2):152–163, 2015.

[224] Lucas McDaniel, Erik Talvi, and Brian Hay. Capture the flag as cyber security introduction. In *2016 49th hawaii international conference on system sciences (hicss)*, pages 5479–5486. IEEE, 2016.

[225] Fabio Massimo Zennaro and Laszlo Erdodi. Modeling penetration testing with reinforcement learning using capture-the-flag challenges: trade-offs between model-free learning and a priori knowledge. *arXiv preprint arXiv:2005.12632*, 2020.

[226] Crispin Cowan, Seth Arnold, Steve Beattie, Chris Wright, and John Viega. Defcon capture the flag: Defending vulnerable code from intense attack. In *Proceedings DARPA Information Survivability Conference and Exposition*, volume 1, pages 120–129. IEEE, 2003.

[227] Matthew A Blake, Gerrit A Sorensen, James K Archibald, and Randal W Beard. Human assisted capture-the-flag in an urban environment. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 2, pages 1167–1172. IEEE, 2004.

[228] Michael Novitzky, Paul Robinette, Michael R Benjamin, Danielle K Gleason, Caileigh Fitzgerald, and Henrik Schmidt. Preliminary interactions of human-robot trust, cognitive load, and robot intelligence levels in a competitive game. In *Companion of the 2018 ACM/IEEE international conference on human-robot interaction*, pages 203–204, 2018.

[229] Gorka Olalde Mendia, Lander Usategui San Juan, Xabier Perez Bascaran, Asier Bilbao Calvo, Alejandro Hernández Cordero, Irati Zamalloa Ugarte, Aday Muniz Rosas, David Mayoral Vilches, Unai Ayucar Carbajo, Laura Alzola Kirschgens, et al. Robotics ctf (rctf), a playground for robot hacking. *arXiv preprint arXiv:1810.02690*, 2018.

[230] Alexandros Merkouris, Varvara Garneli, and Konstantinos Chorianopoulos. Programming human-robot interactions for teaching robotics within a collaborative learning open space: Robots playing capture the flag game: Programming human-robot interactions within a collaborative learning open space. In *CHI Greece 2021: 1st International Conference of the ACM Greek SIGCHI Chapter*, pages 1–5, 2021.

[231] Eisha Akanksha, Neeraj Sharma, Kamal Gulati, et al. Review on reinforcement learning, research evolution and scope of application. In *2021 5th international conference on computing methodologies and communication (ICCMC)*, pages 1416–1423. IEEE, 2021.

[232] Aurélien Géron. Hands-on machine learning with scikit-learn and tensorflow: Concepts. *Tools, and Techniques to build intelligent systems*, 2017.

[233] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[234] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[235] RICHARD Bellman. Dynamic programming, princeton univ. *Press Princeton, New Jersey*, 1957.

[236] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44, 1988.

[237] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[238] Kenji Doya. Reinforcement learning in continuous time and space. *Neural computation*, 12(1):219–245, 2000.

[239] Hado Van Hasselt. Reinforcement learning in continuous state and action spaces. *Reinforcement Learning: State-of-the-Art*, pages 207–251, 2012.

[240] Alessandro Lazaric, Marcello Restelli, and Andrea Bonarini. Reinforcement learning in continuous action spaces through sequential monte carlo methods. *Advances in neural information processing systems*, 20, 2007.

[241] Adarsh Sehgal, Hung La, Sushil Louis, and Hai Nguyen. Deep reinforcement learning using genetic algorithm for parameter optimization. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, pages 596–601. IEEE, 2019.

[242] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*, 2017.

[243] Sean Gillies, Casper van der Wel, Joris Van den Bossche, Mike W. Taves, Joshua Arnott, Brendan C. Ward, et al. Shapely, December 2022. Please cite this software using these metadata.

[244] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Best Papers, São Paulo, Brazil, May 8-12, 2017, Revised Selected Papers 16*, pages 66–83. Springer, 2017.

[245] Soummya Kar, José MF Moura, and H Vincent Poor. Qd-learning: A collaborative distributed strategy for multi-agent reinforcement learning through consensus + innovations. *IEEE Transactions on Signal Processing*, 61(7):1848–1862, 2013.

[246] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. Multi-agent reinforcement learning: An overview. *Innovations in multi-agent systems and applications-1*, pages 183–221, 2010.

[247] Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE transactions on cybernetics*, 50(9):3826–3839, 2020.

[248] Benjamin Golub and Matthew O Jackson. Naive learning in social networks and the wisdom of crowds. *American Economic Journal: Microeconomics*, 2(1):112–149, 2010.

APPENDIX A: SUPPLEMENT TO CHAPTER 2

In this supplement we provide a description of our model using the updated Overview, Design concepts and Details (ODD) protocol [94].

Overview, Design concepts, and Details

Purpose and patterns

The purpose of our model is to produce new, unobserved GPS locations of white-tailed deer that are consistent with recorded GPS location data of white-tailed deer in the Midwest. There are three patterns our model aims to reproduce.

1. The first pattern is the distribution of jumps (distance between two positions recorded one after another) should follow a Laplace distribution as observed in our recorded data.

2. The deer in our data move in one or more distinct spatial locations and occasionally jump between them. We refer to these spatial locations as basins and the jumps between them as basin hops. When animals are in one of these spatial locations, their movements are constrained to a localized region. Therefore the second pattern is reproducing movements in a single basin.

3. The final pattern is reproducing the hops between the basins.

Entities, state variables, and scales

Our model consists of the following three entities.

1. A deer agent that represents a single deer that moves individually in a basin,

2. a herd collective that represents a group of deer that undergo basin hops together, and

3. a world environment containing the herds and deer. The world initializes and runs each simulation.

The state variables for each of the entities are summarized in Table 8.1, Table 8.2, and Table 8.3.

The spatial scale of our model is a single UTM zone as our GPS data employed the UTM coordinate system and all data are in a single zone. We generated positions in smaller areas (approximately 18 km^2). We chose this smaller region to examine herd movements that are close to each other and could overlap. A single time-step in our simulation is 5 hours, since that was the resolution our GPS was measured. In our example simulations we ran our model for 1000 steps (approximately 208 days).

Variable name	Type and units	Description
x	float, dynamic; m	The UTM Easting of a deer.
y	float, dynamic; m	The UTM Northing of a deer.
c1x	float, static; unitless	Movement model parameter. Controls size of basin in combination with c1y.
c1y	float, static; unitless	Movement model parameter. Controls size of basin in combination with c1x.
bx	float, static; m^{-1}	Laplace distribution scale parameter for UTM Easting jump distribution.
by	float, static; m^{-1}	Laplace distribution scale parameter for UTM Northing jump distribution.
cov_mat	float array, static; m^2	Covariance matrix of UTM Easting and UTM Northing data.

Table 8.1: Deer state variables.

Variable name	Type and units	Description
num_deer	int, static; deer	The number of deer in the herd.
cx	float, dynamic; m	UTM Easting of center of basin.
cy	float, dynamic; m	UTM Northing of center of basin.
num_crit_jumps_data	int, jumps	Scale for Poisson process.
xtrds	float array, static;m	Basin hop trend removed from UTM Easting data.
ytrds	float array, static; m	Basin hop trend removed from UTM Northing data.
deer_list	Deer list, static; deer object	List of deer that are in the herd.
jump_times	int array, static; time-steps	Time-steps which basin hops occur.
time	int, dynamic; time-step	Herd's current time-step.

Table 8.2: Herd state variables.

Variable name	Type and units	Description
num_herds	int, static; herds	Number of herds in a simulation.
herd_sizes	int array, static; deer	Number of deer in each herd.
herd_pos	int tuple list, static; (m,m)	Initial position of each herd.
herd_list	herd list, static; herd object	List of herds in simulation.

Table 8.3: World state variables.

Process overview and scheduling

We designed our model to produce GPS locations of white-tailed deer in the Midwest measured at approximately 5 hour intervals, as this was the location and time-scale of our data. There are two movement processes that we wished to capture, localized movement (within

basins) and uncommon large movements (basin hops).

Our data does not track every deer in a region, and thus we assumed that groups of deer (herds) existed and our data tracked one deer in a group. We assumed that all deer in a shared herd were constrained to the same area (basin), but had independent movements from each other. When a basin-hop occurred, we assumed all deer in the herd underwent the large movement together. These processes are captured through the herd's and deer's "move" submodel.

The schedule of each time-step is as follows.

1. The world calls its "update" submodel, which includes:
    (a) Each herd calls their "move" submodel, which includes:
        i. Each deer calls their "move" submodel, in which they update their individual positions (x,y) according to the parameters of their herd and a random component.
        ii. The herd checks if a basin-hop should occur on the current time-step. If yes:
            A. A basin-hop is sampled and all deer in the herd are moved according to the sampled jump.
            B. The center of the basin is moved according to the sampled jump, updating cx and cy.
        iii. time is incremented by one.

Design concepts

Basic principals

In designing our model we aimed to highlight the importance of EDA in model creation. This concept is widely applicable to agent-based models of different phenomenon. We additionally wanted to create a model that was able to reproduce three aspects of our data, non-Gaussian distributed step distributions, localized movement (within basins), and uncommon large movements (basin-hops). It is common to use Gaussian random walks of some flavor and/or Levy flights as models for animal movement, but through our EDA we found these models were not consistent with our data. Therefore, we used our data as a guide to build a model that was more representative of our data. Namely, our model produces GPS locations of white-tailed deer using a non-Gaussian random walk in combination with a basin-hopping model to produce realistic simulated GPS positions. This model could be extended to study other phenomenon related to deer movement, for example the spread of CWD.

Emergence

They key result of our model are the generated GPS locations. Specifically, jumps (differences between adjacent locations) distributions being Laplace distributed, local movements (inside basins), and uncommon large movements (basin-hops). In our model, the distribution of jumps, and two movement patterns are imposed by the deer's movement update rule. From the deer's movement rule, the spatial distribution of deer in a simulation emerge.

Adaptation

Our model has one adaptive behavior, whether or not a herd performs a basin-hop. This behavior is modeled with indirect objective seeking. There are two rules herds follow to produce basin-hops. First, herds sample multiple time-steps that they will perform a basin-hop at. These distributions are trained on real data to produce a number of jumps similar to the number observed in our data. The second rule is when a herd's time is equal to one of the basin-hop times, it samples a basin hop from a kernel density estimate trained on the observed basin hops, and move all of the deer in its deer_list and its center (cx,cy) according to the sampled jump.

Objectives

Our model only has indirect objective seeking, so there are no objectives.

Learning

The model includes no learning.

Prediction

The adaptive behavior of basin-hopping utilizes implicit prediction. There are many reasons that deer might disperse, for example finding a new food source or a dispersal event. Our model is not attempting to explain how these predictions are actually made, but are modeling them to produce results that are similar to those observed in our data.

Sensing

Deer are the only agents in this model. They are able to perfectly sense all of their state variable. Their position (x,y) and movement parameters (c1x, c1y) are used directly to update their positions, and the remaining variables are used to produce a random component to their movement. Deer are also able to perfectly sense the center of a the herd they belong

to (cx, cy) but not any other herd. They are attracted to the center of the herd they belong to.

Interaction

Deer do not directly interact with each other, as their inner-basin movements are independent. Though deer who belong to a common herd are constrained to the same localized regions (basins) as they are attracted towards the center of the herd. They also undergo basin-hops together.

Stochasticity

Our model uses stochasticity in many ways. In the initialization, herds initial center could be set using a random number generator. Herds are then randomly assigned state variables by sampling distributions trained on our data to include variability in the herds. Using these variables, they sample a number of basin-hops they will perform from an Poisson distribution and when the basin-hops will occur from an exponential distribution. Once the herds have been created, they populate themselves with deer. The deer's initial positions are generated randomly by adding separate random numbers, sampled from a normal distribution with zero mean and unit variance, to the herds center (cx,cy). These additional random numbers keep all of the deer from starting at the same location. Deer in different herds are randomly assigned state variables, but deer in a single herd have the same state variables (excluding x and y)

There is also stochasticity used when the model updates. When a deer moves, it adds a random component to its new position sampling a distribution constructed from our data. This keeps deer from making the same moves, and from deer just moving to the center of the basin they are in. Additionally, when a herd undergoes a basin-hop it samples a random jump from a distribution constructed from our data. We sample these jumps to add variability in our model and produce results that differ from our data, but are consistent with the data.

Collectives

We have one explicit collective in our model, herds. Herds are groups of deer that move in the same localized areas (basins) and undergo basin hops together. Herds have their own state variables (see Table 8.2). We included herds as a collective to approximate the movement of deer that were not tracked in our data. We assumed that the deer that happened to be tracked had some number of other untracked deer they stayed near. As discussed above the

deer in a herd stay in the same basin and undergo basin hops together. It was convenient to model groups of deer as a collective instead of each individual deer on their own.

Observation

The purpose of our model is to produce new, realistic GPS locations of white-tailed deer. Therefore the output of our model is the positions of each deer and the herd they belong to. These data can be used to make density distribution plots of each herd, show the GPS locations of each herd, and the individual GPS locations of each deer. World has a submodel "draw" to draw each of the herds GPS locations for all deer. Herds has a submodel "draw" to draw the GPS locations of each deer in the herd. Finally, deer has a submodel "draw" that draws its GPS locations.

Initialization

A total number of herds, their sizes, and centers are defined. These parameters along with distributions of c1x, c1y, bx, by, cov_mat, num_crit_jumps_data, xtrds, ytrds are passed to an instantiation of world.

Each herd is passed their corresponding herd size and center, and all distributions which were passed to world. The heard then randomly samples the distributions for its state variables. Then the herd constructs a kernel density estimate from xtrds and ytrds to be used to sample basin-hops later. Next the herd instantiates the total number of deer passed from world to deer_list. It passes a random initial location (see Sec. 8), and samples the deers state variables. Each deer in a herd have the same state variables, except x and y. Next the herd sets the parameter time to zero, and samples the number and times of basin hops.

For each deer in a herd, it was passes a random initial location and state variables. These are stored to be used in updating positions.

Input data

The model does not use input data to represent time-varying processes.

Submodels

World

__init__ This submodel instantiates a simulation by spawning all of the herds to be simulated.

draw This submodel calls the draw method for all of the herds, which draws the GPS locations of all the deer.

update This submodel calls the "move" method for all of the herds.

Herd

__init__ This submodel instantiates a herd. It samples the state variables for itself, and those to be passed to the deer that belongs to the herd. It then spawns all of the deer that belong to the herd.

get_params This submodel samples all of the distributions that are passed to a herd by world. This is done by generating a uniform random number between zero and the total number of samples in the distributions. Then the parameters in the position that corresponds to that random number are returned.

draw This submodel draws the GPS location of all deer in the herd.

make_kde This submodel smooths xtrds and ytrds for a herd. Then combines xtrds and ytrds as a list of two dimensional points. Then each adjacent point is subtracted to arrive at a list of jumps. Next the average distance between all jumps in list is found. Half of this average distance was used as the bandwidth for a Gaussian kernel density estimate (KDE) fit on the list of jump. The resulting KDE can be sampled to generate basin-hops.

move This submodel calls the move function for every deer in the herd. Then it checks if a basin-hop should occur on the current time-step. If so it samples a jump from the KDE produced by "make_kde". This jump is added to the center of the herd (cx,cy) and every deer's position in the herd. Finally, time is incremented by one to advance the current time-step.

lasso_times The submodel samples a number of simulated basin-hops from a Poisson distribution. The argument of the Poisson distribution is the state variable num_crit_jumps_data. After it generates the number of simulated basin-hops, it samples a time-step for each step to occur. The cumulative sum of these times is returned.

Deer

__init__ This submodel instantiates a deer.

draw This submodel plots the GPS location of a simulated deer.

rng This submodel uses cov_mat to generate normally distributed jumps. It then uses a copula (this requires the state variables bx and by) to transform the normally distributed jumps to Laplace distributed ones.

move This submodel updates a deers position. The deer move towards the center of the basin they belong to, and sample "rng" to add a random component to their new position.

APPENDIX B: SUPPLEMENT TO CHAPTER 4

DeGroot Model

In this Supplement, we analyze and adapt a second model, the DeGroot model, [166] to each of the policies discussed in the main document. The DeGroot model was chosen for its mathematical transparency, with the goal of assessing its ability to describe the effects of disinformation-mitigation policies. In Subsection 8, we review the basic properties of the DeGroot model. In the following subsections, the model is then modified to include disinformation-mitigation strategies. Section 8, Section 8, and Section 8 contain figures illustrating the effects of content moderation, education, and counter-campaigns, respectively, in this model.

DeGroot Model

The model employed in the main document requires moderately intensive computational effort to explore the wide ranges of anti-disinformation policies, their model implementations and social graphs. It would be convenient to have an analytic model of the spread of disinformation that provides insight and intuition with minimal computational effort. A model that employs a linear update, contains social-network structure and has an equivalent of a committed agent is the DeGroot model. [166] We adapt this model to include the three interventions policies (education, content moderation, and counter campaigns) we consider in this work and their effects on disinformation dynamics.

In the DeGroot model, we consider a group of N agents connected by a network, which can be represented by an adjacency matrix $\mathbf{A} = [a_{ij}]$. Each agent forms its belief as a weighted average of the beliefs of its neighbors. Formally, if $\mathbf{x}(t) = [x_1(t), x_2(t), \ldots, x_N(t)]^T$ represents the beliefs of the N agents at time t, then the belief of agent i at time $t+1$ is given by the discrete-time update

$$x_i(t+1) = \sum_{j=1}^{N} t_{ij} x_j(t), \tag{8.1}$$

or in matrix notation,

$$\mathbf{x}(t+k) = \mathbf{T}^k \mathbf{x}(t). \tag{8.2}$$

Here, $\mathbf{T} = [t_{ij}]$ is referred to as the trust matrix, with t_{ij} representing the (constant) weight that agent i assigns to agent j when updating its belief; in other words, t_{ij} quantifies the extent to which agent i trusts agent j. The trust matrix $\mathbf{T}$ is derived from the adjacency matrix A by normalizing each row to sum to one; i.e., $t_{ij} = a_{ij} / \sum_{k=1}^{N} a_{ik}$. Starting from

some initial set of beliefs $\mathbf{x}(0)$, this model repeatedly updates each agent's belief based on a weighted sum of all other agents' beliefs, weighted by levels of trust.

In this framework, we have an interpretable update rule in which t_{ij} represents a trust level that varies from $t_{ij} = 0$ for no trust to $t_{ij} = 1$ for complete trust; the model can be initialized to these discrete values, or trust levels can be allowed to be continuous. In Fig. 8.1, we show examples of symmetric, directed graphs with equal weights, and their corresponding trust matrices. In general, however, the graph represented by $\mathbf{A}$ is weighted and directed: I may listen to you, but you need not listen to me. It is through flexibility in $\mathbf{T}$ that we include committed agents and mitigation strategies in the DeGroot model; in the context of the DeGroot model, committed agents are referred to as "stubborns".[248]

Many details of the model dynamics are easily revealed by considering the structure of $\mathbf{T}$. By construction, $\mathbf{T}$ is a right-stochastic matrix and thus has a spectral radius of 1; this is an important property, as updates are obtained using progressively higher powers of $\mathbf{T}$, as in (8.2). At least one eigenvalue is 1 and corresponds to a right eigenvector whose elements are all 1. Eigenvalues associated with left and right eigenvectors are equal for square matrices; this implies that T has at least one left eigenvector associated with an eigenvalue of 1. If $\mathbf{T}$ can be represented as a directed, weighted graph that is strongly connected (i.e., irreducible), then we can apply the Perron-Frobenius theorem that states that there is a unique left eigenvector $\mathbf{v}$ that sums to one and corresponds to an eigenvalue of 1. This eigenvector determines the final consensus belief; specifically, the final state is given by

$$x_{\text{final}} = \mathbf{v}^T \mathbf{x}(0), \tag{8.3}$$

where we see that $\mathbf{v}$ represents the long-term influence of the agents. This equation identifies $\mathbf{v}$, which is the eigenvector centrality of $\mathbf{A}$, as the relevant parameter for describing the spread of information, including disinformation. Intuitively, eigenvector centrality measures a node's influence in a network by considering both the number and quality of its connections, recursively factoring in the influence of neighboring nodes. It is also immediately clear from (8.3) that initial beliefs $\mathbf{x}(0)$ contribute to the final consensus. A block structure in $\mathbf{T}$ represents independent echo chambers that evolve according to the reducibility class of each block. It is possible that $\mathbf{T}$ is periodic, although some structures (e.g., upper/lower triangular or symmetric) guarantee that it is aperiodic. We can also examine the second largest eigenvalue of $\mathbf{T}$ to obtain the eigenvalue gap, which reveals the speed of convergence; larger gaps are associated with faster convergence.

We now extend the DeGroot model to include the three disinformation-mitigation strategies – education, content moderation, and counter campaigns – that we consider in this work

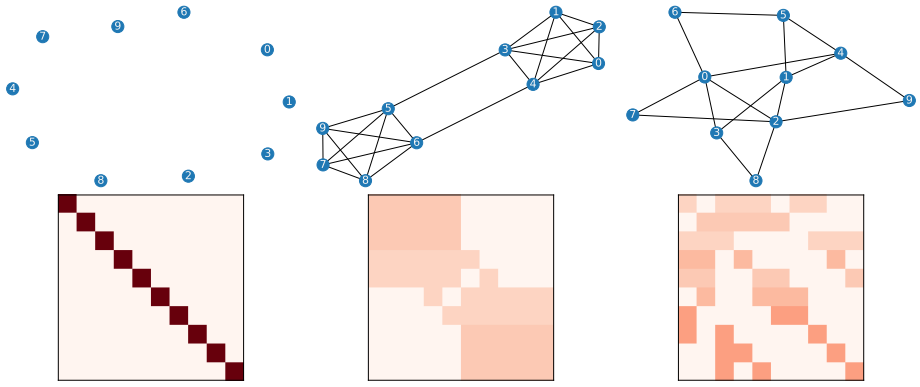


Figure 8.1: Example graphs and their corresponding trust matrices. Despite its simplicity, the DeGroot model contains a complete social network structure as a directed and weighted graph. Social network graphs are shown in the top row, and their corresponding trust matrices are shown in the bottom row.

and their effects on the spread of disinformation in the presence of stubborns. We model these interventions as modifications to the trust matrix and/or the initial condition and analyze their impact through the lens of the eigenvalue spectrum and eigenvectors of the modified trust matrix. For each mitigation strategy, we begin with a (scale-free) Barabasi-Albert graph with 100 nodes. Each edge is then converted into two directed arcs that point in opposite directions, and their weights are determined using social transitivity (described in the main document). This method of generating directed and weighted edges guarantees that the corresponding trust matrix is irreducible, because the graph is clearly strongly connected. Opinions have values between 0 and 1; here, we assume that 0 is the truth and that 1 is disinformation. Values between 0 and 1 indicate a bias towards the truth or disinformation, with the exception of 0.5, which indicates a completely neutral position.

Stubborns are added to the model by selecting an individual i and setting $t_{ii} = 1$, with $t_{ij} = 0$ for $i \neq j$. Importantly, the addition of stubborns make the trust matrix reducible, and consensus is not necessarily reached. However, opinions will converge to a value that depends on the structure of the trust matrix and on agents' initial opinions. One important scenario is that in which all stubborns have the same opinion. In such a case, any node that is reachable from a stubborn will converge to the stubborn's opinion. Using our method for generating trust matrices guarantees that the presence of one or more stubborns with the same opinion, and no stubborns with a different opinion, will eventually converge the network to a consensus of their opinion. Unless otherwise stated, $x_i(0) = 1$ if an individual is stubborn, and is initially 0 otherwise, in each of our scenarios; i.e., stubborns are committed to disinformation, and all other agents initially believe the truth. The goal of mitigation strategies is to minimize x_{final} when a consensus is reached, or if a consensus is not reached,

then the goal is to push opinions towards the truth or to minimize the eigenvalue gap when opinions tend towards disinformation.

Education

Skeptical and attentive education strategies are modeled by altering the t_{ij} based on the type of education received by each agent i. In the skeptical education strategy, the values of the diagonal elements of $\mathbf{T}$ of committed agents are decreased; this reflects a weaker commitment to one's own belief. In the attentive strategy, the t_{ij} of neighbors adjacent to stubborn agents are decreased. To test both education strategies, utilized Barabási-Albert graphs with 100 agents, selecting 10 of these agents to be committed to opinion A, leaving the remainder uncommitted to opinion B. The edges of these graphs are then transformed into two directed edges in both directions, weighted using social transitivity as described in the main document. We generate a trust matrices, $\{\mathbf{T}\}$ by normalizing the adjacency matrices of these graphs, and then we apply our education mitigation strategies to these trust matrices.

The skepticism educational strategy reduces diagonal elements t_{ii} of the trust matrix by a factor $\sigma < 1$, and the remaining elements corresponding to a committed individual's neighbors (determined by the adjacency matrix) are increased by equal parts of σt_{ii}, so that the row still sums to 1. Such a change to $\mathbf{T}$ changes the eigenvalue spectrum. Most importantly, it guarantees that $t_{ii} < 1$, ensuring that the matrix is reducible, and a consensus opinion is reached at a value less than 1. In Fig. 8.2, we show the consensus opinion versus σ. For each value of σ, we executed 50 simulations and averaged over initial conditions and graphs; for each value of σ, the average consensus opinion reached is shown with a black $\mathbf{x}$, and the value each simulation converged to is shown with a grey dot. When there is no education ($\sigma = 0$), the stubborns' opinion takes over, and all simulations converge to disinformation. As the stubborns become more skeptical (σ increases), the average value of consensus decreases and approaches a constant value of approximately 0.2. The variance in the consensus value is approximately constant for all values of $\sigma > 0$.

The attentive strategy is modeled by increasing elements t_{ij} that correspond to non-stubborn individuals i listening to other non-stubborn individuals j with initial opinions of zero, by a factor $\alpha > 1$, and reducing elements $t_{i,k}$ equally such that the rows of $\mathbf{T}$ sum to 1. Such a change biases individuals to listen more to true opinions. The attentive educational strategy does not affect the opinions of stubborn individuals, which means that eventually, the population will still converge to the disinformation opinion. However, the attentive strategy can affect how quickly the model converges, which is measured by the eigenvalue gap. A smaller eigenvalue gap is associated with a longer convergence time. In Fig. 8.3, we examine the eigenvalue gap versus α in 50 simulations. The blue line shows the average

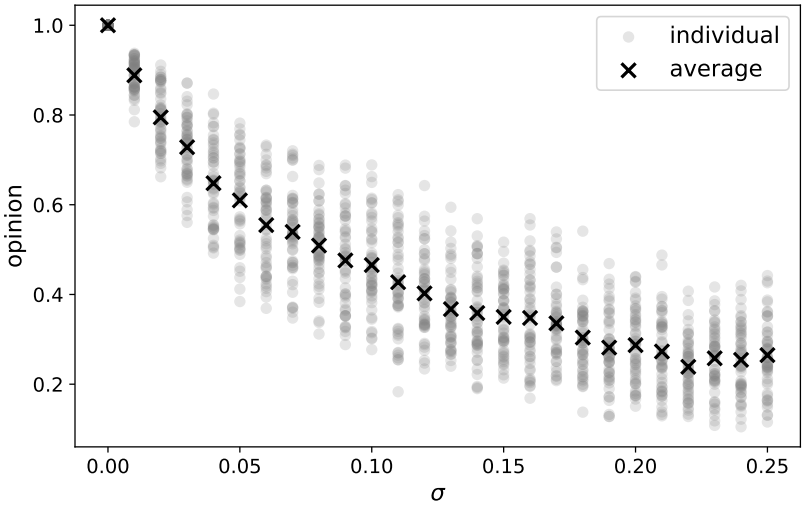


Figure 8.2: Final consensus opinions versus the education parameter σ. For each value of σ, gray circles and dark **x** indicate the value each individual run converged to and the average final opinion over 50 runs, respectively. The final opinion decreases toward the truth with more skepticism education (larger values of σ).

eigenvalue gap in the absence of a mitigation strategy, and the orange line shows the average eigenvalue gap when long-term education is employed. The 95% confidence intervals for both are shown as shaded regions. The attentive education mitigation strategy was applied to the same 50 networks examined in the absence of a mitigation strategy, and thus, both lines share random fluctuations. When no mitigation is applied, the eigenvalue gap is fairly constant with respect to α. Attentive education reduced the eigenvalue gap slightly as α increases, but not substantially.

The results for the skepticism and attentive education strategies we obtained using a DeGroot model are consistent with what we found when studying such strategies using the binary agreement model. As we found using that model, reducing people's commitment to disinformation can largely counteract the stubborn's ability to sway individuals' opinions. However, solely biasing people towards the truth is not sufficient to overcome the influence of stubborns.

Content Moderation

In our model of disinformation spread, we first create a Barabási-Albert graph with 100 agents, selecting 10 of these agents to be committed to opinion A, leaving the remainder uncommitted to opinion B. The edges of this graph are then transformed into two directed edges in both directions, weighted using social transitivity as described in the main document. We create three copies of this graph for distinct analyses. In the first copy, we randomly

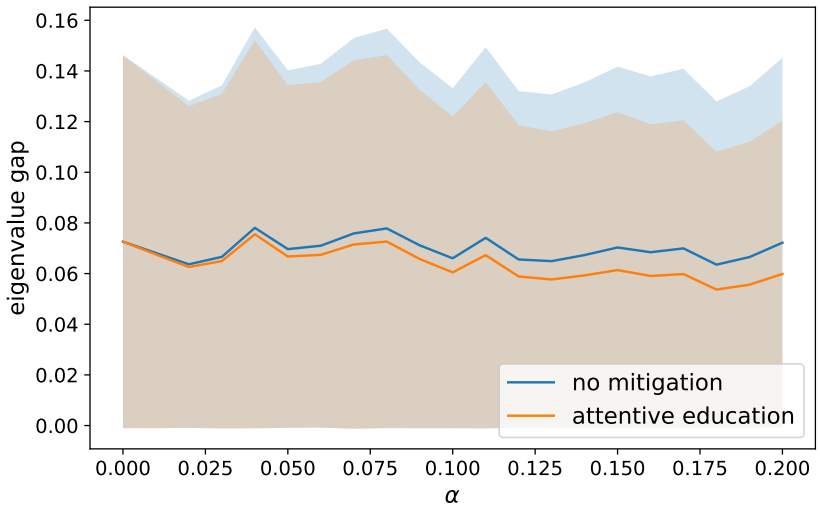


Figure 8.3: Eigenvalue gap versus the education parameter α. The blue line shows the eigenvalue gap when no education is employed, while the orange line shows the eigenvalue gap for increasing amounts of attentive education. The eigenvalue gap reduces slightly as α increases meaning the rate that the model converges to the disinformation is slightly slowed.

select n committed nodes to remove, where n varies from 0 to 9. In the second copy, we systematically remove n committed agents based on their eigenvector centrality. To do this, we compute eigenvector centrality, remove the node with the highest centrality, recompute the centrality, and repeat this process until n agents have been removed. The third copy remains unaltered, serving as a control. For each of these three graphs, we generate a trust matrix by normalizing the adjacency matrix and then calculate the eigenvalues of the trust matrices. We also compute the difference in the two largest eigenvalues, ignoring any repeated values (e.g., if the eigenvalues are 1, 1, 0.75, then the eigenvalue gap is 0.25). This entire process is repeated 50 times. For each value of n, ranging from 0 to 9, we average the 50 runs and plot this average as a solid line, also calculating a 95% confidence interval to create a band around the solid lines. This methodology allows us to explore the effects of content moderation through both random and strategic removal of influential nodes on the dynamic behavior of the system, as characterized by the eigenvalue gap. Removing individuals from the graph alters the dominant eigenvector and the eigenvalue spectrum of $\mathbf{T}$, thereby shifting control away from the removed agents. However, as discussed previously, if we do not remove all of the stubborns, then the population opinion will eventually converge to the disinformation opinion. As before, we can then examine the eigenvalue gap to determine whether the rate of convergence changes.

In Fig. 8.4, we show the eigenvalue gap versus the number of committed individuals

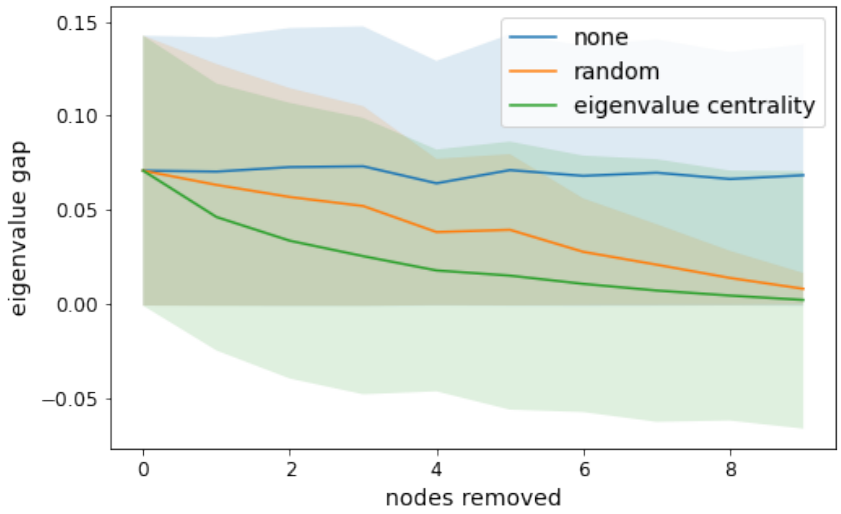


Figure 8.4: Eigenvalue gap versus number of committed agents removed. The eigenvalue gap versus removing agents randomly is shown in orange, and for removing agents based on eigenvalue centrality in green. As a baseline we show the eigenvalue gap when not removing agents in blue. Solid lines are the average of 50 simulations, while shaded regions are 95% confidence intervals. Removing agents with either strategy reduces the eigenvalue gap, however, a targeted approach based on eigenvalue centrality reduces the eigenvalue gap a a faster rate.

removed from the model. Results for removing individuals randomly are shown with a solid orange line, and results for removing individuals based on their level of influence are shown with a solid green line. We also show results for removing no individuals with a solid blue line, for comparison. The results shown with all three solid lines were obtained using the same graphs and are averages over 50 simulations. The shaded regions indicate 95% confidence intervals. When no individuals are removed, the eigenvalue gap is approximately 0.07. When stubborns are removed from the graph with either approach, the eigenvalue gap tends towards zero. However, when stubborns are removed based on eigenvalue centrality, the eigenvalue gap converges towards zero faster than when stubborns are removed randomly. Therefore, in this model, a targeted approach to removing stubborns slows the spread of disinformation more than does removing individuals randomly.

Counter Campaigns

Counter campaigns are modeled by introducing another group of stubborns who are committed to the truth. The social networks we consider contain 3 subgroups of agents: 1) agents spreading disinformation (n_A nodes), 2) agents running a counter campaign for truth (n_B nodes), and 3) regular agents comprising the remaining nodes. We model the committed

agents as "stubborn" in the DeGroot model by setting their self-weights to 1. This ensures that they do not update their opinions over time. The initial opinion vectors for the groups are $x_i(0) = 1$ if you are committed to A, and 0 otherwise.

In Fig. 8.5 we show many examples of counter-campaigns of various sizes. As in our previous simulations, we began with a Barabási-Albert graph with 100 agents, selecting 10 of these agents to be committed to opinion A, and n_B agents to be committed to B, and the remaining to be uncommitted to B. The edges of this graph are then transformed into two directed edges in both directions, weighted using social transitivity. Each subplot of Fig. 8.5 shows results for a values of n_B between 0 and 10, i.e., for a different size of counter-campaign. In each subplot, each individual's opinion versus the number of iterations is shown as a grey line. The committed agents' opinions can be seen as horizontal lines with values 0 and 1. Uncommitted individuals' opinions begin at zero and grow quickly to values between 0 and 1. As discussed previously, when all of the stubborns are committed to disinformation, all individuals eventually become committed to the disinformation opinion, as can be seen in the upper left subplot. However, as the size of the counter-campaign grows, uncommitted agents' opinions converge to opinions that are closer to the truth. When $n_B = n_A$, uncommitted agents' opinions converge to values near 0.5.

Conclusion

Understanding and combating the spread of disinformation in social networks is a critical challenge. Through the lens of the DeGroot model, we have examined three potential interventions and their impacts on the dynamics of belief formation. Our results highlight the importance of network topology and the distribution of trust in shaping these dynamics. In comparison with the more complete model in the main document, the DeGroot model allows for extremely fast exploration of parameter space, mainly related to the cost of diagonalizing T.

The results we obtained using the DeGroot model resemble the results of the binary agreement model in the main document. Namely, a skepticism education strategy was by far the most effective way to combat disinformation, while an attentive one had little to no affect. In our content moderation policies, we again saw if we do not remove all of the committed agents, eventually the committed agents will convince everyone of their opinion. However, in the DeGroot model our targeted approach performed better than our random approach, however, they both performed similarly. Counter-campaigns seemed to have a stronger effect in the DeGroot model compared to in the binary agreement model. However the initial positions of the counter-campaign agents were placed randomly when exploring the DeGroot model, which possibly gives those agents a further reach than those conisdered in the binary agreement model.

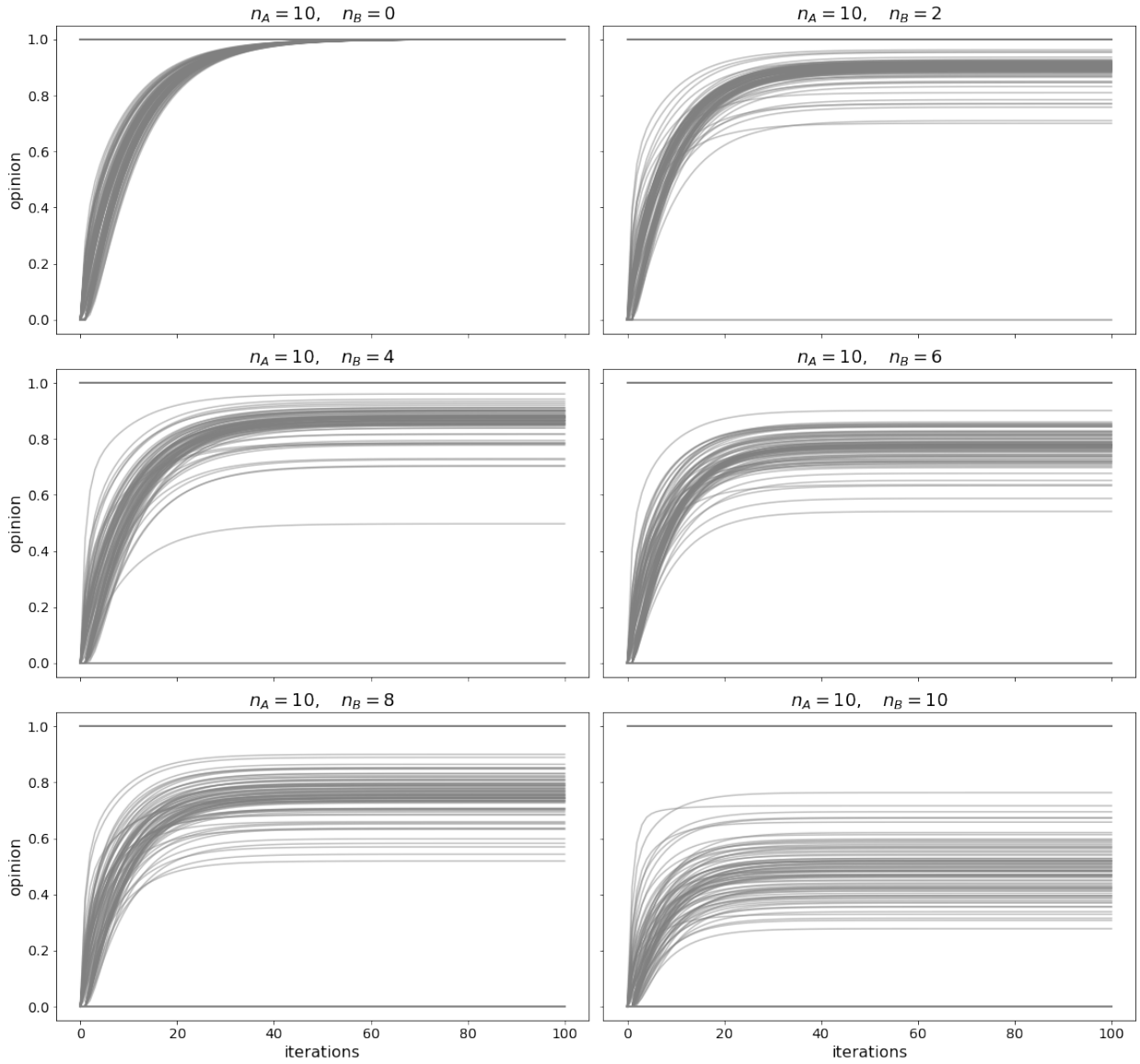


Figure 8.5: Agents opinions versus time in the presence of counter-campaigns. Agent's opinions versus the number of iterations of the DeGroot model are shown in grey for various sizes of counter-campaigns. For each size of counter-campaign (n_B), the size of the minority committed to the disinformation is held constant ($n_A = 10$). As the counter-campaign grows, uncommitted agents converge to opinions closer to the truth. When the counter-campaign is equal in size to the minority committed to disinformation, uncommitted agents converge to opinions near .5.

The DeGroot model allowed us to quickly explore some of our mitigation strategies, but its simplicity limits the applicability of results derived from using it. Utilizing the binary agreement model in the main document allowed us to incorporate more complex attributes, such as nonlinearities. Nonetheless, further research is needed to develop more sophisticated models and strategies for combating disinformation in social networks.

Content Moderation Figures

This section contains all of the results for content moderation applied to various artificial social networks. Each plot corresponds to the graph type listed in the caption. Results for removing nodes based on their level of influence are shown with solid lines, and those for removing nodes randomly are shown with dashed lines. The colors of the lines represent the percentage of the total nodes that were removed. Each opaque line shows the average of 5 simulations, and each transparent line shows results for a single simulation.

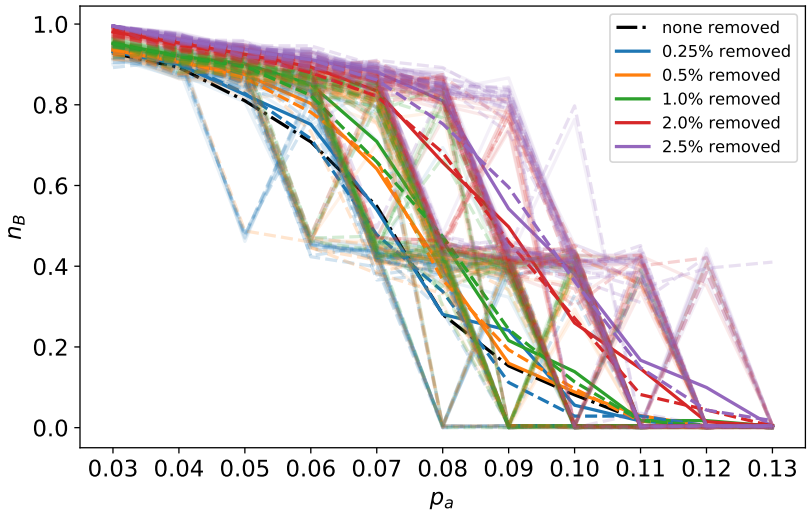


Figure 8.6: Barbell graph.

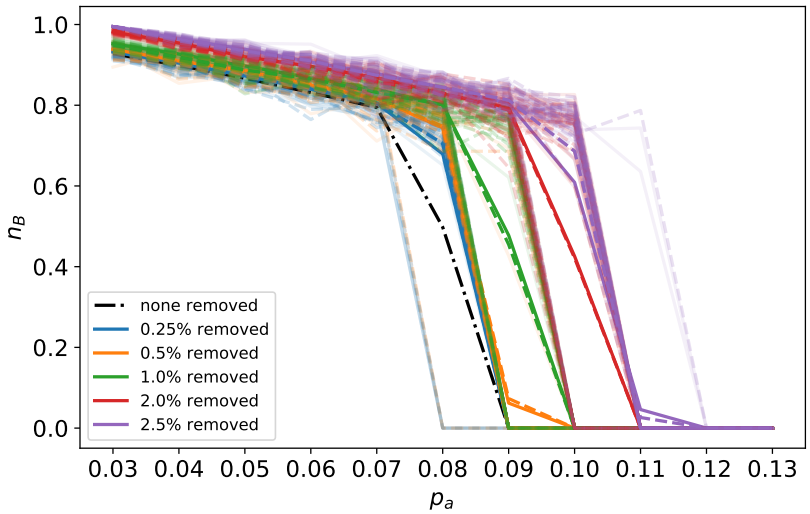


Figure 8.7: Complete graph.

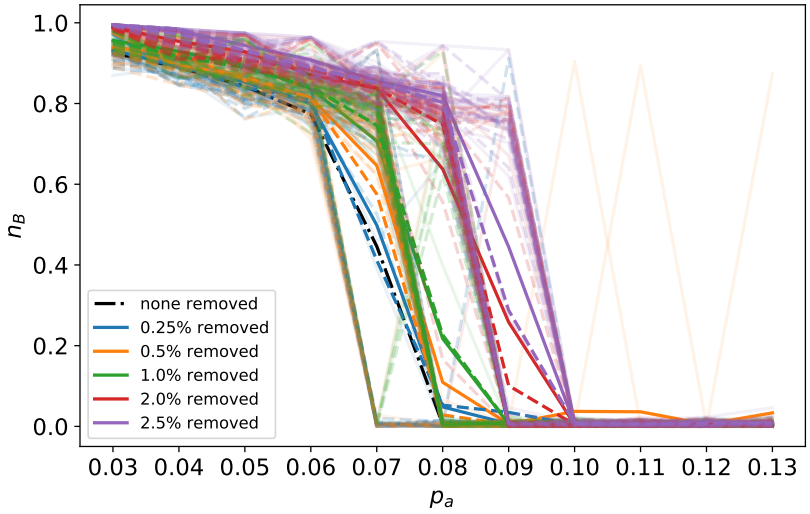


Figure 8.8: Erdos-Renyi graph with $p = .02$.

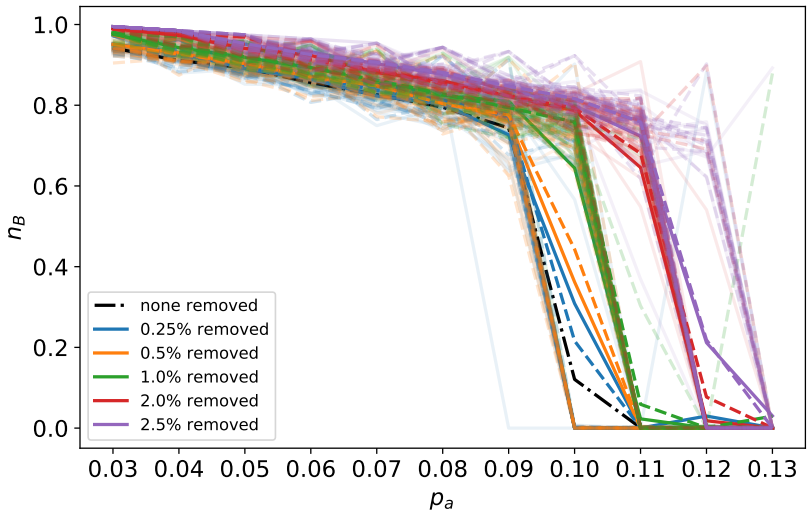


Figure 8.9: Erdos-Renyi graph with $p = .12$.

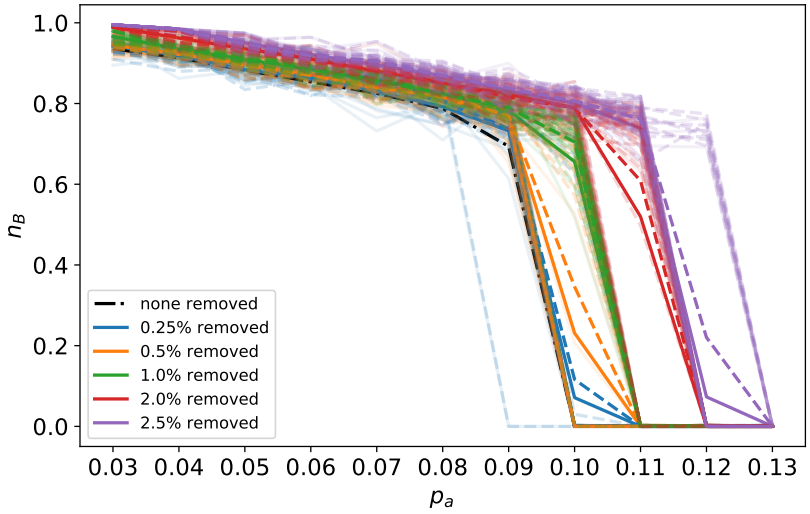


Figure 8.10: Erdos-Renyi graph with $p = .25$.

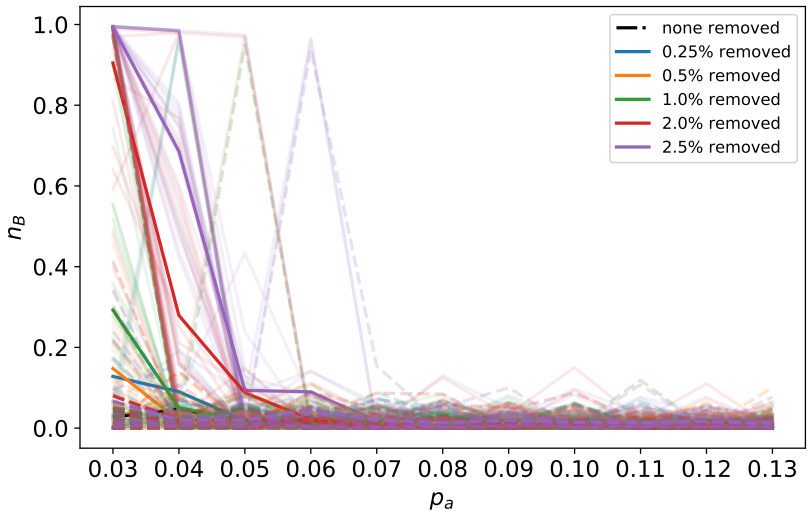


Figure 8.11: Grid graph.

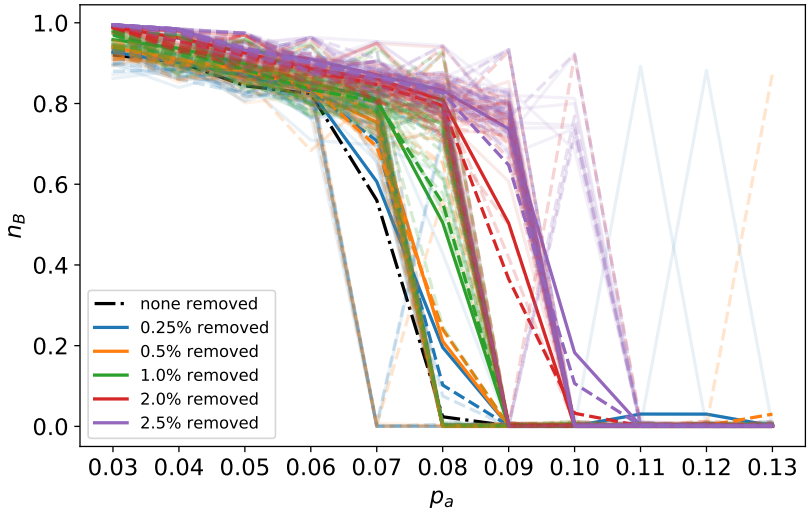


Figure 8.12: Watts-Strogatz random graph with $k = 8$ and $p = 1$.

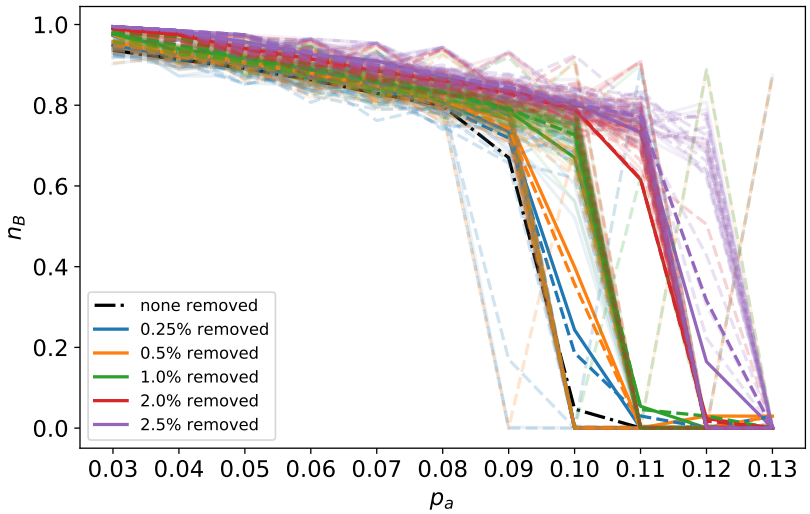


Figure 8.13: Watts-Strogatz random graph with $p = 1$ and $k = 48$.

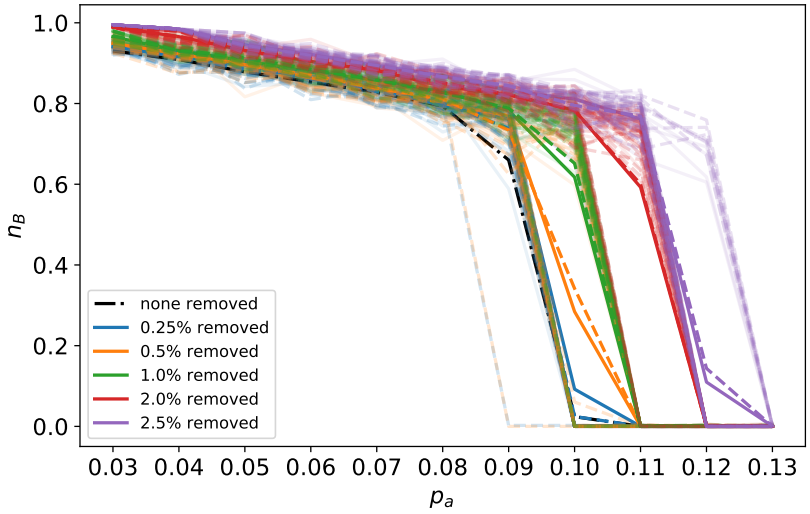


Figure 8.14: Watts-Strogatz small world graph with $p = 1$ and $k = 100$.

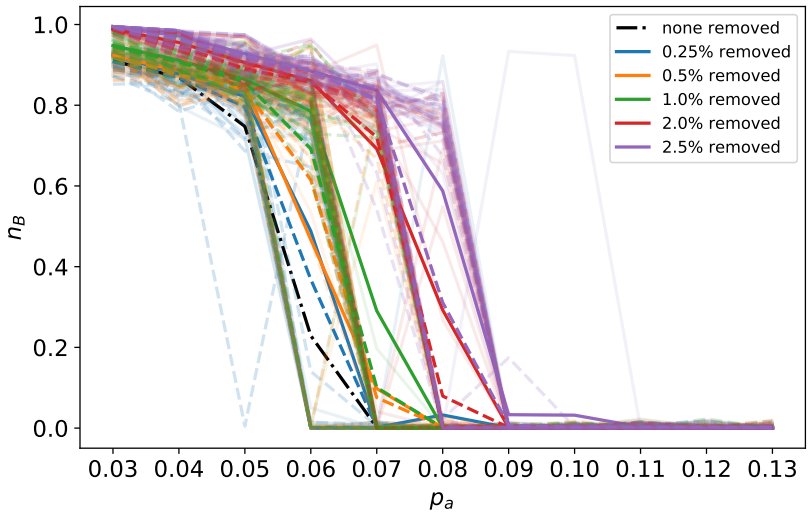


Figure 8.15: Watts-Strogatz small world graph with $p = .5$ and $k = 8$.

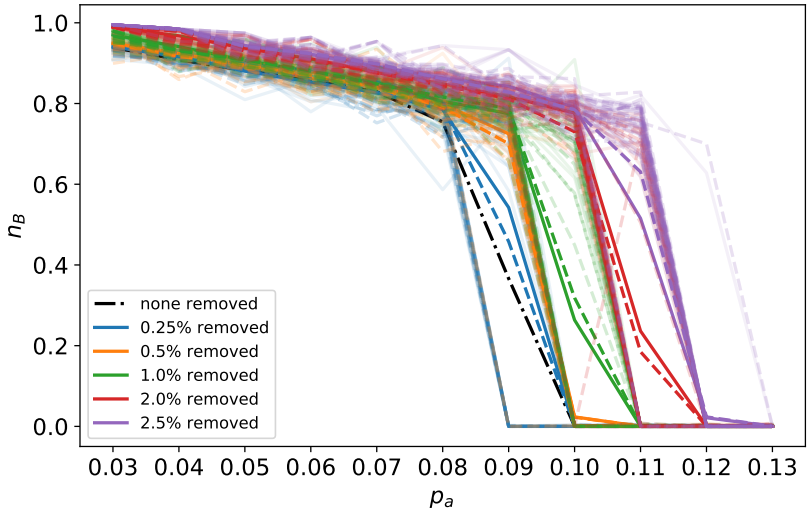


Figure 8.16: Watts-Strogatz small world graph with $p = .5$ and $k = 48$.

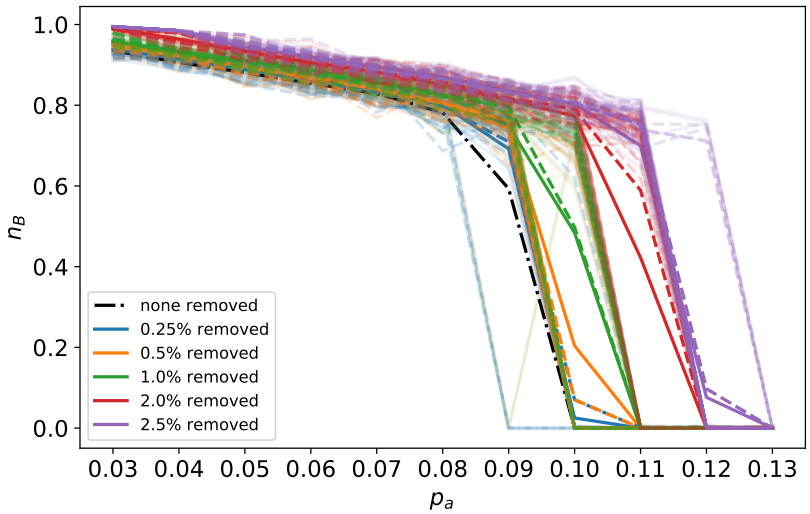


Figure 8.17: Watts-Strogatz small world graph with $p = .5$ and $k = 100$.

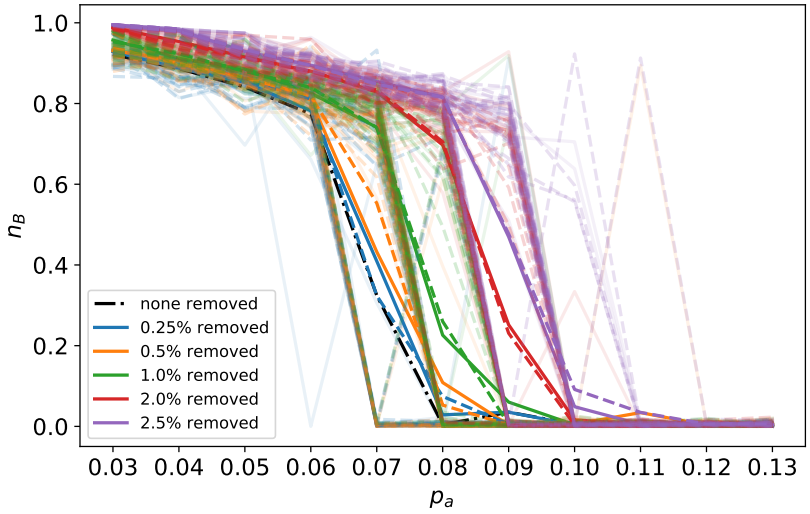


Figure 8.18: Barabasi-Albert graph with $m = 4$.

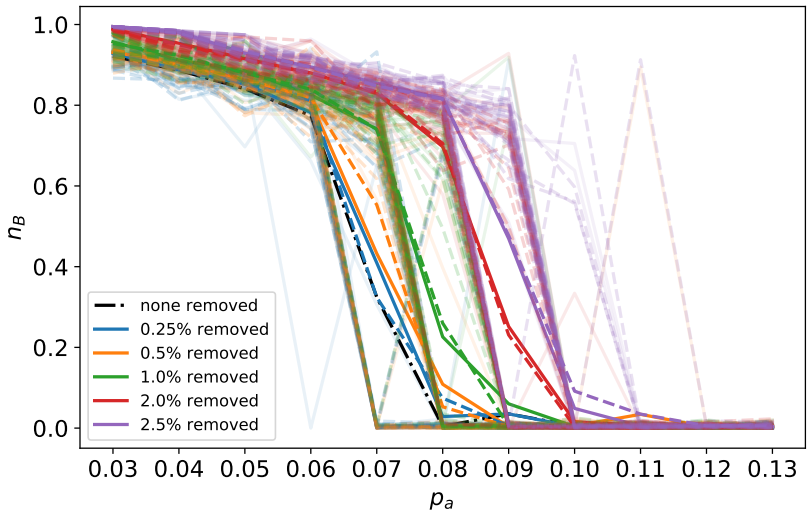


Figure 8.19: Barabasi-Albert graph with $m = 4$.

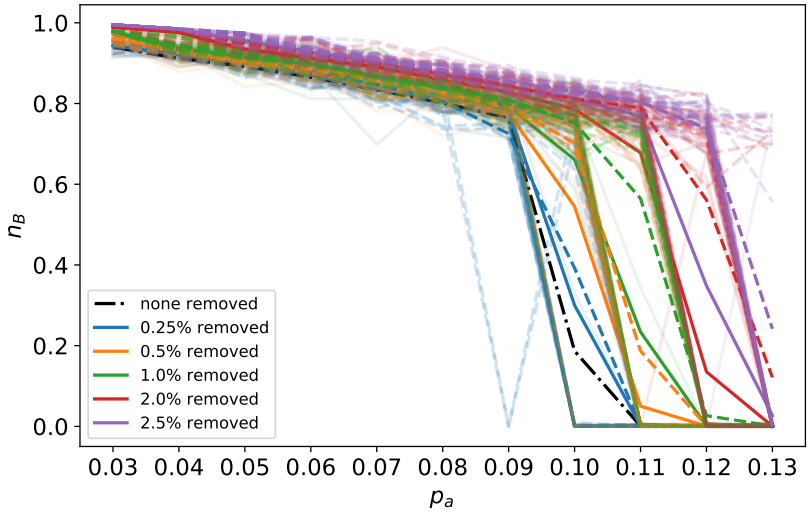


Figure 8.20: Barabasi-Albert graph with $m = 24$.

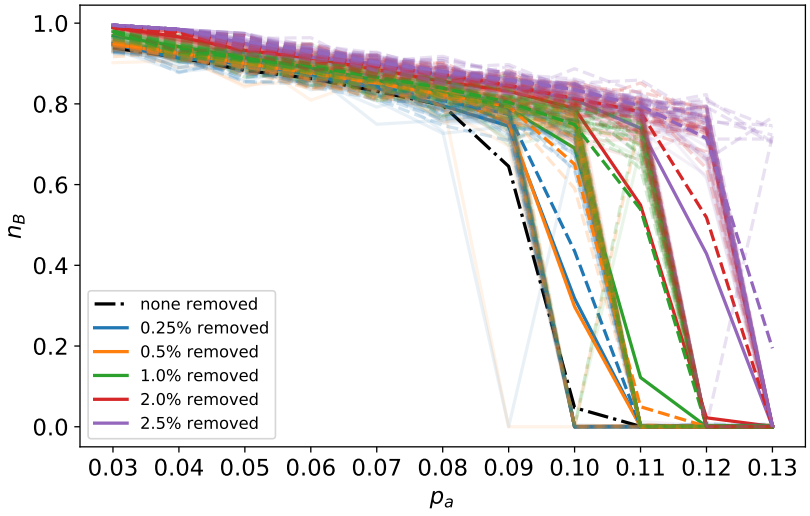


Figure 8.21: Barabasi-Albert graph with $m = 50$.

Education Figures

This section contains all of the results for applying early and late education to various artificial social networks. Each plot corresponds to the graph type listed in the caption. We show the effects of early education with an orange dashed line, late education with a green dotted line, a combination of both with a solid blue line, and no education with a black dot-dashed line. Each opaque line shows the average of 30 runs, and transparent lines show results for individual runs.

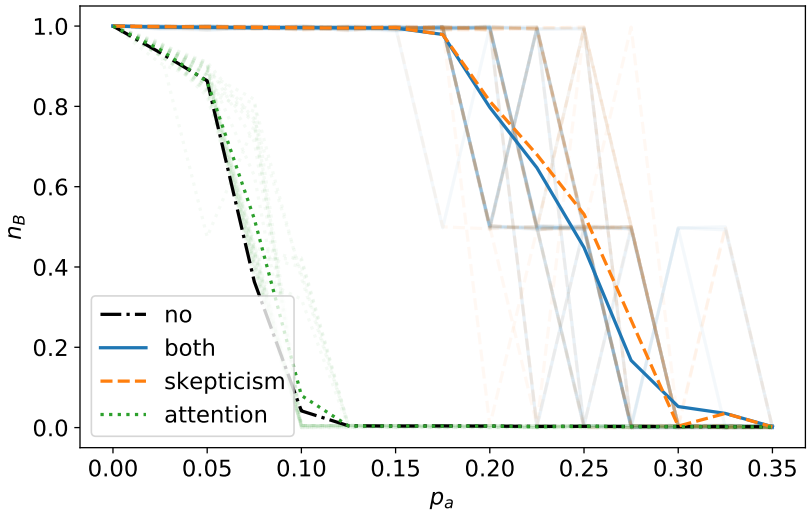


Figure 8.22: Barbell graph.

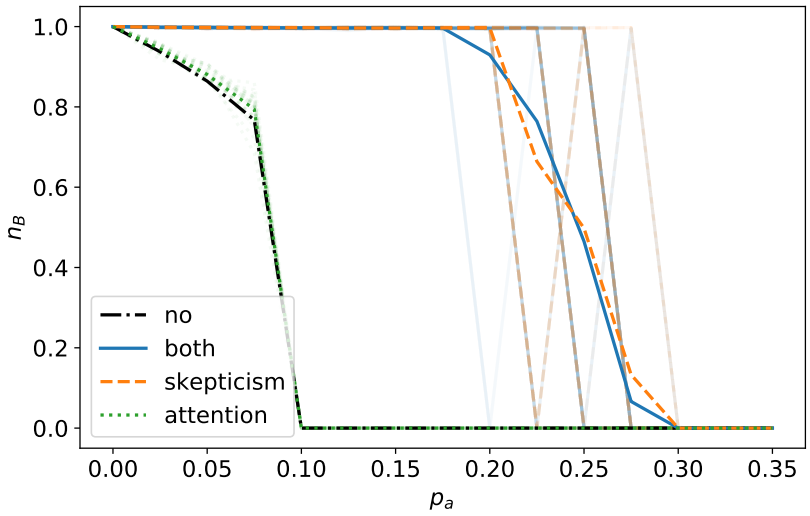


Figure 8.23: Complete graph.

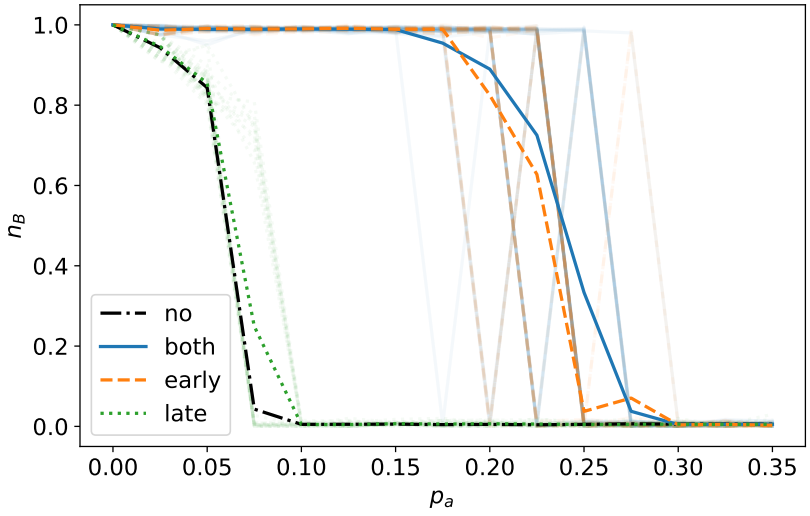


Figure 8.24: Erdos-Renyi graph with $p = .02$.

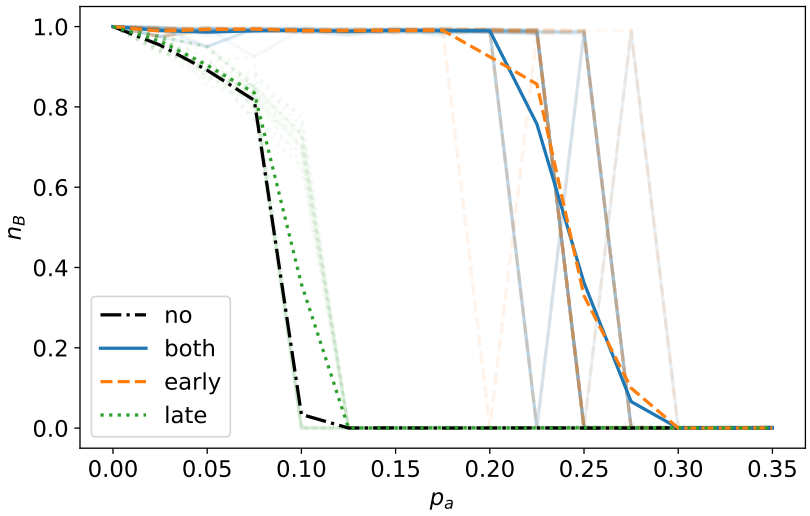


Figure 8.25: Erdos-Renyi graph with $p = .12$.

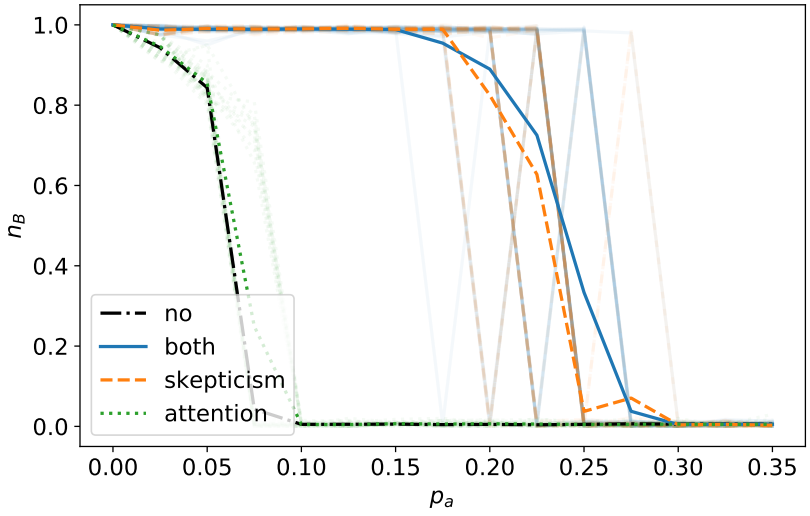


Figure 8.26: Erdos-Renyi graph with $p = .25$.

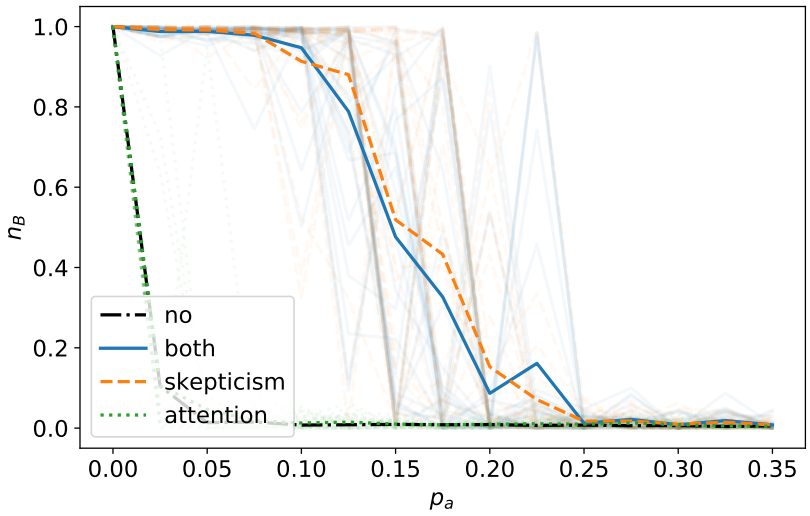


Figure 8.27: Grid graph.

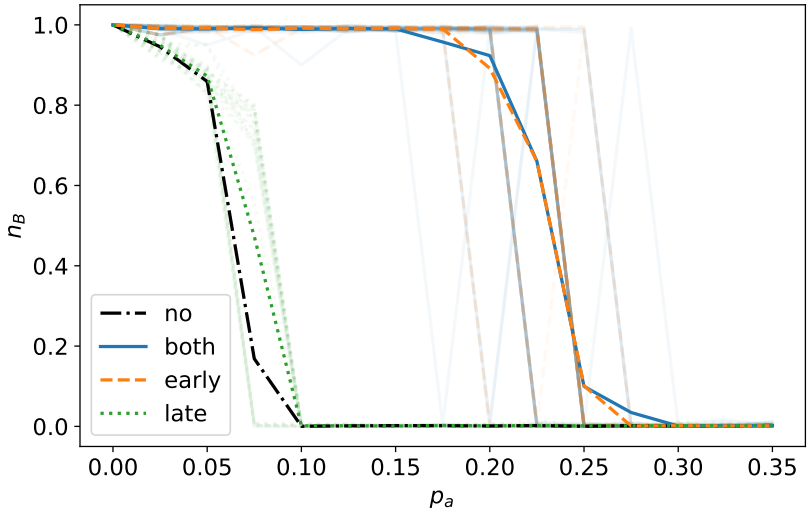


Figure 8.28: Watts-Strogatz random graph with $p = 1$ and $k = 8$.

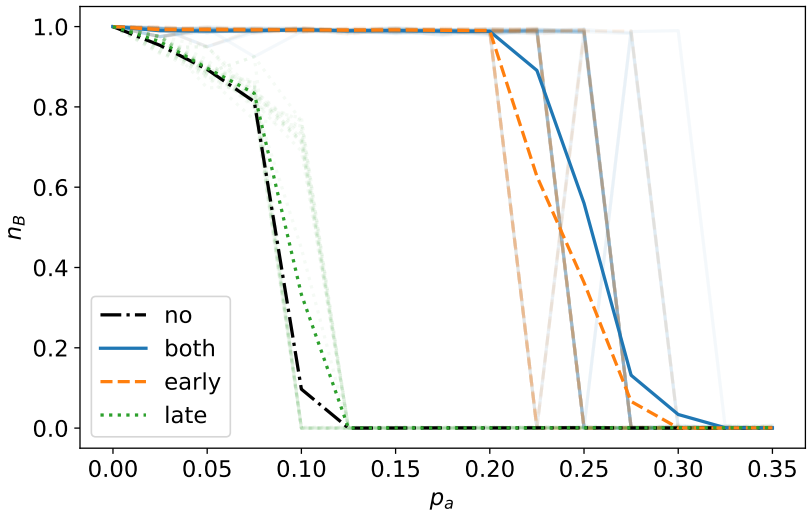


Figure 8.29: Watts-Strogatz random graph with $p = 1$ and $k = 48$.

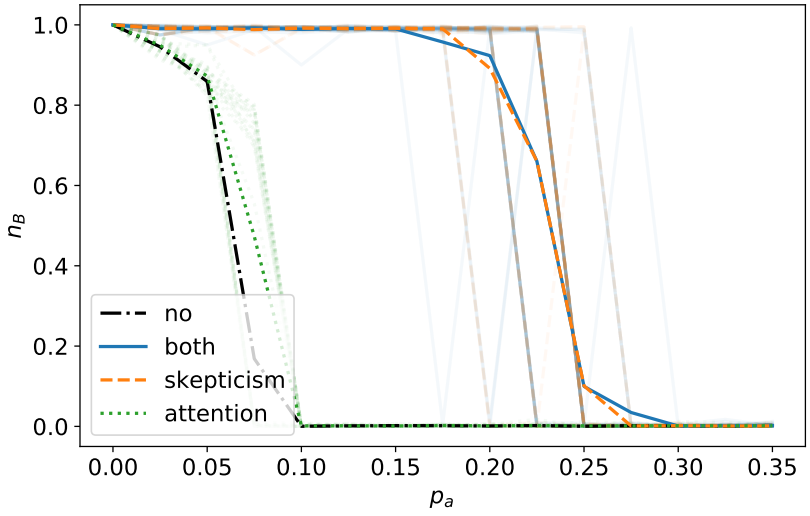


Figure 8.30: WSR25.

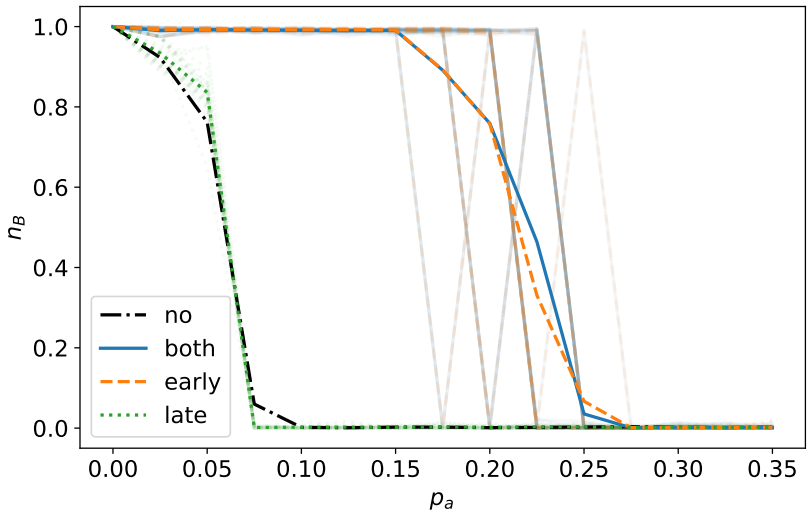


Figure 8.31: Watts-Strogatz small world graph with $p = .5$ and $k = 8$.

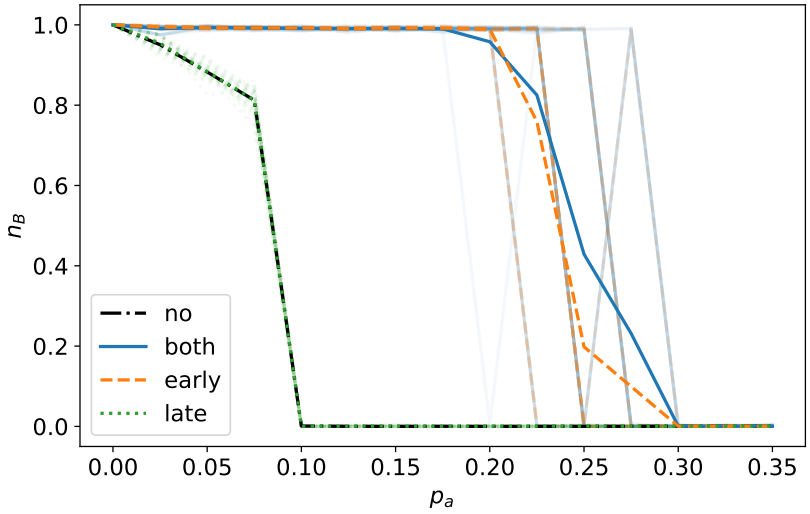


Figure 8.32: Watts-Strogatz small world graph with $p = .5$ and $k = 48$.

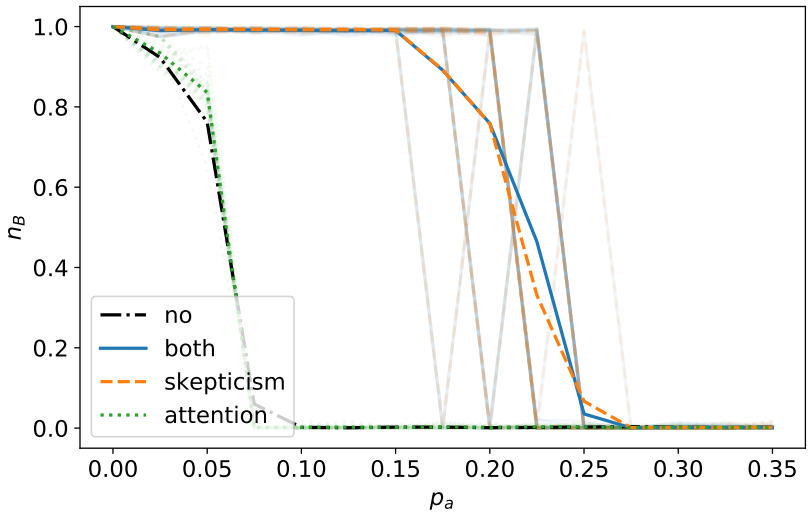


Figure 8.33: Watts-Strogatz small world graph with $p = .5$ and $k = 100$.

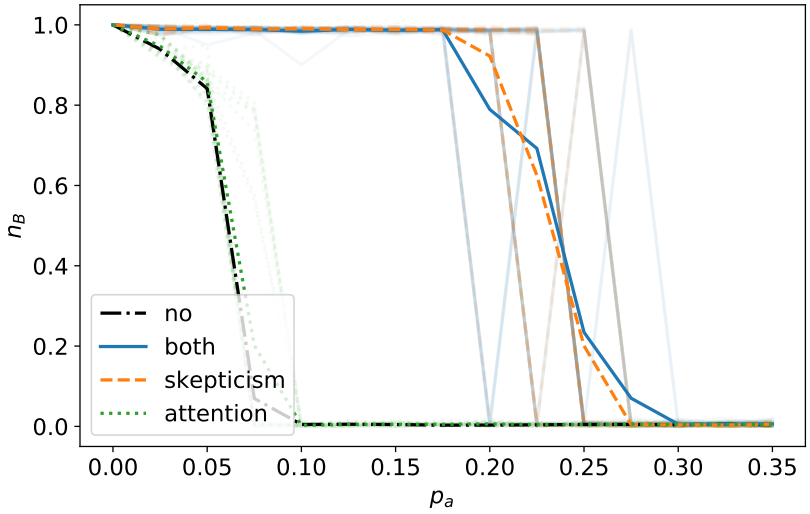


Figure 8.34: Barabasi-Albert graph with $m = 4$.

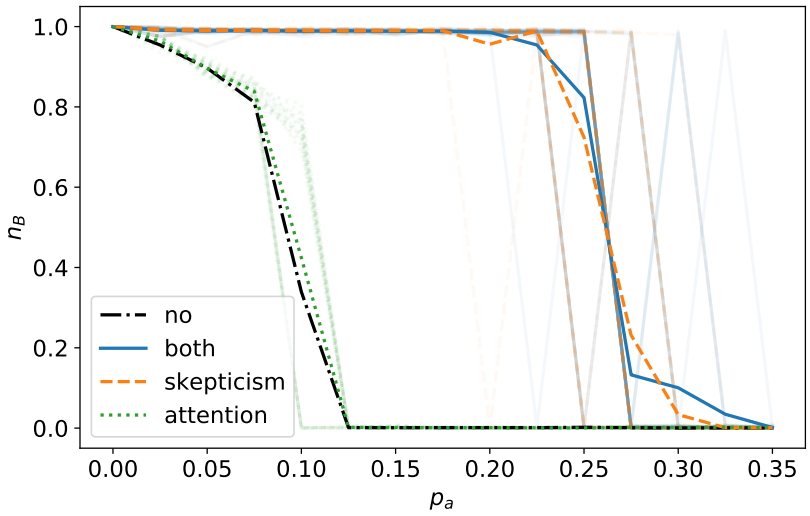


Figure 8.35: Barabasi-Albert graph with $m = 24$.

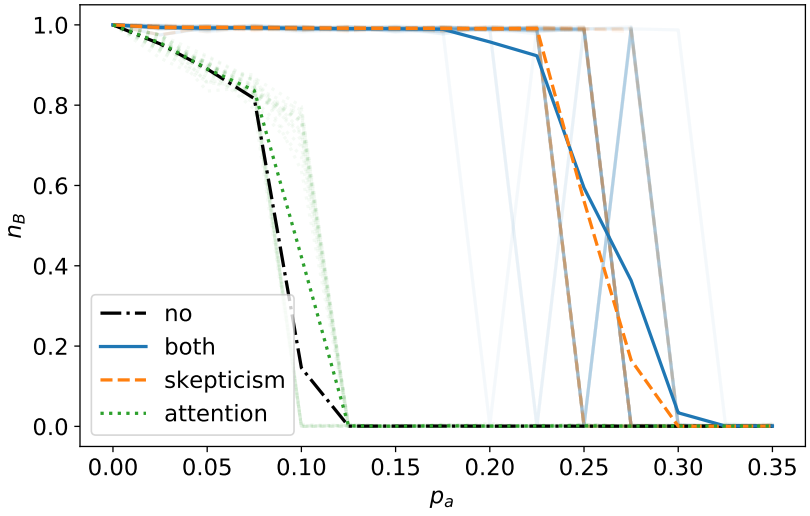


Figure 8.36: Barabasi-Albert graph with $m = 50$.

Counter-Campaign Figures

This section contains all of the results for applying counter-campaigns to various artificial social networks. Each plot corresponds to the graph type listed in the caption. As before, we show the results for no intervention with a black dot-dashed line. We show the effects of a small counter-campaign ($p_b = .05$) with a solid blue line and of a large counter-campaign ($p_b = .15$) with a dashed orange line. Each opaque line shows the average of 10 simulations, and transparent lines show the results for individual runs.

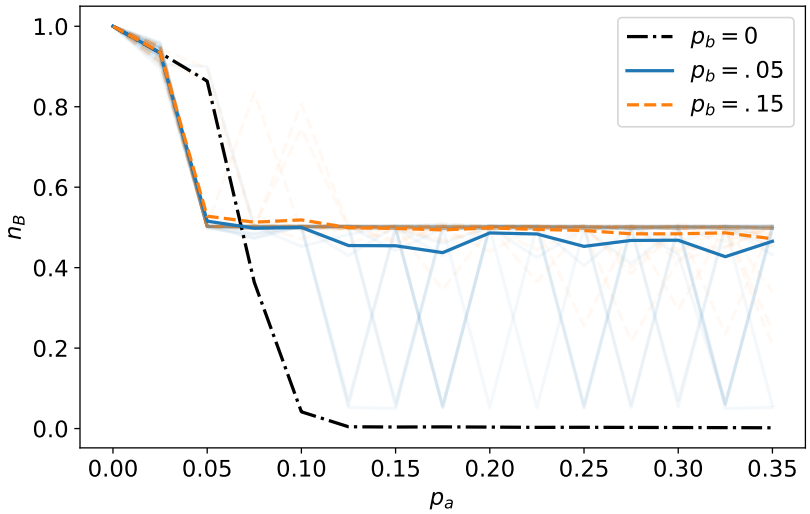


Figure 8.37: Barbell graph.

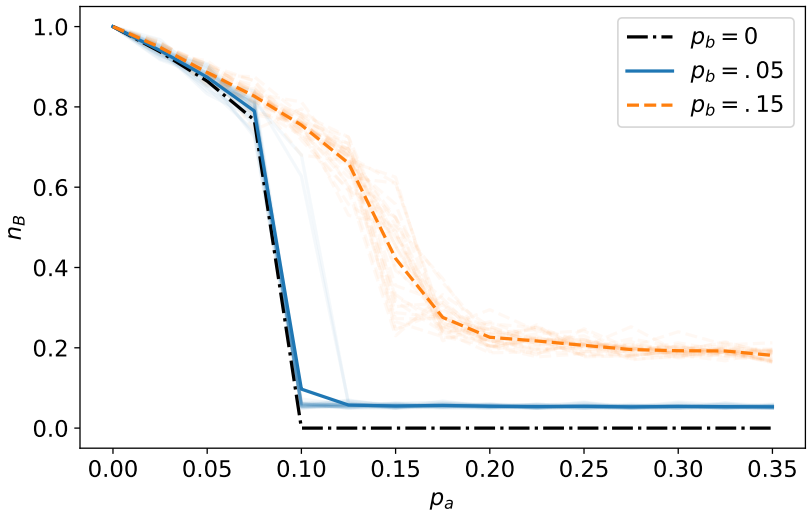


Figure 8.38: Complete graph.

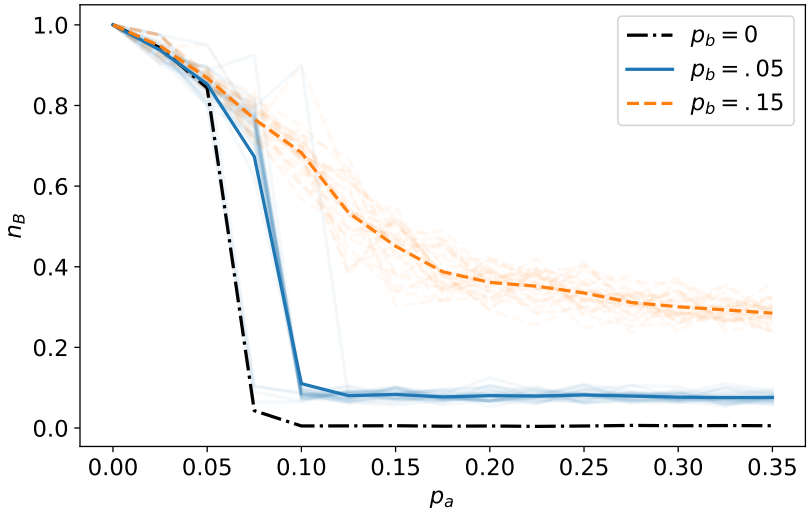


Figure 8.39: Erdos-Renyi graph with $p = .02$.

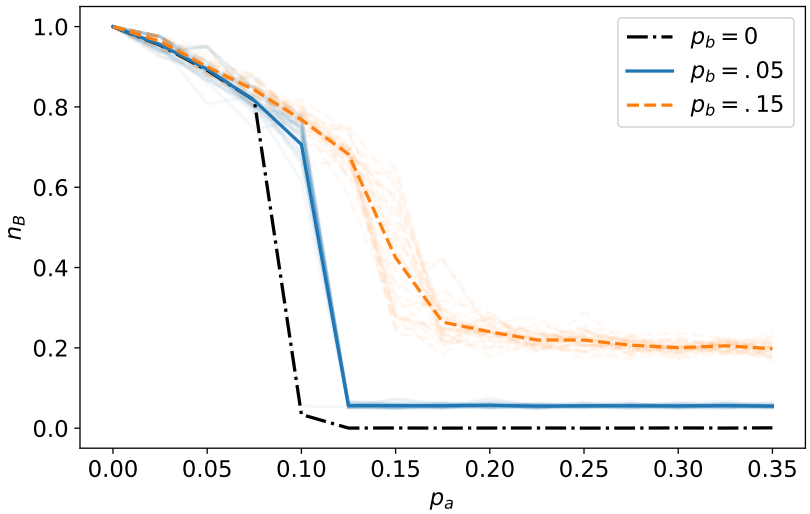


Figure 8.40: Erdos-Renyi graph with $p = .12$.

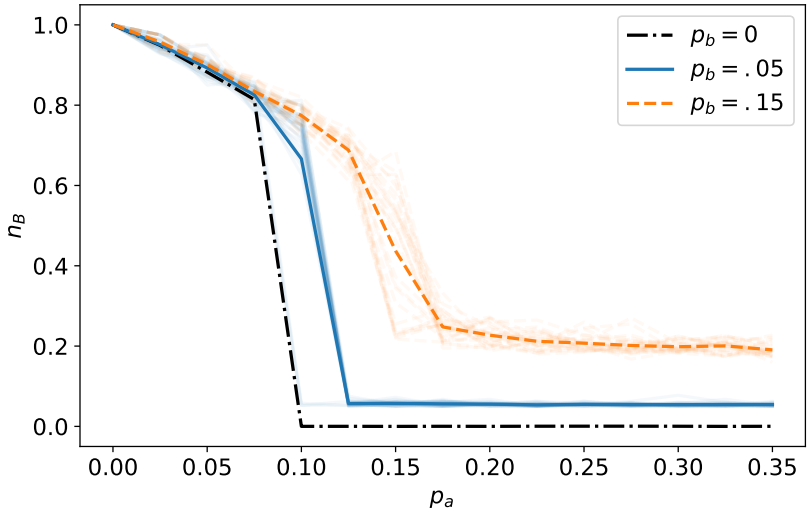


Figure 8.41: Erdos-Renyi graph with $p = .25$.

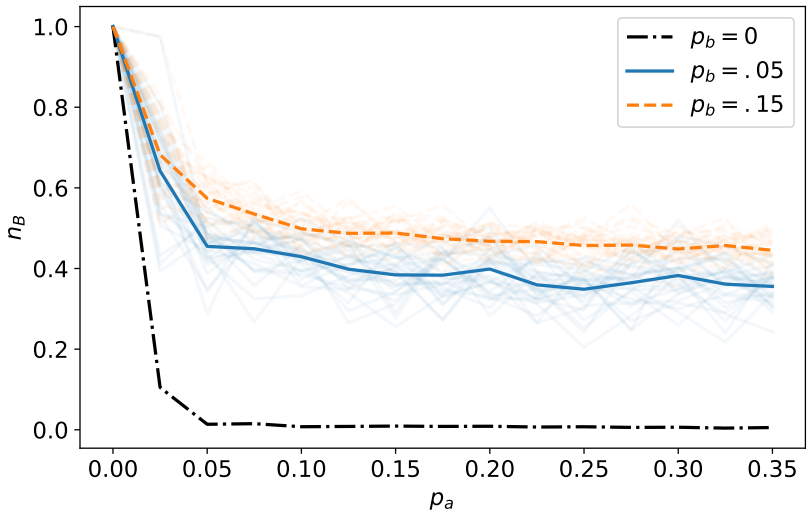


Figure 8.42: Grid graph.

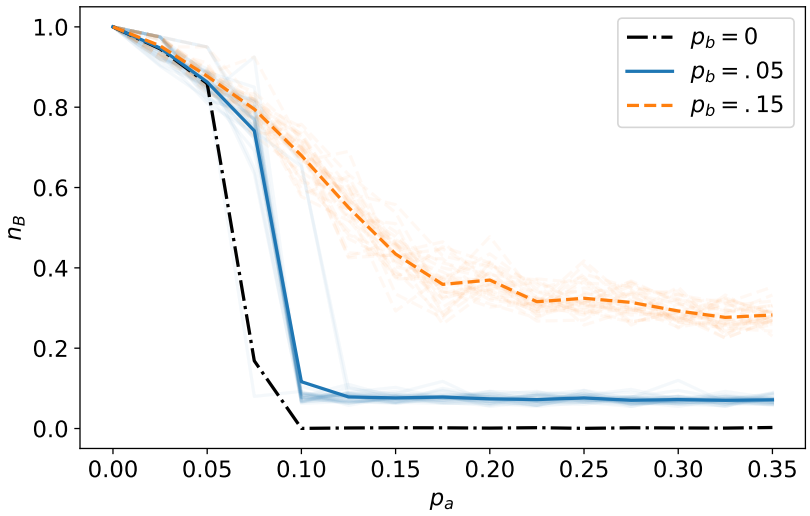


Figure 8.43: Watts-Strogatz random graph with $p = 1$ and $k = 8$.

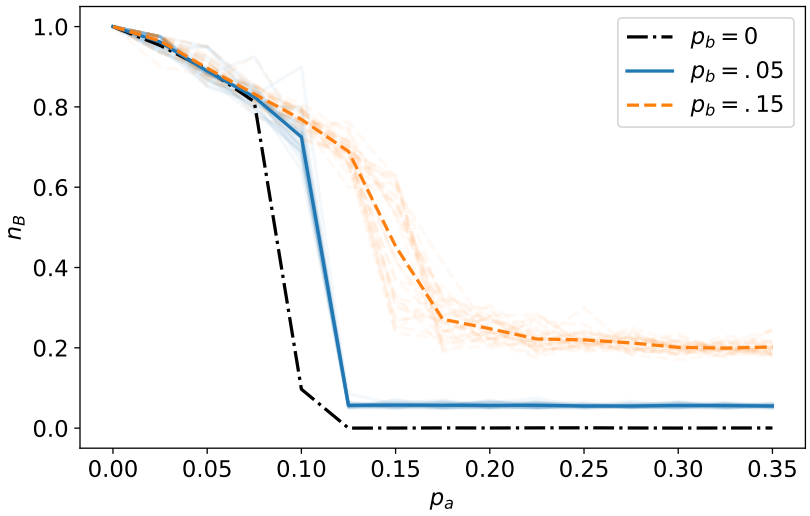


Figure 8.44: Watts-Strogatz random graph with $p = 1$ and $k = 48$.

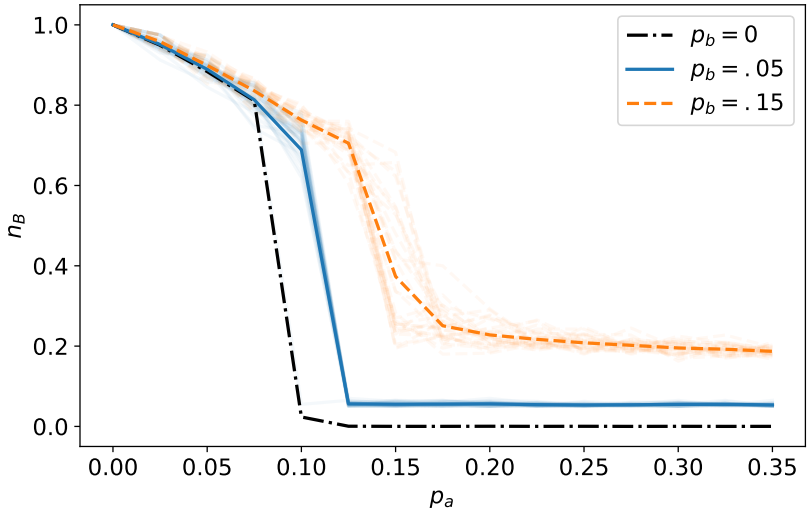


Figure 8.45: WSR25.

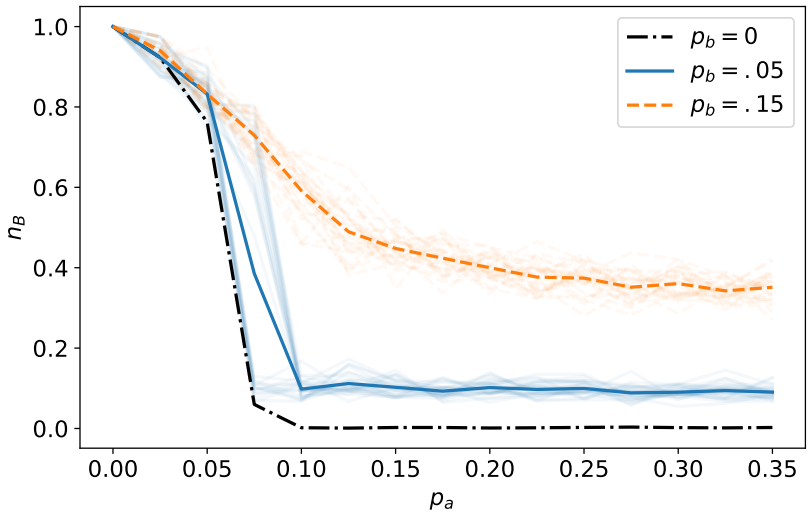


Figure 8.46: Watts-Strogatz small world graph with $p = .5$ and $k = 8$.

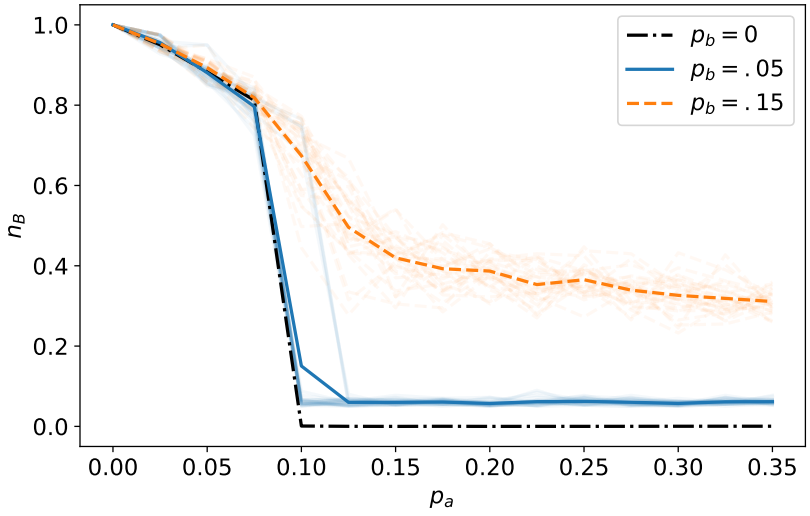


Figure 8.47: Watts-Strogatz small world graph with $p = .5$ and $k = 48$.

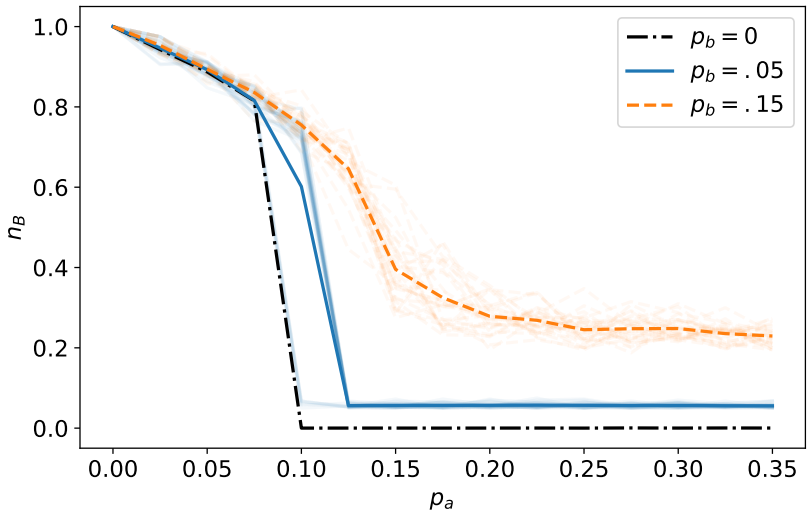


Figure 8.48: Watts-Strogatz small world graph with $p = .5$ and $k = 100$.

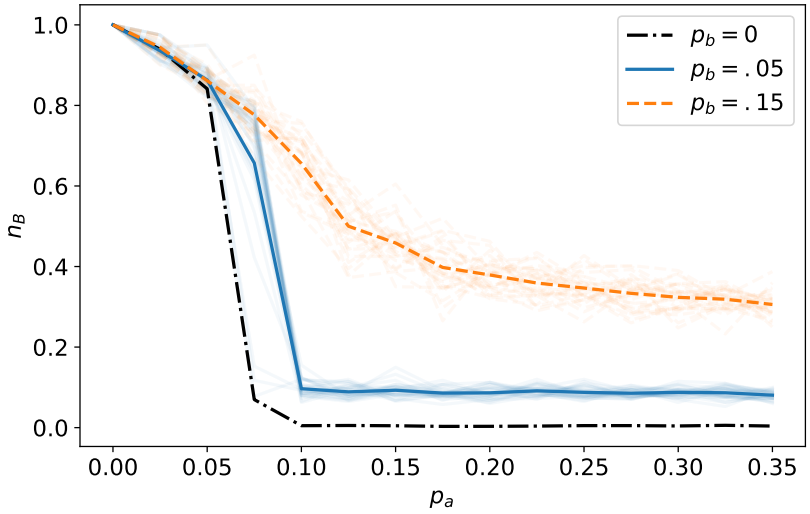


Figure 8.49: Barabasi-Albert graph with $m = 4$.

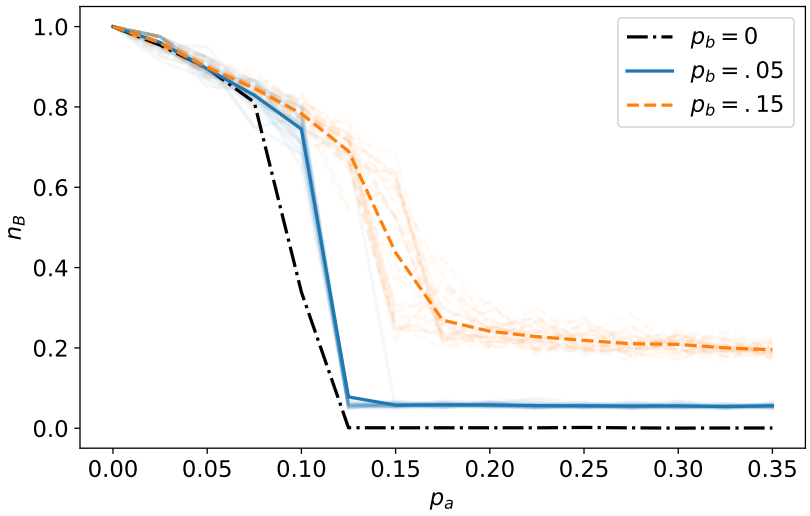


Figure 8.50: Barabasi-Albert graph with $m = 24$.

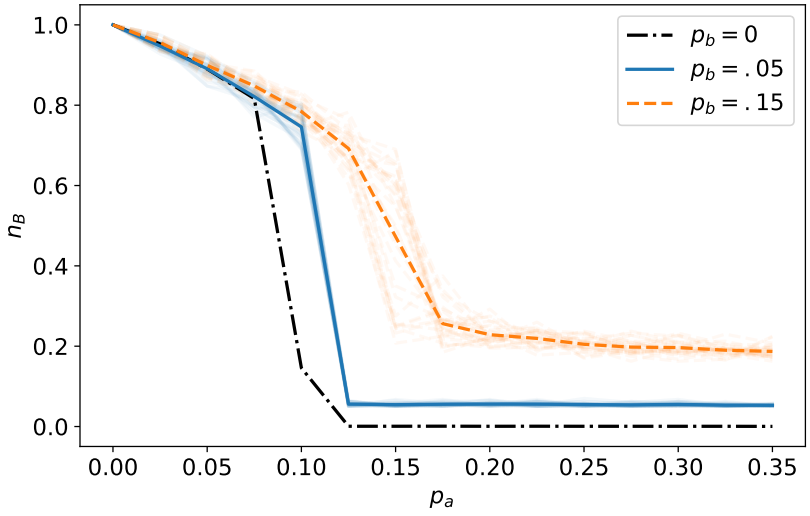


Figure 8.51: Barabasi-Albert graph with $m = 50$.