2019

Evolutionary Model Discovery: Automating Causal Inference for Generative Models of Human Social Behavior

Chathika Gunaratne
University of Central Florida

EVOLUTIONARY MODEL DISCOVERY: AUTOMATING CAUSAL INFERENCE FOR GENERATIVE MODELS OF HUMAN SOCIAL BEHAVIOR

by

CHATHIKA GUNARATNE
B.Sc. University of Colombo, 2012
M.S. University of Central Florida, 2016

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the School of Modeling, Simulation, and Training
in the College of Engineering & Computer Science
at the University of Central Florida
Orlando, Florida

Fall Term
2019

Major Professor: Ivan Garibay

ABSTRACT

The desire to understand the causes of complex societal phenomena is fundamental to the social sciences. Society, at a macro-scale has many measurable characteristics in the form of statistical distributions and aggregate measures; data which is increasingly abundant with the proliferation of online social media, mobile devices, and the internet of things. However, the decision-making processes and limits of the individuals who interact to generate these statistical patterns are often difficult to unravel. Furthermore, multiple causal factors often interact to determine the outcome of a particular behavior. Quantifying the importance of these causal factors and their interactions, which make up a particular decision-making process, towards a societal outcome of interest helps extract explanations that provide a deeper understanding of social behavior.

Holistic, generative modeling techniques, in particular agent-based modeling, are able to 'grow' artificial societies that replicate emergent patterns seen in the real world. Driving the autonomous agents of these models are rules, generalized hypotheses of human behavior, which upon validation against real-world data, help assemble theories of human behavior. Yet often, multiple hypothetical causal factors can be suggested for the construction of these rules. With traditional agent-based modeling, it is often up to the modeler's discretion to decide which combination of factors best represent the rule at hand. Yet, due to the aforementioned lack of insight, the modeled agent rule is often one out of a vast space of possible rules.

In this dissertation, I introduce Evolutionary Model Discovery, a novel framework for automated causal inference, which treats such artificial societies as sandboxes for rule discovery and causal factor importance evaluation. Evolutionary Model Discovery consists of two major phases. Firstly, a rule of interest of a given agent-based model is genetically programmed with combinations of hypothesized factors, attempting to find rules which enable the agent-based model to more closely

mimic real-world phenomena. Secondly, the data produced through genetic programming, regarding the correspondence of factor presence in the rule to fitness, is used to train a random forest regressor for importance evaluation. Besides its scientific contributions, this work has also led to the contribution of two Python open-source software libraries for high performance computing with NetLogo, Evolutionary Model Discovery and NL4Py.

The results of applying Evolutionary Model Discovery for the causal inference of three very different cases of human social behavior are discussed, revisiting the rules underlying two widely studied models in the literature, the Artificial Anasazi and Schelling's Segregation, and an ensemble model of diffusion of information and information overload. First, previously unconsidered factors driving the socio-agricultural behavior of an ancient Pueblo society are discovered, assisting in the construction of a more robust and accurate version of the Artificial Anasazi model. Second, factors that contribute to the coexistence of mixed patterns of segregation and integration are discovered on a recent extension of Schelling's Segregation model. Finally, causal factors important to the prioritization of social media notifications under loss of attention due to information overload are discovered on an ensemble of a model of Extended Working Memory and the Multi-Action Cascade Model of conversation.

ACKNOWLEDGMENTS

The research presented in this dissertation would not have been possible without the funding, data, and through-provoking, DARPA-hard challenge events, provided throughout the DARPA Social-Sim program (DARPA program HR001117S0018 (FA8650-18-C-7823) and the endless computational resources provided through AWS research grants.

It goes without saying that much of this work was done under the close supervision of my PhD advisor and committee chair Dr. Ivan Garibay, who throughout the past years provided me with all the support and guidance I needed to grow into independent researcher. Whether be it a though-provoking discussion over results or the extra-funding needed to attend a summer school for extra skills, he has always supported me with whatever means possible, for which I am indebted. I would like to thank the rest of my committee Dr. Gita Sukthankar, Dr. Stephen Fiore, and Dr. Paul Wiegand for their guidance and advice throughout the dissertation process.

Much of this work features collaborative research conducted with experts in the areas of agent-based modeling and computational social sciences with whom I have had the great honor to work with. I have had the pleasure of working with Prof. Joshua Epstein of New York University, upon who's foundational contributions on 'Generative Social Sciences' and 'Inverse Generative Social Sciences' I have based Evolutionary Model Discovery, and has provided valuable advice throughout. Chapter 3 of this dissertation builds on the work of Dr. Erez Hatna of New York University, who provided his expertise and model of residential segregation for which I am grateful. I am ever grateful to Dr. William Rand of North Carolina State University with whom I have had the great pleasure of working closely on the development of the models described in Chapter 5, lending me his expertise and guiding me towards my career goals.

I would like to thank the folks at the Complex Adaptive Systems Lab whom I have been fortunate

to work alongside. Over the years I have seen the lab grow from three of us to over 30 associated individuals. I was fortunate and very grateful to have a wonderful mentor early on in my PhD career, Dr. Ilhan Akbas previously our Research Associate, who never hesitated to make room in his impossibly busy schedule for those many coffees over which I would dump on him my graduate student woes and he would mentor me with his experience. A big thank you to my fellow PhD students Chathura Jayalath, Nisha Baral, and Chathurani Senevirathna for sharing the many sleepless nights over which we developed the Multi-Action Cascade Model. I am grateful for all the data and compute infrastructure support by Brandon Barnes, and from Ezequiel Gioia of the Office of Research Commercialization. Also, many thanks to Dr. Asli Soyler Akbas for giving me the opportunity to experience how agent-based modeling is put to practice in the industry, during my internship at Universal Studios, Orlando.

Mostly, I am ever grateful to my parents and my sister back in Colombo for their patience and understanding, while I have been in Florida pursuing my dreams. My father for being my inspiration as a child, for surrounding me with science, programming, and philosophy at a very young age. Finally, I am dearly grateful for the unconditional love and support my partner, Praimmika Choopromkeaw has shown me throughout the past few years, patiently supporting me through the many hardships of being a international graduate student.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

API	Application Programming Interface
ABM	Agent-Based Model
EMD	Evolutionary Model Discovery
EWM	Extended Working Memory
GP	Genetic Programming
GUI	Graphical User Interface
JDK	Java Development Kit
JVM	Java Virtual Machine
ODD	Overview, Design, and Details
ODD+D	Overview, Design, and Details plus Decisions
NL4Py	NetLogo for Python
POM	Pattern-Oriented Modeling
MACM	Multi-Action Cascade Model
RMSE	Root Mean Squared Error
UML	Unified Modeling Language

CHAPTER 1: INTRODUCTION

What socio-agricultural factors might have lead to the sudden demise of a flourishing ancient civilization? Why do communities of individuals with no racial bias still maintain pockets of segregation? What drives compulsive information sharing by highly active social media users? Such are the questions that the computational social science strives to answer. Understanding the causes of social phenomena at the level of the individual is crucial for policy decisions. Such insights provide the ability to design interventions to ensure desirable societal outcomes and mitigate undesirable ones. Simulation models that are driven by these generating factors can be used simulate intervention strategies and inform of the expected outcomes in advance minus the cost and time required to observe and test them in the real-world.

However, providing explanations for complex social phenomena is not a trivial task as the decision-making processes of the individuals generating the society-scale phenomena is not explicitly observable. Often, gathering data on individual-level motivations through surveys is tedious, prone to sampling biases, in cases of large-scale phenomena, such as those that occur over online social media, quite difficult to sample. Instead, data observed as population-level outcomes of society, such as community sizes of an ancient civilization measured through archaeological excavations, racial diversity in an urban community, or the distribution of responsiveness to social media notifications, are typically more reliable sources of information. Such sources of data have motivated a series of successes in deep learning and artificial intelligence that focus on prediction. Yet these algorithms lack the ability to provide human-interpretable explanations of the causes of these phenomena.

Society is a complex adaptive system. It consists of autonomous individuals that interact and adapt to the actions of one-another, self-organizing into groups, and producing emergent societal phenomena. As with any complex system, the emergent properties of society are a result of the

interactions of its individuals and cannot be studied through experimentation on human behavior in isolation.

Generative models of society, also known as artificial societies, encapsulate human-interpretable rules of behavior. Such models embrace the holistic view of complex adaptive systems and model social phenomena 'from the bottom up' [5]. Quoting Epstein and Axtell in [5], "... if you haven't grown it, then you haven't explained it...". As with any complex adaptive system, the trajectory of events that occur within a society are highly sensitive to the underlying rules with which the individuals act. This is due to the fact that small changes in behavior of the individual quickly compound into large macro-scale deviations due to the highly interactive nature of such systems. 'Top down' models that are learned off of statistical correlations in data are limited in their ability to follow such non-linear trajectories. In contrast, generative models grow societies along trajectories similar to those in the real-world, and are able to follow the non-linear dynamics that are produced as a result. Agent-based models are an example of such generative models.

Causal and Mechanistic Explanations through Agent-Based Models

The work presented in this dissertation is premised on the view that agent-based models of society are able to provide mechanistic explanations of human behavior, and that mechanistic explanations can be considered as causal explanations for the social sciences. While prediction has historically been a major goal of simulations, explanation has more recently seen increasing interest [6, 7, 8, 9, 10]. There has been debate as to whether prediction and explanation can be considered separate or are interdependent. Epstein considers explanation and prediction distinct; He provides supporting examples such as '*plate tectonics*' help explain earthquakes, but do not allow us to predict their '*time and place of occurrence*' [11]. This was in contrast to earlier views that explanation and prediction were of indistinguishable structure and view explanation merely as a retrodictive

prediction, as is described in the *symmetry thesis* by Hempel and Oppenheim [12]. Thompson and Derr argue that instead of being separable, explanation can lead to better prediction as more knowledge regarding the causes of the phenomena is unraveled [10]. Nonetheless, understanding what the type and scope of explanations agent-based models are able to provide is important.

Elsenbroich categorizes explanations in social science into four types: 1) Covering laws: explanations deducted from applying general laws and considering initial conditions, the result of classical deductive reasoning, 2) Causal explanations: a complete, step-by-step account of how a phenomena came about including all relevant facts [13], 3) Mechanistic explanations: a form of causal explanation, where the full story is not told, instead entities causing a macroscale phenomena and their relevant activities are provided as an explanatory account [14], and 4) Unifying explanations: attributing observations to a general theory that covers several such, similar phenomena [15]. Elsenbroich explains how agent-based models are able to provide mechanistic explanations. They oppose Grune-Yanoff's earlier claim that agent-based models cannot provide causal explanations as causal explanations require knowledge of all facts, and instead provide functional explanations; instead, they argue that full causal knowledge can never be achieved in complex social phenomena, and that explanations that consist of entities and their actions that lead to reproduction of aspects of macroscale patterns in data can be considered mechanistic, and thereby causal.

Agent-based models are not only mechanistic representations of the real-world, but the mechanisms they encode are human-interpretable. According to Machamer et al., mechanisms are entities and their activities that activate to lead a system from its initial conditions to a particular end state [14]. In agent-based models, agents represent these entities, while agent rules are specified to encode the activities of the entities that are hypothesized to generate the macroscale phenomena. These rules are typically specified as utility functions, threshold functions, or decision trees, and implemented in computer software. These implementations can easily be interpreted by a researcher, a comparative advantage when considering highly predictive deep-learning or machine

learning algorithms for example, where despite the accuracy and precision of prediction, often the neural networks fitted to patterns in the data cannot be used to provide a causal explanation of *why* the outcomes are as they are. I demonstrate the ease with which agent rules can represent human-interpretable causal narratives in Chapter 3.

Yet, I argue that agent-based models are rarely manipulated to their full explanatory potential. If, according to Elsenbroich and Epstein, the explanatory potential of ABM's lie in the specification of their rules, then ideally, the white-box manipulation of the causal factors constituting the rules would be crucial in the exploration of the space of alternate mechanisms for the most plausible mechanistic explanation. Yet, this is rarely done.

The true explanatory power of *agent-based models* (ABMs) lie in the flexibility and ease with which these behavior rules can be encapsulated within agents. An agent's rules can typically be reduced to a mathematical function of utility maximization (or cost minimization) on the possible decisions given the agent's state, memory, and sensory input. These functions define how the agent decides to act under different conditions. In the computational social sciences, it is common for these functions to be organized to represent social drives, or factors of the human mental process(es). Often, agents are driven by multiple such rules governing the execution of multiple behavior choices. In such cases, agent behavior rules are organized into sub-models of agent behavior.

These rules may be parameterized, allowing for some flexibility in behavior. For example, parameters could fix weights in the behavior rule functions, controlling the level of impact of a certain human decision making factor on the behavior selection of the agent. When these parameters are provided as a distribution, the model developer is able to simulate a heterogeneous population of agents in terms of the level of impact of decision making factors. Calibration of these parameters is then performed either through a grid search of the parameter space [3] or more powerful

optimization algorithms [16, 17].

However, the underlying behavior rule is not affected through this type of parameter variation. By modeling a different agent behavior rule, the simulation output can potentially be vastly altered. Critics have claimed this to be a fallacy in agent-based modeling, stating that an Agent-Based Model merely embodies the model developer's or model development team's ideology of the human rationale [18]. Instead of exploring various behavior rules representing different human mental processes, the model developer would assume a behavior rule set and attempt to calibrate their model to find the best parameter set that allows them to replicate the desired macro-patterns to the lowest error possible when compare to real world data. In other words, researchers often find themselves modeling one out of a large space of possible hypotheses of the underlying decision-making process and then calibrating this rule until it fits real-world data. Yet, no comparison to alternative hypotheses is made, so there is no way to falsify the behavior hypothesized by the researcher in the first place.

Pattern-oriented modeling (POM) [19], was developed to address this issue. In POM, Grimm et al, recommend that multiple instances of a single ABM be produced and compared on their ability to replicate patterns seen in the real-world. Each instance would encapsulate a competing possible explanation of the underlying generative behavior. Then by identifying the models that are able to most closely simulate patterns seen in real-world data, the researcher is able to compare and identify explanations that most likely describe the mechanism that may be at work in the actual decision-making process. However, POM as originally described is manual and limited due to several reasons, which I will discuss below.

Why is Modeling Multiple Theories of Agent Behavior Difficult?

Creating multiple ABMs embodying alternate, plausible explanations as agent behavior rules without automation is difficult and infeasible due to the following reasons.

- **Repeated manual implementation leads to increased probability for programmatic error.** Modeling multiple hypotheses as specified by POM requires the re-implementation of multiple versions of the same Agent-Based Model embodying alternate hypotheses of human behavior. This increases the chance of introducing programmatic bugs in the code. It also increases the complexity of the total codebase. If a bug is introduced into an initial version of the Agent-Based Model this bug will have propagated to all the alternate ABMs and will have to be corrected throughout the entire collection. The entire collection must then be retested and validated. If parameters were calibrated, calibration, which is usually an expensive computation in itself, must be redone. This results in the management of a huge codebase and the requirement of a large team of developers to maintain the project as it evolves.

- **It is difficult to manually maintain a structure of comparable rule components representing factors of human decision making, across many versions of the same agent-based model.** A theory of human social behavior can be represented as the combination of factors driving the human decision making process. A clear modular representation of these factors allows for easy cross-analysis of the many hypotheses being tested to asses factor impact on the macro-properties being simulated. Manually maintaining this regular agent cognition structure while developing multiple versions of them same agent-based model quickly becomes a difficult task.

- **Limitations to model implementation resources (programmers in particular.) leads to**

modeling and testing of a limited or biased sample of all possible factor combinations. Manually developing each agent-based model associated to every combination of factors, is not practical. As shown in [8, 20] the factors driving human behavior can be arranged as a mathematical function of utility. This results in a vast number of possible configurations they can be arranged in, resulting in an equally vast number of hypotheses of human behavior to be modeled and tested. Therefore, model developers commonly resort to modeling a single combination of factors, or a limited set of combinations, as theories of human behavior.

- **Due to lack of domain expertise, researchers many models have a disconnect between theories of human social behavior and data.** Often, theories used to model human decision making within agents are backed with scientific knowledge from domain experts. Yet, it has been shown that often additional theories that consider further causal factors may exist [21, 22, 23, 20, 24, 25] that support the explanation of a macro-phenomena. In particular, according to Weisberg's interpretation of Levin's strategy of model building [25], it is the underlying common components of causality captured across each alternate theory that allows for each of them to produce the same solution. Therefore, human biases may govern the selection of theory choice when constructing the agent behavior rules. Despite many Agent-Based Modeling efforts for the computational social sciences usually being multi-disciplinary efforts with input from diverse teams of domain experts, there may be instances when the team lacks complete domain knowledge. In such a situation model developers could greatly benefit from a collection of human decision making factors.

EMD: Overcoming the Difficulties of 'Many Modeling' of Agent-Based Models to Exploit their Explanatory Power

The aim of this dissertation is to enhance explanatory potential of agent-based models by enabling the exploration of the vast space of possible functions of factors of human behavior. This is done through the genetic programming of agent rules, followed by factor importance evaluation through random forest regression. The modification of agent rules may lead to the emergence of patterns previously unseen, as demonstrated in the three case studies that follow. Therefore, an agent-based model embodying a combination of factors, that is able to replicate target patterns in data, provides a plausible explanation of the social phenomena under investigation.

Evolutionary Model Discovery answers the four issues described in Sec. 1 as follows. The use of genetic programming, automates the programming of multiple models, reducing the chances for programmatic error. Genetic programming works by choosing modules of model code from a primitive set. Each primitive encapsulates a factor of human behavior. The genetic program then evolves generations of syntax trees, combining multiple factors through operators. The function that is represented by a syntax tree calculates a utility (or cost) that the agent perceives they would incur if they take a particular decision. This utility (or cost) function will then determine which behavior the agent will choose to perform, dependent on its state, interactions, memory, and environment sensors. Further description of this process is given in Sec. 3.

The modularization of agent rules into hypothesized factors of human decision making based on theories of human behavior answers the second difficulty of maintaining a structure of comparable rule components. In Agent_Zero, Epstein derives, from neuro-cognitive foundations, that an agent cognitive architecture must consider emotional drives, ortho-rational drives, and social forces on information received by the agent through the individuals it received information from [26]. Similarly, Social/Psycho-Social theories have been used to drive agents towards replicating collective

social phenomena [8, 20]. Evolutionary Model Discovery evolves plausible candidate models, able to replicate desired patterns of collective social behavior. The presence of causal factors of these candidate models is then assessed on the ability the give the model to reproduce the phenomena of interest.

Evolutionary Model Discovery is provided as an open-source library implementation [27] to use with the NetLogo ABM software [28], answering third difficulty. This library discussed in Chapter 7, allows researchers to easily code factors in the NetLogo modeling language, and tag them as factors using annotated comments, which are automatically interpreted for the genetic program. Evolutionary Model Discovery then provides a Python API for the specification of model setup conditions, parameters of the genetic program, and the fitness function.

Finally, by allowing concepts from multiple theories to combine as factors, Evolutionary Model Discovery answers the fourth difficulty. Depending on the operators provided to the genetic program for factor manipulation, complex factor interactions can also be explored. Instead of training a structure that is not easily interpreted back to an explainable form, such as a neural network, Evolutionary Model Discovery works with chunks of human-interpretable concepts implemented as rules, promoting the use of theory in agent-based models towards mechanistic explanations.

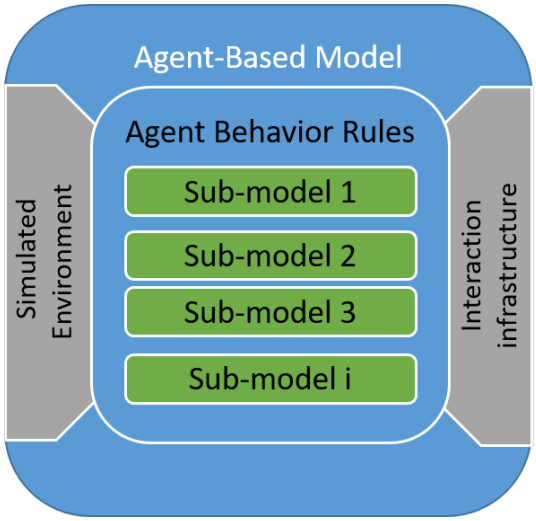


Figure 1.1: A conceptual view of the components of an agent-based model. Agent-based models consist of one or more sub-models that specify autonomous rules of behavior. The agents are allowed to interact with a simulated environment and provided an interaction infrastructure.

Statement of Contributions

Multiple scientific and engineering contributions are made in this dissertation. Primarily, Evolutionary Model Discovery provides a novel technique through which complex social systems can be given human-interpretable explanations in terms of their generating factors. This framework has shed light on the causal factors generating three very different cases of human social behavior:

1. Previously unconsidered factors driving the socio-agricultural behavior of an ancient ancestral Pueblo civilization are discovered, constructing a more robust and accurate version of the Artificial Anasazi model.

2. Factors leading to the coexistence of mixed patterns of segregation and integration are discovered on a recent extension of Schelling's Segregation model.

3. Factors determining the prioritization of social media notifications under loss of attention

due to information overload are discovered on an ensemble of a model of Extended Working Memory and the Multi-Action Cascade Model of conversation.

A major reason automated rule exploration has not been popular for the causal inference of agent-based modeling is due to the vast search space of possibilities. Theoretically, this search space may be infinitely large. Also, despite parameter calibration being a regular part of most agent-based modeling projects, there has been no standardized tool for rule exploration, let alone causal factor importance evaluation. This work presented in this dissertation has done the ground work for such a tool, through the EvolutionaryModelDiscovery Python package for factor importance analysis through parallelized genetic programming of NetLogo models. In order to achieve this goal, the following engineering contributions were made:

1. NL4Py, an open-source toolkit for the parallel execution of NetLogo models through Python was developed and release on the Python Package Index. https://github.com/chathika/NL4Py

2. EvolutionaryModelDiscovery, a implementation of the framework described in this dissertation, for the causal inference of NetLogo models through random forest importance evaluation of genetically programmed agent-based models with parallel computing. https://github.com/chathika/EvolutionaryModelDiscovery

NL4Py, in particular is currently in use by the community and has 7 major releases at the time of writing, with users actively contributing with issue posts and feature suggestions on the GitHub project page.

Statement of Originality

Parts of this work have been included in conference presentations and preprints under review for journal publications. Other than the work discussed in the following manuscripts, the rest of this dissertation has not been published publicly at the time of writing:

- Gunaratne, C., & Garibay, I. (2017, July). Alternate Social Theory Discovery using Genetic Programming: Towards Better Understanding the Artificial Anasazi. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (pp. 115-122). ACM.

- Gunaratne, C., Senevirathna, C., Jayalath, C., Baral, N., Rand, W., & Garibay, I. (2019) A Multi-Action Cascade Model of Conversation. In *5th International Conference on Computational Social Science.* Amsterdam, NL.

- Gunaratne, C., Garibay, I., & Dang, N. (2019). Evolutionary model discovery of causal factors behind the socio-agricultural behavior of the ancestral Pueblo. *arXiv preprint arXiv:1802.00435. In review: PLOS One.*

- Gunaratne, C., Baral, N., Rand, W., Garibay, I., Jayalath, C., & Senevirathna, C. (2019). A Theory of Extended Working Memory and its Role in Online Conversation Dynamics. *arXiv preprint arXiv:1910.09686. In review: Computational and Mathematical Organization Theory.*

- Baral, N., Gunaratne, C., Jayalath, C., Rand, W., Senevirathna, C., & Garibay, I. (2019) Negative Influence Gradients Lead to Lowered Attention Span on Social Networks. In *Conference of the Computational Social Science Society of the Americas (CSS).* Santa Fe, NM.

- Gunaratne, C., & Garibay, I. (2019). NL4Py: Agent-Based Modeling in Python with Parallelizable NetLogo Workspaces. arXiv preprint arXiv:1808.03292. In review: *Simulation*

Modelling Practice and Theory.

- Gunaratne, C., Rand, W., & Garibay, I. (2020). Inferring Mechanistic Explanations of Response Prioritization on Social Media under Information Overload. Submitted to the *19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020)*. Auckland, NZ.

CHAPTER 2: LITERATURE REVIEW

The field of Computational Social Sciences has benefited heavily from Agent-Based Modeling as a tool allowing social scientists and policy makers to embody rules of human behavior and interaction within artificial agents, letting them play out scenarios of society, and measuring emergent macro-phenomena as simulation output. Prime examples include, explaining the population dynamics of ancient societies [29, 2, 3, 30], modeling job markets [31], innovation and economic growth of society and ecosystems [32, 33, 34, 35, 36, 37], urban expansion [38, 39] , evaluating control strategies for the prevention of disease spread [40, 41], and the evolution of languages [42], social norms and culture[43, 44]. Advances in data driven validation of Agent-Based Models have further enhanced the practical applicability of simulation output, through model calibration [45, 46, 47, 48, 49, 50]. Particularly helpful are data-driven calibration tools that come with simulation packages such as OptQuest, an ensemble of Tabu Search, Neural Networks, and Scatter-Search that comes with AnyLogic and Arena [51] and the BehaviorSearch calibration tool that comes with NetLogo, equipped with Standard Genetic Algorithms, Hill-Climber, Stimulated Annealing, and Random Search [16]. These tools treat agent-based models as black-boxes, tuning the parameters they expose to users with the objective of minimizing error between simulation output and patterns in target datasets. There have been efforts to explore the state space of agent-based models with other machine learning methods [52, 45], by treating the model as a black-box and tuning its parameters.

However, there are no such software tools or frameworks in the literature for white-box exploration of agent rules and evaluation of potential causal factors derived from theory. Yet, recently Georges et al. report automating model selection by trading agents using least absolute shrinkage and selection operator (LASSO) on a polynomial form of hypothesized variables and variable interactions [53], similar to the genetic programming of agent rules in Evolutionary Model Discovery. Deep-

learning of agent behavior rules has been suggested [54], which has the potential to improve the predictive power of agent-based models, but undermines the explanatory potential by replacing human-interpretable rules with layers of neural networks. Rand discusses the importance of theory interpretable models [6], highlighting successes of rule induction [55] and causal state modeling [56, 57, 58] towards the discovery of agent rules, learned off of individual-scale data.

Genetic Programming has been used in the past to evolve agent rules for multi-agent systems since [59] where an 'artificial ant' was programmed, through evolution, to navigate an *in Silico* San Mateo trail and is still in a preferable approach in this area [60, 61]. Yet, Genetic Programming has been sparingly for rule-exploration of Agent-Based Modeling within the computational social sciences, some of the prominent studies being: evolving crowd evacuation [62], human-environment action decision making [63], understanding bounded rationality in human decision making [64], and understanding human decision making in behavior finance [65]. An [66] recognizes the capability of genetic programming to discover 'rules of thumb' often defined by domain experts but also calls for the need of more studies into 'why and when' Genetic Programming can best benefit causal exploration in modeling. Manson recognizes the need for standardization of the human decision making process [63], allowing methods like Genetic Programming to be intuitively applied to enhance the explanatory ability of generative Agent-Based Models. In particular, Manson used symbolic regression of factors of land-use as the representation for the Genetic Programming in [63]. This absence of a standard human decision making representation for generative Agent-Based Modeling, ready for Genetic Programming, has limited the use of this methodology in the computational social sciences.

However, with standardization efforts such as the ODD (Overview, Design, and Details) documentation standard [67] being used regularly, there is more transparency and record of the factors and social theories leading to the successful simulation of plausible collective human behavior. The ODD protocol has been used in at least 137 models out of the 447 models in the OpenABM library,

a live compendium of agent-based modeling research [1]. The ODD protocol, and its descendant for models of human decision making ODD+D [68], have sections for the specification of sub-models where model developers are able to elaborate on the agent behavior rules modeled. The body of ODD and ODD+D documentation has grown sufficiently to draw inferences on common structure used in modeling the human decision making process.

Adding to this, has been the creation of agent cognitive architectures to specifically capture social, emotional and rational theories of human behavior founded on neuro-cognitive principles, such as Agent_Zero [26] that aim specifically to model generative aspects of human society. Importantly, Agent_Zero expresses the quantification of emotional decision making, rational decision making, and social influence towards the computation of a social utility. Similarly, Whitmeyer et al. [20, 69] quantify three theories of obedience: Legitimacy, Coercion, and Representative, three theories of Social Influence: None, Social Influence, Resistance to Repression, and four theories of Adaptation/Psychological change: Cognitive Dissonance, Results-Based, Homophily, and Socialization. Together, they make 144 combinations of theories, each representable by a utility function, of human social behavior, driving agents to simulate allegiance patterns in Afghanistan. Similar to this work, Davis and O'Mahony [8] quantify social, emotional, and rational factors of human decision making to produce 'factor trees'.

Koza also demonstrates the ability to evolve programs leading to complex emergent behavior due to the interactions of agents embodying simple rules [59]. Genetic Programming, as defined by Koza, aims to produce programs which are able to evolve new, highly fit programs with regards to the task or tasks they are supposed to solve. Genetic Improvement, the improvement of existing software through automated search has seen an increasing trend in recent years with many successes in the improvement of industrial level programs [70, 71, 72].

[1] This statistic was done by mining ODD documents from OpenABM using a web-crawler. Code available at: https://gitlab.com/chathika/OpenABMCrawler

Genetic Programming has its own pros and cons, a primary disadvantage being its susceptibility to 'bloat'. Langdon defines bloat as increasing redundancy in generated code caused by introduction of introns, genes or GP nodes whose phenotypes have no added value to the behavior expected of the program [73, 74]. Instead, as the Genetic Program nears convergence, the probability of adding an intron to the program syntax tree surpasses the probability of making a mutation or crossover that will further improve the fitness of the resulting program. Despite having a potential of helping the evolutionary process by increasing the number of 'safe' crossover points on the syntax tree, bloat introduces a significant challenge to answering one of the research questions of this study: the relationship between evolved logic depth and task complexity, an issue overlooked in Mason's isometric relationship of bounded rationality to genetic programming syntax depth [64]. This is a disadvantage as true logic depth, accounting for introns, would potentially give a measure of bounded rationality. An agent decision process evolved to fit actual patterns in data would then potentially provide an estimate of the actual limits of human cognition. Langdon summarizes three approaches to avoiding bloat: 1) setting a maximum tree depth, 2) Including program size in the fitness function to introduction selection of parsimonious syntax trees, and 3) tailored genetic operators [73].

Layered Learning [75] has been proposed towards the generalization of solutions evolved through Genetic Programming. With Layered Learning the problem space is decomposed into subtasks which are subjected to learning through the Genetic Program incrementally. This is concept of avoiding over-fitting borrows from the Machine Learning traditions of using validation and test data sets in addition to training data. In particular, Layered Learning has been applied in multi-agent problems such as for agents playing the keep-away soccer game [76, 77].

Several Genetic Programming software exist, including ECJ by Sean Luke et al [78] where GP nodes are written as Java class files, Clojush by Spector et al for the Clojure language [79], DEAP, a distributed evolutionary algorithms software written in Python [80], and FlexGP written in Java

from ALFA Group, MIT [81]. Koza identifies 3 ways of parallelizing genetic programming without the use of sub-populations: distributing by fitness case, distributing by individual syntax trees, and distributing by runs [59]. FCube [82] and CCube [83] are software enabling Genetic Programming distribution over cloud services. FCube in particular, is able to distribute FlexGP learners across Amazon Web Services EC2 instance.

CHAPTER 3: THE EVOLUTIONARY MODEL DISCOVERY FRAMEWORK

Agent-based models consist of autonomous agents driven by rules. The scientific analogy made is that these rules represent hypotheses of human behavior. When the macro-outcomes of the agent-based model are validated against real-world data, these hypotheses declared as not falsifiable. From an engineering standpoint, these rules are often functions of sensory input to the agents through the environment they exist in, sensory input from other agents, or their own internal state. Typically, in the design of an agent-based model, the researcher decides which of these variables are important when defining the rules with which the agents act. Rules are often specified as utility functions, thresholds, or decision trees, with operators combining the variables into a single decision-making unit. Due to the complex nature of the resulting agent interactions, the emergent macro-output of an agent-based model is highly sensitive to the selection of variables and operators, and the design of the rules.

Evolutionary Model Discovery focuses on identifying the sensory inputs from the environment and other agents, and internal state variables, collectively referred to as causal factors, that have the best ability to generate the macro-phenomena being simulated by the agent-based model. This requires isolating the rule to be explored and providing a set of hypothesized causal factors as input. The fitness of the simulations under varying *presence* of each factor within the isolated rule is the experimented with. This is performed through the following two stages.

1. Alternate forms of the rule are genetically programmed into the agent-based model, with the objective of improving the fitness of the simulations to a metric, qualitative measurement or comparison to data, specified by the researcher.

2. The data generated through the evolution of the agent-based model relating factor presence to fitness is then used to train a random forest regressor. Feature importance analysis is then used to determine the importance of the hypothesized factors towards the prediction of the model fitness.

The factor importance data along with the marginal fitness produced under varying presence of each factor in the rule is then used to determine the optimal structure/s of the rule.

Mechanistic Explanation and Causal Factors of Human-Decision Making

Definition 1 *An **agent-based mechanism** M can be defined as follows:*

$$\begin{aligned} M_F &= G^T(N_0) \\ \textit{Where,} \qquad G(N_{t+1}) &= (u_{x_0,F}(N_t) \circ u_{x_1,F}(N_t) .... \circ u_{x_{|N|-1},F}(N_t)) \end{aligned} \tag{3.1}$$

Where, M is modeled as T iterations of a step function, G, on a population of agents, N, with each agent, $x \in N$, acting by a function, u, of one or more factors of human-decision making, F, applied to evaluate N, per time step. M is one out of a space of alternate mechanisms $\mathbb{M}$, defined with varying, F ($F \subset \mathbb{F}$), where $\mathbb{F}$ is the set of all factors of human-decision making.[1]

As elaborated on in Chapter 1, agent-based models are mechanistic explanations of social phenomena [7]. By definition [14], a mechanism consists of entities and their activities that take a system from its initial conditions, ϕ_0, into its observed state, ϕ_t, over time, T. In an agent based model the entities are the agents, x, of a population of, N. The activity of, x, is defined by the agent rules that

[1] G^T denotes each T iterations of function G. $\circ$ denotes functional composition, an abstraction of the operations used to calculate the macro-state of the simulation from agent actions.

can be represented as a function, u, of a set of factors of human-decision making, F, applied on the current state of the population, or $u_{x_1,F}(N_t)$. Each simulation step, t, the composition of these activities, $G(N_{t+1})$, defines the next macro-state. Iteration of $G(N_{t+1})$ T number of times, should then transition the simulation to $G(N_T)$

Definition 2 *A **factor of human decision-making** $F_i \in F$, where F is the modeler's set of hypothesized causal factors and operators, is defined as in Eq (3.2).*

$$F_i = (C, R, P \quad | \quad \theta_R, \theta_R P_k \in \Theta \quad \forall k = 1...n) \tag{3.2}$$

$$\exists k,\ \theta_R R_{f_i} = \theta_R P_{f_j,k} \tag{3.3}$$

Where C is the set of commands defined within F that are applied on the n number of input parameters P to produce an output return value R, where the type of each parameter θ_{P_j} and the type of the return value θ_R are each an element of the set Θ of all possible parameter and return types defined by the modeler. A factor is considered an operator if C resembles an operation on one or more factors, which it accepts as parameters, rather than resembling a decision-making step. In order for a factor or operator F_i to accept another F_j as an input, the condition Eq (3.3) must be met.

An agent behavior rule u is represented as a tree of factors combined under this condition. Depending on Θ and the factor definitions, the space of behavior rules can be infinitely large. To prevent the construction of such undesirably large trees a maximum depth for all u are specified. There must be at least one F_i of which $\theta_{R_{F_i}}$ is the return type expected by the entire agent behavior rule.

Given Definition 1 of an agent-based mechanism and Definition 2 of a factor of decision-making, we theorize what constitutes a mechanistic explanation.

Theorem 1 *An **agent-based mechanistic explanation**, $M_F(\Gamma_{\phi_0} \mapsto \Gamma_{\phi_T})$, of the cause of a system, Γ, with initial macro-state, ϕ_0, to enter a macro-state, ϕ_T, at time, T, must be driven by a step function, G^*, that considers the set of factors, F^* ($F^* \in \mathbb{F}$), that drive the activities of the real-world entities of that system (equation 3.4).*

$$\begin{aligned} M_F(\Gamma_{\phi_0} \mapsto \Gamma_{\phi_T}) &= G^{*T}(N_0) \\ G^*(N_{t+1}) &= (u_{x_0,F^*}(N_t) \circ u_{x_1,F^*}(N_t) .... \circ u_{x_{|N|-1},F^*}(N_t)) \end{aligned} \tag{3.4}$$

$M_F(\Gamma_{\phi_0} \mapsto \Gamma_{\phi_T})$ can be represented by a data-validated model, M^*, with a step function, G^*. G^* encodes u with $F^* \in \mathbb{F}$, such that the simulations make minimal deviations from the actual trajectory of the real system over time. Assuming, total deviation from the data is calculated with some loss function, L, F^* is defined in equation 3.5.

$$F^* = \underset{F \subset \mathbb{F}}{\arg\min} \sum_{t=0}^{T} (L((\mathbb{E}G(N_t), (\Gamma_{\phi_t}))) \tag{3.5}$$

A **mechanistic explanation**, M_{F^*}, would then follow the trajectory of the real-world system the closest out of the possible space of mechanisms, M.

Definition 3 *Presence of a factor or interaction of factors in an agent rule, u, is denoted by its coefficient in u. For example:*

In,

$$u(N) = aF_1(N) + bF_2(N) + ab(F_1(N) \circ F_2(N)) \tag{3.6}$$

Factor presence of F_1 is $p_{F_1} = a$, F_2 is $p_{F_2} = b$, and $F_1 \circ F_2$ is $p_{F_1 \circ F_2} = ab$.

Theorem 2 *The **importance of of causal factors** of human-decision making to a valid mechanistic explanation, M^*, of the macro-phenomena is proportional to the sensitivity of the simulations' deviation from data, with changing factor presence:*

$$Importance \quad F_i \propto \frac{var(\mathbb{E}M^*_{\Pi_{p_{F_i}}})}{var(\mathbb{E}M^*)} \tag{3.7}$$

Where, var indicates variance, and $M^*_{\Pi_{F_i}}$ is M^* perturbed with random permutations of the presence of F_i.

Given the above theorems 1 and 2, I make the following claim and test it empirically, with the cases presented in Chapters 4 to 6:

Claim 1 *An agent-based mechanism, M^*, encoding the casual factors, F^*, of a macro-phenomena at their optimal presence values $p^*_{F_i}$ will produce highly robust simulations with the least deviation from the real-world trajectory of $\Gamma_{\phi_0} \mapsto \Gamma_{\phi_T}$.*

Due to the vastness of $\mathbb{M}$ finding F^* and M^* is difficult. Often, there may not exist a single global optimum due to model stochasticity. Therefore, Evolutionary Model Discovery attempts to computationally approach the best approximations of F^* and M^*.

Genetically Programming Agent Rules with Hypothesized Causal Factors

During the first stage of Evolutionary Model Discovery, models driven by alternate decision making processes consisting of combinations of elements of F are evolved through genetic programming [59, 84, 70]. Genetic programming performs automated program implementation and is a

suitable approach towards automating the rule discovery process [63, 64, 62, 85]. Genetic programming evolves generations of programs through crossover and mutation operators performed on a representation consisting of primitives and terminals that combine to define program statements. Primitives are defined as a set of functions that encode program statements and may be strongly typed to only accept child and parent primitives that are compatible with the arguments and return statements accepted by its program statement. Primitives with no arguments are considered terminals. Programs in a generation that have a closer fit to data are more likely to be selected for reproduction through crossover and mutation to populate the next generation of programs.

Representation

Defining the representation structure for primitives is an important part of evolutionary algorithms. Syntax tree representations are among the most commonly used for genetic programming [59]. Tree representations allow for easily defining limits on how primitives can combine through strong typing, which is important in ensuring the compilability of the generated programs. In the case of Evolutionary Model Discovery the terminals, primitives making up the leaves of the generated trees, are the factors while these factors are combined through multiple operators. Strong typing ensures that this structure is maintained while crossover and mutation operators function.

From Narrative to GP Syntax Tree

In order to explain how causal factors, representing theories and constructs of human behavior can be included in the genetic programming of agent rules, consider the following example. The following is a narrative explaining the behavior of two GitHub users:

> *A*lbert and Betsy are two GitHub users. Albert wants help regarding a project, but is shy to post publicly, and doesn't want to embarrass himself. Meanwhile, Betsy is eager to show off her programming skills, not necessarily to be helpful. After a direct request from Albert, Betsy makes a publicly visible contribution to Albert's project as a response. [2]

Several, causal factors can be identified from analyzing keywords, which have been underlined, of this narrative. Consider a set of causal factors, hypothesized to drive the actions described above: friendship (f), status (s), risk (r), achievement (a). Albert's behavior is driven through positive s and negative r, with a moderate contribution of positive f. On the other hand Betsy's behavior is driven by positive s and positive a, without necessarily being driven by r or f.

Accordingly, this behavior can be represented in the syntax tree shown in Fig. 3.1. This tree represents the process through which utility is assigned to a set of perceived possible behaviors. In the case of GitHub these behaviors are typically restricted to creation of repositories, contribution to or sharing existing repositories, or commenting or posting issues to existing repositories. The set of perceived behaviors may be inherently limited between different individuals and given different situations. For instance Albert is not a good programmer and, therefore, does not have the option of contributing code to existing repositories.

Evaluating Importance and Optimal Presence of Causal Factors with Random Forests

The second stage of Evolutionary Model Discovery is identifying important causal factors and their optimal presence within the behavior being studied. Random forest feature importance methods in combination with non-parametric statistical tests are used for this purpose.

[2] This example is an extended adaptation from a communication with Dr. Joseph Whitmeyer regarding the DARPA SocialSim project

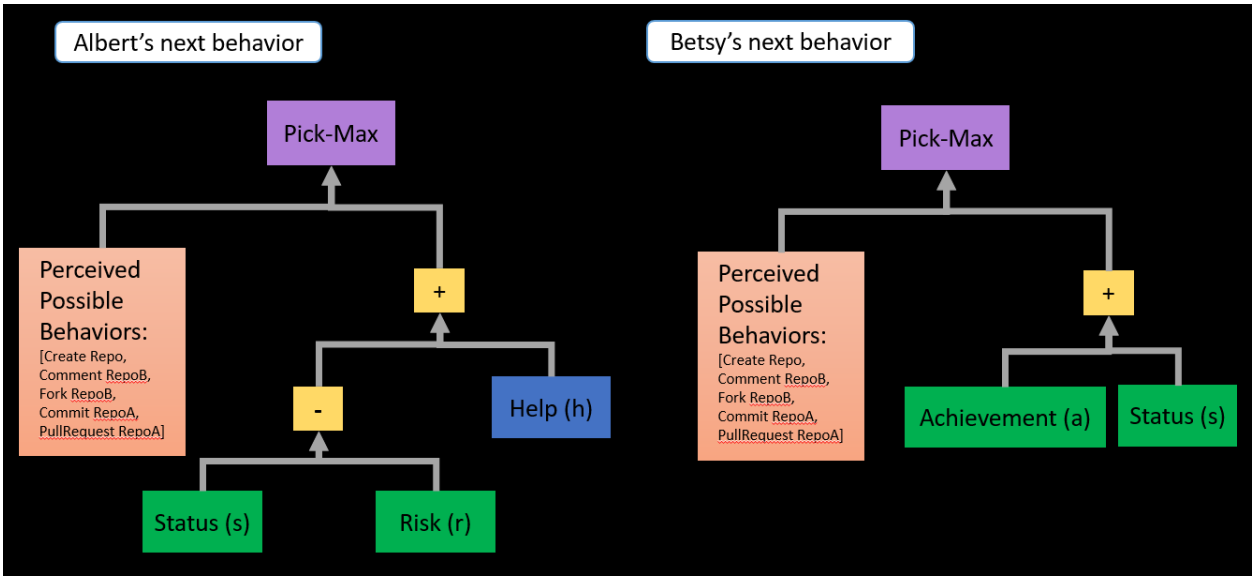


Figure 3.1: An example syntax tree representation of the causal factors and operators driving the behavior of two GitHub users with different motives.

Random forests are powerful ensemble machine learning algorithms for predictive analysis. Predictions of ensemble models are the combined prediction of a set of weak learners. This technique has shown to provide much better results than using a single prediction algorithm [86]. Prediction error of machine learning algorithms exists in a trade-off of the prediction bias and the prediction variance. Ensemble learning algorithms dilute prediction bias and variance through the combination of many high bias single weak learners. In the case of random forests, the weak learners are decision trees and their predictions are combined through *bagging* and random feature sampling. Bagging refers to the individual training of each weak learner, or tree in the case of random forests, on bootstrapped samples of the training data. Bootstrapped sampling refers to the approximation of additional data required to train independent weak learners, learner trees. By sub-sampling the feature set for training independent learners, a random forest is also able to asses the importance of features and feature interactions towards prediction, and is a powerful tool for feature importance analysis and engineering.

Multiple techniques for feature importance evaluation through random forests exist, including gini importance, permutation accuracy importance, and functional analysis of variance. The two most common factor importance measurement techniques for random forests are gini importance (or mean decrease in impurity), and permutation importance (or mean decrease in accuracy)[87, 88, 86]. Unlike gini and permutation importance, fANOVA [89, 90] uses variance based measurements and is unsuitable for use in Evolutionary Model Discovery considering the heteroskedasticity of the fitness data produced by the evolutionary process as the genetic program finds more optimal solutions.

Unfortunately, unlike fANOVA, both gini and permutation importance are only able to identify first order importance of features, but not the importance of feature interactions, instances where joint occurrence of features together are more important than when considered alone. A simple example of the value to feature interactions is demonstrated when considering the importance of two inputs related to an output through an XOR gate. Considered alone, the first order importance of both features is misleadingly high. However, the reality is that the output can be completely defined by the interaction of the two inputs. Fortunately, an algorithm for estimating joint contributions in random forests was designed by Saabas [91, 92]. This algorithm is able to extract the importance of sets of features existing together within a decision tree towards the prediction accuracy of the random forest. Saabas' joint contribution and it's implementation *TreeInterpreter* have been succesfully applied in assessing the importance of feature interactions in a large number of recent studies [93, 94, 95, 96, 97, 98, 99].

The second stage of Evolutionary Model Discovery, involves applying these feature importance evaluation techniques to understand the causal factors and causal factor interactions that were important in the generation of the target macro-phenomena of the agent-based model. Accordingly, a random forest regressor is trained on the factor presence to fitness data produced by the genetic program. Gini importance and permutation accuracy importance for each factor is then calculated.

Joint contributions for factors and factor interactions are also contributed. Due to super-linear relationship between the number factors considered in an interaction and the calculation of joint contribution, in the cases presented in this dissertation, I only consider factor interactions of up to three factors.

Both gini and permutation accuracy importance provide multiple estimates of the importance of a particular feature. In many cases the importance of two or more features may not be distinguishable without statistical tests of significance. Respecting the heteroskedasticity of the factor presence to fitness data, single-tailed Mann-Whitney U tests are used to compare the importance values of each factor pair, say A and B, for all factors, under the alternate hypothesis importance of $A >$ importance of B, for a significance 0.05.

Finally, results of the three techniques, gini importance, permutation accuracy importance, and joint contribution, are used to identify the factors whose change in presence most affects the ability to predict the model fitness. The thus identified causal factors can now be considered to construct a more robust agent rule. In order to do so, the optimal presence of the most important factors must be identified through the data generated by the genetic program. Single-tailed Mann-Whitney U tests are again employed to systematically compare levels of presence of each causal factor, say A and B, under the alternate hypothesis that marginal simulation fitness under $A >$ marginal simulation fitness under B.

Now, the knowledge of the most important causal factors and their respective optimal presence in agent rule can be used to construct versions of the agent-based model that are more robust to parameters and accurately reproduces the target macro-phenomena.

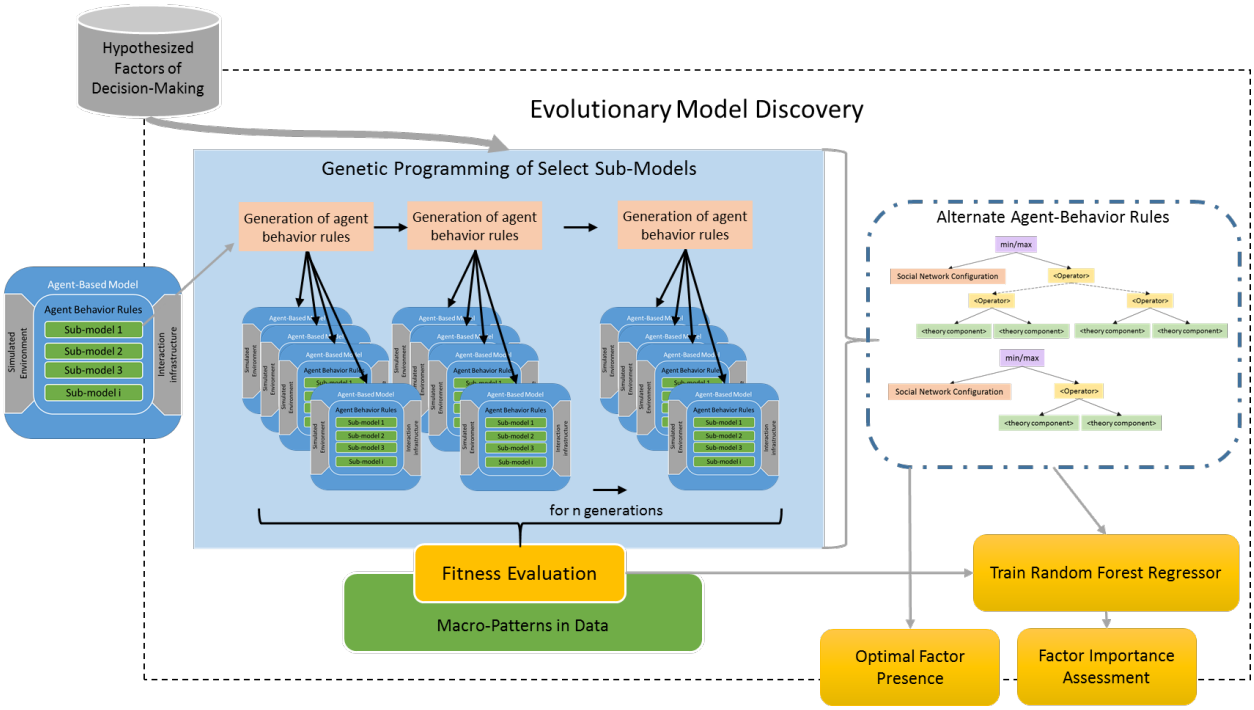


Figure 3.2: Schematic Architecture of Evolutionary Model Discovery

Parallelization and Computational Complexity

Theorem 3 *The time complexity O(T) for the genetic programming of agent-based models of maximum factor tree depth, d, and m number of agents, over g generations, each with n gp-individuals requiring e evaluations each, increases quadratically with the ABM scale and exponentially with logical depth:*

$$O(T) = O(gnem^2 2^{2d}) \tag{3.8}$$

Both genetic programming and random forests have the advantage of being well suited for parallel computation. Genetic programs, similar to many evolutionary algorithms, evolve generations of individuals, referred to gp-individuals to avoid confusion with individuals as humans. This process requires the evaluation of the fitness of each individual. In the case of Evolutionary Model

Discovery, evaluation of gp-individuals means the simulation of the respective agent-based model programmed by that individual. Evaluation of genetically programmed agent rules may be computationally expensive. For each of g generations, each gp-individual of the genetic program of population of size n must be written into the agent-based model as a rule and evaluated multiple times to allow for stochasticity of output, say e evaluations. The aggregated output will then be used for selection for reproduction of the next generation. Assuming the genetic programming uses a simple one-point crossover, point mutation, and tournament selection, and a maximum tree depth of d for the generated syntax trees, the time complexity of the genetic programming, G, without the fitness evaluation would be as shown in Eq. 3.9, as every unit increase in tree depth doubles the number of theory components.

$$O(G) = O(g(n2^d + n2^d + n)) = O(gn2^d) \tag{3.9}$$

Each e evaluations, of the agent-based model would perform runs with populations of m agents of logical depth d, for a maximum 2^d causal factors each, matching that of the respective gp-individual. Assuming each causal factor takes $O(m)$ time (calculating once across all neighbors), the time complexity of execution of the generated agent-based model, A, would be as shown in Eq. 3.10.

$$O(A) = O(m * (m * 2^d)) = O(m^2 2^d) \tag{3.10}$$

Performing e evaluations per gp-individual, the total time complexity of the genetic programming of agent-based models, T, is given in Eq. 3.11.

$$\begin{aligned} O(T) &= O(G * e * A) \\ &= O(gn2^d * e * m^2 2^d) \\ &= O(gnem^2 2^{2d}) \end{aligned} \tag{3.11}$$

Therefore, the runtime of EMD will typically increase quadratically with the agent-based model scale/population size of the agents and exponentially with unit increases in the logical depth.

The implementation of the genetic program for the experiments that follow, was 'embarrassingly parallel'; i.e. it parallelized the evaluations of the agent-based models. The time complexity was thus reduced to $O(gm^2 2^{2d})$, with a computing core dedicated to the evaluation of each gp-indivudal. In future work, parellelizing the agent-based model itself, in particular, the execution of the causal factors would be desirable. However, the technology to support parallel factor execution on top of parallel evaluations at the time of writing is under development.

The time complexity of random forests is discussed in depth by Louppe [86]. Time complexity of the CART algorithm for random forest training $O(XZY^2 log(Y))$, where X is the number of weak learner trees, Z is the number of features, in this case the number of unique factors and factor interactions, and Y the number of samples. In our case $Y = gne$, prior to bootstrapping. Therefore, the time complexity of training the random forest with CART is:

$$O(XZY^2 log(Y)) = O(XZ(gne)^2 log(gne)) \tag{3.12}$$

According to Louppe, the worst case time complexity of prediction through a random forest is $O(XY)$, which reduces to:

$$O(XY) = O(Xgne) \tag{3.13}$$

Implementation

The Evolutionary Model Discovery framework has been implemented as an open-source Python library, for causal inference of NetLogo models, to be used by the computational social science community. A detailed description of this software is provided in Sec. 7, and is available for quick installation through pip package manager for Python.

Several open-source technologies were used for the development of Evolutionary Model Discovery. Evolutionary Model Discovery is able to evolve NetLogo models and the NetLogo controlling API was heavily used in the implementation of this framework [28]. The DEAP library [100] in combination with [101] was used to implement the parallelized genetic program. In order to accommodate for the high computing resources required to expore such a vast search space as described in the next sections, all model runs had to be performed on an AWS EC2 cloud environment. The Scikit-learn [102] library was used for the implementation of random forests and gini importance measurement. Permutation importance was implemented using the ELI5 library [103] was used for permutation accuracy importance, and tree interpreter [92] for joint contribution measurement.

CHAPTER 4: CASE STUDY 1: SOCIO-AGRICULTURAL BEHAVIOR OF THE ANCESTRAL PUEBLO

What socio-agricultural factors might have lead to the sudden demise of a flourishing ancient civilization? Researchers have no direct insight into the decisions made by people of past societies. However, with the thorough assessment of simulation models against archaeological data, we can infer possible decisions that could have led to patterns that are observed in the data. The Artificial Anasazi is one of the first important demonstrations of the success of this technique. I demonstrate how Evolutionary Model Discovery can relax certain assumptions made in the model through exploration of the possible space of rules and highlight factors, which were already present in the original model, that might have proven more important to the patterns seen in the archaeological data.

The Artificial Anasazi model

The Artificial Anasazi is an agent-based model of an ancestral Pueblo community, the Kayenta Anasazi during the years of 800 AD to 1350 AD [1, 2]. This model was initially developed as part of a larger effort to study the ancestral Pueblo civilization that occupied the Long House Valley region. Ancestral Pueblo typically lived in cliff dwellings carved into the face of the rock (Figures 4.1 and 4.2). They depended on agriculture for nutrition and farmed on the valley floor below, typically consisting of rich alluvial soil from the river flood plains. The ABM is implemented in NetLogo [28, 1]. Archaeological excavations provide annual population time series data as estimated counts of households that existed in the valley during the period of study. Annual data on water sources and estimated soil dryness (Adjusted Palmer Drought Severity Index) for each grid location on the map are provided. The model used a normal distribution to map relative

quality of soil over the map. The agent-based model simulates the rise and fall of households over a geographic map of the valley over time and produces a time series of annual household count. The original purpose of the Artificial Anasazi was to test if environmental factors could have triggered the sudden disappearance of the Anasazi from the Long House Valley around 1350 AD.

Figure 4.1: Ancestral Pueblo ruins on the valley floor at the Bandalier National Monument, a site similar to the Long House valley. (Photo credits: Chathika Gunaratne)

Critics of the Artificial Anasazi have argued that the agent-based model itself is but a single candidate explanation of the social phenomenon at hand, the rise and fall of the Anasazi population over time [18]. However, this can be viewed as an advantage as the Artificial Anasazi can be used as a test-bed to discover multiple plausible explanations of the population dynamics of the Long Valley at the time. Testing combinations of hypothesized factors that may have influenced actual decision-making processes of the individuals results in a vast search space of plausible Artificial Anasazi behavior results. We concentrated on a particular sub-model of the Artificial Anasazi: the

farm plot selection strategy. The households perform farm plot selection under two conditions: 1) when a new child household is hatched by a household that has enough resources to increase its family size, or 2) when the current farm plot is unable to produce enough yield to satisfy the nutrition needs of the household anymore. The original model, hypothesizes that the households simply selected the next closest available farm plot to the household's current farm plot during farm plot selection, i.e., minimizing over distance. A patch must be free of farms or households and not be located inside a water body to be available. Consequently, the original farm selection strategy ignores other sensory data available to the households regarding the land and the state of other households in the valley.

Figure 4.2: Cliff dwellings of the ancestral Pueblo at the Bandalier National Monument. (Photo credits: Chathika Gunaratne)

Hypothesized Alternate Factors Influencing Farm Plot Selection

Human social behavior is rarely entirely rational. Accordingly, our hypothesis proposed that the farm selection decisions of the ancestral Pueblo were complex, and took into account the state of the potential farm plots available to them and the social influences of other households around them. Agent_Zero [9] models the human decision making process into three dimensions: social, emotional and rational. Similarly, factors hypothesized to influence the farm plot selection process within these dimensions were defined. The social component is expressed through four mutually exclusive social connectivity configurations through which the agent could receive information on a subset of potential farm plots, s, out of the entire set of potential farm plots in the valley, S_{All}. The received information is then processed through a utility function $f(x)$ defined as a combination of factors and operators, F, which consider both the internal state of the household and the conditions of the farm plot and its surroundings in order to determine the next farm plot $x' \in s \subset S_{All}$ as in Eq (4.1).

$$x' = \underset{x \in s \subset S_{All}}{\operatorname{argmax}} f(x) \tag{4.1}$$

Households in the original Artificial Anasazi model consider a single factor, distance, which will be referred to as F_{Dist}, and choose the potential farm plot with minimal distance to their current farm location. No further factors are considered in the decision making process. Furthermore, the original model assumes that the households have complete information of the valley, and every potential farm plot is compared. Therefore, the farm selection process of the original Artificial Anasazi can be represented as in Eq (4.1).

$$x' = \underset{x \in S_{All}}{\operatorname{argmax}}(-F_{Dist}(x)) \tag{4.2}$$

Arguing that the farm selection decision may have been more complex, considering a variety of other factors, an extended factor set is proposed, consisting of four social and five rational factors, namely: homophily by age (F_{HAge}), homophily by agricultural productivity (F_{HAgri}), social presence (F_{Soc}), migration from current zone (F_{Mig}), comparison of quality (F_{Qual}), comparison of dryness (F_{Dry}), comparison of yield (F_{Yeild}), comparison of water availability (F_{Water}), and comparison of distance (F_{Dist}). Additionally, the numerical operators $+$ and $-$ are included in F, for the aggregation of sub-scores reported by the social/emotional and rational factors.

Four hypothesized configurations of social connectivity were included F. These configurations determined the subset of all viable farm plots that were to be considered by the households for comparison.

- **Full information (S_{All})**: Households had complete knowledge of all potential farm plots in the valley. Full information was used by agents in the original version of the model, assuming that each household knew and compared every potential farm plot in the Long House Valley.
- **Family inherited information (S_{Fam})**: Households solely depended on information available through their 'family'. Families are defined as a household's parent household, sibling households, any surviving grandparents, and the household itself.
- **Nearest-neighbor information (S_{Neigh})**: agents only consider the farm plots known to their neighboring households within a fixed radius of their current location.
- **Best performers** S_{Perf}: Households only consider potential farm plots known to the best performing households, demonstrating a leadership dynamic.

Four social/emotional factors were included in F: two types of homophily (the tendency for social entities to congregate among those with similar traits), need for social presence, and one of fleeing/migration. Each social/emotional factor returned a sub-score representing the desirability of

each evaluated farm plot. Sub-scores were normalized within the factors, to lie in the range of 0 to 1, for fair comparison.

- **Homophily by age** (F_{HAge}): Households prefer to select farm plots near other households that are of similar age, where age is measured as the number of simulation steps the household has survived since splitting from its parent.
- **Homophily by agricultural productivity** (F_{HAgri}): Households tend to select farm plots near other households with a similar corn stock to itself.
- **Social presence** (F_{Soc}): Agents score potential farm plots with many nearby households higher than those in isolation.
- **Fleeing/migration** (F_{Mig}): Agents score potential farm plots that are in a completely different zone than the current one with a full sub-score, while patches in the same zone receive a sub-score of zero.

Five Rational factors considered for the farm selection process were logical comparisons of sensory data on the potential farm plots already available to the households in the original model. Similar to the social/emotional factors, rational factors also returned a normalized sub-score of farm plot desirability between 0 and 1.

- **Comparison of quality** (F_{Qual}): Higher sub-scores were reported for potential farm plots with higher quality of land.
- **Comparison of dryness** (F_{Dry}): Higher sub-scores were reported for potential farm plots with higher dryness of land.
- **Comparison of yield** (F_{Yeild}): Higher sub-scores were reported for potential farm plots that were known to have higher yield in the previous year.

- **Water availability (F_{Water})**: Higher sub-scores were reported for potential farm plots with more nearby water sources.
- **Comparison of distance (F_{Dist})**: Higher sub-scores were reported for potential farm plots that were closer to the current farm plot location.

The addition ($+$) and subtraction ($-$) operators were included as primitives of the genetic program to construct rules from the hypothesized factors.

Experiments

Twenty genetic programming runs were executed with the objective of minimizing the (RMSE) between the simulated household count to the actual household count over 550 simulation ticks of the Artificial Anasazi. Details on the RMSE calculation can be found in [85]. In order to ensure robustness of the evolved rules, the parameters of the ABM were randomly initialized with values $\pm 5\%$ about the optimal parameter values found through Stonedahl's calibration of the Artificial Anazasi through a genetic algorithm [16] (ie: water source distance = $(10.925, 12.075)$, death age span = $(9.5, 10.5)$, min fertility = $(0.1615, 0.1785)$, base nutrition need = $(175.75, 194.25)$, fertility span = $(0.0285, 0.0315)$, min fertility ends age = $(27.55, 30.45)$, harvest variance = $(0.418, 0.462)$, harvest adjustment = $(0.608, 0.672)$, maize gift to child = $(0.4465, 0.4935)$, min death age = $(38.0, 42.0)$, fertility ends age span = $(4.75, 5.25)$). Each genetic program run was executed for 100 generations over populations of 50 individuals. Syntax trees of minimum depth 4 and maximum depth 10 were used to avoid trees exhibiting bloat. The Half-and-Half tree builder was used for initialization [59].

Finally, new farm selection strategies were designed taking into account the insights gained through Evolutionary Model Discovery. The robustness of the Artificial Anasazi with these new strategies were tested against the original model by comparing the RMSE of 100 runs of each model under

randomized initialization of parameters within the ranges above.

Results

The resulting best farm selection strategies evolved by the genetic program by run are provided in Table 6.1 along with their respective RMSE values. 15 of the runs produced RMSE values lower than the current best RMSE in the literature obtained through parameter calibration of the Artificial Anasazi model with the original farm plot selection by closeness (733.6) [4]. All best scoring rules for each run utilized S_{All}, i.e., the model produced best results when the agents had full information regarding available farm plots as shown in Fig. 4.3, comparing S_{All}, S_{Fam}, S_{Neigh}, and S_{Perf} over the complete factor presence to fitness data. One-tailed Mann-Whitney U tests comparing the fitness of all rules by their social connectivity configurations confirmed that rules with S_{All} had significantly ($\alpha = 0.05$) lower RMSE than the other three configurations: $\text{argmax}_{x \in S_{All}} f(x) < \text{argmax}_{x \in S_{Fam}} f(x)$ ($p = 2.045 \times 10^{-113}$), $\text{argmax}_{x \in S_{All}} f(x) < \text{argmax}_{x \in S_{Neigh}} f(x)$ ($p = 4.856 \times 10^{-154}$), $\text{argmax}_{x \in S_{All}} f(x) < \text{argmax}_{x \in S_{Perf}} f(x)$ ($p = 1.983 \times 10^{-57}$). Also, rules with S_{Neigh} were shown to have significantly (alpha=0.05) lower RMSE than those with S_{Fam} and S_{Perf}: $\text{argmax}_{x \in S_{Neigh}} f(x) < \text{argmax}_{x \in S_{Fam}} f(x)$ ($p = 3.535 \times 10^{-14}$), $\text{argmax}_{x \in S_{Neigh}} f(x) < \text{argmax}_{x \in S_{Perf}} f(x)$ ($p = 2.339^{-24}$). Finally, rules with S_{Fam} were shown to have significantly (alpha=0.05) lower RMSE than rules with S_{Perf}: $\text{argmax}_{x \in S_{Fam}} f(x) < \text{argmax}_{x \in S_{Perf}} f(x)$ ($p = 0.012$). Accordingly, the rest of the analyses detailed in this paper were performed on rules where the social connectivity configuration was S_{All}.

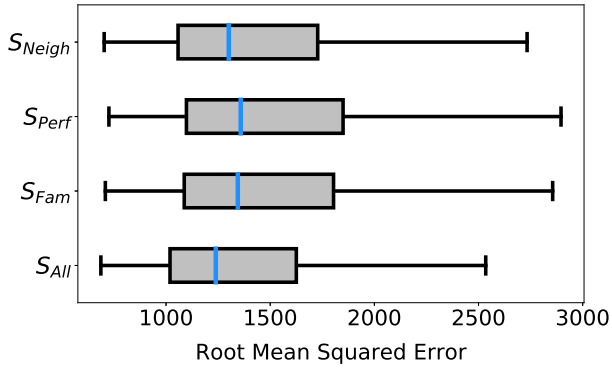


Figure 4.3: Best fit to data was obtained under S_{All}. Comparison of the RMSE produced by the Artificial Anasazi model when agents had full information (S_{All}), information through family households (S_{Fam}), information through the households with most agricultural success (S_{Perf}), or information through neighboring households (S_{Neigh}). Models that used S_{All} produced the lowest RMSE overall $\text{argmax}_{x \in S_{All}} f(x) < \text{argmax}_{x \in S_{Fam}} f(x)$ ($p = 2.045 \times 10^{-113}$), $\text{argmax}_{x \in S_{All}} f(x) < \text{argmax}_{x \in S_{Neigh}} f(x)$ ($p = 4.856 \times 10^{-154}$), $\text{argmax}_{x \in S_{All}} f(x) < \text{argmax}_{x \in S_{Perf}} f(x)$ ($p = 1.983 \times 10^{-57}$).

Table 4.1: The candidate farm selection strategies of models produced by the Evolutionary Model Discovery process along with their best fitness as reported by the genetic programming search.

GP Run	Best scoring rule	Best Fitness
0	$\underset{x \in S_{All}}{\text{argmax}}(F_{Mig}(x))$	753.43
1	$\underset{x \in S_{All}}{\text{argmax}}(-F_{Dist}(x) - F_{Dry}(x) + 2 * F_{Mig}(x))$	755.27
2	$\underset{x \in S_{All}}{\text{argmax}}(F_{Yield}(x) + F_{HAgri}(x))$	709.50
3	$\underset{x \in S_{All}}{\text{argmax}}(F_{Mig}(x) - F_{HAgri}(x))$	738.95
4	$\underset{x \in S_{All}}{\text{argmax}}(F_{Mig}(x))$	730.48
5	$\underset{x \in S_{All}}{\text{argmax}}(F_{Dist}(x))$	752.52
6	$\underset{x \in S_{All}}{\text{argmax}}(F_{Dist}(x))$	728.29
7	$\underset{x \in S_{All}}{\text{argmax}}(F_{Yield}(x))$	714.21
8	$\underset{x \in S_{All}}{\text{argmax}}(F_{Dist}(x) - F_{Dry}(x))$	734.25
9	$\underset{x \in S_{All}}{\text{argmax}}(4 * F_{Dist}(x) + F_{Dry}(x) + F_{Qual}(x) + F_{Water}(x) + F_{Soc}(x) + F_{HAge}(x))$	701.21
10	$\underset{x \in S_{All}}{\text{argmax}}(F_{Dist}(x) + F_{Qual}(x) + F_{Water}(x) - F_{Yield}(x) + F_{Mig}(x) + F_{Soc}(x))$	720.285
11	$\underset{x \in S_{All}}{\text{argmax}}(F_{Mig}(x))$	723.63
12	$\underset{x \in S_{All}}{\text{argmax}}(F_{Dist}(x) + F_{Qual}(x) + 2 * F_{Yield}(x) + 2 * F_{Mig}(x) + F_{Soc}(x) + F_{HAgri}(x))$	687.12
13	$\underset{x \in S_{All}}{\text{argmax}}(F_{Dist}(x) + F_{Soc}(x))$	732.19
14	$\underset{x \in S_{All}}{\text{argmax}}(F_{Qual}(x))$	728.77
15	$\underset{x \in S_{All}}{\text{argmax}}(F_{Qual}(x))$	706.28
16	$\underset{x \in S_{All}}{\text{argmax}}(F_{Dist}(x) + 2 * F_{Qual}(x) + F_{Yield}(x) + F_{Soc}(x) + 3 * F_{HAge}(x))$	715.96
17	$\underset{x \in S_{All}}{\text{argmax}}(F_{Mig}(x))$	715.47
18	$\underset{x \in S_{All}}{\text{argmax}}(-F_{Dist}(x) + F_{Soc}(x) - F_{HAgri}(x))$	701.44
19	$\underset{x \in S_{All}}{\text{argmax}}(F_{Qual}(x) + F_{Mig}(x) + F_{Soc}(x))$	701.30

Fig. 4.4 displays the distribution of RMSE against factor presence, for presence values that were recorded in at least 200 rules across the 20 genetic program runs. Negative correlations to RMSE (higher fitness) are seen between F_{Dist}, F_{Qual}, F_{Water}, F_{Yield}, F_{Mig}, F_{Soc}, and F_{Age}, and in general the genetic program favored the positive presence of these factors, and evolved more rules with these factors having a positive effect on farm selection. F_{Dry} on the other hand had a negative correlation to RMSE for presence less than 2.

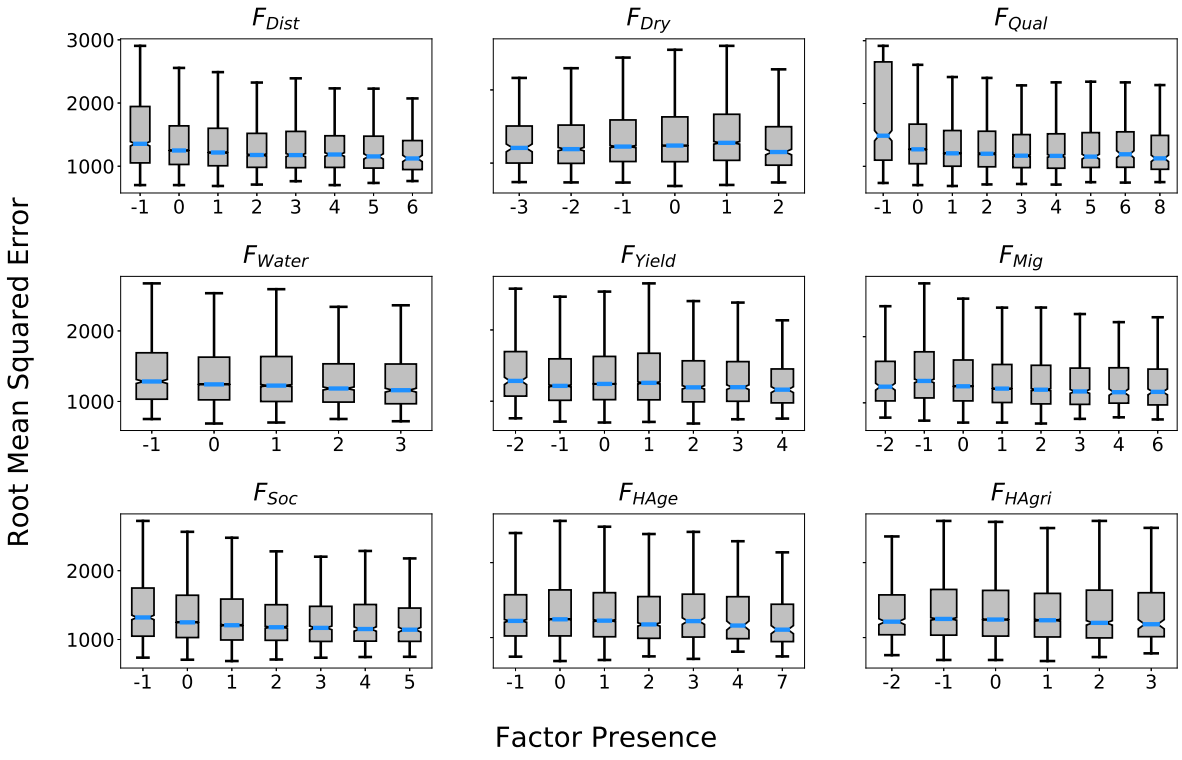


Figure 4.4: RMSE vs Factor Presence under S_{All}. RMSE distributions by factor presence produced by Evolutionary Model Discovery of the farm selection strategy of the Artificial Anasazi under S_{All}. Only presence values that appeared at least 200 times in the genetic program are displayed. Most factors display negative correlations to RMSE, while F_{Dry} shows a positive correlation.

The random forest fit the factor presence to fitness data best for a forest of 520 regression trees, testing from 10 to 1000 trees with a train/test split 90%-10%. Accordingly, a forest of 520 trees was used for factor importance determination. Factor importance under S_{All} obtained through both the gini importance and permutation accuracy importance techniques can be seen in Fig. 4.5. Gini importance generally had less precise estimations than permutation accuracy importance. Yet both techniques indicated F_{Qual} as the factor of highest importance towards RMSE prediction. F_{Soc}, F_{Mig}, and F_{Dist} also scored higher importance values than the other factors hypothesized. Fig. 4.6 displays the p-values of one-tailed Mann Whitney U tests (alpha=0.05), comparing the permutation importance of each factor A against every other factor B, testing the alternate hypothesis: importance of $A >$ importance of B. According to the results, 7 of the 9 factors showed significant difference and could be ordered in terms of permutation accuracy importance as F_{Qual}, F_{Soc}, F_{Dist}, F_{Mig}, F_{Water}, F_{Yield}, F_{HAgri}, F_{HAge}, and F_{Dry}.

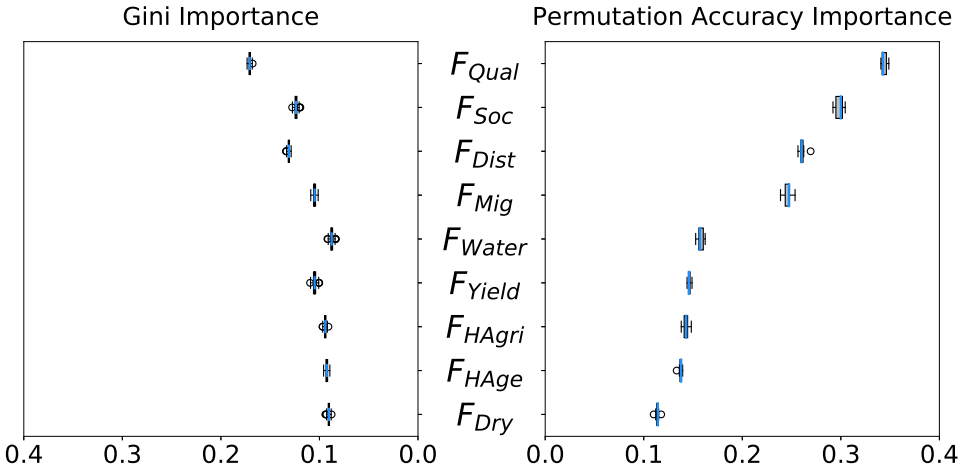


Figure 4.5: F_{Qual}, F_{Soc}, F_{Dist}, and F_{Mig} have highest Gini and Permutation Accuracy Importance. Gini importance and permutation accuracy importance of the hypothesized factors towards a random forest's ability to predict the models' RMSE. Gini importance results are less decisive than permutation accuracy importance. Both techniques agree that F_{Qual}, F_{Soc}, F_{Dist}, and F_{Mig} are the most important factors.

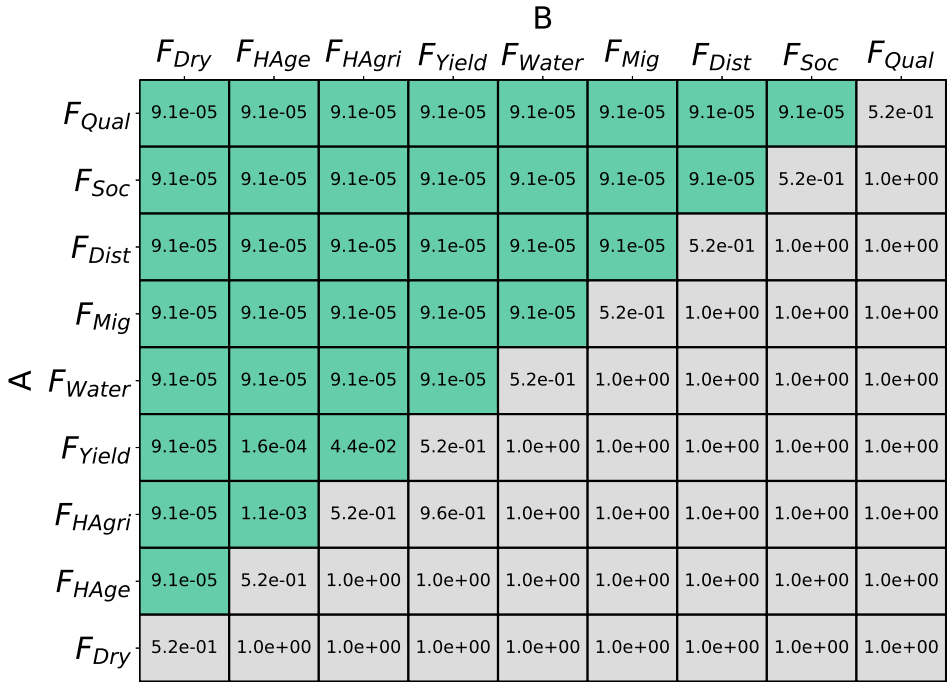


Figure 4.6: Statistical confirmation of the existence of order by importance among causal factors. Results from systematic Mann-Whitney U tests on the permutation accuracy importance results. The cells contain p-values for the alternate hypothesis that $A > B$ (null hypothesis $A = B$). Green cells indicate agreement of the alternate hypothesis. The results indicate a clear ordering of the factors by importance.

Fig. 4.7 compares the top ten joint contributions towards RMSE prediction of the random forest by individual factors, and joint contributions of factors considered in pairs and triples. Again, F_{Qual} demonstrates far higher importance than any other factor or factor interaction. The factor pairs (F_{Qual}, F_{Mig}) and (F_{Qual}, F_{Soc}) also demonstrate high importance, followed by $(F_{Qual}, F_{Mig}, F_{Soc})$, $(F_{Dry}, F_{Qual}, F_{Mig})$, and $(F_{Dist}, F_{Qual}, F_{Soc})$. Overall, F_{Qual} is present in all highest scoring joint contributions. Despite F_{Dry} having very low individual importance, F_{Dry} showed higher importance when considered in combination with F_{Qual} and F_{Mig}.

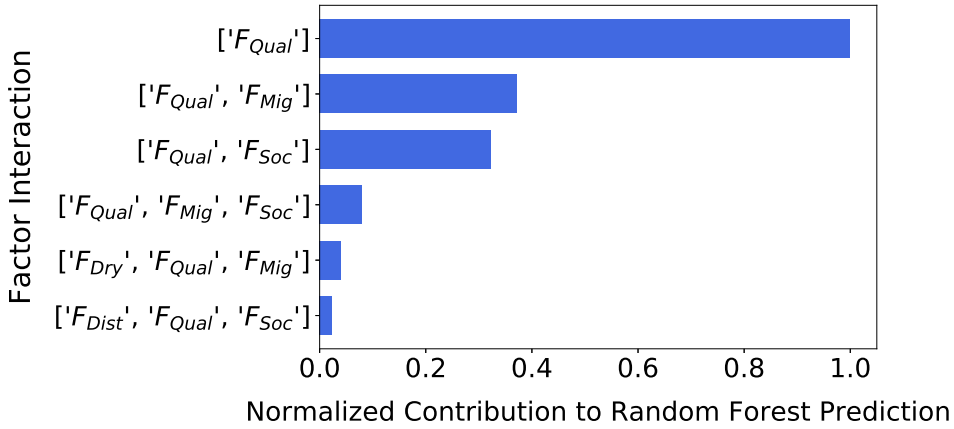


Figure 4.7: F_{Qual}, [F_{Qual},F_{Mig}], and [$F_{Qual}F_{Soc}$] have highest joint contribution to farm plot selection. Ordered barchart of highest normalized joint contribution scores of factors and interactions of three or less under S_{All}. Again, F_{Qual} shows a far larger contribution to the random forest's ability to predict model RMSE than other factors and factor interactions, and is present in all of the highest contributing interactions. Interactions [F_{Qual},F_{Mig}] and [F_{Qual},F_{Soc}] also demonstrate high joint contribution.

Considering the evidence of F_{Qual}, F_{Soc}, F_{Mig}, F_{Dist}, and F_{Dry} as important factors, Fig. 4.8 demonstrates Mann Whitney U tests conducted for each factor F_i, for the alternate hypothesis that RMSE when presence of F_i was A, is less than the RMSE when presence of F_i was B in rules with S_{All}. Models with positive presence of F_{Qual}, F_{Soc}, F_{Dist}, and F_{Mig} showed significantly higher fitness (with the exception of when presence of F_{Mig} = -2). Models with strong positive or negative presence of F_{Dry} showed lower RMSE overall, most likely a result of F_{Dry}'s interaction with F_{Qual}, F_{Soc}, or F_{Mig}. The lowest median RMSE for (F_{Qual}, F_{Soc}) was 985 at presence of F_{Soc} at 5 and presence of F_{Qual} at; the lowest median RMSE for (F_{Qual}, F_{Mig}) was 997 at presence of F_{Mig} at 3 and presence of F_{Qual} at 5.

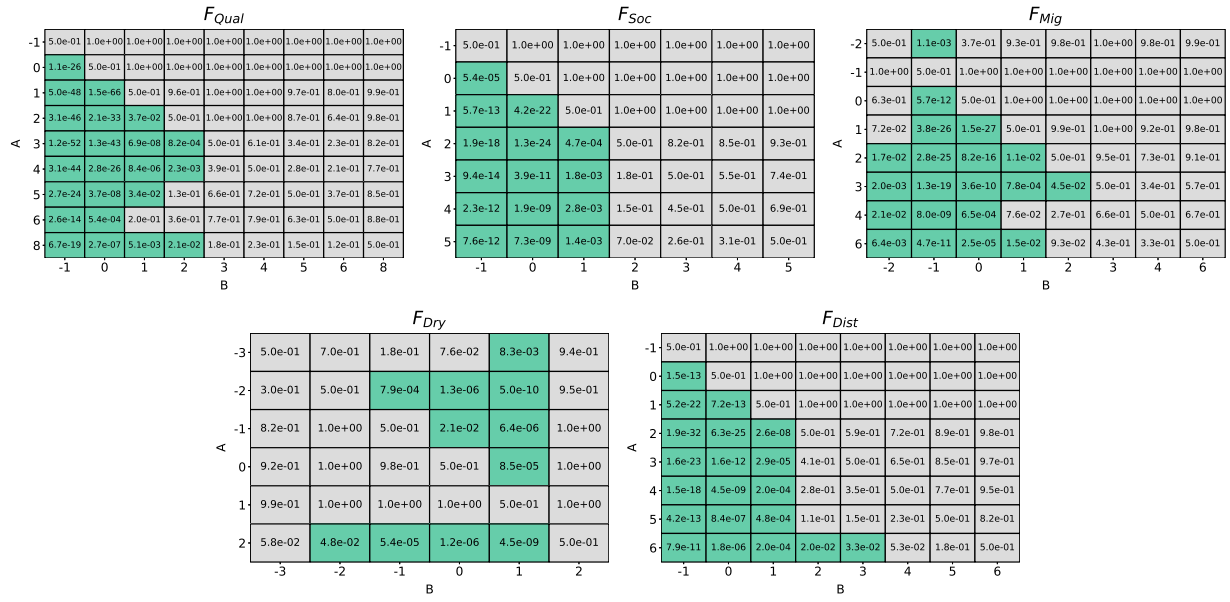


Figure 4.8: Optimal presence scores for causal factors with highest importance. Results from systematic one-tailed Mann-Whitney U tests between presence values of the 5 most important factors for the alternate hypothesis: RMSE for presence $A <$ RMSE for presence B (null hypothesis: RMSE for presence $A =$ RMSE for presence B) for $\alpha = 0.05$. Green cells indicate agreement of the alternate hypothesis. Results indicate that for F_{Qual}, F_{Soc}, F_{Mig}, and F_{Dist} RMSE is generally lower for higher, positive presence. For F_{Dry}, both negative and higher positive presence may provide low RMSE scores.

Finally, rules following the three highest joint contributions were constructed using the best values for each factor concerned: $\text{argmax}_{x \in S_{All}}(F_{Qual}(x))$, $\text{argmax}_{x \in S_{All}}(5F_{Soc}(x) + 6F_{Qual}(x))$, and $\text{argmax}_{x \in S_{All}}(3F_{Mig}(x) + 5F_{Qual}(x))$, and RMSE was compared against the original farm selection strategy $\text{argmax}_{x \in S_{All}}(-F_{Dist}(x))$ for 100 runs each under random initialization of parameters within the ranges specified in section 4. Fig. 4.9 shows that all three of these rules derived through Evolutionary Model Discovery have significantly lower RMSE than that of the original farm selection strategy under randomized parameter initialization.

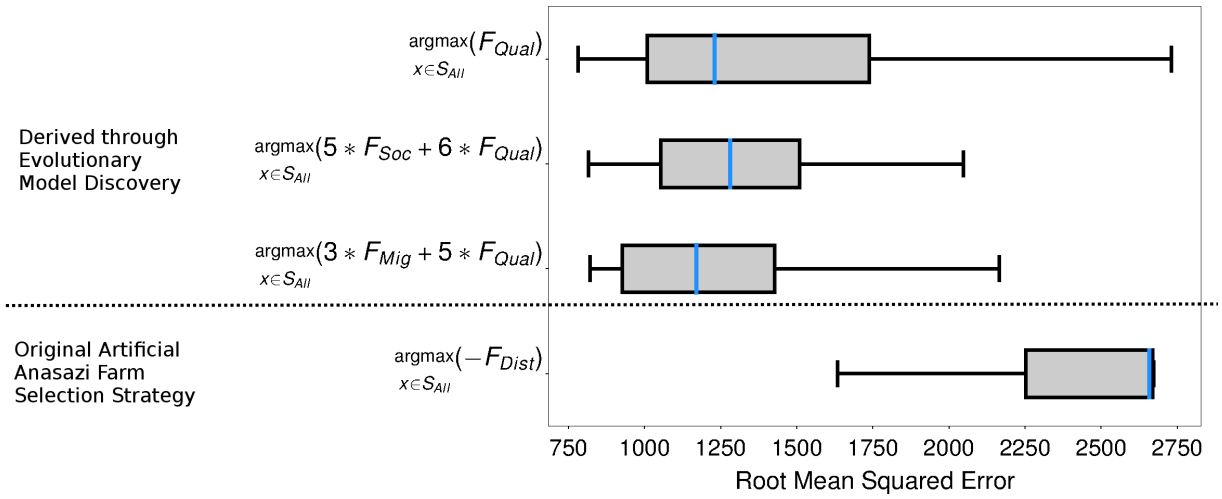


Figure 4.9: Models designed through Evolutionary Model Discovery insights are significantly more robust. Comparison between the RMSE of 100 runs of three models with farm selection strategies designed taking into consideration the insights from Evolutionary Model Discovery, 1) $\text{argmax}_{x \in S_{All}}(F_{Qual}(x))$, 2) $\text{argmax}_{x \in S_{All}}(5F_{Soc}(x) + 6F_{Qual}(x))$, and 3) $\text{argmax}_{x \in S_{All}}(3F_{Mig}(x) + 5F_{Qual}(x))$, against 100 runs of the original farm selection strategy $\text{argmax}_{x \in S_{All}}(-F_{Dist}(x))$ in [1, 2, 3, 4], under random initialization of parameters. The three farm selection strategies derived from Evolutionary Model Discovery are far more robust under random parameter initialization and show significantly better RMSE scores compared to the original model.

Discussion

Mechanistic Explanation *Upon failure of a farm plot, the ancestral Pueblo households of the Long House valley, were likely to consider the whole valley in search of new land to farm on, preferring areas that indicated higher soil quality, higher social presence, and farming further away from areas where farm plots failed previously.*

Applying Evolutionary Model Discovery on the Artificial Anasazi, I show that the socio-agricultural behavior of the ancestral Pueblo of the Long House Valley was more deliberative and informed than originally assumed. Our results indicate that, contrary to the original farm selection behavior, where households would select the next closest possible plot of land once their present farm

was depleted, the households most likely selected potential farming land with higher soil quality (F_{Qual}). Further, it was highly likely that the households had good knowledge of the potential arable land throughout the valley, since S_{All} was the best social connectivity configuration for information spread. Also, the desire to congregate into communities was indicated, as positive desire for social presence (F_{Soc}) was the second most important factor, and acting on information on arable land known to neighboring households (S_{Neigh}) was the second most successful social connectivity configuration. Further, instead of choosing closer potential farm plots ($-F_{Dist}$), choosing farm plots that were further away from the households current farm plot (F_{Dist}) or moving to a completely different zone in the region (F_{Mig}) was found to be a more likely behavior. Finally, versions of the Artificial Anasazi where farm plot selection was driven by seeking higher quality land, higher quality land with more social presence, and higher quality land in different zones, all proved to be significantly more robust than the decision to move to the next closest available plot of land (Fig. 4.9).

CHAPTER 5: CASE STUDY 2: MIXED PATTERNS OF RESIDENTIAL SEGREGATION AND INTEGRATION

Why do communities of individuals with no racial bias still maintain pockets of segregation? Among the earliest phenomena recognized as an unexpected, emergent result of complex interactions at an individual-scale, this phenomena was originally studied by Thomas Schelling in his seminal, model of residential segregation [104]. One of the earliest demonstrations of the value of Agent-Based Modeling in the Computational Social Sciences, Schelling's model of segregation [104] modeled the dynamics of residential segregation among two races sharing a common spatial residential region. Through his model, Schelling demonstrated how populations of individuals who did not prefer a majority of their neighbors to be of similar race, could still end up in segregated neighborhoods. Each agent prefers to have F percent of its immediate neighbors be of similar race. In the original model, F is a homogeneous parameter of the Agent-Based Model. The only other parameter of the Agent-Based Model is the density of agents D in the residential region. An agent that is able to satisfy this condition is labeled as 'happy'. If an agent is unhappy, i.e. if there are not enough neighbors of its own race as its immediate neighbors of its current location c to satisfy F, the agent decides to move its residence to a random new location. The rule of racial preference for any agent a used in Schelling's Segregation can be represented as follows:

$$\text{Relocate}_a \iff H_a \neq 1$$

$$H_a = f_a(c) > F$$

Where H is the boolean state of happiness of the agent and $f_a(c)$ is the percentage of similar neighbors immediately surrounding the agent a at location c. In the original Schelling's Segregation, the above inequality is the sole determinant of an agent's decision to relocate. This rule expresses the

agent's internal disposition to its neighborhood. It does not consider the influence of the disposition of other agents in its neighborhood.

Hatna's Model of Mixed Segregation-Integration Patterns

Hatna and Benenson [105], extend the original Schelling's Segregation model in several ways. Most importantly, The desirability of a residential location on the grid is measured through a utility function, $u_{a,i}$, eq. 5.1.

$$u_{a,i} = \begin{cases} \frac{f_a(i)}{F_a}, & \text{if } f_a(i) < F_a \wedge F_a > 0, \\ 1, & \text{otherwise.} \end{cases} \tag{5.1}$$

Where, i is a location considered by agent a, F_a is its personal tolerance to the fraction of racially similar neighbors, and f_a is its current fraction of friends. The agents would relocate to a patch i if its utility, $u_{a,i}$, exceeded the utility of its current patch, c, as shown in eq. 5.2.

$$Relocate \iff (\exists i u_{a,i} > u_{a,c}) \vee (u_{a,c} = 1 \wedge p < m) \tag{5.2}$$

Where, p is a uniform random noise function and m the probability of moving despite complete satisfaction, a parameter of the model. The utility function eq. 5.1 was still premised on the Schelling's original thesis of racial preference. Additionally, they separate the processes of needing to leave one's current household and the decision to move to a new residential location. This allows agent's who desire to move to a new location to resist moving unless they find a more desirable location to move to. Further, this model, allows agent's who are completely satisfied with their current location to still move to a new location at a certain probability, acting as a noise function.

They allow for heterogeneous tolerance experimenting with ratios of high and low tolerance agents and distributions of tolerance, in addition to ratios of the abundance of both races.

Hatna and Benenson are able to demonstrate that with these extensions, slightly mixed patterns of segregation and integration are able to co-exist within a population for specific parameter values. They provide evidence through spatio-demographic analysis of several cities, showing that such mixed patterns are more likely to exist in urban areas. They develop a metric, the C-index [106], which is able to quantify the equal existence of segregated and integrated areas on a single map. In [105], they provide extensive analysis of the racial tolerance distributions and race abundance ratios that are able to produce residential patterns with high c-indices. It is seen that C-index is highest when both races are in equal proportion and there are equal numbers of highly racially intolerant and highly racially tolerant individuals.

However, racial preference is not the sole factor defining the desirability of a residence. Perhaps other factors could lead to stronger mixed patterns with higher c-indices. Several factors such as financial suitability, safety and stability of the neighborhood, or familiarity with the neighborhood, usually affect the decision to select a place of resident, to name a few. The question however, is how important these factors are towards the emergence of mixed patterns of segregation to form? With minimal modification to the Hatna's model of segregation I test the importance of racial preference against several other hypothesized factors on their ability to produce mixed patterns of segregation and integration.

Causal Factors for Mixed Patterns of Segregation and Integration

The following factors are hypothesized to affect the desirability of a new location of residence. Each factor provides a utility sub-score between 0 and 1 and takes in a location of interest or the

current residential location as a parameter.

- **Fraction of racial similarity** F_{Race}: The fraction of agents in the neighborhood who are of the same race as the agent. This is the factor considered in both Schelling's seminal work [104] and Hatna's more recent model [105]

- **Mean neighborhood's tolerance** $F_{TolMean}$: Represents the typical tolerance to the other race shown in the neighborhood. Measured as the mean racial tolerance of all the agents in the neighborhood.

- **Diversity of neighborhood's tolerance** F_{TolDiv}: Represents the diversity of tolerance shown in the neighborhood towards the other race. Measured as the variance of the racial tolerance of all the agents in the neighborhood.

- **Neighborhood isolation** F_{Isol}: Represents the isolation of the neighborhood. The normalized number of unoccupied locations in the neighborhood.

- **Length of residence** F_{Res}: Represents the time spent in the neighborhood. Measures the number of time steps this agent has resided at the location.

- **Mean satisfaction of neighborhood residents** $F_{SatMean}$: Scores higher if the residents of the neighborhood of the location are themselves satisfied. Measured by taking the mean of the home utility of the residents in the neighborhood of the location being considered.

- **Diversity of satisfaction of neighborhood residents** F_{SatDiv}: Scores higher if the residents of the neighborhood show more diverse levels of satisfaction. Measured by taking the variance of the home utility of the residents in the neighborhood of the location being considered.

- **Distance** F_{Dist}: Measures the distance the location is from the agent's current home patch.

- **Tendency to move** F_{Move}: Scores higher if the agent has moved residential locations more frequently in the past. If the location considered is the home patch, measured as 1 - the number of residences the agent has had in the past divided by the number of ticks. If the location considered is not the home patch then is measured as the number of residences the agent has had in the past divided by the number of ticks.

The addition $(+)$ and subtraction $(-)$ operators were included as primitives of the genetic program to construct rules from the hypothesized factors.

Experiments

Hatna's NetLogo implementation of residential segregation [105] was extended for Evolutionary Model Discovery. The utility function measuring desirability was tagged as the entry point for EMD with implementations of the factors defined in 5.

In [105], C-index is found to be highest when equal proportions of the two races (blue and green) were present, along with equal proportions of less tolerant and highly tolerant agents of both races. Accordingly, the fraction of blue agents was set to 0.5, and tolerance distributions of both blue and green races was set to 50% 0.125 and 50% 0.833 to match the parameter values found in [105]. Density was allowed to vary uniform randomly in the range [0.5...0.9] in increments of 0.05. The probability of a happy agent relocating was set to 0.01. Neighborhoods were considered as the grid locations 1 hop away from a patch. The model was run for 100 ticks per simulation.

The genetic program parameters for EMD was set as follows. Mutation rate was set to 0.1, crossover rate to 0.8, minimum depth set to 2, maximum depth set to 6, and population size was set to 50 models per generation. The results over 5 genetic program runs were used for factor importance analysis through random forest regression.

Results

The results of running Evolutionary Model Discovery on Hatna's model of segregation are discussed in this section. Fig. 5.1 displays the marginal distributions of C-index produced by varying values of the presence of each factor. In this case, unlike that of the Artificial Anasazi Sec.4, or as later discussed in the case of notification prioritization Sec. 6, a more limited region of the factor space was explored by the genetic program. This is indicated by the fact that most factors only had 2 or 3 unique presence values which appeared at least in 100 samples of the data produced by the genetic program. Higher C-index is desired for the existence of mixed patterns. Preference for race and preference for isolation seem to have the largest impact on the C-index. Diversity of satisfaction and diversity of tolerance also seemed to have a slight effect on the C-index with the C-index maximal for $F_{SatDiv} = -1$ and $F_{TolDiv} = 1$. However, further statistical analysis was required to confirm these findings.

Table 5.1: Best 20 rules that produced the highest C-index, greatest mixing of segregated of integrated residential locations.

Rule	C-Index
$u_{a,i} = F_{Race}(i) - F_{Move}(i) + 2F_{TolDiv}(i)$	0.2999
$u_{a,i} = F_{Race}(i) - F_{Move}(i) + F_{Isol}(i) + F_{TolDiv}(i)$	0.2999
$u_{a,i} = F_{Race}(i) - F_{Move}(i) + F_{Isol}(i) + F_{TolDiv}(i)$	0.2999
$u_{a,i} = 2F_{Race}(i) - F_{Move}(i) + F_{Isol}(i) + F_{TolDiv}(i)$	0.2999
$u_{a,i} = 2F_{Race}(i) - 2F_{Move}(i) + 2F_{Isol}(i) + F_{TolDiv}(i)$	0.2999
$u_{a,i} = F_{Race}(i) - F_{Move}(i) + F_{Isol}(i) + F_{TolDiv}(i)$	0.2999
$u_{a,i} = 2F_{Race}(i) + F_{SatMean}(i) - 2F_{Move}(i) + 2F_{Isol}(i)$	0.2999
$u_{a,i} = F_{Race}(i) - F_{Move}(i) + F_{Isol}(i) + F_{TolDiv}(i)$	0.2999
$u_{a,i} = F_{Race}(i) - F_{Move}(i) + F_{Isol}(i) + F_{TolDiv}(i)$	0.2999
$u_{a,i} = F_{Race}(i) + F_{Isol}(i) + F_{TolDiv}(i)$	0.2999
$u_{a,i} = F_{Race}(i) - F_{Move}(i) + F_{Isol}(i) + F_{TolDiv}(i)$	0.2910
$u_{a,i} = F_{Race}(i) - 2F_{Move}(i) + F_{Isol}(i) + F_{TolDiv}(i)$	0.2903
$u_{a,i} = 2F_{Race}(i) - F_{Move}(i) + F_{Isol}(i)$	0.2887
$u_{a,i} = F_{Race}(i) - F_{Move}(i) + F_{Isol}(i) + F_{TolDiv}(i)$	0.2880
$u_{a,i} = F_{Race}(i) - F_{Move}(i) + F_{Isol}(i) + F_{TolDiv}(i)$	0.2876
$u_{a,i} = F_{Race}(i) - 2F_{Move}(i) + 2F_{Isol}(i) + F_{TolDiv}(i)$	0.2868
$u_{a,i} = 2F_{Race}(i) - F_{Move}(i) + F_{Isol}(i) + F_{TolDiv}(i)$	0.2868
$u_{a,i} = F_{Race}(i) - F_{Move}(i) + F_{Isol}(i) + F_{TolDiv}(i)$	0.2857
$u_{a,i} = F_{Race}(i) - 4F_{Move}(i) + 3F_{Isol}(i) + 2F_{TolDiv}(i)$	0.2849
$u_{a,i} = F_{Race}(i) - F_{Move}(i) + F_{Isol}(i) + F_{TolDiv}(i)$	0.2841

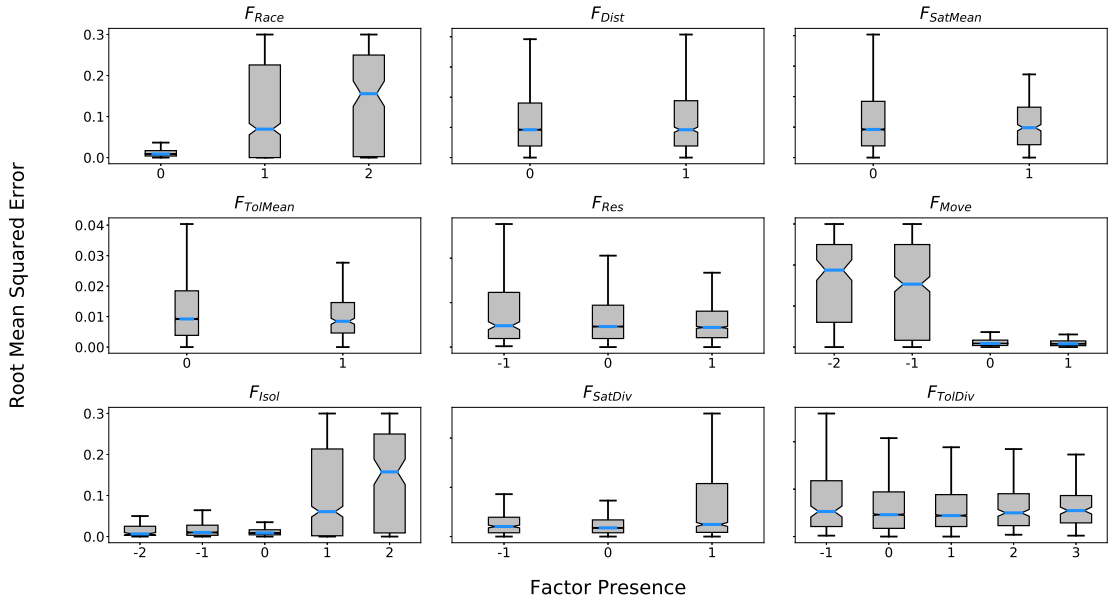


Figure 5.1: Marginal distributions of C-index with varying presence of the hypothesized factors driving mixed patterns of segregation (only presence values for which the genetic program produced at least 100 samples are considered). Higher C-index indicates . F_{Race} and F_{Isol} show the largest variation in C-index. Moderately negative (-1) F_{SatDiv} and positive F_{TolDiv} also show higher C-index. Other factors considered do not show any visible relationships and require statistical tests of significance.

The results of first order random forest importance evaluation on the data produced by the genetic program are displayed in Fig. 5.2. There was some disagreement between the two techniques, compared to Permutation importance, Gini importance underestimates the improtance of F_{TolDiv} and F_{Move}. However, both methods agree that F_{Race}, F_{Isol}, and F_{TolDiv} have are among the most important factors for mixed pattern generation. Permutation importance rated F_{TolDiv} as the most important factor and F_{Move} as the second most important factor, though a significant uncertainty was attributed to F_{Move}. F_{SatDiv}, F_{Res}, $F_{TolMean}$, $F_{SatMean}$, and F_{Dist} had relatively low importance.

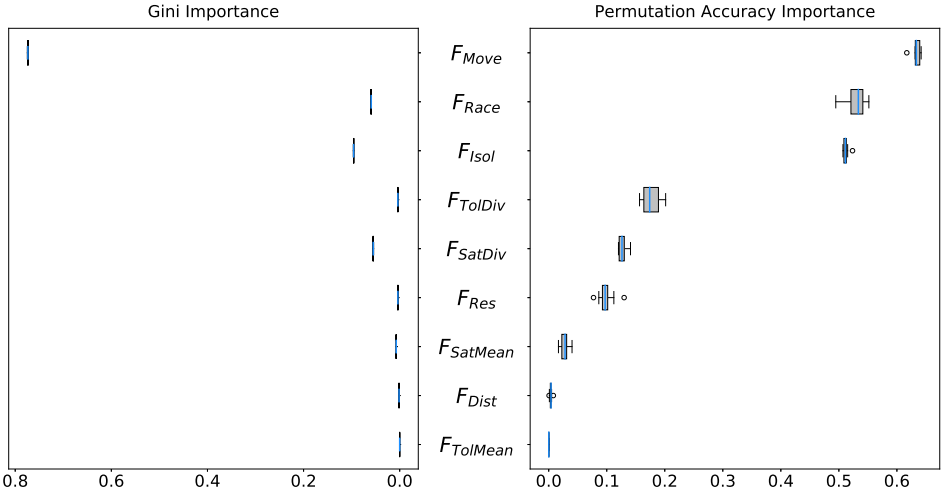


Figure 5.2: Gini and Permutation importance values for factors hypothesized to generate mixed patterns of segregation and integration. There is some disagreement between the two techniques, yet F_{Race}, F_{Isol}, and F_{TolDiv} are give high importance by both techniques. Permutation importance indicates that F_{Move} also has high importance but with high uncertainty on this measurement.

The p-values of Mann-Whitney U tests comparing the permutation importance of each factor are shown in Fig. 5.3 for the alternate hypothesis that $A > B$ for all factors A and B (null hypothesis $A = B$, $\alpha = 0.05$). Green cells indicated where there was no evidence for supporting the null hypothesis at a significance of $\alpha = 0.05$. Despite being unable to distinguish between the importance of F_{Race} and F_{Isol}, and F_{Res} and $F_{TolMean}$, there was a reasonable ordering of the importance of

the hypothesized causal factors; in descending order of importance: F_{TolDiv},F_{Move},F_{Race} and F_{Isol}, F_{SatDiv}, F_{Res} and $F_{TolMean}$, $F_{SatMean}$, and F_{Dist}.

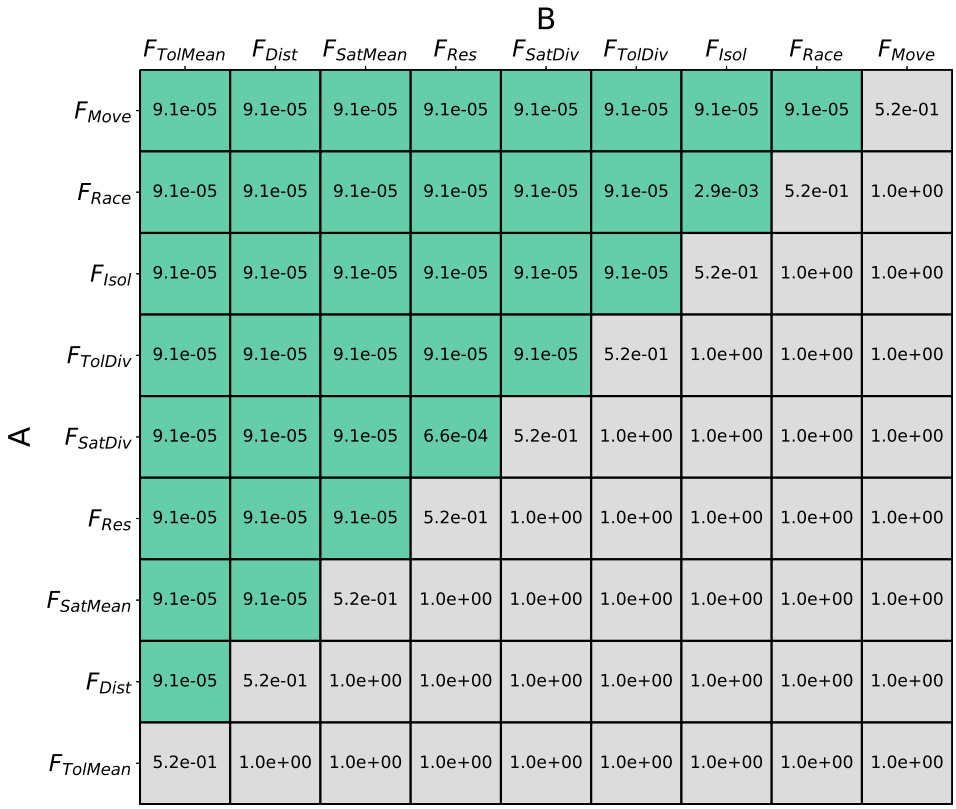


Figure 5.3: P-values of Mann-Whitney U tests comparing significance of difference in permutation importance of factors for mixed patterns (alternate hypothesis that $A > B$; null hypothesis $A = B$, $\alpha = 0.05$). Green cells indicate agreement of the alternate hypothesis. Results indicate an statistically confirmed ordering of factors by importance, except of the a lack of difference between F_{Race} and F_{Isol}, and F_{Res} and $F_{TolMean}$.

Joint contributions of interactions of three or less factors are shown in 5.4. Interestingly, the presence of the three factors F_{Race}, F_{Isol}, and F_{TolDiv} together provided the highest contribution to the predictions made by the random forest. F_{Isol} alone showed relatively modest importance as well.

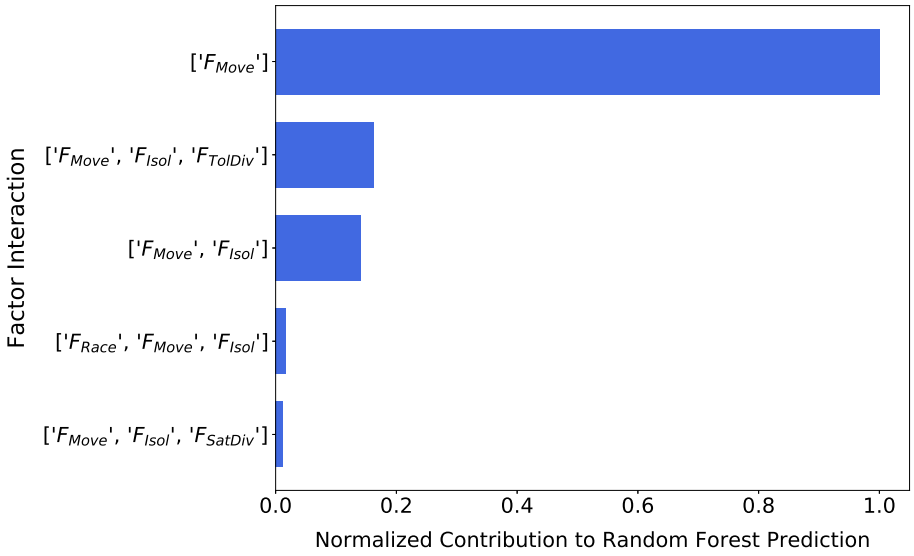


Figure 5.4: Joint contributions of factors towards the generation of mixed patterns of segregation and integration. The factor interaction $[F_{Race}, F_{Isol}, F_{TolDiv}]$ show the highest joint importance. Moderate importance is shown by F_{Isol} alone.

Considering the previous results the top five factors causing mixed patterns to form in Hatna's model of segregation were F_{Race}, F_{TolDiv}, F_{Isol}, F_{Move}, and F_{SatDiv}. Fig. 5.5 shows the p-values of conducting pairwise Mann-Whitney U tests for significant difference in C-index produced by the different presence values considered for each of these factors. The alternate hypothesis of this test was RMSE for presence $A <$ RMSE for presence B (null hypothesis: RMSE values are equal) and $\alpha = 0.05$. Green cells indicate instances were there was evidence for falsification of the null hypothesis. Mixed patterns were more likely to form when both F_{Race} and F_{Isol} were not present in the model, F_{TolDiv} was 1,, F_{SatDiv} was -1. There was insufficient evidence to make a conclusion on the best value for F_{Move}.

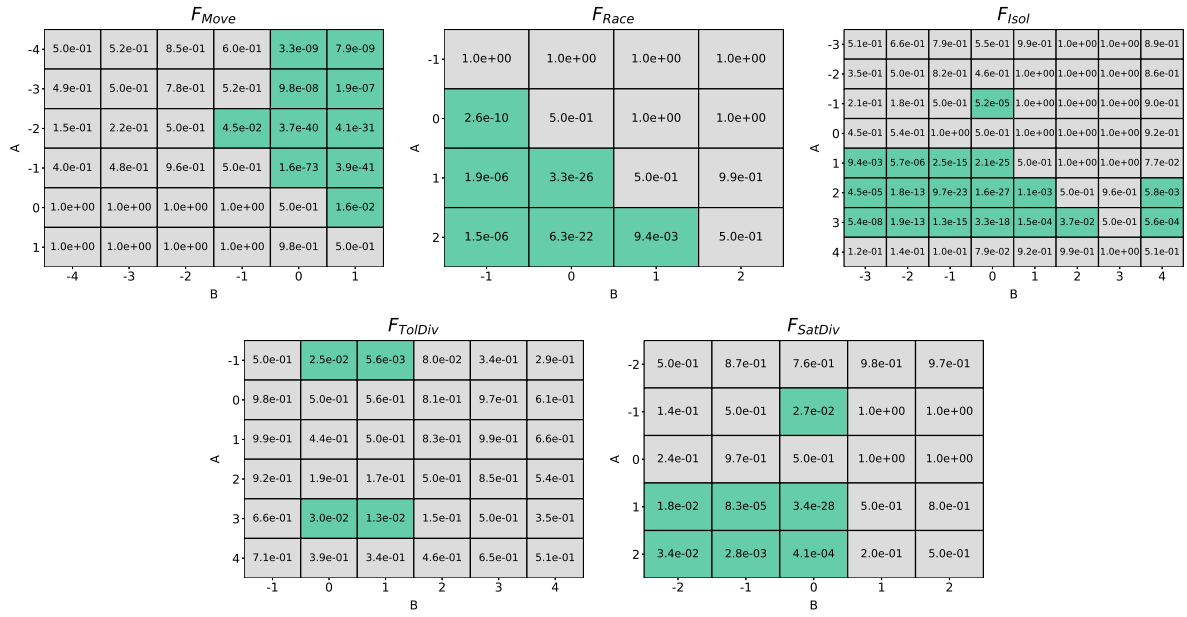


Figure 5.5: Optimal presence scores of the top 5 causal factors important to the generation of mixed patterns of segregation and integration. Shown above are p-values of Mann-Whitney U tests on pairwise comparisons of the marginal C-index produced user presence of A and B for the alternate hypothesis RMSE for presence $A <$ RMSE for presence B (null hypothesis: RMSE values are equal), $\alpha = 0.05$. Green squares indicate that the test showed was unable to falsify the null hypothesis. Results demonstrate that preference for diversity in tolerance and avoidance of areas with high variance in neighborhood satisfaction is important, while preference of race or isolation can inhibit emergence of mixed patterns.

Taking into account the results above, models including F_{Move}, F_{Isol}, F_{TolDiv}, and F_{SatDiv} with their optimal presence values were created and compared against the original utility function containing only F_{Race}. In particular, models containing the rules: $u_{a,i} = -2F_{Move}(i) + 3F_{Isol} + 3F_{TolDiv}(i)$, $u_{a,i} = -2F_{Move}(i) + 3F_{Isol}(i) + 2F_{Race}(i)$, and $u_{a,i} = -2F_{Move}(i) + 2F_{Race}(i) + 3F_{Isol}(i) + 3F_{TolDiv}(i) + 2F_{SatDiv}(i)$, where compared against the original rule $u_{a,i} = F_{Race}(i)$. Fig. 5.6 compares c-indices of 100 simulations of each model under random D in the range $[0.5, 0.9]$. The models created with insights from Evolutionary Model Discovery showed the ability to produce higher C-index values despite random parameter settings. The rule $u_{a,i} = -2F_{Move}(i) + 3F_{Isol}(i) + 2F_{Race}(i)$ in particular

was able to allow the model to generate significantly higher C-index values than the other rules.

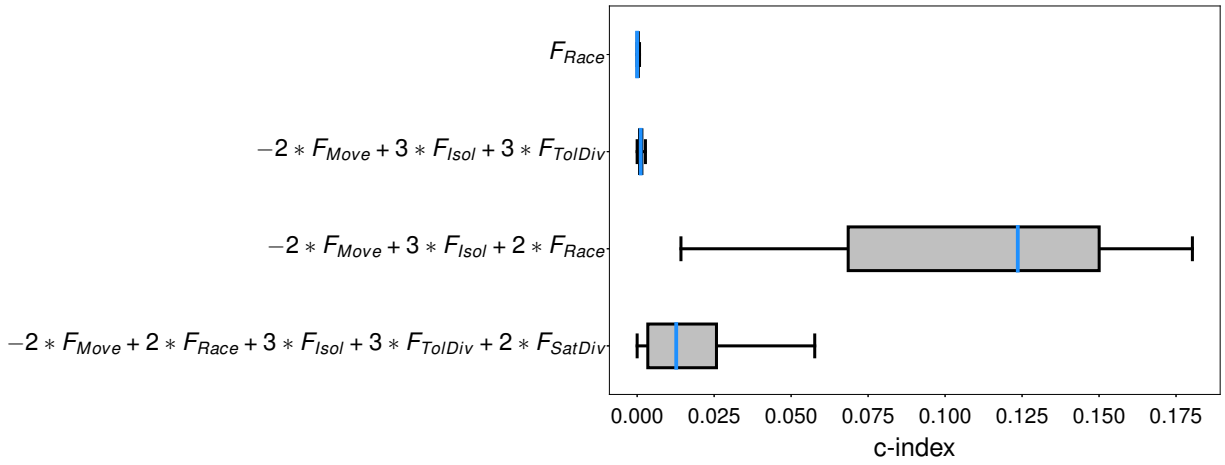


Figure 5.6: 100 Runs of Hatna-Benenson Segregation with residence utility rules inferred through Evolutionary Model Discovery with randomized parameter initialization. The rule $u_{a,i} = -2F_{Move}(i) + 3F_{Isol}(i) + 2F_{Race}(i)$ in particular was able to allow the model to generate significantly higher C-index values than the other rules.

Discussion

Mechanistic Explanation: *Mixed patterns of segregation and integration cannot be explained by individuals evaluating residential locations on a single factor alone, and instead is the result of individuals evaluating residential locations on an interplay between resistance to moving from their current location, preference for less crowded neighborhoods, and preference for higher racial similarity, prior to moving.*

In this chapter Evolutionary Model Discovery was applied to attempt to discover factors causing the emergence of mixed patterns of coexisting residential segregation and integration. Hatna and Benenson have shown that in reality urban communities are more likely to not either be completely segregated nor completely integrated, but have coexisting mixed patterns [105, 106]. Agents of

Hatna and Beneson's model of segregation embody the same rule for evaluating the desirability of a potential neighborhood patch for relocation as originally specified by Schelling [104]. Desirability of a potential residential location is solely estimated through racial similarity, i.e. locations where the number of neighbors of similar race are higher are preferred.

In this experiment I have tested multiple factors which I hypothesize to affect the desirability of a residential location, and as a result, generate the emergence of mixed patterns. Out of the tested factors, reluctance to move, similar racial preference, and desire for isolation/less crowding were important in the generation of mixed patterns. Additionally, acceptance of neighborhoods with diverse tolerance and conflicting satisfaction levels showed moderate importance. Models that are constructed with these insights consistently have higher C-indices than when only racial preference was considered. The residential desirability produced highest C-indices when reluctance to move, similar racial preference, and desire for isolation/less crowding were considered together when making the decision to move or not. Given that mixed patterns are common in urban environments, these findings are reflective of the thought processes of urban residents consisting of the above desires.

The ease of visualizing the macro-state of this model, i.e. the patterns of segregation, allowed for closer examination, exposing a narrative-like explanation of the contribution of the more important factors towards the emergence of mixed patterns. Fig. 5.7, shows how incrementally adding factors to the residential location desirability function, affects the macro-state in surprising ways. F_{Move} and F_{Isol} by themselves produce integrated, unsettled patterns, with low C-indices. Including both factors at optimal presence, i.e. $u_{a,i} = -2F_{Move}(i) + 3F_{Isol}(i)$, produces a tight unstable, and constantly shifting mass of agents, compressed by a bubble of isolation, highly integrated with even lower C-index. F_{Race} by itself, i.e. the original rule $u_{a,i} = F_{Race}(i)$, produces static segregated neighborhoods surrounded by boundaries, as is already known. However, when F_{Race} is added to $u_{a,i} = -2F_{Move}(i) + 3F_{Isol}(i)$ with its optimal presence to give the rule

$u_{a,i} = -2F_{Move}(i) + 3F_{Isol}(i) + 2F_{Race}(i)$, neither a bubble of isolation or segregated patterns occur. Instead, the boundaries disappear and communities with areas of segregation and some regions of integration, i.e. mixed patterns, quickly emerge and settle into a stable, static state, with far higher C-index. This demonstrates how there may be instances where desired emergent patterns might only be generated when multiple causal factors are considered in the decision-making process.

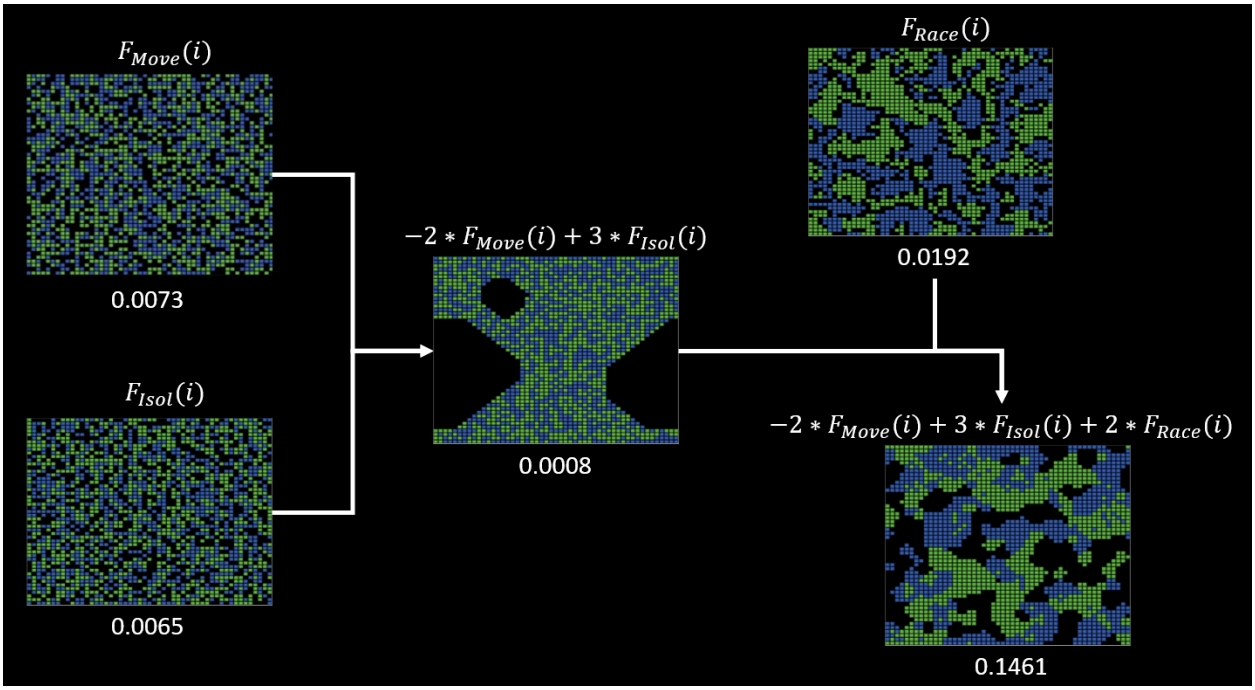


Figure 5.7: A narrative-like explanation of the emergence of mixed patterns (density = 0.75). Noisy integration, tight integration, and segregation with borders can result from considering reluctance to move, desire for less crowding, and preference for racial similarity separately. Yet, when considered within the same rule, stable, static mixed patterns easily emerge.

CHAPTER 6: CASE STUDY 3: PRIORITIZATION OF RESPONSES UNDER INFORMATION OVERLOAD ON ONLINE SOCIAL MEDIA

What drives compulsive information sharing by highly active social media users? Agent-based diffusion of information models [107, 108, 109, 110] have been shedding light to our understanding of the dynamics of information cascades throughout the literature. Understanding the factors that cause information to go viral online is of high interest to policy makers as this information can be easily used for the successful dissemination of polarizing propaganda and (dis)information. The users of online social media are (mostly; many automated bot accounts exist on social media sites) human and are subject to limits of cognition and attention span.

A Theory of Extended Working Memory and Implications of Information Overload

Due to the biological limits of human cognition, humans are susceptible to information overload. Extensive research in lab experiments from psychology and neuro-science have shown that humans posses a working memory [111, 112], in addition to their long-term memory, which functions as a temporary storage within which individuals keep information for immediate processing. The information retrieved from long-term memory is recalled and stored in working memory as chunks of information [113]. However, under information overload, these chunks are reduced to single units of information [111]. The number of units of information that can be held in working memory has been shown to be limited due to the limit of cognitive processing. This limit was originally estimated to be 7 +/- 2 [114] but in more recent studies shown to be around 4 units of information [115, 116].

The theory of extend self posits that humans have a tendency to depend on familiar physical and

virtual objects as a means to re-embody oneself, including their memories and identity [117]. Examples of extended self can range from a depending on instructions written in a personal journal for future reference [118], to online social media profiles [119, 120]. In recent years, increasing numbers of individuals have become hooked to the use of social media, not only as their main form of communication, but to re-embody themselves, re-defining themselves with doctored selfies, catchy taglines, and targeted shares of popular opinion. Motivations for this behavior include the removal of inhibitions and freedom of expression through anonymity [121, 122, 123], fear of missing out (or FOMO) [124], and in some cases due to being the sole means of self expression for oppressed individuals [125].

Considering the premise that online social media allows for the formation of an extended self, the processes of information overload and memory capacity should reflect in one's extended memory and cognition. Extended working memory can be defined as the information stored in a technological scaffold, such the notification feeds of a social media profile, that are accessed in order to perform tasks. The extended working memory capacity can then be defined as the number of messages in the extended working memory that individuals are able to respond to, without actively searching through their notification feed (Actively searching through one's notification feed could be considered as accessing one's long-term memory). However, unlike working memory, the attention associated with extended working memory may be more volatile, depending on the cognition devoted to the social media profile, i.e., the extended self, by the original self. Similar to working memory, large in-flows of distracting information on the notification feeds may cause information overload in an individual. Unlike one's biological working self, this may lead to an individual 'detaching' from their extended self. That is, the overload experienced through an individual's extended social media self may force a loss in the attention span an individual dedicates to their incoming notifications, allowing more notifications to go unanswered. I refer to attention span as M_t, as it may, therefore, vary with time, t. I model the suppression of attention as having a power-

law relationship to the information overload experienced by the user, with an exponent, α, which I refer to as the rate of attention suppression. Of course, there would be an upper limit to extended working memory, i.e., extended working memory capacity (M_{max}), determined by both properties of the social media platform and biological limits, which would be observed under unoverloaded conditions.

In the following sections I discuss how this theory of extended working memory is modeled and combined with a model of conversation in order to model information dynamics of conversation participants experiencing information overload through a heterogeneous network of influence.

The Multi-Action Cascade Model of conversation

The (Multi-Action Cascade Model) MACM [126] derives from traditional diffusion of information models such as the independent cascade model [107, 108, 109, 110], but it is unique, as it is the first of its kind to simulate diffusion of information in the form of conversations, following the principles of conversation theory [127], instead of merely simulating the binary adoption of a topic or opinion. The MACM is based on four premises:

- **Premise 1**: Diffusion of information over online social media occurs through conversations. Individuals participate in conversations due to the following factors: 1) influence of other participants, 2) influence from information sources exogenous to the conversation, or 3) the internal need to participate in conversation.
- **Premise 2**: Conversation participants can perform three types of actions: 1) Initiation of a new conversation, 2) contribution to an existing conversation, 3) sharing existing information from a conversation.

- **Premise 3**: Given a particular topic of interest, the influences $q \in Q$,$p \in P$, and $i \in I$ can be determined from event timeseries data, by measuring the ratio of information flow from the timeseries of events of the influencing user's action type to the timeseries of events of the influenced user's action type [128, 126].

MACM agents exist on a network of endogenous influence probabilities that govern the probability that an agent's neighbor takes a particular action (out of the actions listed in Premise 2) provided information that the agent has itself performed an action. When a MACM agent performs a particular action, they produce a message, representative of a social media notification, that indicates which user performed the action, the action type, and the conversation the action is being performed on. These messages are propagated to neighboring agents to which the acting agent has an influence probability greater than 0 over the receiving agent performing any given action type. The receiving agents then act on the incoming messages according with a probability indicated by the influence probability the sender agent has over it, q. MACM messages are thus akin to information chunks, as referred to in the literature on cognitive models of short-term memory [114, 113]. Once an MACM agent receives a message, it can then decide to initiate a new conversation on the topic, contribute to the sender's conversation, or share the sender's conversation with other agents. For the experiments in this paper, q between the agents in the simulations presented in this paper are derived by considering diadic relationships between social media users, and calculating the ratios of information flow over time from one user, as the influencer, to another, as the influenced, to the total information produced by the influencer agents as described in [126]. p and i are established similarly, but for the purposes of the experiments in this paper, considering the endogenous influence probabilities q is sufficient. The GPU-based implementation of the MACM in python can be found in [129].

Modeling Extended Working Memory

We integrate the extended working memory sub-model into MACM based on a further three premises:

- **Premise 4**: Information exists in extended working memory in the form of chunks and the number of chunks can be recalled for immediate processing is quantified by the current attention span, M_t, of one's extended self [114, 130, 115, 113].
- **Premise 5**: Attention span, and effectively responsiveness, to messages has an upper bound due to cognitive limits M_{max} [115, 131].
- **Premise 6**: Extended selves' experience the effects of information overload once the number of incoming messages has exceeded their extended working memory capacity M_{max}, after which the suppression of M_t follows a power-law relationship of exponent α with the amount of incoming information overload experienced [131, 132, 133]

Fig. 6.1 summarizes our model of extended working memory and the reduction of attention span under information overload. Individuals modeled in the MACM are often subject to receiving more than a single message every time step. Often, the influence probability of the sender is insufficient for the users to act on the particular piece of information within the same time step it was received. Accordingly, all MACM agents have an actionable information queue which buffers incoming messages until they are processed and removed in a last-in-first-out manner. If new information is received by an agent while the actionable information queue is full, then the oldest messages are pushed out of the queue in a first-in-first-out manner to make room for the incoming messages.

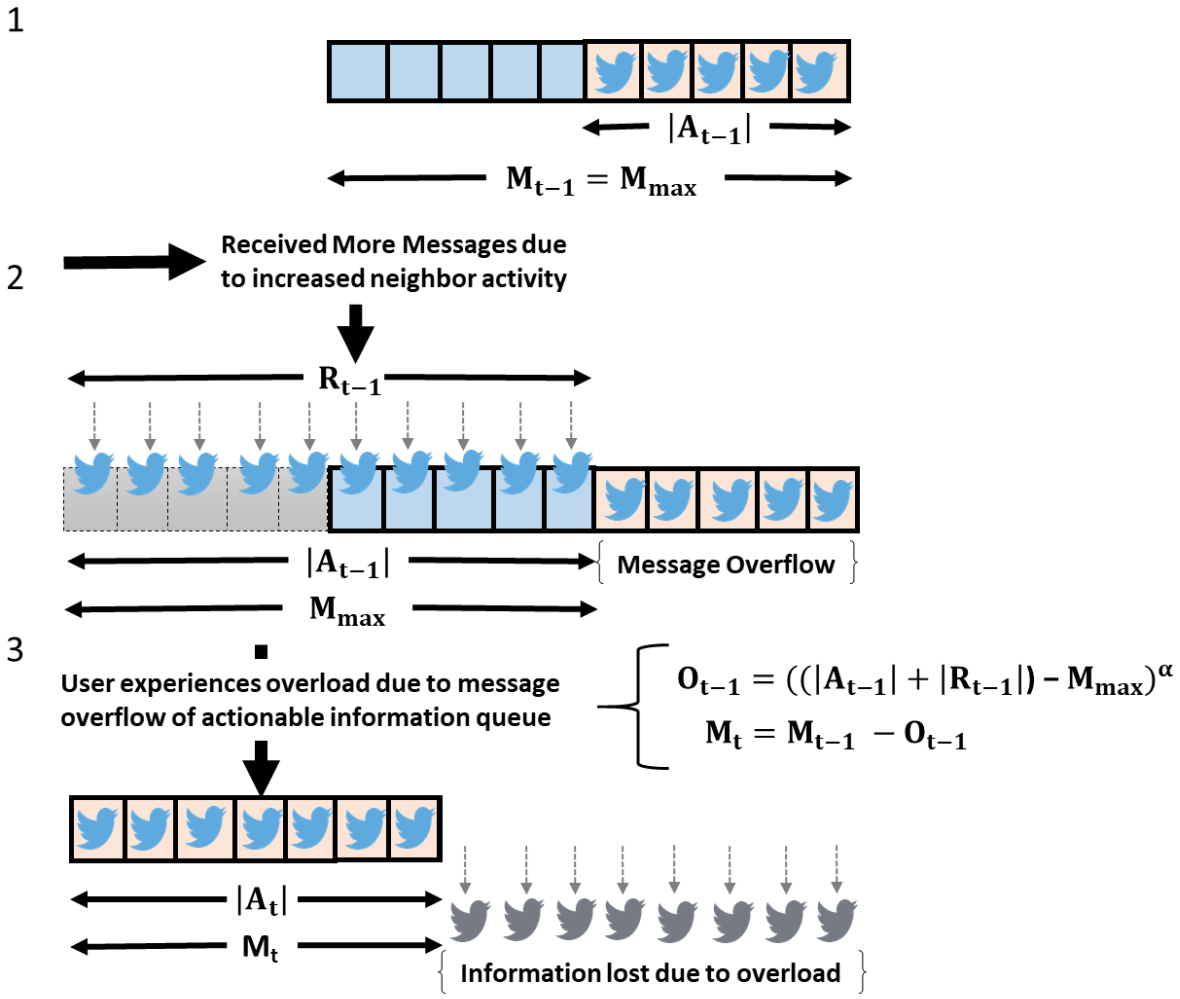


Figure 6.1: An illustrated demonstration of the actionable information queue and the process of information overloading. Step 1) Actionable information stores incoming information, is accessed in a last-in-first-out fashion. Its capacity is synonymous to the individual's current attention span, M_t. 2) Received information is added to the front of the actionable information. A user is overloaded if M_t is exceeded, in which case a new M_t is calculated based on the extent of overload experienced. 3) Excess messages are dropped in a first-in-first-out fashion, removing the oldest messages first.

In other words, when applied to a social media context, the current actionable information queue capacity at a given time t, (M_t), represents the attention span of a conversation participant to incoming notifications. In order to simulate information overload, I allow (M_t) to varying according

to the rate of information inflow to the agent. At every time step, the current attention span (M_t) of each individual is re-calculated based on the overload (O_{t-1}) experienced by the user due to excessive information in-flow during the previous time step raised to the power of a α ($0 <= \alpha <= 1$) as shown in eq.6.1. A parameter of the model, α represents the power-law in [131], which represents an agent's susceptibility to loss in responsiveness under information overload.

We model the impact of information overload on attention span to notifications by deriving the overloading mechanism discovered between reduced user responsiveness under information overload on Twitter described in [131]. The overload experienced by an individual at time t, O_{t-1} is calculated as the number of excess messages, beyond the current attention span of the user, as shown in eq. 6.2, i.e. by how many messages does the sum of number of messages that were received from the previous time step (R_{t-1}) and the number of messages left over on the actionable information queue from the previous time step ($|A_{t-1}|$) exceed the extended working memory capacity, M_{max}. O_{t-1} has a lower limit of 0. [131] estimate the value of M_{max} to be 30.

$$M_t = \begin{cases} M_{t-1} - O_{t-1}^{\alpha}, & \text{if } O_{t-1}^{\alpha} <= M_{t-1} \\ 0, & \text{otherwise} \end{cases} \tag{6.1}$$

$$O_{t-1} = \begin{cases} (|A_{t-1}| + |R_{t-1}|) - M_{max}, & \text{if } |A_{t-1}| + |R_{t-1}| >= M_{max} \\ 0, & \text{otherwise} \end{cases} \tag{6.2}$$

If $M_t < |A_{t-1}| + |R_{t-1}|$, the oldest messages in the actionable information queue are removed during at t until $|A_t| = M_t$.

Causal Factors for Notification Response Prioritization

The original model of extended working memory assumes that when experiencing information overload, individuals tend to focus their attention towards information that they have received more recently. In other words, notifications that are received more recently are given higher priority for response. However, it is necessary to test whether there is sufficient evidence that message recency is in fact the factor that drives message prioritization. There could exist other factors, pertaining to qualities of the influencer or the information content that the notification displays that affect this prioritization process.

In order to test the importance of message recency, I propose seven other factors that may possibly affect the prioritization of notifications for response. All eight hypothesized factors are listed below:

- **Conversation popularity** $F_{ConvPop}$: Conversation popularity represents the global popularity of a particular information cascade. It is measured by the normalized number of users that have responded to the conversation created by the root message that was created by the original poster.
- **Conversation size** $F_{ConvSize}$: Conversation size represents the global volume of a particular information cascade. It is measured by the normalized number of responses that have accumulated to the root message that was created by the original poster.
- **Initiators popularity** $F_{InitPop}$: Initiator's popularity represents the global popularity of the conversation initiator. This is measured as the number of times messages by this individual had been responded too by any individual, normalized by the corresponding maximum in the data.

- **User common interests** F_{Intr}: User common interests measures the number of times the individual has participated in a conversation which the other user has participated in.
- **User reciprocity** F_{Recip}: Absolute reciprocity (both positive and negative) between two users. This was measured as the number of times the individual has responded to an another individual, normalized by the corresponding maximum in the data.
- **URL domain popularity** F_{URLPop}: URL domain popularity represents the global popularity of any URL domains that were mentioned by all users. It is measured as the normalized count of messages that have a reference to the URL domain.
- **URL Domain familiarity** F_{URLFam}: URL domain familiarity represents the local popularity of any URL domains, measured as the normalized number of references this individual has made to the URL domain in their past messages.
- **Information expertise** F_{Info}: Information expertise represents how often this user mentions a particular piece of information, normalized by the corresponding maximum in the data.
- **Recency** F_{Recn}: Recency was measured as the reciprocal of the amount of time that had passed since the message was originally received by the individual.

Unlike the previous two cases, $\times$ and $\div$ were also included in the genetic program, in addition to $+$ and $-$, which allowed for the evolution of more complicate factor interactions.

Experimental Setup

Evolutionary Model Discovery was run on the model of extended working memory and the multi-action model of conversation. Data used for this experiment was obtained through the DARPA

SocialSim program [1]. This data consisting of timestamped events of user profiles engaged in discussion and development related to cyber vulnerabilities and exploits over three social media platforms, GitHub, Reddit, and Twitter. Each event in the dataset consisted of the following information: 1) time of event, 2) anonymized user identifier, 2) the anonymized event identifier of the immediate parent event to which this event responds to if any, 3) keywords identifying the information being discussed by the individual, and 4) domains of any external URLs referred to in the event. The training period was from February 1st 2017 to April 1st 2017, while the simulation period was from April 1st 2017 to May 1st 2017. The eight most responsive Twitter users in terms of their retweet count during the training period were considered as influencees for the experiment. The influencers of these influencees and their hourly probabilities of influence per action-action relationship per influencee were measured using pairwise measurements of marginal transfer entropy on the training data according to the MACM [126]. The GPU implementation of MACM initialization in [129] was used for this purpose. The event data throughout training and simulation period were used to extract data related to users, conversations, URL domains, and topics that were required to calculate the measures required for the eight factors described in section 6. The events performed by the influencers during the simulation period were extracted from the data and loaded in as MACM messages into the model, while the events performed as responses to these messages were simulated through MACM and EWM.

The MACM and EWM were implemented in NetLogo. A NetLogo procedure for the calculation of message utility was included as an extension to the EWM model. This routine was tagged as the behavior for evolution and evaluation through EMD with implementations of the factors described in section 6. The time resolution of the model was set to hours to match the probabilities extracted during the initialization process and was run for 720 ticks simulating the entire month of April,

[1] All user information was completely anonymized in compliance to IRB standards. For more details please see Appendix A

2017.

The following metric was used to measure how closely the simulation matched the real world data in terms of user responsiveness to messages (equation 6.3). The real-world events performed by the eight influencees in response to the influencers during the period for simulation were isolated. The number of responses per influencer message, in the real-world data, were then measured. The same was performed on the messages produced through the simulations. The model fitness was calculated as the squared root of the sum of the squared errors between the total number of responses by the selected user U in the data $R_U^{real}(m)$ for each message received through their influencers, $m \in m_{all}$, against that of the simulation $R_U^{sim}(m)$

$$RMSE = \sqrt{\frac{1}{|m_{all}|} \sum_{m \in m_{all}} (R_U^{real}(m) - R_U^{sim}(m))^2} \tag{6.3}$$

Parameters of the genetic program for EMD were set as follows. Mutation rate was set to 0.1, crossover rate set to 0.8, tree depth was set between 2 and 10, and population size of 50 individuals were used, with no fitness caching. Each model was run for 720 time steps. The genetic program was run for 50 generations.

Results

In this section the results of applying Evolutionary Model Discovery to identify factors driving the prioritization of Twitter notifications for response among the selected cryptocurrency interest-community are discussed. Fig. 6.2 displays the marginal distributions of the RMSE error of responsiveness under varying values of presence for all nine hypothesized factors. Only factors presence values for which at least 100 samples were present in the data produced by the genetic

program have been displayed and have been considered throughout the rest of the analysis. Merely through observation of the marginal distributions of RMSE, it can be observed that URL related factors, F_{URLFam} and F_{URLPop} were best when absent and very limited exploration of these factors were performed by the genetic program. The other factors had at least five values of factor presence for which at least 100 instances were generated through the genetic program. The RMSE under negative presence of $F_{InitPop}$ was seen to be much lower than when this factor was present positively. In contrast, the RMSE under positive presence of F_{Recn} was seen to be much lower than when this factor was present negatively. Relationships between the marginal RMSE and presence for the other factors were difficult to distinguish visually and required further statistical testing, as performed below.

Table 6.1: Best 20 rules that produced the lowest RMSE in Responsiveness to the real-world data.

Rule	RMSE
$u_a = -F_{InitPop} + F_{Recn}$	7.6597
$u_a = -F_{InitPop}$	7.6669
$u_a = -4F_{InitPop} + 3F_{Recn}$	7.6674
$u_a = F_{Recn}$	7.6706
$u_a = F_{Recn} - 1F_{Intr}$	7.6750
$u_a = -2F_{InitPop} + F_{Recn} + 2F_{Recip}$	7.6751
$u_a = F_{ConvPop} + 2F_{Recn}$	7.6751
$u_a = 2F_{Info} - 4F_{InitPop} + 2F_{Recn}$	7.6758
$u_a = -2F_{InitPop}$	7.6761
$u_a = F_{Recn}$	7.6765
$u_a = -F_{InitPop} + F_{Recn}$	7.6768
$u_a = -2F_{InitPop} + 3F_{Recn}$	7.6769
$u_a = [{}'F'_{InitPop}, {}'F'_{Recn}]$	7.6776
$u_a = -F_{InitPop} + F_{Recn}$	7.6783
$u_a = -2F_{InitPop} + F_{Recn} + F_{Intr}$	7.6783
$u_a = [{}'F'_{InitPop}, {}'F'_{Recn}]$	7.6786
$u_a = -F_{InitPop} + F_{Recn}$	7.6787
$u_a = -F_{InitPop} + F_{Recn}$	7.6792
$u_a = -2F_{InitPop} - 1F_{Intr}$	7.6798
$u_a = -4F_{InitPop} + 2F_{Recip}$	7.6805

The first-order gini and permutation importance of the top 10 factors and factor interactions found by training the random forest on the factor presence data are shown in Fig. 6.3. Unlike the case of mixed patterns of segregation in Sec 5, there was a considerable agreement between the results of both techniques. F_{Recn} was by far the most important factor when predicting the RMSE of the models generated by the genetic program, and $F_{InitPop}$ followed with high importance values under both techniques. The interaction between these two factors $[F_{Recn}, F_{InitPop}]$ followed with very similar importance to $F_{InitPop}$. Other factor and factor interactions showed relatively lower importance; F_{Intr} was next with more than 8 times less than that of F_{Recn}, followed by the interactions $[F_{URLFam}, F_{InitPop}, F_{Recn}]$, $[F_{Intr}, F_{Recip}]$, and $[F_{ConvSize}, F_{InitPop}, F_{Recn}]$. $F_{ConvSize}$ followed by $F_{ConvPop}$, was next in terms of decreasing importance, followed by the interaction $[F_{InitPop}, F_{Recn}, F_{Intr}]$. F_{Info} and F_{URLPop} were not among the 10 factors and factors interactions with highest importance.

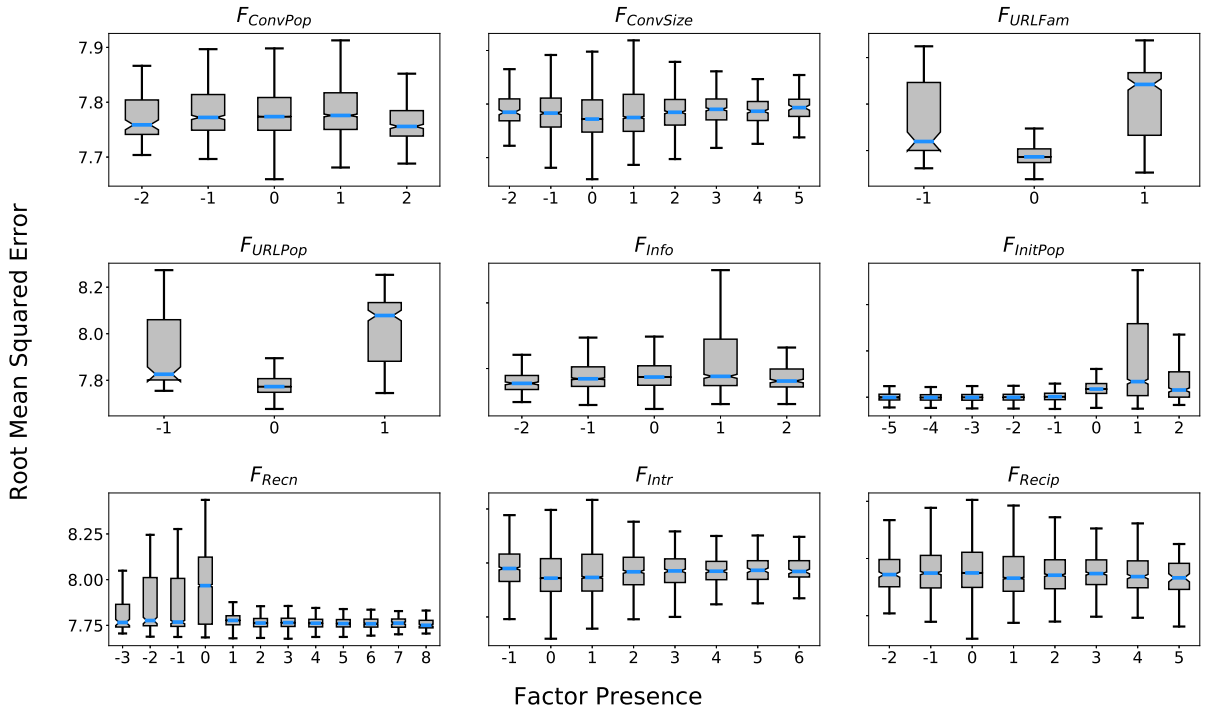


Figure 6.2: Marginal RMSE distributions for factors hypothesized to affect response prioritization to notifications under information overload. RMSE is generally lower with positive presence of F_{Recn} and negative presence of $F_{InitPop}$. RMSE is lowest when factors considering URLs present in the content, F_{URLFam} and F_{URLPop}, are not considered. Slight correlations are indicated by F_{Info} and F_{Recip} but further statistical tests for significance are required and have been performed below.

Pairwise Mann-Whitney U tests helped further confirm this ordering of factor importance. Mann-Whitney U tests were conducted for the alternate hypothesis: permutation importance of a factor A > permutation importance of a factor B (null hypothesis permutation importance of a factor A = permutation importance of a factor B) for a significance level of 0.05. Fig. 6.4 shows the p-values for these pairwise tests. The results confirm that there clear ordering to the importance of the hypothesized factors; in descending order of importance: F_{Recn}, $F_{InitPop}$, F_{Intr}, $[F_{URLFam}, F_{InitPop}, F_{Recn}]$, $[F_{Intr}, F_{Recip}]$, $[F_{ConvSize}, F_{InitPop}, F_{Recn}]$, $F_{ConvSize}$, $F_{ConvPop}$, $[F_{InitPop}, F_{Recn}, F_{Intr}]$

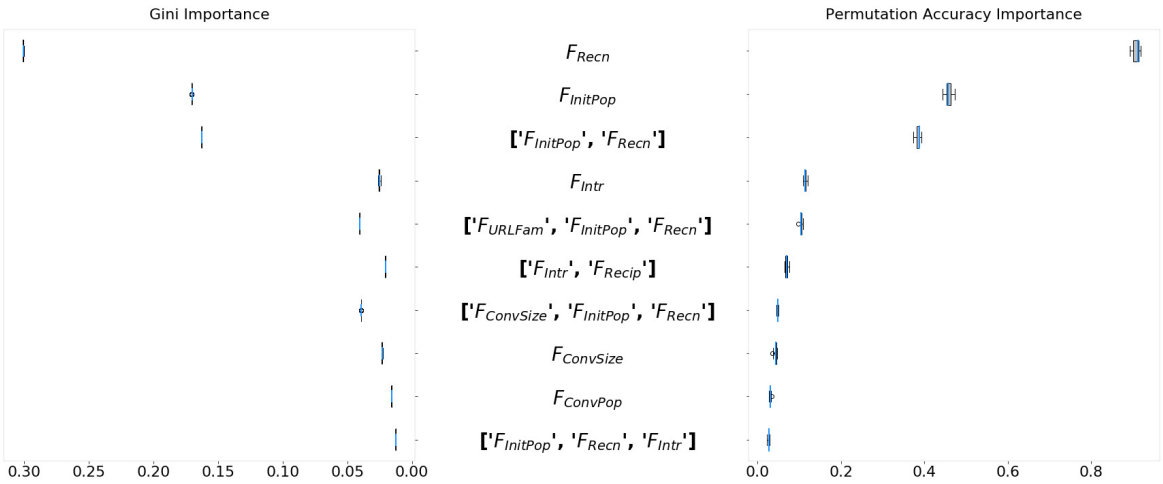


Figure 6.3: Comparison of Gini and Permuation Accuracy importance of factors hypothesized to drive response prioritization under information overload. A general order of importance is agreed on by both techniques, with the exception of the importance of the interactions $[F_{URLFam}, F_{InitPop}, F_{Recn}]$ and $[F_{ConvSize}, F_{InitPop}, F_{Recn}]$.

The joint contributions of interactions of three or less factors are displayed in Fig. 6.5. Again F_{Recn} by itself shows the highest importance, even greater than interactions of other factors. Interestingly, F_{URLPop}, when considered with the two factors of highest first-order importance F_{Recn} and $F_{InitPop}$, showed moderate importance. Interactions of other factors with these two factors followed with relatively lower joint contribution to prediction of the fitness data.

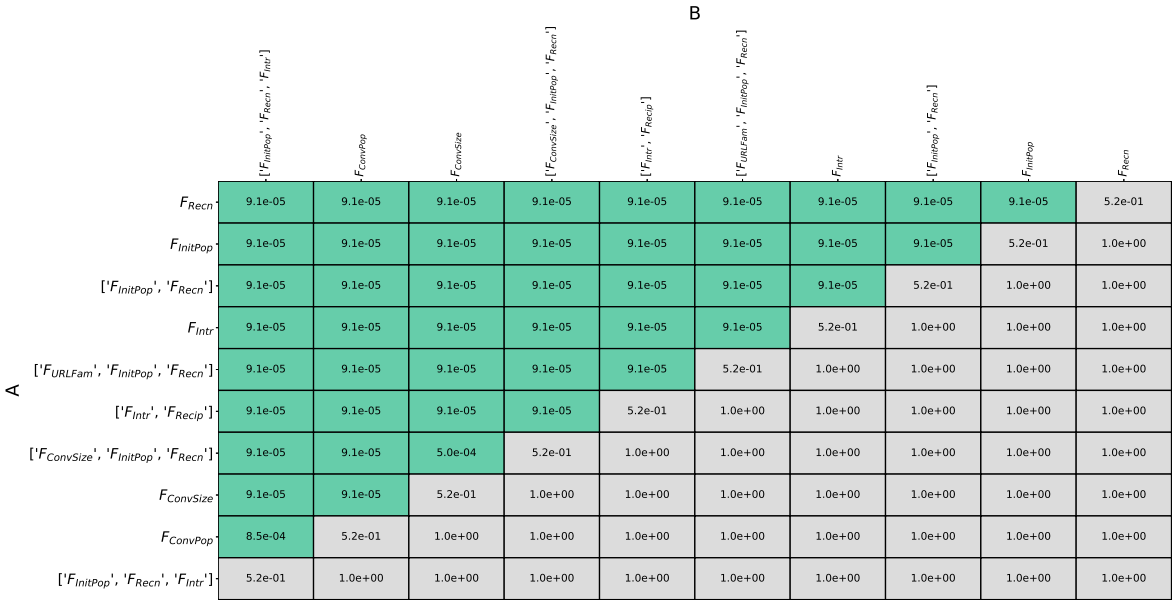


Figure 6.4: P-values of systematic Mann-Whitney U-test on the first-order permutation accuracy importance of factors hypothesized to drive the prioritization of response under information overload. The cells contain p-values for one-tailed Mann-Whitney U tests of the alternate hypothesis that permutation importance of $A >$ permutation importance of B (null hypothesis: permutation importance of $A =$ permutation importance of B) at $\alpha = 0.05$. Green cells indicate agreement of the alternate hypothesis. A clear order of descending importance is confirmed F_{Recn}, $F_{InitPop}$, $F_{ConvSize}$, F_{Recip}, F_{Intr}, F_{Info}, $F_{ConvoPop}$, F_{URLPop}, and F_{URLFam}.

Given the factor importance results, F_{Recn}, $F_{InitPop}$, $F_{ConvSize}$, F_{URLPop}, and F_{Recip} are considered the top five important factors for the generation of mixed patterns of segregation and integration. The optimal values for these factors are determined through pairwise Mann-Whitney U tests of the presence values considered for each of these factors. The alternate hypothesis of these tests were RMSE for presence A < RMSE for presence B (null hypothesis: RMSE values are equal) for $\alpha = 0.05$. The p-values for these tests are displayed in Fig. 6.6. Green cells indicate instances where there was evidence for the falsification of the null hypothesis. The results indicate that positive F_{Recn}, negative $F_{InitPop}$ were important for the prioritization of notifications for response under overload. The optimal values for $F_{ConvSize}$ were centered around 0, and the optimal value for

F_{URLPop} was 0. F_{Recip} was generally better with positive presence, particularly at values of 1, 2, and 4.

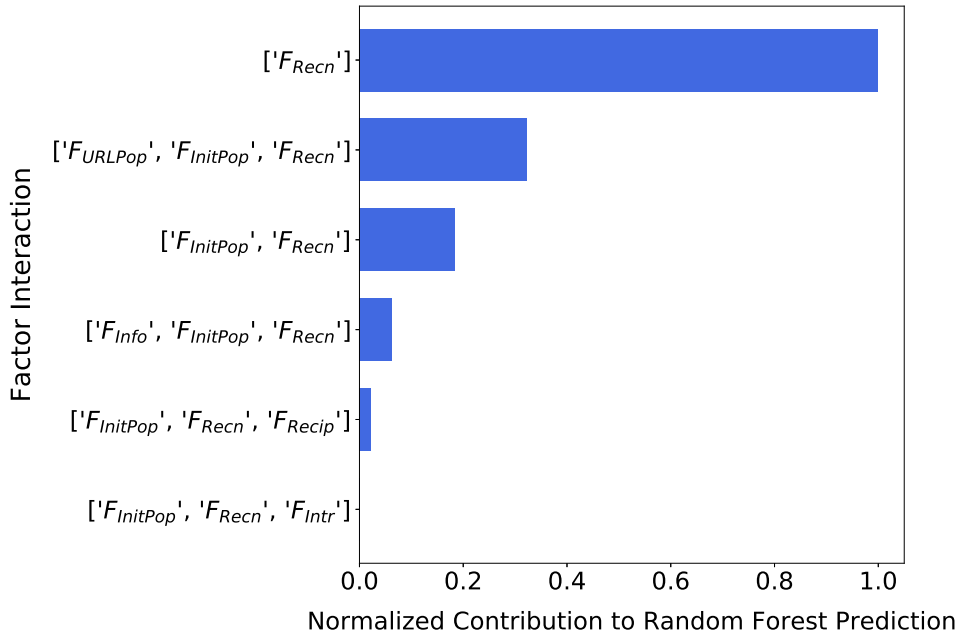


Figure 6.5: Joint contributions of interactions of three or less factors. Still, F_{Recn} on its own by far has the highest joint contribution when predicting model fitness. The interaction of three factors $[F_{Recn}, F_{InitPop}, F_{URLPop}]$ has the second highest joint contribution, despite F_{URLPop} by itself being among the least important factors. The interaction of the two factors with the highest first-order importance $[F_{Recn}, F_{InitPop}]$ has the third highest joint contribution.

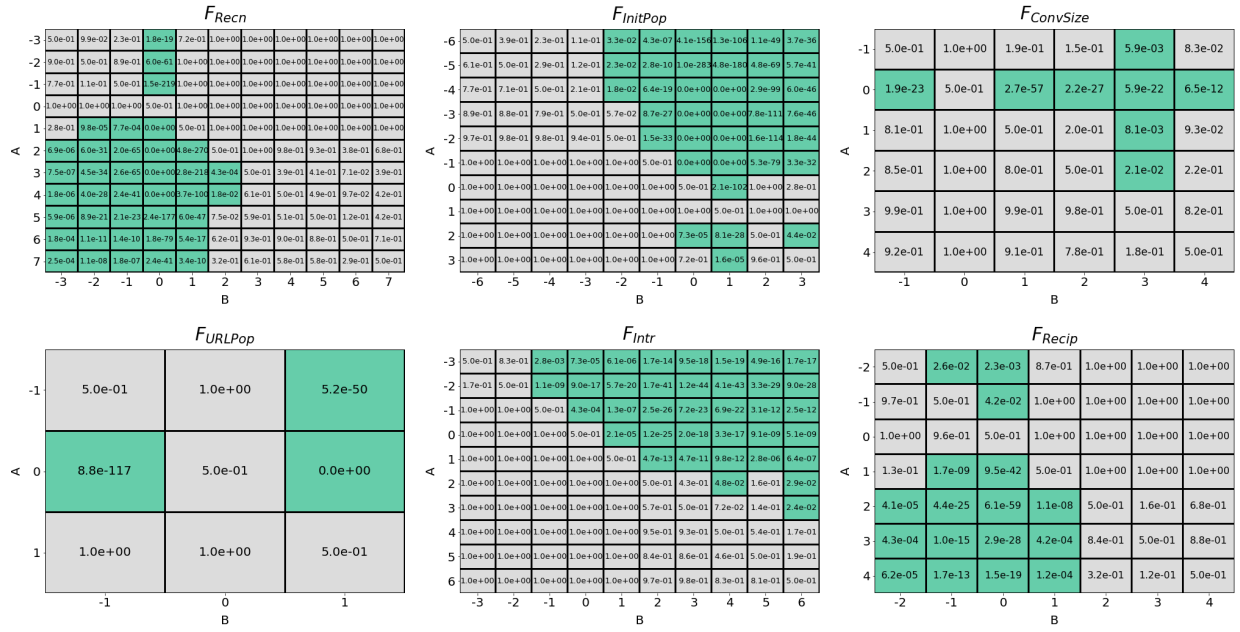


Figure 6.6: Optimal presence scores for causal factors with highest importance. P-values of systematic one-tailed Mann-Whitney U tests between presence values of the 5 most important factors for the alternate hypothesis: RMSE for presence $A <$ RMSE for presence B (null hypothesis: RMSE for presence $A =$ RMSE yfor presence B) for significance level 0.05. Green cells indicate agreement of the alternate hypothesis. Results indicate that for positive presence of F_{Recn} and F_{Recip}, and negative presence of $F_{InitPop}$, RMSE is generally higher than when considered negatively. RMSE is generally highest when $F_{ConvSize}$ and F_{URLPop} are not present in the prioritization process at presence 0.

Models with response prioritization rules derived from the results above are constructed with F_{Recn}, $F_{InitPop}$, $F_{ConvSize}$, F_{URLPop}, and F_{Recip} and compared against the original prioritization rule F_{Recn}. In particular, the rules $u_a = 4F_{Recn} - 3F_{InitPop} - 2F_{Intr} + 2F_{Recip}$ and $u_a = 4F_{Recn} - 3F_{InitPop} + 2F_{Recip}$ were tested against $u_a = F_{Recn}$. As the parameters of the model M_{max} and α were know through analysis calibration [134], the values 30 and 0.8 were used, respectively. As shown in Fig. 6.7, models where F_{Recen} interacts with $F_{InitPop}$, F_{Intr}, F_{Recip} have lower RMSE in responsiveness see in the actual data.

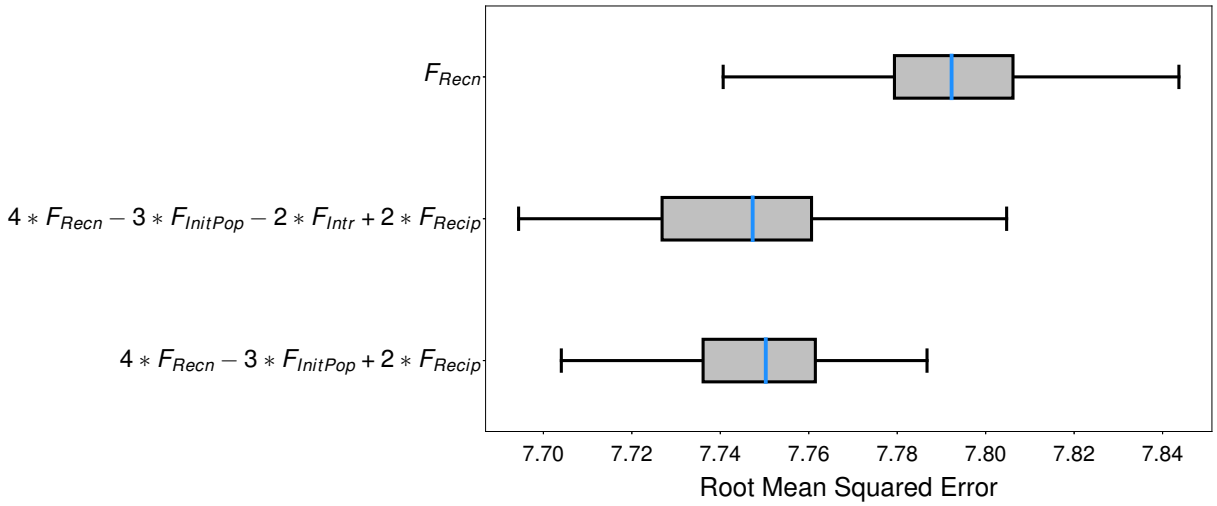


Figure 6.7: 100 Runs of MACM-EWM with response prioritization rules inferred through Evolutionary Model Discovery. Models where F_{Recen} interacts with $F_{InitPop}$, F_{Intr}, F_{Recip} have lower RMSE in responsiveness see in the actual data.

Discussion

Mechanistic Explanation *Users experiencing information overload on social media prioritize responses mainly by the recency with which they had been received, but also are more likely to respond to messages on conversations initiated by globally less popular users, and messages from individuals whom they have less in common with and yet have a history of responding to.*

In this chapter I have applied Evolutionary Model Discovery to identify and compare the factors driving the prioritization of Twitter notifications for response under information overload. There is much evidence surrounding the fact information overload can hinder responsiveness. Results of lab experiments in the psychology literature show that under information overload an individual's attention span is reduced, focusing on immediate tasks at hand. The theory of extended self posits that an individual can manifest themselves through physical or virtual objects that they use

regularly, without any conscious decision to do so. Most social media users have shown to treat their social media profiles as extended selves. The theory of extended memory combines these concepts, relating information buffers such as notification lists of social media profiles to one's extended working memory. Under information overload, however, attention to one's extended self, or social media profile can be lost, leading to a loss of attention, and thereby a loss of responsiveness.

Under information overload users have to prioritize incoming messages to which the would respond. The model of extended working memory places information on the actionable information queue, a last in first out stack for processing, while as new information arrives under overload older messages are truncated in a first in first out order. The original model of extended working memory considers the recency of received messages as the sole factor for message prioritization, hence the first in first out order of messages as excessive messages are received.

I have proposed several alternate factors that are hypothesized to affect the prioritization of message on the actionable information queue under information overload. Out of the factors considered, recency is shown to be the factor of highest importance, with messages received more recently being prioritized for response. Global popularity of the conversation initiator is also of high importance, however, interestingly, conversations initiated by users with high global popularity are less likely to be responded to by highly active individuals. This result is similar to the findings by Hodas et al. [135] and Lerman et al. [136] that globally popular information is shared relatively less among a user's immediate neighborhood. Conversation size and popularity of URL domains included in the content of the notifications are not factors that are considered for message prioritization by highly active Twitter users functioning under information overload. Reciprocity, having a history of mutual response to one another, increases the chance of a highly active user responding to a message from another user, despite being under information overload. In particular, like case 2 Chapter 5, the interaction of the main causal factor, recency, with other factors, global unpopularity, lack of common interests, and reciprocity, better explains the phenomena at hand.

CHAPTER 7: OPEN SOURCE SOFTWARE CONTRIBUTIONS

EvolutionaryModelDiscovery

The framework described in this paper has been implemented an open-source Python library. The EvolutionaryModelDiscovery software is a highly-parallelizable library for the genetic programming of NetLogo models followed by causal factor importance evaluation through random forests.

Documentation, Source, and Installation

The documentation for the project is available online at:
https://evolutionarymodeldiscovery.readthedocs.io/en/latest/

The source code is available on GitHub: https://github.com/chathika/evolutionarymodeldiscovery.

The Python library can be easily installed using pip package manager with the following command:

```
>>> pip install EvolutionaryModelDiscovery
```

EvolutionaryModelDiscovery Annotations

EvolutionaryModelDiscovery is able to identify components to be evolved via genetic programming on the NetLogo model through annotated NetLogo comments. All lines to be processed by EvolutionaryModelDiscovery must be tagged with the @EMD annotation in a comment in the immediately prior line of the NetLogo model code. The rule to be evolved must be tagged with the

@EvolveNextLine tag. An example is shown in Fig 7.1. Typically, this annotation is followed by the *.nls* file that contains the hypothesized causal factors, if they are in a separate file, and the final return type expected by the root of the evolved GP tree, with the annotations @*Factors-File*= and @*return-type*=, respectively.

```
209 ;; turtle procedure
210 ;; the turtle evaluates the utility of a given patch
211 to-report calc-utility [patch-to-evaluate]
212   ;let fraction calc-fraction-of-friends patch-to-evaluate
213   ;let min-desired-fraction tolerance
214
215   ;ifelse fraction < min-desired-fraction    [
216   ;  report fraction / min-desired-fraction
217   ;]
218   ;[
219   ;  report 1; 1 represents an happy turtle
220   ;]
221   set patch-being-evaluated patch-to-evaluate
222   let utility-here 0
223   carefully [set utility-here
224   ;; @EMD @EvolveNextLine @Factors-File="util/functions.nls" @return-type=float
225   calc-fraction-of-friends get-patch-to-evaluate
226   ][set utility-here 0]
227   report utility-here
228 end
```

Figure 7.1: The entry point for EvolutionaryModelDiscovery is the line where the rule to be evolved is specified in the NetLogo model. This can be indicated by adding the @*EvolveNextLine* annotation in a comment directly above the line as shown. This may be followed by @*Factors-File*= providing a string with the location of a .nls file that contains the causal factor reports, and the required annotation *return-type*= that indicates the return type expected by the root of evolved GP trees.

Hypothesized causal factors can also be implemented as NetLogo reporters within the original model itself by tagging them with the @*Factor* annotation. An example is shown in Fig. 7.2. Factors must be accompanied with typing of the return types and all, if any, parameters through the @*return-type*= and @*parameter-type*= annotations. Operators are similarly defined with the @*Operator* annotation, followed by the @*return-type*= and @*parameter-type*= annotations, an example is shown in Fig. 7.3, but in addition require a @*structure*= annotation which is a comma sperate list of '+' or '-' signs indicating a positive or negative contribution to presence by the parameter at the corresponding position, respectively.

```
;; turtle procedure
;; reports the mean tolerance in the patch neighborhood
;; @EMD @factor @return-type=float @parameter-type=patch
to-report mean-neighborhood-tolerance [patch-to-evaluate]
  let neighbor-patches [neighboring-patches] of patch-to-evaluate
  let neighbor-turtles-tolerances (list)
  foreach neighbor-patches [
    [neighbor-patch] ->
    if [resident] of neighbor-patch != nobody [
      let tolerance-turtle-here [[tolerance] of resident] of neighbor-patch
      set neighbor-turtles-tolerances lput tolerance-turtle-here neighbor-turtles-tolerances
    ]
  ]
  if (length neighbor-turtles-tolerances = 0 ) [report 0]
  let mean-neighbor-tolerance mean neighbor-turtles-tolerances
  report mean-neighbor-tolerance ; already normalized since tolerance is between 0 and 1
end

```

Figure 7.2: An example of the implementation of a causal factor for EvolutionaryModelDiscovery in NetLogo with the @*Factor* annotation. In addition, *return-type*= must specified, followed by @*parameter-type* specified for each parameter, if parameters exist.

```
;; @EMD @operator @return-type=float @parameter-type=float @parameter-type=float @structure=+,+
to-report combine [a b]
  report (a  + b) / 2
end
```

Figure 7.3: Example of an operator defined as a NetLogo reporter and tagged with the @*Operator* annotation for EvolutionaryModelDiscovery. @*return-type*= must be defined, followed by *parameter-type*= for each parameter, if parameters exist. In addition, the @*structure*= annotation must be specified, providing the contributions of each parameter in order, as '+' or '-', separated by commas, respectively.

Strong Typing

In order to ensure compilation of the genetically programmed NetLogo models, EvolutionaryModelDiscovery enforces strong typing. As discussed above the rule to be evolved, the causal factors, and operators, must all have types specified. Components will only be attached to each other in child-parent fashion if the return type of the child matches the parameter type of the parent. If no such primitives are available, then an exception will be thrown. Types are specified on-the-fly as the EMD entry point specified through @*return-type*=, and at causal factors and operators through the @*return-type*= and @*parameter-type*= annotations. The types are user defined and any strings

valid in both Python and NetLogo can be used. No explicitly declaration of types is required and EvolutionaryModelDiscovery will automatically identify the typing structure for the genetic program primitives through the annotation system. At the time of writing the library does not support dynamic typing.

Running EvolutionaryModelDiscovery

The library can be imported and set up as shown in Fig 7.4.

```
from EvolutionaryModelDiscovery import *
import numpy as np
modelPath = "SimpleSchellingTwoSubgroups_HatnaAdaption.nlogo"
setup = [
    'set neighborhood-distance 1',
    'set empty-cells-to-evaluate-frac 1',
    'set prob-of-relocation-attempt-by-happy random 0.01',
    'set fraction-of-blue 0.5',
    'set density 0.5 + (0.05 * random 10)',
    'set tolerance-dist-blue "0.125,0.5\\n0.833,0.5"',
    'set tolerance-dist-green "0.125,0.5\\n0.833,0.5"',
    'setup']
measurements = ["c-index"]
ticks = 100
emd = EvolutionaryModelDiscovery("/opt/netlogo/", modelPath,setup, measurements, ticks)
```

Figure 7.4: Example of Python commands required to set up EvolutionaryModelDiscovery of a NetLogo model.

Several parameters related to the genetic program can be set as in Fig 7.5. An objective function for the genetic program must be set by defining the objective function callback, as shown in Fig 7.6. The NetLogo model can then be evolved as shown in the example in Fig. 7.7.

For parallel execution, the script should be saved and run using the SCOOP library with the following command:

```
>>> python –m scoop RunEMD.py
```

```
emd.setMutationRate(0.1)
emd.setCrossoverRate(0.8)
emd.setGenerations(20)
emd.setReplications(5)
emd.setDepth(2,6)
emd.setPopulationSize(50)
emd.setIsMinimize(False)
```

Figure 7.5: Example of Python commands that may be used to configure the genetic program of EvolutionaryModelDiscovery.

```
def cindexObjective(results):
    #print(results.iloc[-1][0])
    return np.mean(results.iloc[-1])

emd.setObjectiveFunction(cindexObjective)
```

Figure 7.6: Example of Python commands that can be must be used to specify the objective function for the genetic programming of the NetLogo function.

```
if __name__ == '__main__':
    import warnings
    warnings.filterwarnings("ignore", category=DeprecationWarning)
    emd.evolve()
```

Figure 7.7: Example Python command to begin the genetic programming of the NetLogo model with EvolutionaryModelDiscovery. The *if __name__ = '__main__':* condition is a standard Python best practice to avoid issues related to the absence of a fork implementation on certain operating systems when using parallelization.

NL4Py

NL4Py is a NetLogo controller for Python, developed with the goals of usability, rapid parallel execution, and model parameter access in mind. In addition, NL4Py is platform independent, supporting Windows, MacOS, and Linux, and supports both Python 2 and 3. Unlike PyNetLogo, which uses the Java native interface (JNI) framework to access the JVM, NL4Py, inspired by [137], employs a client-server architecture via Py4J [138], a Python-Java bridging package. NL4Py automatically downloads and hosts a NetLogoControllerServer Java archive (JAR) executable that handles the parallel execution of NetLogo workspaces internally as Java threads. The client-server architecture allows NL4Py to package and hide the excessive programming required to maintain multiple parallely executing NetLogo models, which may even have to be queried regularly depending on the usecase. Further, the NetLogoControllerServer ensures thread safety and handles JVM memory allocation/garbage collection, reducing these burdens from the Python application developer. In short, NL4Py performs the parallelization of NetLogo workspaces on the JVM, instead of leaving it to the user's Python application, unlike PyNetLogo.

Software Architecture

NL4Py uses a client-server architecture and consists of two main components, the NL4Py client written in Python and the NetLogoControllerServer JAR executable written in Java, as shown in Fig. 7.8. The client code communicates to the NetLogoControllerServer through a socket enabled by the Py4J library. The entire NL4Py package is hosted on the Python package index and is automatically downloaded through the pip installer and sets up the NetLogoControllerServer in the Pip package installation directory. The client-server architecture allows NetLogoHeadlessWorkspaces to be run in parallel as Java threads on the NetLogoControllerServer, independent of the user's

Python application code. This eliminates the need for users to have to manage the connection to the JVM, thread/process creation, and garbage collection of multiple headless workspaces from their Python application code.

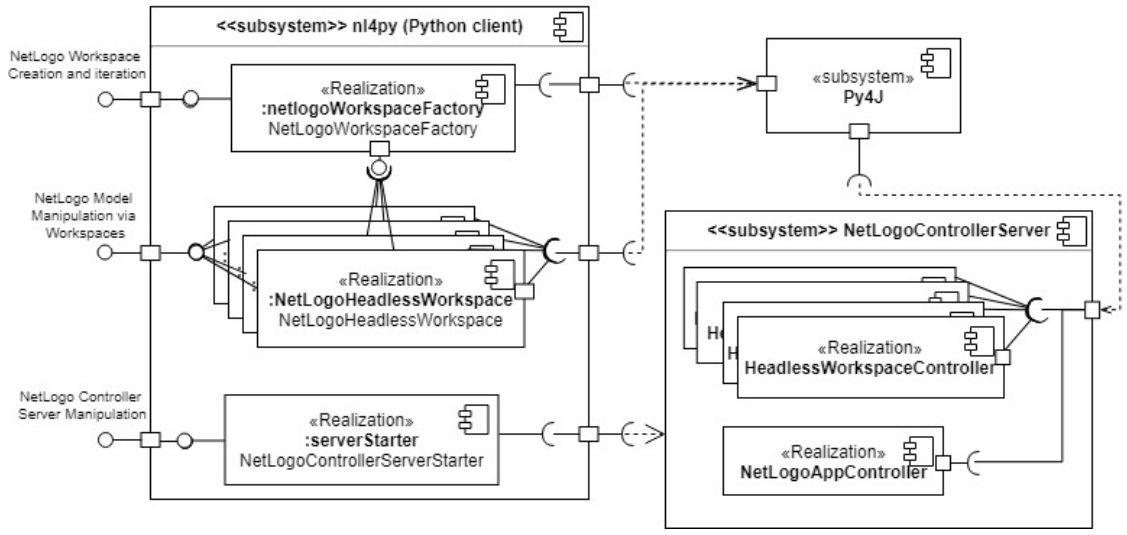


Figure 7.8: UML Component Diagram of NL4Py

NetLogo provides headless workspaces through its controlling API, which can be controlled through Java or Scala application. NetLogo headless workspaces are inherently thread safe. NL4Py, uses this to its advantage by pushing concurrency to the JVM via the NetLogoContollerServer. The NL4Py Python client provides thread-safe NetLogoHeadlessWorkspace objects to the Python application developer, created according to the factory design pattern. Each NetLogoHeadless-Workspace object is mapped to a HeadlessWorkspaceController object on the NetLogoController-Server, which is responsible for starting and stopping the NetLogo model, sending commands to the model, fetching results from reporters to the model, querying parameters, and scheduling reporters over model execution. NL4Py relieves the Python application of thread/process creation, by ensuring that the procedures with long execution times are non-blocking, i.e., results for procedures such as scheduled reporters, whose results must wait till the end of a model run, return

immediately and results can be queried later at a custom time by the user application, without blocking the Python application.

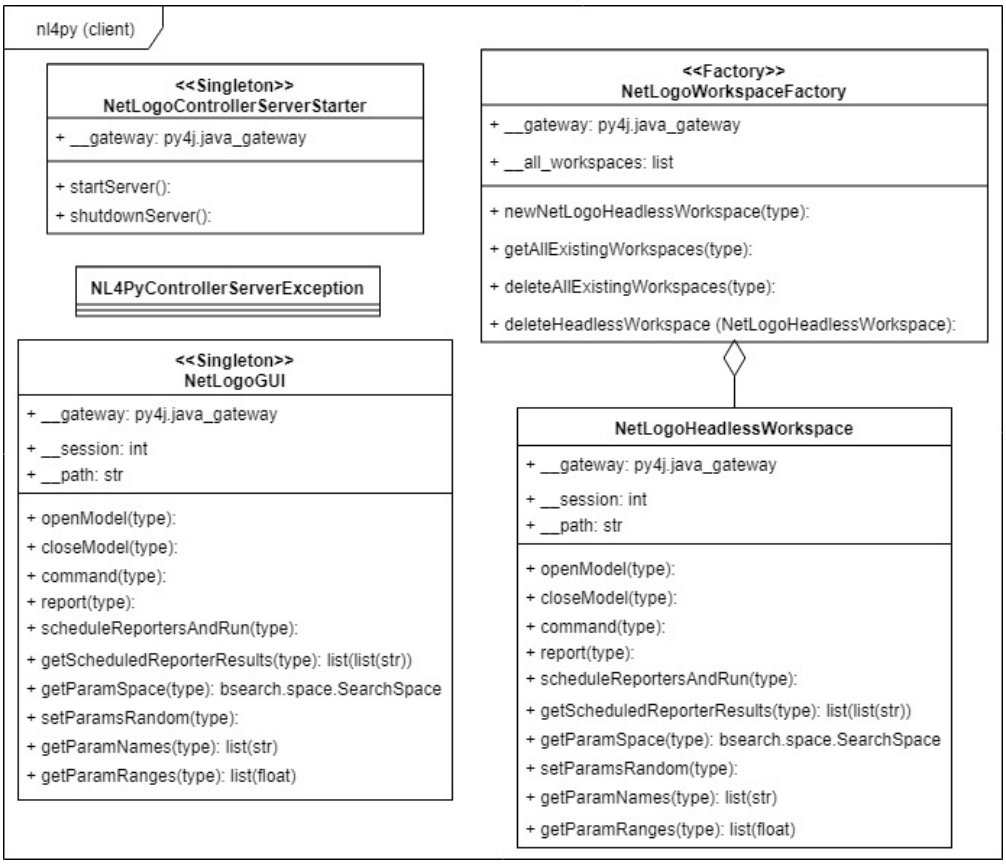


Figure 7.9: UML Class Diagram of NL4Py Python client.

Fig. 7.9 and Fig. 7.10 explain the internal organization of the NL4Py Python client and NetLogoControllerServer, respectively. The NL4Py client consists of a NetLogoControllerServerStarter object, the NetLogoWorkspaceFactory object, and a NetLogoGUI object respectively. Each object has access to the JVM through the Java_gateway offered by Py4J. The NetLogoWorkspaceFactory, is able to create and manage multiple NetLogoHeadlessWorkspace objects, each which maps to a HeadlessWorkspaceController object on the NetLogoControllerServer. On the NetLogoControllerServer, each HeadlessWorkspaceController executes its own command thread for communi-

cation with the NetLogoHeadlessWorkspace object through the NetLogo controlling API. Through this command thread, each HeadlessWorkspaceController is able to pass commands and schedule and query from reporters to the NetLogo model concurrently.

Controlling NetLogo in Python with NL4Py

This section describes how users can setup NL4Py and control multiple NetLogo model runs from within their Python script with the help of NL4Py.

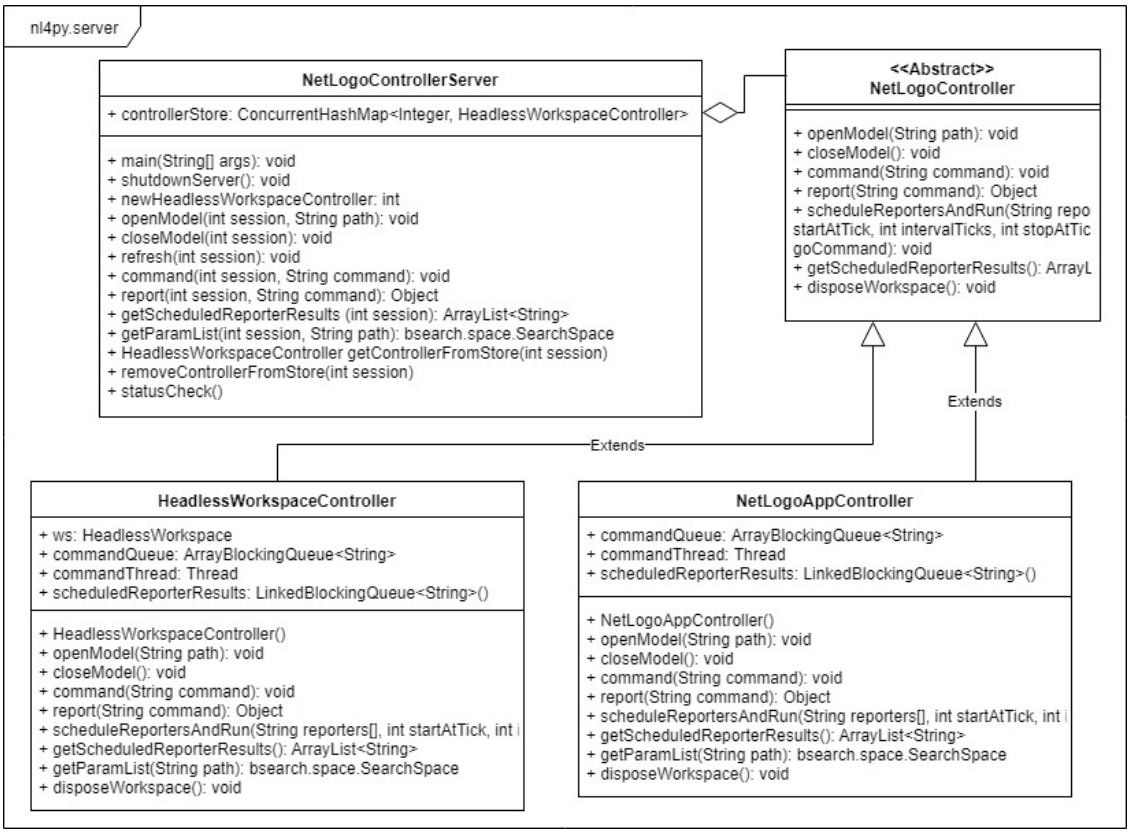


Figure 7.10: UML Class Diagram of NL4Py NetLogoControllerServer.

Installation

NL4Py is made available on the Python package index [139] for easy installation and version control. At the time of writing, NL4Py is in release version 0.5.0 [140]. Pip tools (Python's package manager) can be used to install NL4Py using the following command:

```
>>> pip install NL4Py
```

Requirements

NL4Py works with NetLogo 6.0.2 or higher and requires Java development kit (JDK) 8 or higher to be installed. Other Python dependencies such as Py4J will be installed automatically with pip tools. NL4Py has been tested on both Python 2.7 and Python 3.6 on Windows 10, MacOSX 10.10 and Ubuntu operating systems.

Using the NL4Py API

NL4Py allows both NetLogo *HeadlessWorkspace* creation and control, and also NetLogo GUI-enabled application control. In this section, I describe how users can control NetLogo in both GUI and headless modes with NL4Py.

Starting and stopping the NetLogoControllerServer

The first step to controlling NetLogo through NL4Py is by importing NL4Py and starting the *NetLogoController* server. This can be done with the following commands:

```
>>> import nl4py
```

```
>>> nl4py.startServer(path_to_netlogo)
```

The function requires the path to the top level directory of the NetLogo installation as a string argument. The *NetLogoControllerServer* is then started and ready for requests from the NL4Py client through Python for NetLogo controlling. In complement to this, the *NetLogoControllerServer* can be shutdown, in order to free computational resources using the following command:

```
>>> nl4py.stopServer()
```

Using NetLogo in GUI mode

In order to start and control the NetLogo application in GUI mode, users can execute the following command in their Python script:

```
nl4py.NetLogoApp()
```

Users can then use the *NetLogoApp()* functions to send commands, execute reporters, and schedule reporters to the NetLogo Application. These functions are described in the next section.

Using NetLogo headless workspaces

However, most optimization work requires vast parallel runs of NetLogo models and GUI mode unnecessarily burdens computational resources during such an analysis. NetLogo provides headless workspaces for these purposes, which are essentially NetLogo models running without the GUI enabled. *HeadlessWorkspaces* tend to run more efficiently, since there is no computation required for rendering visualizations BehaviorSearch, the model calibration tool that is packaged with NetLogo, uses headless workspaces driven by optimization algorithms and is a prime example of their utility.

NL4Py provides API controls for Python developers to create NetLogo headless workspaces, open and close models on these workspaces, get and set parameters to the models, send NetLogo commands these models, and schedule and execute reporters to query the simulation state at regular intervals or at the end of a simulation run. Resulting *NetLogoHeadlessWorkspace* objects can then be used to open and control NetLogo models from within Python.

*NetLogoHeadlessWorkspace*s can be created with the following function:

```
nl4py.newNetLogoHeadlessWorkspace()
```

Additionally, users can get a list of all the existing *NetLogoHeadlessWorkspace*s, delete all the existing *NetLogoHeadlessWorkspace*s, and delete a single *NetLogoHeadlessWorkspace* with the following commands, respectively:

```
nl4py.deleteAllHeadlessWorkspaces()
nl4py.getAllHeadlessWorkspaces()
nl4py.deleteHeadlessWorkspace(nl4py.NetLogoHeadlessWorkspace)
```

Opening and closing models

The following commands can be then used to open and close models on *NetLogoHeadlessWorkspace*s, respectively:

```
nl4py.NetLogoHeadlessWorkspace.openModel("path_to_model")
nl4py.NetLogoHeadlessWorkspace.closeModel()
```

Similarly, the same can be done on the NetLogo GUI application with the following:

```
nl4py.NetLogoGUI.openModel(path_to_model)
nl4py.NetLogoGUI.closeModel()
```

Commands and basic reporters

NL4Py provides users the ability to execute NetLogo commands and reporters from within their Python application. The *command* function takes NetLogo syntax as strings and executes the command on the respective workspace. The *report* function takes in NetLogo syntax as strings, executes the reporter, and returns the results. In the case of a failed reporter due to a NetLogo exception, *report* will report the exception. Return values from *report* are typically strings and must be cast into their correct data types accordingly. On *NetLogoHeadlessWorkspace*s the following execute commands and reporters:

```
nl4py.NetLogoHeadlessWorkspace.command(netlogo_command_string)
nl4py.NetLogoHeadlessWorkspace.report(netlogo_reporter_string)
```

Similarly, the equivalent for NetLogo GUI application control:

```
nl4py.NetLogoGUI.openModel(path_to_model)
nl4py.NetLogoGUI.closeModel()
```

Working with parameters

NL4Py allows users to query a NetLogo model's parameter names, get the suggested ranges as set on the NetLogo interface objects (slider min/max values, list values, etc), and set the parameters to random values. The three following methods provide these functions for NetLogoHeadless-Workspaces, respectively:

```
nl4py.NetLogoHeadlessWorkspace.setParamsRandom()
nl4py.NetLogoHeadlessWorkspace.getParamNames()
nl4py.NetLogoHeadlessWorkspace.getParamRanges()
```

Similarly, these functions are available for the NetLogo GUI application:

```
nl4py.NetLogoGUI.setParamsRandom()
nl4py.NetLogoGUI.getParamNames()
nl4py.NetLogoGUI.getParamRanges()
```

Reporter scheduling

In certain instances, a user may require to record the simulation state at regular intervals, often for every simulation tick, over a given period of time. For this, NL4Py provides scheduled reporters. Multiple reporters can be specified as a Python list of strings of NetLogo commands. This reporter list along with the start tick, stop tick, and interval of ticks required between each reporter execution can be passed into into *scheduleReporterAndRun* for this purpose. Optionally, a custom *go* command can be supplied to *scheduleReportersAndRun* in the case that the NetLogo model's execution procedure has a name different to the standard *go*. NL4Py then schedules the reporters to execute on the respective NetLogo workspaces and store results on the *NetLogoControllerServer* from start time till stop time at every interval number of ticks.

On *NetLogoHeadlessWorkspace*s, this method signature is:

```
nl4py.NetLogoHeadlessWorkspace.scheduleReportersAndRun(
    reporters_array, startAtTick=0, intervalTicks=1, stopAtTick=-1,
    goCommand="go")
```

Similarly, on the NetLogo GUI Application mode this is:

```
nl4py.NetLogoGUI.scheduleReportersAndRun(reporters_array,
    startAtTick=0,intervalTicks=1,stopAtTick=-1,goCommand="go")
```

The results stored on the *NetLogoControllerServer* can be queried at anytime during or after the scheduled reporters execution, with *getScheduledReporterResults*. This function returns a Numpy array of lists of the reporter results, for each execution thus far. This function is non-blocking to prevent imposing unnecessary wait times on the user's application. If the model has not finished execution, then an empty array will be returned. On *NetLogoHeadlessWorkspace*s this function is:

```
nl4py.NetLogoHeadlessWorkspace.getScheduledReporterResults()
```

Similarly, in the NetLogo GUI application:

```
nl4py.NetLogoGUI.getScheduledReporterResults()
```

CHAPTER 8: CONCLUSION

Despite being an excellent tool for the construction and analysis of human-interpretable, mechanistic explanations of social phenomena, ABMs risk premature assumptions when modeling individuals' decision-making processes. Parameter calibration alone cannot adequately explore the causal factors and their possible interactions in order to infer more accurate decision-making processes. This is primarily due to the absence of a systematic method for behavior inference and discovery. I address this issue with the development of Evolutionary Model Discovery, a framework for automated causal inference in agent-based artificial societies. Given a set of factors of decision-making hypothesized to generate the societal phenomena of interest, Evolutionary Model Discovery is able to quantify, compare, and discover the optimal presence of the causal factors for the robust and accurate replication of the target phenomena. By combining automated program generation of genetic programming with feature importance evaluation of random forests, Evolutionary Model Discovery is able to quantify the importance of these factors to the decision-making process that results in society-level phenomena simulated by the ABM. This allows for the construction of agent rules that more accurately represent the actual decision-making process of individuals and result in more models that deviate less from the actual trajectory of the system being studied and remain robust even under random parameter initialization, as stated in Claim 1 in Chapter 3.

Evolutionary Model Discovery was applied to three separate cases where the causes of a societal phenomena were not directly understood due to the disappearance of a culture, hidden motives, and/or complex thought processes at play. Results from applying Evolutionary Model Discovery to the Artificial Anasazi show highlight the danger of constructing socio-behavioral models without a complete exploration of the space of possible rules. Contrary to the original model, the most important factors driving farm plot selection of the ancestral Pueblo community were found to be

desire for higher quality soil and social presence, instead of the closeness of the next plot. closeness actually hindered the model from robustness to randomized parameter settings. In hindsight, this result indicates quite intuitive behavior of a society that was aware of the more arable areas of land that would yield better crops, but at the same time wish to agglomerate into tighter neighborhoods. These results seem to agree with the archaeological evidence from ruins that exist today demonstrating that the ancestral Pueblo lived in dwellings that were often located close together, centered around their respective farm plots.

Secondly, Evolutionary Model Discovery of factors leading to the emergence of mixed patterns of segregation and integration, indicate that the factor driving the desirability of residential locations, racial preference alone, as modeled originally by Schelling [104] and used in later extension of the model [105] is insufficient and may hinder the production of robust models of mixed patterns. In fact, it was shown that no single factor could generate such patterns well. Instead the interplay of multiple factors: reluctance to move, preference of less crowded neighborhoods, and preference for higher racial similarity easily produced patterns with higher mixing, under random urban density settings. There is evidence that mixed patterns of segregation and integration might also be caused due to acceptance of diverse tolerance to race in prospective neighborhoods and of neighborhoods with an imbalance or conflict in residential satisfaction.

Thirdly, Evolutionary Model Discovery identified factors driving highly active, overloaded users to prefer to respond to certain social media notifications over others. The strongest factor driving notification prioritization was message recency; messages received more recently were more likely to respond to. This signals a decay of responsiveness to notifications the longer they remain unresponded to with the passage of time, which has been observed in studies such as [131]. Notifications of conversations that were received regarding conversations initiated by profiles that had lower global popularity received higher priority of response under overload by highly active users. This may explained the observation in analytical studies that show that globally popular

information is often less abundant in local neighborhoods [135, 136]. Due to scale-freeness of social networks, highly popular users are also less abundant globally, and this could mean that overloaded users prefer to respond to conversations started by their friends/followers who are most likely globally less popular. Confirming this, overloaded highly active user's were found to be more likely to respond to friends with higher reciprocity; i.e. those with whom they have a history of direct response to. Conversation size and presence of URLs in the content were irrelevant to response prioritization.

These results provide empirical support to Claim 1 and demonstrate the successful application of Evolutionary Model Discovery in finding mechanistic explanations of complex societal phenomena, while respecting the fact that such systems must be studied holistically, with consideration of the non-linear trajectories and high sensitivity to initial conditions that led to the emergence of the phenomena at hand. Evolutionary Model Discovery embraces the arguments made by Grune-Yanoff [18] and Elsenbroich [7] and extends the vision of Epstein [43] towards human-interpretable explanation of societal phenomena, by seeking mechanistic explanations as defined by Machamer et al. [14].

Future Work

This work opens the doors to many new avenues of study. First and more straightforward is the application of Evolutionary Model Discovery in other agent-based modeling projects. Many data/stylized-fact -driven agent-based models exist in the literature that would benefit from further scrutinizing of the causal factors behind the societal phenomena they seek to simulate.

Secondly, in Sec. 3, I prove how the logic depth of the genetic program tree forming the behavior rule has the highest cost on computational complexity of the genetic programming of agent-based

models. An improvement that could be made to this framework is, first, establishing that this is reflected in practice, and next, incorporating such a strategy into the current implementation. Unfortunately, NetLogo does not support parallel computing. In fact, parallelization of agent-based models is not well supported by most software, with a few exceptions of RepastHPC or FLAME-GPU. However, the code of models implemented in such software are in C or C++, compiled programming languages, unlike the lisp-like scripting of NetLogo, making it difficult to implement a compilable genetic program representation. This would be quite a challenging, yet beneficial, avenue for extension of the current implementaiton of Evolutionary Model Discovery.

APPENDIX A: IRB OUTCOME LETTER

University of Central Florida Institutional Review Board
Office of Research & Commercialization
12201 Research Parkway, Suite 501
Orlando, Florida 32826-3246
Telephone: 407-823-2901, 407-882-2012 or 407-882-2276
www.research.ucf.edu/compliance/irb.html

NOT HUMAN RESEARCH DETERMINATION

From: **UCF Institutional Review Board #1**
FWA00000351, IRB00001138

To: **Ivan I Garibay, Alexander Mantzaris, Gita Reese Sukthankar, Stephen M Fiore**

Date: **February 21, 2018**

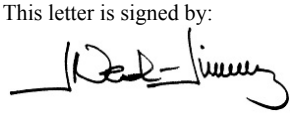
Dear Researcher:

On 02/21/2018, the IRB determined that the following proposed activity is not human research as defined by DHHS regulations at 45 CFR 46 or FDA regulations at 21 CFR 50/56:

Type of Review:	Not Human Research Determination
Project Title:	Deep Agent: A Framework for Information Spread and Evolution in Social Networks
Investigator:	Ivan I Garibay
IRB ID:	SBE-18-13732
Funding Agency:	DARPA
Grant Title:	Deep Agent: A Framework for Information Spread and Evolution in Social Networks
Research ID:	1062483

University of Central Florida IRB review and approval is not required. This determination applies only to the activities described in the IRB submission and does not apply should any changes be made. If changes are to be made and there are questions about whether these activities are research involving human subjects, please contact the IRB office to discuss the proposed changes.

This letter is signed by:



Signature applied by Jennifer Neal-Jimenez on 02/21/2018 04:32:37 PM EST

Designated Reviewer

APPENDIX B: FURTHER CONSIDERATIONS

Evolvability

The driving force behind evolutionary search is the selection pressure experienced due to increases in fitness with changes in the genes encoding the primitives structure of the gp-individuals. Primitives that form 'flat' search spaces in terms of fitness, i.e. have no gradients of fitness with changing values of genes, do not have any considerable selection pressure. Such a landscape has no driving force for the evolutionary search and can be searched with a trivial random search. Therefore, it is important to establish whether such selection pressure existed in the three cases explored above.

Two indicators of evolvability are considered, convergence and selection pressure. The gradual convergence towards a common solution by all gp-individuals, indicates that the genetic program tends to exploit regions of higher fitness in the search space. On the contrary no convergence indicates that there was no tendency among the population of gp-individuals to evolve towards higher fit solutions. Increasing tendency to select certain primitives over generations, indicates higher selection pressure towards the variables represented by those primitives; i.e. if the absolute presence of a certain primitive increases over time, it is indicative there are global or local optimal solutions that can be reached through its selection. On the other hand, no indication of change in absolute selection indicate that there are no gradients of fitness achieved through the selection of the primitive; i.e. the represented factor does not lead to higher fit solutions.

Figs. B.1, B.2, and B.3 demonstrate the convergence of cases 1, 2, and 3 explained in the previous chapters. The average convergence with the 95% confidence interval is shown. Genetic program of the farm plot selection in the Artificial Anasazi converges relatively slowly in comparison to the other two cases, at around 75 generations. Case 2, evolution of the residential location utility function converges at a quicker pace, within around 10 generations. Response prioritization converges relatively quite quickly, again at around 20 generations. Considering the fact that there less distinction among high importance factors in the Artificial Anasazi case (F_{Qual}, F_{Soc}, F_{Dist}, and

F_{Mig}), more in the response prioritization case (F_{Move}, F_{Race}, and F_{Isol}), and even more (F_{Recn} and $F_{InitPop}$ were by far more important than other factors), this indicates that the search space for Case 1 was harder to navigate than Case 2, and Case 2 harder than Case 3, because for the first case more primitives provided high fitness when selected than when compared to the latter case.

This is confirmed when looking at the plots on primitive selection by generation for the three cases Figs. B.4, B.5, and B.6, respectively. Most factors appear to have reasonable positive or negative selection bias as generations progress in Case 1 (Fig. B.4); e.g.: F_{Qual}, F_{Mig}, F_{Dist}, F_{Yield}, and F_{Soc} show higher presence as generations progress, and provide lower RMSE when present positively, while F_{HAgri} progressively is searched for negative presence providing lower RMSE. Overall most primitives are searched over a reasonable range of presence. In case 2 (Fig. B.5), F_{Race}, F_{Move}, F_{Isol}, and F_{TolDiv} are selected gradually selected either in a positive of negative bias, but only F_{Race}, F_{Move}, and F_{Isol} show large improvements in C-Index, under higher absolute selection. In Case 3, even a smaller number of primitives demonstrate a significant change in selection with the progression of generations. $F_{InitPop}$, F_{Recn}, and F_{Intr} show to exert more selection pressure, being selected more often as generations progress, and provide lower RMSE at higher absolute presence. These results also reflect that fact that the best solutions in Case 1 generally have are greater number of causal factors, than Case 2, and even more than for Case 3.

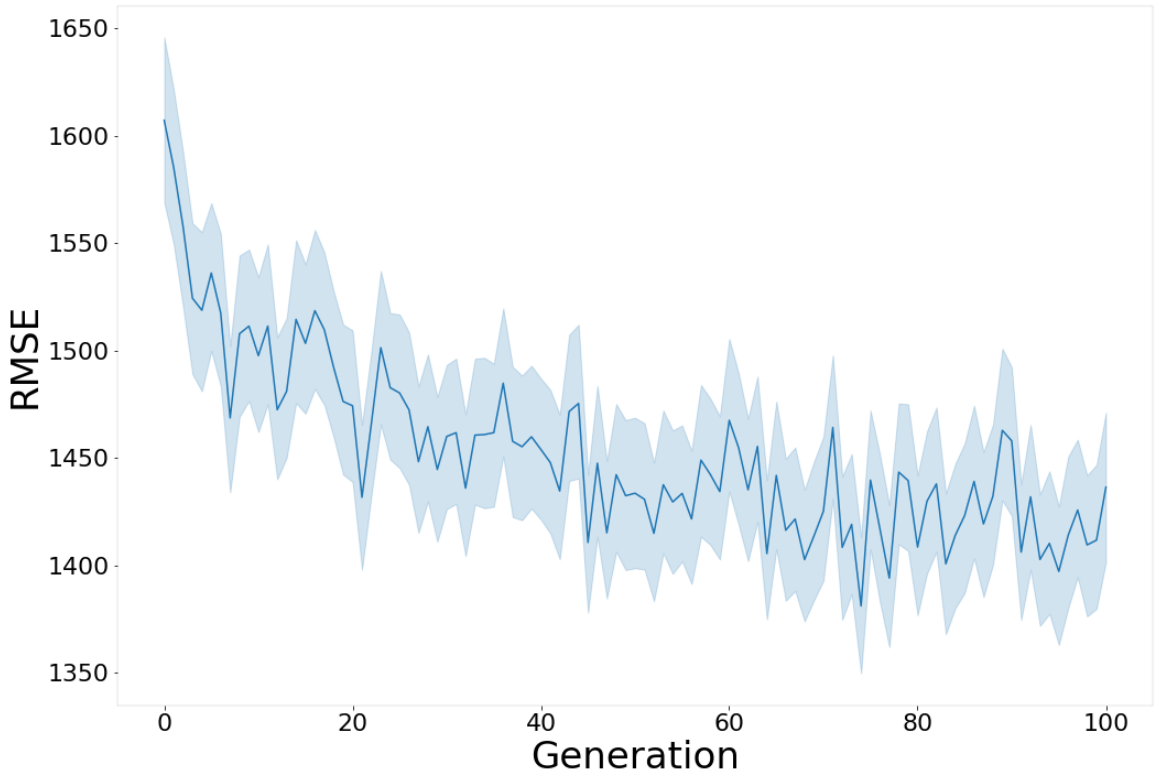


Figure B.1: Convergence of the genetic programming of the farm selection decision-making rule in the Artificial Anasazi. Average RMSE and the 95% confidence interval are shown.

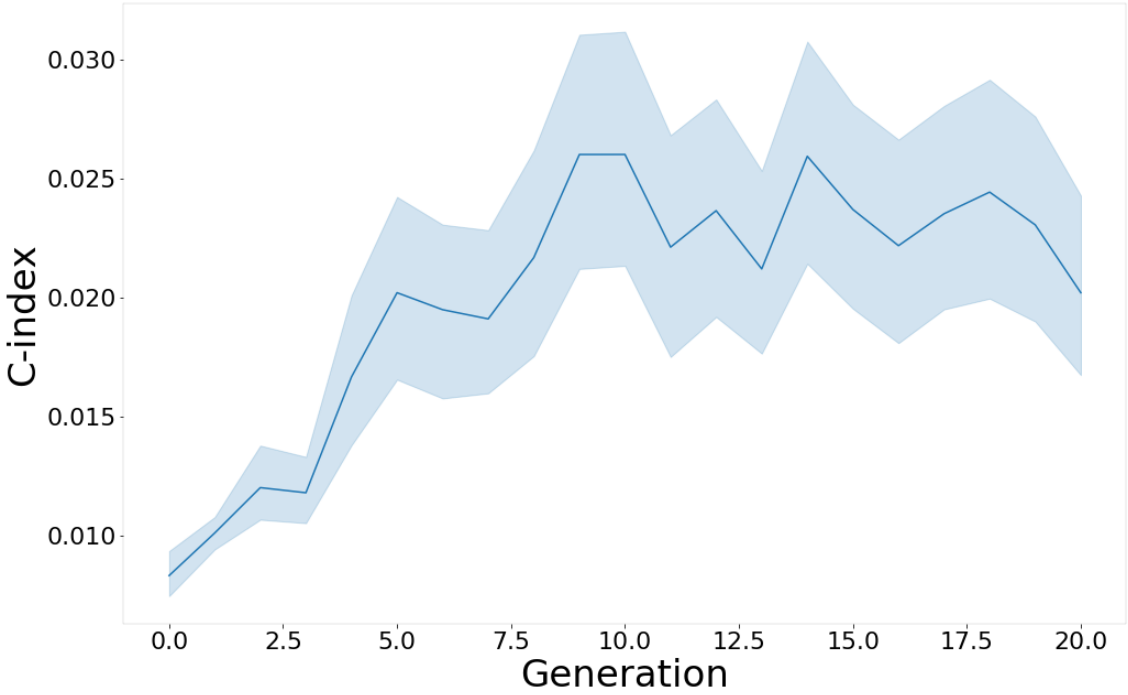


Figure B.2: Convergence of the genetic programming of the residential location utility function in Hatna and Benenson's model of segregation. Average C-index and the 95% confidence interval are shown.

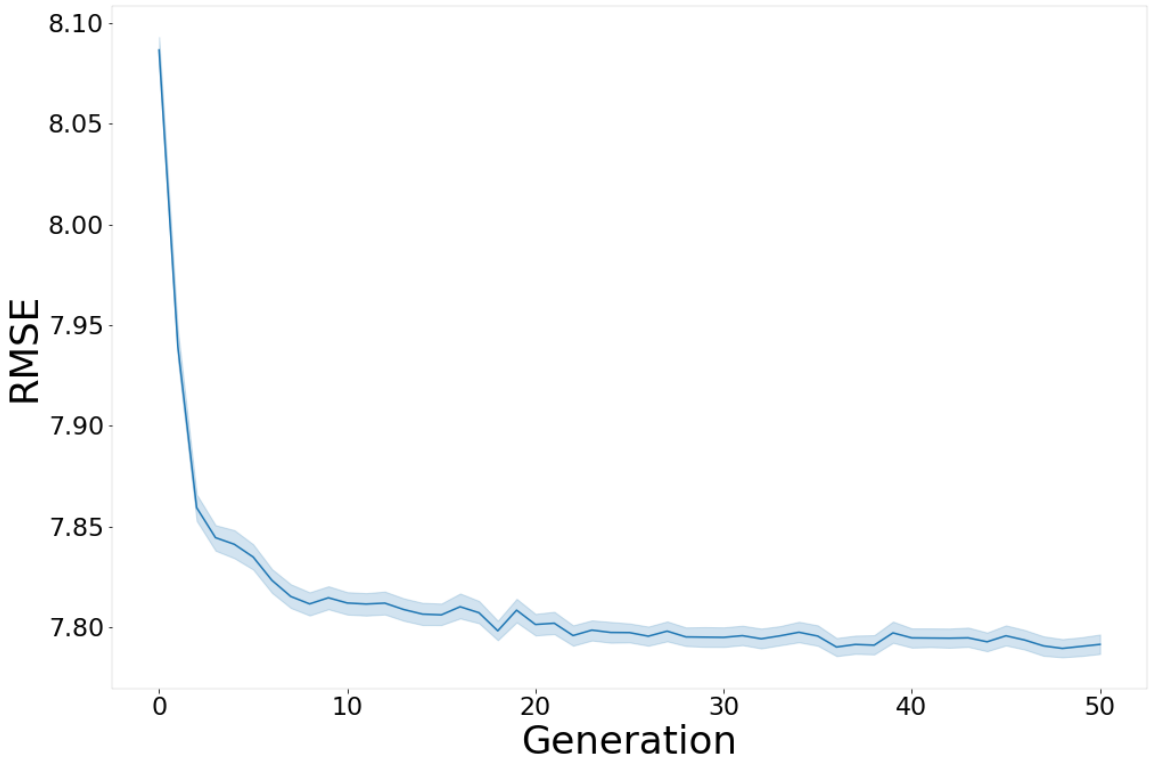


Figure B.3: Convergence of the genetic programming of the response prioritization utility function in the model of extended working memory. Average RMSE and the 95% confidence interval are shown.

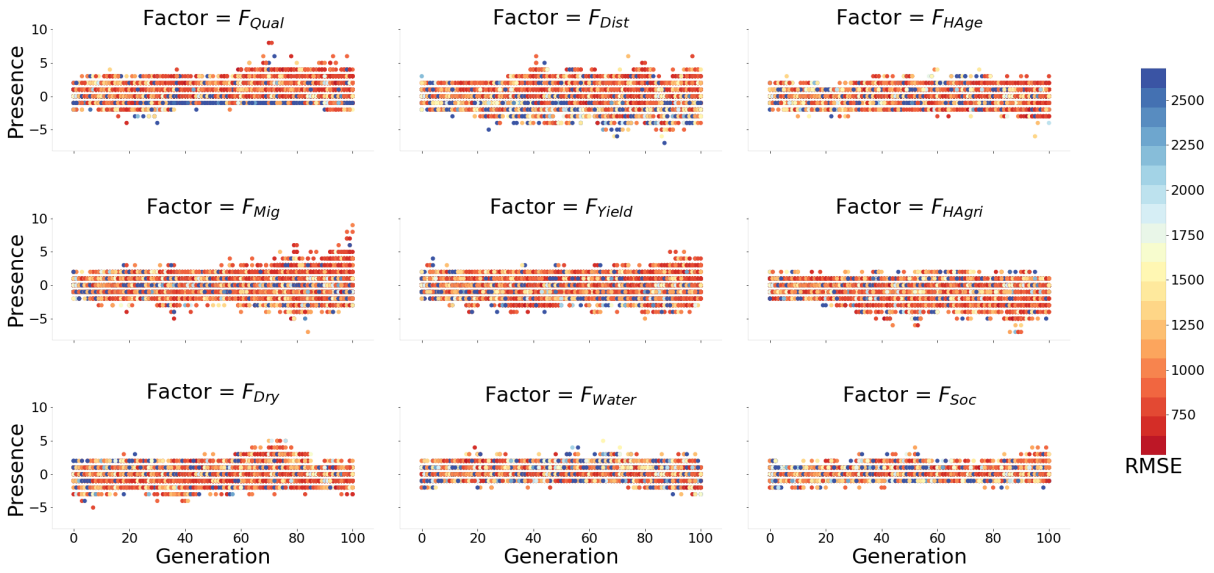


Figure B.4: Primitive selection of the genetic programming of the farm selection strategy in the Artificial Anasazi. Factor presence and the respective RMSE are shown over generations.

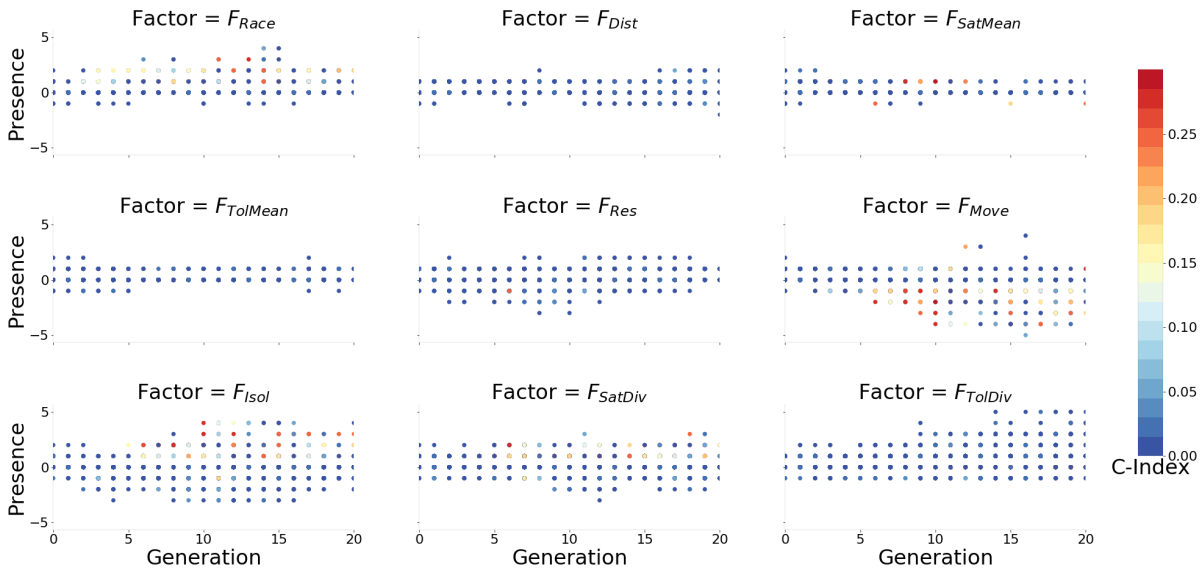


Figure B.5: Primitive selection of the genetic programming of the residential location utility function of Hatna's model of segregation. Factor presence and the respective RMSE are shown over generations.

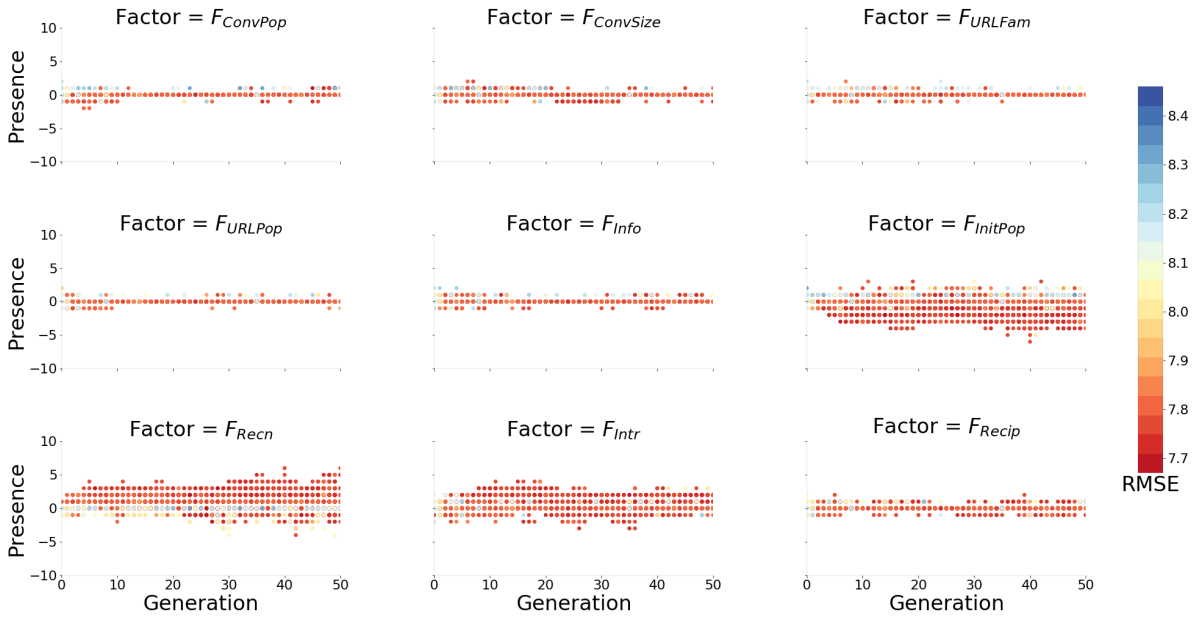


Figure B.6: Primitive selection of the genetic programming of the response prioritization of the model of extended working memory. Factor presence and the respective RMSE are shown over generations.

Model Size and Bloat

The size of the syntax trees represented by a gp-individual provides an estimate of how complicated the task being modeled actually is. More primitives on a decision tree that provides relatively higher fitness, indicates a task that requires more information. However, genetic programming is susceptible to *bloating*, where primitives accumulate over generations with no improvements to fitness. Bloating is more probable if evolution continues beyond convergence. Several strategies have been suggested to control bloating, out of which setting a maximum tree depth has been used in the experiments above.

Figs. B.7, B.8, and B.9 show the fitness and simplified model size, or logic depth, with the progression of generations for all gp-individuals for the 3 cases considered in the previous chapters respectively. In case 1 (Fig. B.7), there seems to be an increase in model size after generation 40. Though there's somewhat of an decrease in RMSE, many large solutions also exist which have higher RMSE, an indicaiton of bloat. In case 2 (Fig. B.8), larger models are evolved past generation 10, some of which achieve higher C-indices. Yet many of the largest solutions remain with low C-indices, again indicating bloat. In case 3 (Fig. B.9), large solutions start appearing gradually after generation 20, and there is a sudden set of very large rules around generation 48 to 50 that have low RMSE. Yet, even in case 3, a few large solutions do exist that have high RMSE, despite their size, indicating some bloat.

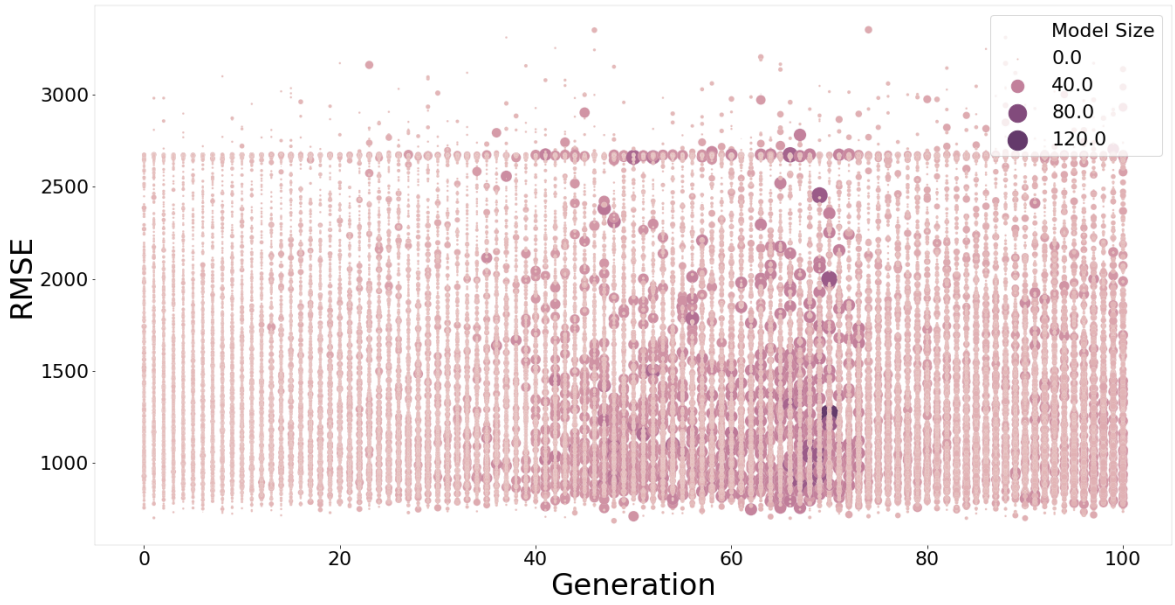


Figure B.7: Simplified model size (size and color of points) and fitness over the progression of generations of all gp-individuals for genetically programming farm selection decision-making rule in the Artificial Anasazi.

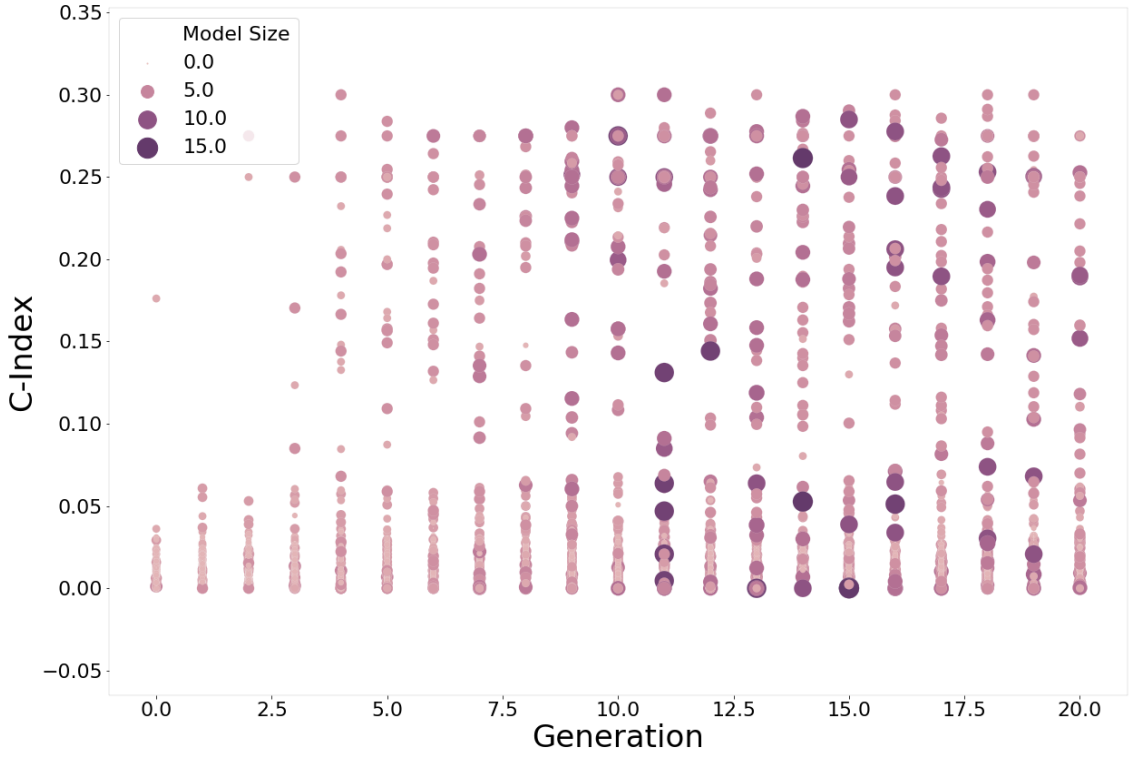


Figure B.8: Simplified model size (size and color of points) and fitness over the progression of generations of all gp-individuals for genetically programming residential location utility function in Hatna and Benenson's model of segregation.

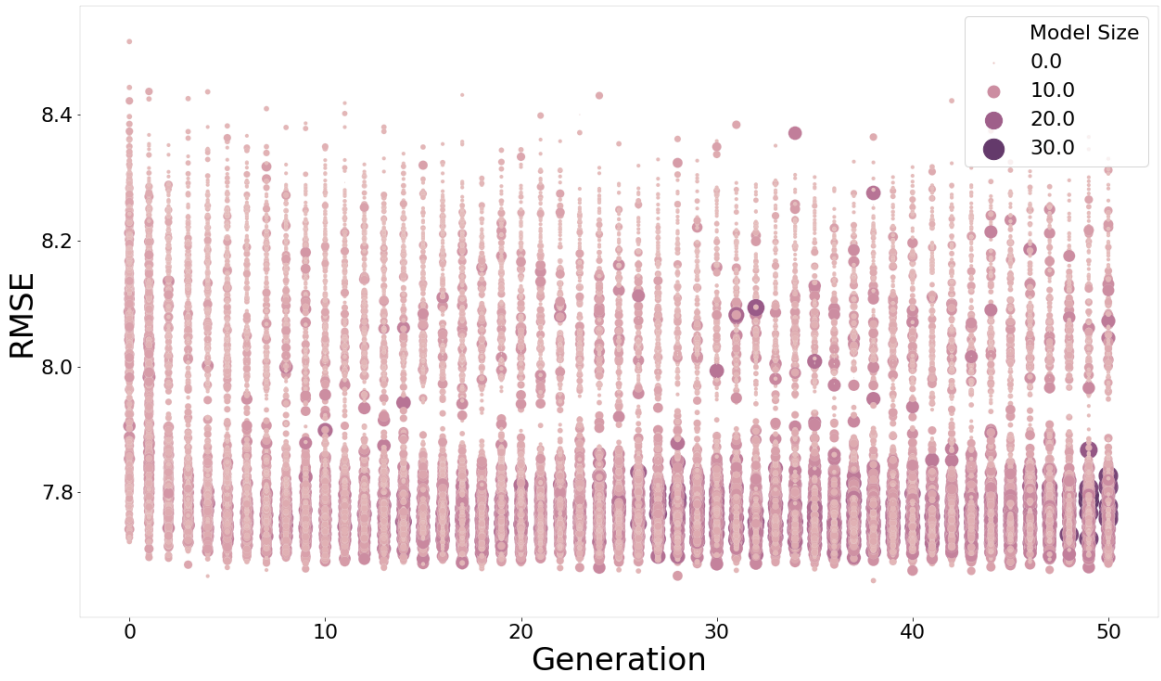


Figure B.9: Simplified model size (size and color of points) and fitness over the progression of generations of all gp-individuals for genetically programming response prioritization utility function in the model of extended working memory.

LIST OF REFERENCES

[1] F. Stonedahl and U. Wilensky, "Netlogo artificial anasazi model," Center for Connected Learning and Computer-Based Modeling, Northwestern University, 2010. [Online]. Available: http://ccl.northwestern.edu/netlogo/models/ArtificialAnasazi

[2] J. S. Dean, G. J. Gumerman, J. M. Epstein, R. L. Axtell, A. C. Swedlund, M. T. Parker, and S. McCarroll, "Understanding anasazi culture change through agent-based modeling," *Dynamics in human and primate societies: Agent-based modeling of social and spatial processes*, pp. 179–205, 2000.

[3] M. A. Janssen, "Understanding artificial anasazi," *Journal of Artificial Societies and Social Simulation*, vol. 12, no. 4, p. 13, 2009.

[4] F. Stonedahl and U. Wilensky, "Evolutionary robustness checking in the artificial anasazi model." in *AAAI Fall Symposium: Complex Adaptive Systems*, 2010, pp. 120–129.

[5] J. M. Epstein and R. Axtell, *Growing Artificial Societies: Social Science from the Bottom Up*. Brookings Institution Press, 1996.

[6] W. Rand, "Theory-interpretable, data-driven agent-based modeling," *Social-Behavioral Modeling for Complex Systems*, pp. 337–357, 2019.

[7] C. Elsenbroich, "Explanation in agent-based modelling: Functions, causality or mechanisms?" *Journal of Artificial Societies and Social Simulation*, vol. 15, no. 3, p. 1, 2012.

[8] P. K. Davis and A. O'Mahony, "A computational model of public support for insurgency and terrorism: A prototype for more-general social-science modeling," Rand National Defense Research Institute, Santa Monica, CA, Tech. Rep., 2013.

[9] J. M. Epstein, "Agent-based computational models and generative social science," *Complexity*, vol. 4, no. 5, pp. 41–60, 1999.

[10] N. S. Thompson and P. Derr, "Contra epstein, good explanations predict," *Journal of Artificial Societies and Social Simulation*, vol. 12, no. 1, p. 9, 2009.

[11] J. M. Epstein, "Why model?" *Journal of Artificial Societies and Social Simulation*, vol. 11, no. 4, p. 12, 2008.

[12] C. G. Hempel and P. Oppenheim, "Studies in the logic of explanation," *Philosophy of science*, vol. 15, no. 2, pp. 135–175, 1948.

[13] W. C. Salmon, *Causality and explanation*. Oxford University Press, 1998.

[14] P. Machamer, L. Darden, and C. F. Craver, "Thinking about mechanisms," *Philosophy of science*, vol. 67, no. 1, pp. 1–25, 2000.

[15] P. Kitcher, "Explanation, conjunction, and unification," *The Journal of Philosophy*, vol. 73, no. 8, pp. 207–212, 1976.

[16] F. Stonedahl and U. Wilensky, "Behaviorsearch [computer software]," *Center for Connected Learning and Computer Based Modeling, Northwestern University, Evanston, IL. Available online: http://www. behaviorsearch. org*, 2010.

[17] M. Laguna and R. Marti, "The optquest callable library," in *Optimization software class libraries*. Springer, 2003, pp. 193–218.

[18] T. Grüne-Yanoff, "The explanatory potential of artificial societies," *Synthese*, vol. 169, no. 3, pp. 539–555, 2009.

[19] V. Grimm, E. Revilla, U. Berger, F. Jeltsch, W. M. Mooij, S. F. Railsback, H.-H. Thulke, J. Weiner, T. Wiegand, and D. L. DeAngelis, "Pattern-oriented modeling of agent-based complex systems: lessons from ecology," *science*, vol. 310, no. 5750, pp. 987–991, 2005.

[20] J. M. Whitmeyer, M. Khouja, T. Carmichael, A. Saric, C. Eichelberger, M. Sun, and M. Hadzikadic, "A computer simulation laboratory for social theories," in *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on*, vol. 2. IEEE, 2008, pp. 512–515.

[21] E. Economo, L. Hong, and S. E. Page, "Social structure, endogenous diversity, and collective accuracy," *Journal of Economic Behavior & Organization*, vol. 125, pp. 212–231, 2016.

[22] J. Bednar and S. E. Page, "Complex adaptive systems and comparative politics: Modeling the interaction between institutions and culture," *Chinese Political Science Review*, vol. 1, no. 3, pp. 448–471, 2016.

[23] L. Hong and S. E. Page, "Problem solving by heterogeneous agents," *Journal of economic theory*, vol. 97, no. 1, pp. 123–163, 2001.

[24] M. Weisberg, "Forty years of 'the strategy': Levins on model building and idealization," *Biology and Philosophy*, vol. 21, no. 5, pp. 623–645, 2006.

[25] R. Levins, "The strategy of model building in population biology," *American scientist*, vol. 54, no. 4, pp. 421–431, 1966.

[26] J. M. Epstein, *Agent Zero : toward Neurocognitive Foundations for Generative Social Science*. Princeton, NJ : Princeton University Press, 2013.

[27] C. Gunaratne, "Evolutionary model discovery documentation," 2019. [Online]. Available: https://evolutionarymodeldiscovery.readthedocs.io/en/latest/

[28] U. Wilensky, "Netlogo," 1999. [Online]. Available: http://ccl.northwestern.edu/netlogo/

[29] R. L. Axtell, J. M. Epstein, J. S. Dean, G. J. Gumerman, A. C. Swedlund, J. Harburger, S. Chakravarty, R. Hammond, J. Parker, and M. Parker, "Population growth and collapse in a multiagent model of the kayenta anasazi in long house valley," *Proceedings of the National Academy of Sciences*, vol. 99, no. suppl 3, pp. 7275–7279, 2002.

[30] T. A. Kohler, D. Cockburn, P. L. Hooper, R. K. Bocinsky, and Z. Kobti, "The coevolution of group size and leadership: An agent-based public goods model for prehispanic pueblo societies," *Advances in Complex Systems*, vol. 15, no. 01n02, p. 1150007, 2012.

[31] R. L. Axtell, "120 million agents self-organize into 6 million firms: a model of the us private sector," in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 806–816.

[32] J. D. Farmer and D. Foley, "The economy needs agent-based modelling," *Nature*, vol. 460, no. 7256, p. 685, 2009.

[33] L. Tesfatsion, "Agent-based computational economics: modeling economies as complex adaptive systems," *Information Sciences*, vol. 149, no. 4, pp. 262–268, 2003.

[34] C. Deissenberg, S. Van Der Hoog, and H. Dawid, "Eurace: A massively parallel agent-based model of the european economy," *Applied Mathematics and Computation*, vol. 204, no. 2, pp. 541–552, 2008.

[35] H. Dawid and M. Neugart, "Agent-based models for economic policy design," *Eastern Economic Journal*, vol. 37, no. 1, pp. 44–50, 2011.

[36] S. Heckbert, T. Baynes, and A. Reeson, "Agent-based modeling in ecological economics," *Annals of the New York Academy of Sciences*, vol. 1185, no. 1, pp. 39–53, 2010.

[37] I. Garibay, C. Gunaratne, M. I. Akbas, and O. Ozmen, "The importance of product space complexity in agent-based computational economics," 2016.

[38] J. J. Arsanjani, M. Helbich, and E. de Noronha Vaz, "Spatiotemporal simulation of urban growth patterns using agent-based modeling: The case of tehran," *Cities*, vol. 32, pp. 33–42, 2013.

[39] D. Weisburd, A. A. Braga, E. R. Groff, and A. Wooditch, "Can hot spots policing reduce crime in urban areas? a," *Criminology*, vol. 55, no. 1, pp. 137–173, 2017.

[40] E. Hunter, B. Mac Namee, J. Kelleher *et al.*, "A taxonomy for agent-based models in human infectious disease epidemiology," *Journal of Artificial Societies and Social Simulation*, vol. 20, no. 3, pp. 1–2, 2017.

[41] D. S. Burke, J. M. Epstein, D. A. Cummings, J. I. Parker, K. C. Cline, R. M. Singa, and S. Chakravarty, "Individual-based computational modeling of smallpox epidemic control strategies," *Academic Emergency Medicine*, vol. 13, no. 11, pp. 1142–1149, 2006.

[42] R. M. Axelrod, *The complexity of cooperation: Agent-based models of competition and collaboration*. Princeton University Press, 1997.

[43] J. M. Epstein, *Generative social science: Studies in agent-based computational modeling*. Princeton University Press, 2006.

[44] E. R. Smith and F. R. Conrey, "Agent-based modeling: A new approach for theory building in social psychology," *Personality and social psychology review*, vol. 11, no. 1, pp. 87–104, 2007.

[45] F. Lamperti, A. Roventini, and A. Sani, "Agent-based model calibration using machine learning surrogates," *Journal of Economic Dynamics and Control*, vol. 90, pp. 366–389, 2018.

[46] D. G. Brown, S. Page, R. Riolo, M. Zellner, and W. Rand, "Path dependence and the validation of agent-based spatial models of land use," *International Journal of Geographical Information Science*, vol. 19, no. 2, pp. 153–174, 2005.

[47] S. Moss, "Alternative approaches to the empirical validation of agent-based models," *Journal of Artificial Societies and Social Simulation*, vol. 11, no. 1, p. 5, 2008.

[48] C. Bianchi, P. Cirillo, M. Gallegati, and P. A. Vagliasindi, "Validating and calibrating agent-based models: a case study," *Computational Economics*, vol. 30, no. 3, pp. 245–264, 2007.

[49] C. M. Macal, "Model verification and validation," *University of Chicago*, pp. 1–21, 2005.

[50] F. J. Stonedahl, "Genetic algorithms for the exploration of parameter spaces in agent-based models," 2011.

[51] J. P. Kleijnen and J. Wan, "Optimization of simulated systems: Optquest and alternatives," *Simulation Modelling Practice and Theory*, vol. 15, no. 3, pp. 354–362, 2007.

[52] M. Edali and G. Yücel, "Exploring the behavior space of agent-based simulation models using random forest metamodels and sequential sampling," *Simulation Modelling Practice and Theory*, vol. 92, pp. 62–81, 2019.

[53] C. Georges and J. Pereira, "Market stability with machine learning agents," *Available at SSRN 3374666*, 2019.

[54] S. van der Hoog, "Deep learning in (and of) agent-based models: A prospectus," *arXiv preprint arXiv:1706.06302*, 2017.

[55] Y. Lu, K. Kawamura, and M. L. Zellner, "Exploring the influence of urban form on work travel behavior with agent-based modeling," *Transportation Research Record*, vol. 2082, no. 1, pp. 132–140, 2008.

[56] D. Darmon, J. Sylvester, M. Girvan, and W. Rand, "Predictability of user behavior in social media: Bottom-up v. top-down modeling," in *2013 International Conference on Social Computing*. IEEE, 2013, pp. 102–107.

[57] J. Harada, D. Darmon, M. Girvan, and W. Rand, "Forecasting high tide: Predicting times of elevated activity in online social media," in *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*. ACM, 2015, pp. 504–507.

[58] A. Ariyaratne, "Modeling agent behavior through past actions: Simulating twitter users," Ph.D. dissertation, University of Maryland, College Park, 2016.

[59] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*. MIT press, 1992, vol. 1.

[60] R. Escobedo, D. Dutykh, C. Muro, L. Spector, and R. Coppinger, "Group size effect on the success of wolves hunting," *arXiv preprint arXiv:1508.00684*, 2015.

[61] C. Muro, R. Escobedo, L. Spector, and R. Coppinger, "Wolf-pack (canis lupus) hunting strategies emerge from simple rules in computational simulations," *Behavioural Processes*, vol. 88, no. 3, pp. 192 – 197, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0376635711001884

[62] J. Zhong, L. Luo, W. Cai, and M. Lees, "Automatic rule identification for agent-based crowd models through gene expression programming," in *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2014, pp. 1125–1132.

[63] S. M. Manson, "Agent-based modeling and genetic programming for modeling land change in the southern yucatan peninsular region of mexico," *Agriculture, ecosystems & environment*, vol. 111, no. 1, pp. 47–62, 2005.

[64] ——, "Bounded rationality in agent-based models: experiments with evolutionary programs," *International Journal of Geographical Information Science*, vol. 20, no. 9, pp. 991–1012, 2006.

[65] S.-H. Chen and C.-H. Yeh, "Genetic programming in the agent-based modeling of stock markets," in *Proceedings of the Fifth International Conference on Computing in Economics and Finance, Boston College, MA, USA*, 1999.

[66] L. An, "Modeling human decisions in coupled human and natural systems: review of agent-based models," *Ecological Modelling*, vol. 229, pp. 25–36, 2012.

[67] V. Grimm, U. Berger, D. L. DeAngelis, J. G. Polhill, J. Giske, and S. F. Railsback, "The odd protocol: a review and first update," *Ecological modelling*, vol. 221, no. 23, pp. 2760–2768, 2010.

[68] B. Müller, F. Bohn, G. Dreßler, J. Groeneveld, C. Klassert, R. Martin, M. Schlüter, J. Schulze, H. Weise, and N. Schwarz, "Describing human decisions in agent-based models–odd+ d, an extension of the odd protocol," *Environmental Modelling & Software*, vol. 48, pp. 37–48, 2013.

[69] J. Whitmeyer, *The ACSES Model of Afghanistan: The Model Operation, Synthetic Population, Calibration, and Surprises*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 59–83. [Online]. Available: https://doi.org/10.1007/978-3-642-39295-5_6

[70] J. Petke, S. O. Haraldsson, M. Harman, W. B. Langdon, D. R. White, and J. R. Woodward, "Genetic improvement of software: a comprehensive survey," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 3, pp. 415–432, 2017.

[71] W. B. Langdon, B. Y. H. Lam, J. Petke, and M. Harman, "Improving cuda dna analysis software with genetic programming," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2015, pp. 1063–1070.

[72] J. Petke, M. Harman, W. B. Langdon, and W. Weimer, *Using Genetic Improvement and Code Transplants to Specialise a C++ Program to a Problem Class*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 137–149. [Online]. Available: https://doi.org/10.1007/978-3-662-44303-3_12

[73] W. B. Langdon and R. Poli, "Fitness causes bloat: Mutation," in *European Conference on Genetic Programming*. Springer, 1998, pp. 37–48.

[74] W. Banzhaf and W. B. Langdon, "Some considerations on the reason for bloat," *Genetic Programming and Evolvable Machines*, vol. 3, no. 1, pp. 81–91, 2002.

[75] N. T. Hien, N. X. Hoai, and B. McKay, "A study on genetic programming with layered learning and incremental sampling," in *Evolutionary Computation (CEC), 2011 IEEE Congress on*. IEEE, 2011, pp. 1179–1185.

[76] P. Stone and M. Veloso, *Layered Learning*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 369–381. [Online]. Available: https://doi.org/10.1007/3-540-45164-1_38

[77] S. Gustafson and W. Hsu, "Layered learning in genetic programming for a cooperative robot soccer problem," *Genetic Programming*, pp. 291–301, 2001.

[78] S. Luke, L. Panait, G. Balan, S. Paus, Z. Skolicki, J. Bassett, R. Hubley, and A. Chircop, "Ecj: A java-based evolutionary computation research system," *Downloadable versions and documentation can be found at the following url: http://cs.gmu.edu/eclab/projects/ecj*, 2006.

[79] E. Pantridge and L. Spector, "Pyshgp: Pushgp in python," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 2017, pp. 1255–1262.

[80] D. Rainville, F.-A. Fortin, M.-A. Gardner, M. Parizeau, C. Gagné *et al.*, "Deap: enabling nimbler evolutions," *ACM SIGEVOlution*, vol. 6, no. 2, pp. 17–26, 2014.

[81] K. Veeramachaneni, I. Arnaldo, O. Derby, and U.-M. O'Reilly, "Flexgp," *Journal of Grid Computing*, vol. 13, no. 3, pp. 391–407, 2015.

[82] I. Arnaldo, K. Veeramachaneni, A. Song, and U.-M. O'Reilly, "Bring your own learner: A cloud-based, data-parallel commons for machine learning," *IEEE Computational Intelligence Magazine*, vol. 10, no. 1, pp. 20–32, 2015.

[83] P. Salza, E. Hemberg, F. Ferrucci, and U.-M. O'Reilly, "Towards evolutionary machine learning comparison, competition, and collaboration with a multi-cloud platform," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 2017, pp. 1263–1270.

[84] W. B. Langdon and M. Harman, "Optimizing existing software with genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 1, pp. 118–135, 2014.

[85] C. Gunaratne and I. Garibay, "Alternate social theory discovery using genetic programming: towards better understanding the artificial anasazi," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2017, pp. 115–122.

[86] G. Louppe, "Understanding random forests: From theory to practice," Ph.D. dissertation, Université de Liège, Liège, Belgique, 2014, chapter 6.1.2: Importances in forests.

[87] L. Breiman, "Manual on setting up, using, and understanding random forests v3. 1," *Statistics Department University of California Berkeley, CA, USA*, vol. 1, 2002.

[88] R. Genuer, J.-M. Poggi, and C. Tuleau-Malot, "Variable selection using random forests," *Pattern Recognition Letters*, vol. 31, no. 14, pp. 2225–2236, 2010.

[89] F. Hutter, H. Hoos, and K. Leyton-Brown, "An efficient approach for assessing hyperparameter importance," in *International Conference on Machine Learning*, 2014, pp. 754–762.

[90] N. Dang and P. De Causmaecker, "Analysis of algorithm components and parameters: some case studies," 2018, accepted at the International Conference on Learning and Intelligent Optimization (LION12).

[91] A. Saabas, "Interpreting random forests," 2014. [Online]. Available: https://blog.datadive.net/interpreting-random-forests/

[92] ——, "andosa/treeinterpreter," May 2019. [Online]. Available: https://github.com/andosa/treeinterpreter

[93] S. M. Lundberg, G. G. Erion, and S.-I. Lee, "Consistent individualized feature attribution for tree ensembles," *arXiv preprint arXiv:1802.03888*, 2018.

[94] E. M. Smith, A. Nantes, A. Hogue, and I. Papas, "Forecasting customer behaviour in constrained e-commerce platforms," in *8th International Conference of Pattern Recognition Systems (ICPRS 2017)*. IET, 2017, pp. 1–8.

[95] M. Beillevaire, "Inside the black box: How to explain individual predictions of a machine learning model : How to automatically generate insights on predictive model outputs, and gain a better understanding on how the model predicts each individual data point." Master's thesis, KTH, School of Electrical Engineering and Computer Science (EECS), 2018.

[96] C. Rea, K. Erickson, R. Granetz, R. Johnson, N. Eidietis, K. Montes, and R. Tinguely, "Initial results of a machine learning-based real time disruption predictor on diii-d," in *Proc. 45th EPS Conf. on Plasma Physics, Europhysics Conf. Abstracts*, vol. 42, 2018.

[97] R. Granetz, C. Rea, K. Montes, R. TINGUELY, N. EIDIETIS, O. MENEGHINI, D. CHEN, B. SHEN, B. XIAO, K. ERICKSON *et al.*, "Machine learning for disruption warning on alcator c-mod, diii-d, and east tokamaks," in *Proc. 27th IAEA Fusion Energy Conference, IAEA, Vienna*, 2018.

[98] X. Morice-Atkinson, B. Hoyle, and D. Bacon, "Learning from the machine: interpreting machine learning algorithms for point-and extended-source classification," *Monthly Notices of the Royal Astronomical Society*, vol. 481, no. 3, pp. 4194–4205, 2018.

[99] E. Bastrakova, "Improving interpretability of complex predictive models," Master's thesis, Universitat Politècnica de Catalunya, 2017.

[100] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, jul 2012.

[101] Y. Hold-Geoffroy, O. Gagnon, and M. Parizeau, "Once you scoop, no need to fork," in *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment*. ACM, 2014, p. 60.

[102] 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*, vol. 12, pp. 2825–2830, 2011.

[103] M. Korobov and K. Lopuhin, "Permutation importance," Apr 2019. [Online]. Available: https://eli5.readthedocs.io/en/latest/blackbox/permutation_importance.html

[104] T. C. Schelling, "Dynamic models of segregation," *Journal of mathematical sociology*, vol. 1, no. 2, pp. 143–186, 1971.

[105] E. Hatna and I. Benenson, "Combining segregation and integration: Schelling model dynamics for heterogeneous population," *arXiv preprint arXiv:1406.5215*, 2014.

[106] ——, "The schelling model of ethnic residential dynamics: Beyond the integrated-segregated dichotomy of patterns," *Journal of Artificial Societies and Social Simulation*, vol. 15, no. 1, p. 6, 2012.

[107] M. Granovetter, "Threshold models of collective behavior," *American journal of sociology*, vol. 83, no. 6, pp. 1420–1443, 1978.

[108] F. M. Bass, "A new product growth for model consumer durables," *Management science*, vol. 15, no. 5, pp. 215–227, 1969.

[109] J. Goldenberg, B. Libai, and E. Muller, "Talk of the network: A complex systems look at the underlying process of word-of-mouth," *Marketing Letters*, vol. 12, no. 3, pp. 211–223, 8 2001. [Online]. Available: https://doi.org/10.1023/A:1011122126881

[110] W. Rand, J. Herrmann, B. Schein, and N. Vodopivec, "An agent-based model of urgent diffusion in social media," *Journal of Artificial Societies and Social Simulation*, vol. 18, no. 2, p. 1, 2015.

[111] N. Cowan, "What are the differences between long-term, short-term, and working memory?" *Progress in brain research*, vol. 169, pp. 323–338, 2008.

[112] A. Baddeley, "Working memory: Theories, models, and controversies," *Annual review of psychology*, vol. 63, pp. 1–29, 2012.

[113] Z. Chen and N. Cowan, "Chunk limits and length limits in immediate recall: a reconciliation." *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 31, no. 6, p. 1235, 2005.

[114] G. A. Miller, "The magical number seven, plus or minus two: Some limits on our capacity for processing information." *Psychological review*, vol. 63, no. 2, p. 81, 1956.

[115] N. Cowan, "The magical number 4 in short-term memory: A reconsideration of mental storage capacity," *Behavioral and brain sciences*, vol. 24, no. 1, pp. 87–114, 2001.

[116] N. Cowan, C. C. Morey, A. M. AuBuchon, C. E. Zwilling, and A. L. Gilchrist, "Seven-year-olds allocate attention like adults unless working memory is overloaded," *Developmental science*, vol. 13, no. 1, pp. 120–133, 2010.

[117] R. W. Belk, "Extended self in a digital world," *Journal of Consumer Research*, vol. 40, no. 3, pp. 477–500, 2013.

[118] R. W. Clowes, "Extended memory," *Routledge handbook on the philosophy of memory*, pp. 243–255, 2017.

[119] U. Schultze, "Embodiment and presence in virtual worlds: a review," *Journal of Information Technology*, vol. 25, no. 4, pp. 434–449, 2010.

[120] R. Belk, "Digital consumption and the extended self," *Journal of Marketing Management*, vol. 30, no. 11-12, pp. 1101–1118, 2014.

[121] J. A. Bargh, K. Y. McKenna, and G. M. Fitzsimons, "Can you see the real me? activation and expression of the "true self" on the internet," *Journal of social issues*, vol. 58, no. 1, pp. 33–48, 2002.

[122] T. L. Taylor, "Living digitally: Embodiment in virtual worlds," in *The social life of avatars*. Springer, 2002, pp. 40–62.

[123] L. P. Tosun, "Motives for facebook use and expressing "true self" on the internet," *Computers in Human Behavior*, vol. 28, no. 4, pp. 1510–1517, 2012.

[124] S. L. Buglass, J. F. Binder, L. R. Betts, and J. D. Underwood, "Motivators of online vulnerability: The impact of social network site use and fomo," *Computers in Human Behavior*, vol. 66, pp. 248–255, 2017.

[125] S. Hongladarom, "Personal identity and the self in the online and offline world," *Minds and Machines*, vol. 21, no. 4, p. 533, 2011.

[126] C. Gunaratne, C. Senevirathna, C. Jayalath, N. Baral, W. Rand, and I. Garibay, "A multi-action cascade model of conversation," in *5th International Conference on Computational Social Science*, 2019. [Online]. Available: http://app.ic2s2.org/app/sessions/9kXqn5btgKKC5yfCvg/details

[127] G. Pask, "Conversation theory," *Applications in Education and Epistemology*, 1976.

[128] T. Schreiber, "Measuring information transfer," *Physical review letters*, vol. 85, no. 2, p. 461, 2000.

[129] C. Gunaratne, "Multi-action cascade model source code," Sep 2019. [Online]. Available: https://github.com/chathika/MACM

[130] A. D. Baddeley and G. Hitch, "Working memory. the psychology of learning and motivation," *New York, NY: Academicp*, 1974.

[131] M. Gomez-Rodriguez, K. P. Gummadi, and B. Schoelkopf, "Quantifying information overload in social media and its impact on social contagions." in *ICWSM*, 2014, pp. 170–179.

[132] K. Koroleva, H. Krasnova, and O. Günther, "'stop spamming me!': exploring information overload on facebook," in *Proceedings of the Association for Information Systems (AMCIS) 2010*, no. 447, 2010. [Online]. Available: https://aisel.aisnet.org/amcis2010/447

[133] P. Li, W. Li, H. Wang, and X. Zhang, "Modeling of information diffusion in twitter-like social networks under information overload," *The Scientific World Journal*, vol. 2014, 2014.

[134] C. Gunaratne, N. Baral, W. Rand, I. Garibay, C. Jayalath, and C. Senevirathna, "A theory of extended working memory and its role in online conversation dynamics," *arXiv preprint arXiv:1910.09686*, 2019.

[135] N. O. Hodas, F. Kooti, and K. Lerman, "Friendship paradox redux: Your friends are more interesting than you," in *Seventh International AAAI Conference on Weblogs and Social Media*, 2013.

[136] K. Lerman and A. Galstyan, "Analysis of social voting patterns on digg," in *Proceedings of the first workshop on Online social networks*. ACM, 2008, pp. 7–12.

[137] D. Masad, *Py2NetLogo*, 2016. [Online]. Available: https://github.com/dmasad/Py2NetLogo

[138] B. Dagenais, *Py4J: A Bridge between Python and Java*, 2009–2015. [Online]. Available: https://www.py4j.org

[139] P. S. Foundation, *PyPI - the Python Package Index*, Accessed 29 May, 2018. [Online]. Available: https://pypi.org/

[140] C. Gunaratne, *NL4Py 0.5.0*, 2018. [Online]. Available: https://pypi.org/project/NL4Py/