# Unreal to Real: Current Abilities and Knowledge Gaps for Measuring the Realism of Social Simulations

Mathew Titus[1], George I. Hagstrom[2], James R. Watson[2*]

May 2021

[1] The Prediction Lab LLC

[2] College of Earth, Ocean and Atmospheric Sciences; Oregon State University

[*] james.watson@oregonstate.edu

## Introduction

Many of the most important systems that govern our modern lives are complex adaptive systems (CASs) ? large, complex, interconnected networks of actors and elements such as the internet, social-networks, ecosystems, transportation infrastructure, or (financial) market systems. Current practices in scientific research have made great progress in utilizing agent-based modeling and gathering ?big data? on simulated analogs of these systems. However, these methods often have difficulty reproducing the most interesting aspects of the systems, those elements which arise due to agent adaptation, system evolution, or feedback loops, and so on. These features can lead to emergent and/or nonlinear phenomena of great interest and they may occur only rarely. Faithfully modeling these systems to enable anticipation of these effects is crucial in order to determine when and where the system as a whole may change course. There is an extreme challenge to characterizing and comparing complex social systems (and the various models we construct for them): unlike purely physical systems, complex social-systems are comprised of numerous and overlapping local causal neighborhoods (or in other words, a group of system elements that work in concert to control another agent or group?s behavior), and there are no clear delineations/borders between subsystems. It is not at all obvious to know how well a given model captures the progression from local interactions to multiscale feedback, coherent group action, or a critical shift in a macroscopic property.

In this project, our aim was to answer the following research questions: 1) Can we develop formal methods/metrics that can be used to rank models/simulations based on how well they capture/reflect/allow for the understanding of the complexity of a given real-world system? 2) In a competition of models, are there methods or metrics to evaluate which model/simulation is closest to the truth? These questions relate to whether a real-world social system that is ?open? can ever be accurately or usefully modeled, where models are in essence ?closed-world? approximations.

Any complex system can be thought of as being comprised of numerous local causal neighborhoods (LCNs), and these modules interact with each other over time and across scales and can come and go as system dynamics evolve. Models are typically created to capture a particular dynamic, and then draw conclusions on the mean behavior of the entire population. But what if non-linearity or hidden structure in the empirical system create a disparity between those low-level interactions which our model represents well and the phenomenon of interest? New methods for comparing systems based on their causal structure, rather than their accuracy in representing any particular aspect of the system, are being developed and this project aims to identify the limits to the state of the art in this respect. We will test methods for identifying exactly which LCNs are being represented, and in doing we will be able to rank models based on their usefulness for answering a given question. More specifically, to answer the research questions above this project will:

1. Develop new approaches to deconstructing data from complex systems (or a model of a complex system) into LCNs;
2. Create new methods for mapping LCNs to one another and
3. Advance new metrics of the goodness and dimensions of fit of models of a given complex system.

These problems have been partially addressed in their own right. Local causal neighborhood or "Markov blanket" detection (Friston 2010, 2012, 2013; Aliferis 2003, 2010; Tsamardinos 2003a, 2003b) and causal Bayesian network discovery (Ellis 2012; Mani 2004; Guyon 2007) have been shown to extract variables? causal dependencies from raw data. This procedure can serve as a generator of group-level features (the Markov blanket and its interior state). These high-level features are the units whose dynamics we want to learn. The tools of unsupervised feature extraction (see e.g. Robnik-Šikonja 2003; Wu 2018; Titus 2019) have been used to perform feature discovery and/or determine feature relevance in the face of a large number of redundant or irrelevant features. These tools then allow us to build up the causal network of a complex dynamical system, extract the interacting units, and learn the policies that control the evolution of the units? states. Finally, by transferring these policies to a target system which may be open, it is

possible to measure a given model?s ability to anticipate causal events at a variety of scales, giving a quantitative measure of model fitness that respects real-world complexity.

In this paper we describe advances to these approaches to deconstructing data from complex and adaptive social-systems. Our overarching goal is to identify the state of the art in terms of our ability to characterize complex adaptive social systems, whether real or modeled, and in doing so providing an ability to answer the research questions stated above. Another major goal is to use this exercise to identify knowledge gaps and research opportunities for DARPA to pursue.

## Research Approach

To objectively and quantitatively compare complex adaptive social-systems we have focused on methods that reveal causal structure. Instead of purely statistical approaches used both classically and in relatively new machine learning approaches, for example that leverage various measures of error or accuracy: R2, mean-square error, receiver-operator characteristic, F1 score... etc., our approach leverages new abilities to detect the causal structure of a give complex system. Specifically, we have explored abilities to describe complex adaptive social-systems in terms of:

- Actors and relationships,
- Policies,
- Actions,
- States.

We assume all complex adaptive social-system are comprised of distinct actors that influence each other through various relationships. This is a standard and basic representation of a complex adaptive social-system. We go further and assume that each actor has a **state** that they occupy at any given point in time, **actions** that they can take, and policies that govern which actions are taken. Policies can be thought of as discrete probability density functions, as in Fig. 1A. An individual's policy (i.e. the probability of taking a certain action) changes over time, as a function of the environment they find themselves in. The **environment** is strictly defined in terms of other actors that are within the subject actor's Local Causal Neighborhood (see Fig. 1B) . This is happening for all actors comprising the system, constantly through time. As a consequence one can imagine the dynamic process described in Fig. 1C.

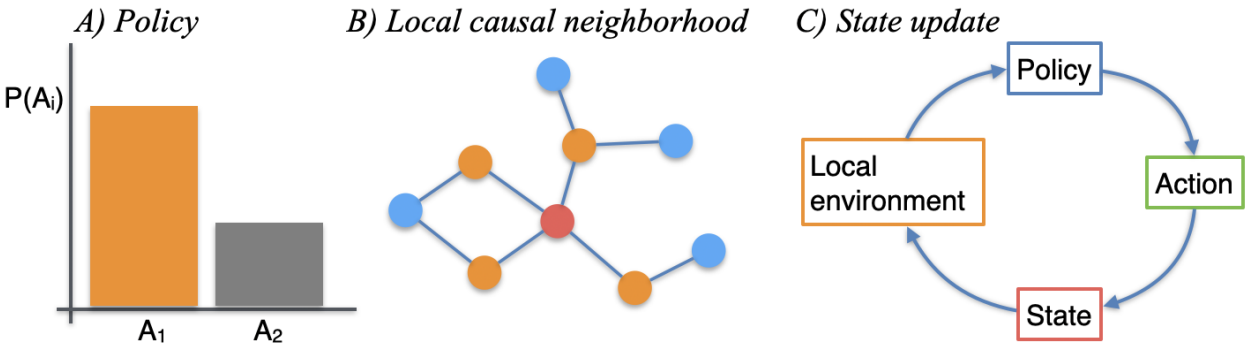


Figure 1: A) Policies for a given actor are simply the probabilities of taken various actions $A_i$. B) The probabilities that comprise a given actor's policy are influenced by other nearby actors – i.e. those within the local causal neighborhood. C) Ultimately, policies determine which actions are taken, which impact the state of each actor, which updates the local environment (i.e. the Local Causal Neighborhood) of each actor. The whole process repeats itself.

Given this framing of the complexity of social-systems, we are able to define a simple typology that spans two axes: the first axis is the dimensionality $D$ of the policy/action space. Essentially this is the number of distinct actions that a given actor may take. The second axis is the size of the system in terms of the number of actors $N$. In this $N$ and $D$ space, we have defined three end-member case-studies: 1) a simple low $N$ and low $D$ flocking model of collective behavior; 2) competitive sports systems with low $N$ but high $D$; 3) online social-networks such as Twitter that have low $D$ but high $N$ and finally 4) real-world system that have both high $D$ and high $N$. In this project, we have used case-studies 1, 2 and 3 to answer the research questions posed above. Certainly, a major step forward will be to apply the methods developed and tested in this project on data from a (large, dirty and uncertain) real-world system (see the discussion section for more on this).


Figure 2: Typology of complex adaptive social-systems based on the dimensionality of the policy/action space and the size of the system (in terms of the number of actors).

# 1 Theory I: Causality and its Measurement

## 1.1 Definitions and background

To start, we review the fundamental objects required to study causal networks, touching on the integral probability metric approach to causal detection.

### 1.1.1 Probability and causation

**Definition 1 (Probability space)** *Let $\Omega$ denote a set with a Borel $\sigma$-algebra $\mathcal{B}$, representing the collection of outcomes that might be realized by the system under study. A probability measure $P_\Omega$, or $P$ if the space is understood, is a Borel measure mapping elements of $\mathcal{B}$ to $[0,1]$ with $P(\emptyset) = 0$, $P(\Omega) = 1$, and subadditivity: $P(\cup_i A_i) = \sum_i P(A_i)$ if the sets $A_i$ are disjoint. The triple $(\Omega, \mathcal{B}, P)$ is a* **probability space**.

In the following capital letters $X$, $Y$, ... will refer to random variables and lowercase letters $x$, $y$, ... will denote sampled values of those random variables. In particular, if $X$ is a real-valued random variable, $X : \Omega \to \mathbb{R}$, then the real value $X(\omega), \omega \in \Omega$ is simply written $x$. We assume the reader is familiar with the notation for conditional probability distributions and use the language "probability distribution of $X$ given $Y$" to refer to the distribution $P(X|Y)$. The symbol $\mathbb{X} := \{X, Y, \dots\}$ will denote the collection of all random variables that we define on the probability space. The capital letters $U$, $V$, and $W$ will refer to **events**, or subsets of $\Omega$ that are measurable in the above sense.

While the individual possible outcomes $\omega$ are the fundamental expression of the system's state, we are interested in a family of variables $\{X_i\}_{i\in\mathcal{I}}$ for an index set $\mathcal{I}$ and how those variables relate to each other. For this reason we will work with the $\sigma$-subalgebra of $\mathcal{B}$ generated by $\mathbb{X} = \{X_i\}_\mathcal{I}$, meaning we restrict our attention to the topology generated by sets of the form $\prod_{i\in\mathcal{I}} X_i^{-1}(B_i)$ where $B_i \subset \mathbb{R}$ is open. We call such sets **observable**, as they are defined in terms of the variables we measure, and in the following values of $\mathbb{X}$ will be referred to as outcomes, observations, or events.

We say that a variable $Y$ **causes** $X$ if the distribution of $X$ given $Y$ varies with the value of $Y$: for some $y_1, y_2 \in \mathbb{R}$,

$$P(X|Y = y_1) \neq P(X|Y = y_2).$$

For example, if $X$ describes whether or not an individual has lung cancer ($X = 0$ if no tumor has formed, $X = 1$ otherwise) and $Y$ describes the number of cigarettes smoked per year by the individual, we find in practice that $P(X = 1|Y)$ increases greatly as $Y$ increases. This indicates that $Y$ (smoking) causes $X$.

Of course, the causation may not be direct, meaning that there may be other variables $Z$ which, if controlled for, would account for the causal link from $Y$ to $X$. (For example, $Z$ could be the number of mutations that have occurred in the lung tissue. For smokers where $Z$ remains on par with the general non-smoking population, those with very large $Y$ value may show no increase in $P(X = 1|Y, Z)$.) In this case we would say that $Y$ is an **indirect cause** of $X$, and we express this mathematically as conditional independence.

**Definition 2** *Two variables $X$ and $Y$ are* **conditionally independent** *given the variable set $Z$, expressed either as $X \perp\!\!\!\perp Y|Z$ or as $I(X, Y|Z)$, if the conditioned distribution $P(X \in U, Y \in V|Z \in W)$ factors as*

$$P\left(X \in U, Y \in V|Z \in W\right) = P(X \in U|Z \in W) \cdot P(Y \in V|Z \in W)$$

*for all values $x$, $y$, and $z$ such that $P(Z \in U) > 0$.*

If no such $Z$ exists within our set of random variables $\mathbb{X}$ we say that $Y$ is a **direct cause** of $X$.

### 1.1.2 Causal networks

We will now identify the elements of our system (random variables $X_1$, $X_2$, ...) as vertices in a directed acyclic graph (DAG). Consider the graph $G = (\mathcal{V}, \mathcal{E})$ with vertices $\mathcal{V}$ and edges $e \in \mathcal{E}$ representing ordered pairs $(v_i, v_j)$ of these vertices. The notation $v \to w$ indicates that the vertices $v$ and $w$ are connected by an edge $(v, w) = e \in \mathcal{E}$ originating at $v$ and terminating at $w$. We associate the system's individual random variables to individual vertices in $\mathcal{V}$ in a one-to-one manner ($X_i \mapsto v_i \in \mathcal{V}$), and recall $P$ is the probability distribution defining the joint distribution of the $X_i$.

**Definition 3** *For a vertex $v$ in $\mathcal{V}$ we write $Pa(v)$ for the* **parents** *of $v$, $\{y \in \mathcal{V} : (y, v) \in \mathcal{E}\}$, and $Ch(v)$ for its* **children**, *$\{y \in \mathcal{V} : (v, y) \in \mathcal{E}\}$. The set $PC(v)$ is simply the union $Pa(v) \cup Ch(v)$.*

The following definition connects the network structure of $G$ to the dependence structure of $P$.

**Definition 4** *The graph $G$ satisfies the* **Markov condition** *(with respect to the joint distribution $P$) if a vertex $v \in \mathcal{V}$ is independent of its non-descendants when conditioned on $Pa(v)$, the parents of $v$. In particular, if $Y \notin PC(X)$, the condition*

$$X \perp\!\!\!\perp Y|PC(X). \tag{1}$$

*holds. In this case the couple $\langle G, P\rangle$ is called a* **Bayesian network**.

Figure 3: Left: An example Bayesian network. Right: the CPN for the same system.

Note that a DAG that satisfies the Markov condition for our system, i.e. a Bayesian network for $(\Omega, \mathcal{B}, P)$, can have many superfluous edges. Indeed, adding another edge to $G$ connecting a variable $Y$ to $X$ simply lowers the number of required conditional independence statements (as $Y$ is now in $PC(X)$), and so the modified network is also a Bayesian network for $P$. For this reason we work with the following definition:

**Definition 5** *A Bayesian network $\langle G, P\rangle$ such that every edge $(v_1, v_2) \in \mathcal{E}$ represents a direct cause is called a* **causal probabilistic network**, *or CPN.*

**Definition 6 (Faithful Distribution)** *The graph $G = (\mathcal{V}, \mathcal{E})$ is faithful to the probability measure $P$ over the random variables $\mathbb{X}$ if and only if its structure captures every independence relation among the elements of $\mathbb{X}$: if $X_1, X_2 \in \mathbb{X}$ are associated to $v_1, v_2 \in \mathcal{V}$, respectively. Then $(v_1, v_2) \notin \mathcal{E}$ if and only if there exists $Z \subset \mathbb{X}$ such that $X_1 \perp\!\!\!\perp X_2 | Z$.*

The joint probability distribution $P$ may have several CPNs, but different networks may all provide faithful representations of the statistical properties of $P$.

### 1.1.3 Integral probability metrics

Recall that testing the conditional independence statement $X \perp\!\!\!\perp Y|Z$ can be done, in essence, by fixing the value of variable(s) $Z$ and showing that the distribution of $X$ is unchanged as $Y$ varies. (In this case, either $Y$ is not a cause of $X$ or it is an indirect cause that is mediated by the variable(s) $Z$, hence $X$ being unaffected as long as $Z$ is fixed.) For this reason we turn to integral probability metrics (IPMs), tools designed to measure the distance between probability distributions.

Symbolically, our task boils down to determining whether the distribution $P(X, Y = y_1|Z = z)$ is the same as $P(X, Y = y_2|Z = z)$ for each choice of $y_1 \neq y_2$ and $z$. In general, this is impractical for most systems which we can only passively observe since we lack the ability to measure samples with $Z$ fixed at the value $z$ while $Y$ varies over a representative sampling of its distribution. However, when observing simulated or controlled environments this is possible, the only cost being the time and energy needed to conduct repeated trials.

The metric on distributions that we consider in the following is the Wasserstein metric:

**Definition 7** *Given a complete, separable metric space $E$ with metric $d_E$ and $\sigma$-algebra $\mathcal{B}$, write $\mathcal{M}$ for the space of all probability distributions on $E$. If $\nu_1$, $\nu_2$ are two distributions in $\mathcal{M}$, define the Wasserstein distance to be*

$$W_d(\nu_1, \nu_2) := \inf_{\pi} \left( \mathbb{E}_\pi \left[ d_\Omega(x, y) \right] \right) \tag{2}$$

*where the infimum is over distributions $\pi$ on $E \times E$ with marginals $\nu_1$ and $\nu_2$ in the first and second coordinates, respectively.*

IPMs originated as a theoretical implement, used in various transport problems [17]. However, by applying the Kantorovich-Rubinstein duality theorem we can express 2 instead as

$$W_d(\nu_1, \nu_2) = \sup_{f \in \mathcal{F}_L} \left| \int_E f d\nu_1 - \int_E f d\nu_2 \right|, \tag{3}$$

where $\mathcal{F}_L$ is the family of Lipschitz functions, $\{f \in C_E : |f(x) - f(y)| \leq d_E(x, y)\}$. As described in [12], this is amenable to estimation through essentially a Monte Carlo sampling method: Let $\{x^{(i)}\}_{i=1}^{N_1}$ be independent samples from $\nu_1$, and $\{x^{(N_1+j)}\}_{j=1}^{N_2}$ independent samples from $\nu_2$. Then writing $\hat{\nu_i}$ for the empirical measure drawn from of $\nu_i$, the estimator

$$\hat{W}_d(\hat{\nu_1}, \hat{\nu_2}) := \sup_{\alpha} \left( \frac{1}{N_1} \sum_{i=1}^{N_1} \alpha_i - \frac{1}{N_2} \sum_{j=1}^{N_2} \alpha_{N_1+j} \right), \tag{4}$$

where the sup is taken over

$$\left\{ \alpha = (\alpha_i)_{i=1}^{N_1+N_2} : |\alpha_i - \alpha_j| \leq d_E(x^{(i)}, x^{(j)}) \text{ for all } i, j \right\}, \tag{5}$$

converges to $W_d(\nu_1, \nu_2)$ as $N_1, N_2 \to \infty$. [12]

Convergence here is a result of the discrete measure $\hat{\nu_1} := \frac{1}{N_1} \sum_{i=1}^{N_1} \delta_{x^{(i)}}$ converging weakly to $\nu_1$:

$$\int_E f d\hat{\nu_1} = \frac{1}{N_1} \sum_{i=1}^{N_1} \int_E f(y) \delta_{x^{(i)}}(y) = \frac{1}{N_1} \sum_{i=1}^{N_1} f(x^{(i)}) \to \int_E f d\nu_1$$

and likewise for $\hat{\nu_2} \to \nu_2$. Note that the constraint on $\alpha$ is precisely the condition that $\alpha$ is the restriction of a Lipschitz function $f$ on $E$ to the points $\{x^{(i)}\}$.

### 1.1.4 The EZK dependency measure

The Wasserstein metric provides a basic measure of distance to investigate how the (family of) distributions $P(X, Y = y | Z = z)$ varies with $y$; the paper [3] defines a new, asymmetric dependency measure based on this IPM, which quantifies the influence of $Y$ on $X$ given $Z$. We call this the

Etesami-Zhang-Kiyavash Dependency Measure, or EZK measure, after the paper's authors. Among other things, this measure is computable for systems where one can perform interventions, and the computational complexity does not increase with the dimension of the space $E$. However, the major benefit of using the EZK measure is that, in addition to detecting direct causal relationships, it explicitly determines the direction of the dependence. We describe now its form and a method of estimation as described in [3].

To begin, we introduce some new notation particular to this section. Let $\mathbb{X} = \{X_i\}$ be the set of observed random variables taking values in the complete, separable metric space $E$. We will write $\underline{x}_{\mathcal{K}} := (\underline{x}_i)_{i\in\mathcal{K}}$ for a set of values in $E$ that will be used to condition the variables $X_i, i \in \mathcal{K}$. We also write $\underline{x}_{\mathcal{K}}(i)$ for the $i^{\text{th}}$ coordinate, $\underline{x}_i$. Here $\mu_i \in \mathcal{M}$ denotes the distribution of $X_i$, and we write $\mu_i(\underline{x}_{\mathcal{K}})$ for the probability distribution of $X_i$ when the measured system state $\mathbb{X}$ is conditioned on the event $\{X_j = \underline{x}_j \text{ for each } j \in \mathcal{K}\}$, $i \notin \mathcal{K}$.

**Definition 8** *Define the set*

$$\mathcal{C}(x,y) = \mathcal{C}^{\mathcal{K}}_{i,j}(x,y) = \left\{ \left(\underline{x}_{\mathcal{K}\cup\{j\}}, \underline{y}_{\mathcal{K}\cup\{j\}}\right) \middle|\ \underline{x}_{\mathcal{K}\cup\{j\}}(m) = \underline{y}_{\mathcal{K}\cup\{j\}}(m) \text{ for } m \in \mathcal{K};\ \underline{x}_{\mathcal{K}\cup\{j\}}(j) = x;\ \underline{y}_{\mathcal{K}\cup\{j\}}(j) = y \right\}$$

*to be ordered pairs of conditioning values that agree on $\mathcal{K}$ but take on the values $x, y \in E$, respectively, in the $j$ coordinate. Then the* **EZK measure** *of the conditional dependence of $X_i$ on $X_j$ conditioned on $\{X_k : k \in \mathcal{K}\}$ is given by*

$$c^{\mathcal{K}}_{i,j} := \sup_{\mathcal{C}(x,y)} \frac{W_d\left(\mu_i\left(\underline{x}_{\mathcal{K}\cup\{j\}}\right), \mu_i\left(\underline{y}_{\mathcal{K}\cup\{j\}}\right)\right)}{d_E(x,y)}. \tag{6}$$

Notice that if $X_i$ and $X_j$ are independent given variables indexed in $\mathcal{K}$, this measure is zero. In the following we consider a value of 1 or more to indicate a notable causal relationship. For examples of the additional favorable traits of this measure and examples of its use, see [3].

As with the Wasserstein IPM estimator of (4), we can construct an approximation of $c^K_{i,j}$ using random sampling. One produces independent samples $\{\underline{z}^{(k)}\}$ of the system $\mathbb{X}$ without conditioning. The values of $X_m, m \in \mathcal{K}\cup\{j\}$ then are recorded as $\left\{\underline{x}^{(k)}_{\mathcal{K}\cup\{j\}}\right\}$, and for each $(k,\ell)$ ordered pair we create the condition set

$$\underline{x}^{(k)}_{\mathcal{K}\cup\{j\}} = \begin{cases} \underline{z}^{(k)}_m \text{ for } m \in \mathcal{K}, \\ \underline{z}^{(k)}_j \text{ for } m = j, \end{cases} \qquad \underline{y}^{(\ell)}_{\mathcal{K}\cup\{j\}} = \begin{cases} \underline{z}^{(k)}_m \text{ for } m \in \mathcal{K}, \\ \underline{z}^{(\ell)}_j \text{ for } m = j. \end{cases}$$

Note that $\underline{y}^{(\ell)}_{\mathcal{K}\cup\{j\}}$ differs from $\underline{x}^{(k)}_{\mathcal{K}\cup\{j\}}$ only at $j$. Then one may calculate $\hat{W}_d\left(\hat{\mu}_i(\underline{x}^{(k)}_{\mathcal{K}\cup\{j\}}), \hat{\mu}_i(\underline{y}^{(\ell)}_{\mathcal{K}\cup\{j\}})\right)$, measuring how much $\mu_i$ changes when $X_j$ changes from $\underline{z}^{(k)}_j$ to $\underline{z}^{(\ell)}_j$, while $\{X_k : k \in \mathcal{K}\}$ is fixed at $\underline{z}^{(k)}_{\mathcal{K}}$.

Suppose we construct $N$ such samples $\left\{\underline{x}^{(k)}_{\mathcal{K}\cup\{j\}}\right\}$. Then [3]claims the estimator

$$\hat{c}^{\mathcal{K}}_{i,j} := \max_{1\le k\neq\ell\le N} \frac{\hat{W}_d\left(\hat{\mu}_i\left(\underline{x}^{(k)}_{\mathcal{K}\cup\{j\}}\right), \hat{\mu}_i\left(\underline{y}^{(\ell)}_{\mathcal{K}\cup\{j\}}\right)\right)}{d_E\left(\underline{z}^{(k)}_j, \underline{z}^{(\ell)}_j\right)} \tag{7}$$

will converge to the EZK measure defined in (6) as $N, N_1, N_2 \to \infty$.

### 1.1.5 Computation of the EZK measure

Unfortunately, as written this estimator exhibits a numerical instability; for samples $\underline{z}^{(k)}$, $\underline{z}^{(\ell)}$ whose realizations of $X_j$ are very close, the small size of the denominator $d_E\left(\underline{z}^{(k)}_j, \underline{z}^{(\ell)}_j\right)$ controls the computed value of $\hat{c}^{\mathcal{K}}_{i,j}$. For simplicity let's take $N_1 = N_2$. If $X_1$ and $X_2$ are independent standard normal variables, and we consider $\mathcal{K} = \emptyset$, then

$$\hat{W}^{(k,\ell)}_d := \hat{W}_d\left(\mu_1\left(\underline{x}^{(k)}_{\{2\}}\right), \mu_1\left(\underline{y}^{(\ell)}_{\{2\}}\right)\right) \tag{8}$$

is an imperfect estimate of an integral against the zero measure, as both arguments of $\hat{W}_d$ are standard normal distributions. Hence the true EZK measure $c^{\emptyset}_{1,2}$ is zero, but in practice our estimator (8) is roughly of order $N_1^{-1/2}$, with fluctuations of the same order.

The denominator $d_{\mathbb{R}}(\underline{z}^{(k)}_2, \underline{z}^{(\ell)}_2)$, on the other hand, is a sample of the difference of two normally distributed variables: $\mathcal{Z}_1 - \mathcal{Z}_2$, with $\mathcal{Z}_i \sim \mathcal{N}(0,1)$. As the sum of two normals is again normal, we find $d_{\mathbb{R}}(\underline{z}^{(k)}_2, \underline{z}^{(\ell)}_2)$ is typically of order $O(1)$, but can be much smaller. In particular, for a small positive constant $0 < \epsilon \ll 1$, $P\left(d_{\mathbb{R}}(\underline{z}^{(k)}_2, \underline{z}^{(\ell)}_2) < \epsilon\right) = O(\epsilon)$.

As a result, if $d_{\mathbb{R}}(\underline{z}^{(k)}_2, \underline{z}^{(\ell)}_2)$ is less than $N_1^{-1/2}$ the estimator will be significantly larger than the theoretical value, and the use of a strict maximum in (7) means that these fluctuations, despite their rarity, will control the value of $\hat{c}^{\mathcal{K}}_{i,j}$. For example, Figure 4 shows the semi-log plot of $\hat{W}^{(k,\ell)}_d / d_{\mathbb{R}}(\underline{z}^{(k)}_2, \underline{z}^{(\ell)}_2)$ against $d_{\mathbb{R}}(\underline{z}^{(k)}_2, \underline{z}^{(\ell)}_2)$ for $N_1 = 50$, $N = 500$. We clearly see that small values of the $d_{\mathbb{R}}$ distance lead to very large values of $\hat{c}^{\emptyset}_{1,2}$

To overcome this, we filter out those ordered pairs $(k,\ell)$ such that $d_E(\underline{z}^{(k)}_j, \underline{z}^{(\ell)}_j)$ is less than $C/\sqrt{N_1 + N_2}$ with $C$ a constant chosen so that the maximum value of $\hat{W}_d$ is of the same order as its median value. Below we take $C = 8$. *NB*: One should be careful not to take $C$ so large that the odds of $(k,\ell)$ being a valid pair become low enough to affect the runtime of the algorithm (due to the burden of resampling $\underline{z}_k$).

Another large improvement made to the general approach outlined in [3] is achieved by restricting ourselves to target variables $X_i$ which take values in a one-dimensional space ($\mathbb{R}$ or $S^1$; we consider $E = \mathbb{R}$ in the following). This allows us to compress the number of constraints: impose an ordering on the $\{x_i\}_{i=1}^{N_1+N_2}$ so that for $i = 1, \ldots, N_1 + N_2 - 1$, $x_i$ is adjacent to the point $x_{i+1}$ in the sense that $(x_i, x_{i+1})$

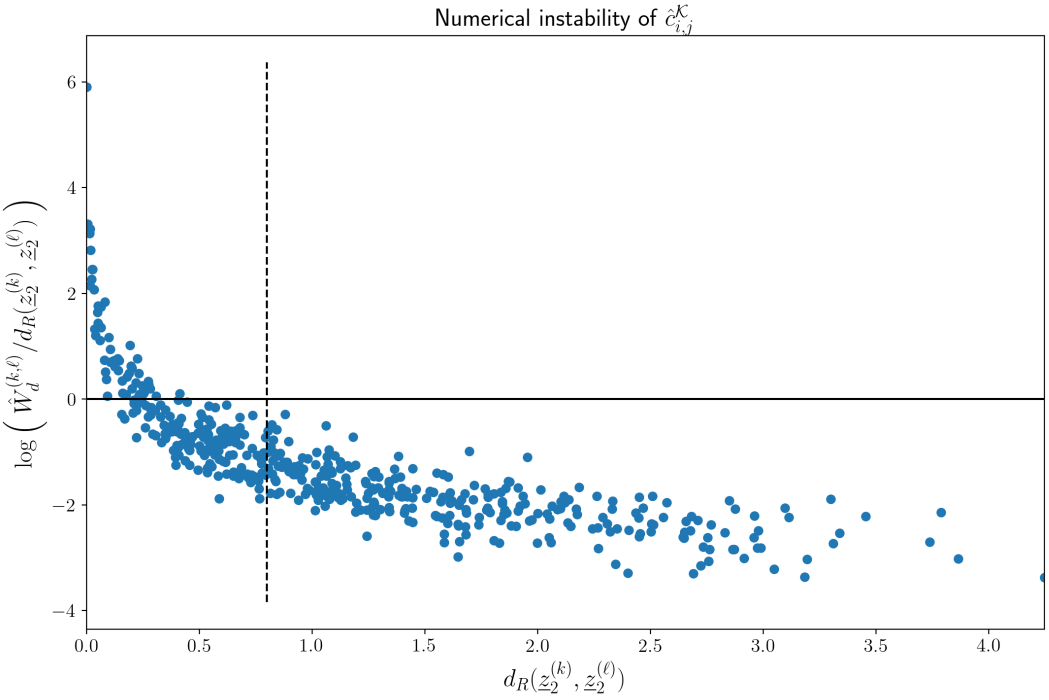


Figure 4: Sample values of $\log\left(\hat{W}_d^{(k,\ell)}/d_{\mathbb{R}}(\underline{z}_2^{(k)},\underline{z}_2^{(\ell)})\right)$, whose maximum defines the estimator $\hat{c}^{\mathcal{K}}_{i,j}$ in [3], plotted against the distance $d_{\mathbb{R}}(\underline{z}_2^{(k)},\underline{z}_2^{(\ell)})$. For this sampling we took $N = 500$ and $N_1 = N_2 = 50$. A positive $y$-value corresponds to an estimate of $\hat{c}^{\mathcal{K}}_{i,j}$ consistent with a causal relationship $X_j \to X_i$. We adjust the algorithm, removing $(k,\ell)$ pairs whose $d_{\mathbb{R}}(\underline{z}_2^{(k)},\underline{z}_2^{(\ell)})$ value falls below $8/\sqrt{N_1+N_2} = 0.8$ (the vertical dashed line).

contains no $x_j$. Then notice that any $\alpha$ satisfying

$$|\alpha_i - \alpha_{i+1}| \leq d_{\mathbb{R}}(x_i, x_{i+1}), \;\; i = 1, \ldots, N_1 + N_2 - 1,$$

is necessarily (a restriction of) a Lipschitz function, and so obeys the full set of constraints (5).

This observation allows us to take the number of constraints on the linear optimization problem (4) from $O((N_1 + N_2)^2)$ to $O(N_1 + N_2)$, significantly reducing the computational complexity. See Algorithm 1 for pseudocode of the algorithm.

Several other computational improvements have been made, e.g. we transform the problem in order to simplify the set of bounds, mollifying the linear programming problem of Algorithm 1. We refer the interested reader to the codebase at [14].

**Algorithm 1** EZK Dependecy Measure

**Input:** $i, j, \mathcal{K}, N, N_1, N_2$

**Output:** $\hat{c}_{i,j}^{\mathcal{K}}$

Initialize $\mathcal{S} = \emptyset$
**while** $|\mathcal{S}| < N$ **do**
  Sample $\underline{z}^{(k)}$ from $\mathbb{X}$
  **for** $\ell < k$ **do**
    **if** $d_E\left(\underline{z}_j^{(k)}, \underline{z}_j^{(\ell)}\right) > 8/\sqrt{N_1 + N_2}$ **then**
      Generate $N_1$ samples from $\mu_i\left(\underline{x}_{\mathcal{K}\cup\{j\}}^{(k)}\right)$ and $N_2$ samples from $\mu_i\left(\underline{y}_{\mathcal{K}\cup\{j\}}^{(\ell)}\right)$.
      Solve for $\hat{W}_d^{(k,\ell)} := \hat{W}_d\left(\mu_i\left(\underline{x}_{\mathcal{K}\cup\{j\}}^{(k)}\right), \mu_i\left(\underline{y}_{\mathcal{K}\cup\{j\}}^{(\ell)}\right)\right)$ using linear programming.
      Record $\hat{W}_d^{(k,\ell)}$ and add $(k, \ell)$ to the set $\mathcal{S}$.
    **end if**
  **end for**
**end while**
Take maximum to find $\hat{c}_{i,j}^{\mathcal{K}}$

## 1.2 The `hiton_ezk` package

The approach of Algorithm 1 gives us a method for testing conditional independence between two variables, and from now on the conditional independence notation will be defined as (abusing notation slightly)

$$X_i \perp\!\!\!\perp X_j | Z = I(X_i, X_j | Z) := \{\hat{c}_{i,j}^{Z} < 1\}. \tag{9}$$

One can build off of this in a number of ways in order to discover the local causal structure of the CPN about a given variable $X_i$; we base our approach off of one of the many variants of the Generalized Local Learning algorithm described in Aliferis etal. [1]. In particular, we use the Interleaved HITON-PC algorithm with symmetry correction.

In short, we fix a variable $X_i$ to learn the local causal neighborhood of (i.e. its parents and children in a faithful DAG $G$) and begin by initializing the two sets of variables OPEN= $\{X_j : j \neq i\}$ and TPC= $\emptyset$. We then test, for each $j \neq i$, whether the two variables are independent: $I(X_i, X_j|\emptyset) = \{\hat{c}^{\emptyset}_{i,j} < 1\}$. If $I(X_i, X_j|\emptyset)$, then the variable $X_j$ is removed from OPEN. Those that are not removed can now be placed in order of descending dependence on $X_i$ (descending $\hat{c}^{\emptyset}_{i,j}$ value).

Next, we repeatedly add the leading variable from OPEN to TPC, the set of potential parents and children, and test the condition $I(X_i, X_j|Z) = \{\hat{c}^{Z}_{i,j} < 1\}$ for each subset $Z \subset$TPC. If there exists some set $Z$ for which $\hat{c}^{Z}_{i,j} < 1$, then we consider $X_i$ causally independent of $X_j$ conditional on $Z$ and remove $X_j$ from TPC. If not, we test each member $X_k$ of TPC for $I(X_i, X_k|Z)$ for all subsets $Z \subset$ TPC containing $X_j$, as adding $X_j$ to TPC allows for new conditional independence tests.

This approach reflects a theorem in [11], which states that there is an edge between $X_i$ and $X_j$ in their Bayesian network if and only if the variables are conditionally dependent given $Z$ for all subsets $Z \subseteq \mathbb{X}\backslash\{X_i, X_j\}$. (*NB*: In practice the number of variables in TPC can become significant, and conditioning on each subset $Z$ in turn becomes onerous, as there are $2^{|\text{TPC}|}$ such sets. Hence, in practice we have an upper limit $M$ on the size of $Z$, and have taken $M = 5$ in this work. However, the combinatorics involved can be alleviated through parallelization.)

We continue this process until OPEN = $\emptyset$ and each member of TPC is tested for conditional independence against the other members of TPC. The end state of the set TPC is then a set containing $PC(X_i)$, though it may potentially contain other variables. See Algorithm 2 for pseudocode.

Once this process has been completed for each variable $X_i$ we can combine the results to construct the entire CPN associated to the system $(\Omega, \mathcal{B}, P)$ and its random variables $\mathbb{X}$ by performing the symmetry correction: writing $\text{TPC}_i$ for the output of Algorithm 2 when applied to $X_i$, we compare $\text{TPC}_j$ for each member $X_j \in \text{TPC}_i$; if $X_i \in \text{TPC}_j$ then either $X_i \to X_j$ or $X_i \leftarrow X_j$, otherwise $X_j \notin PC(X_i)$ (see [1] for further details). See Algorithm 3 for pseudocode.

Our implementation of the algorithms described in Algorithms 1, 2, and 3 have been released in the Python package `hiton_ipm`, which the reader can install with the bash command

```
python3 -m pip install hiton_ipm.
```

**Algorithm 2** Local Causal Network

**Input:** $i, N, N_1, N_2$

**Output:** $\text{TPC}_i$

[1] Set $\text{TPC}_i = \emptyset$, OPEN $= \mathbb{X} \backslash \{X_i\}$.

**for all** $X_j$ in $\mathbb{X} \backslash \{X_i\}$ **do**

Calculate $\hat{c}_{i,j}^{\emptyset}$. ▷ Calling Algorithm 1

**if** $\hat{c}_{i,j}^{\emptyset} < 1$ **then**

Remove $X_j$ from OPEN.

**end if**

**end for**

**while** OPEN $\neq \emptyset$ **do**

Find $j = \arg\max_k \hat{c}_{i,k}^{\emptyset}$.

OPEN $\leftarrow$ OPEN $\backslash \{X_j\}$.

$\text{TPC}_i \leftarrow \text{TPC}_i \cup \{X_j\}$.

**for all** subsets $Z \subset \text{TPC}_i$ **do**

Calculate $\hat{c}_{i,j}^{Z}$. ▷ Calling Algorithm 1

**if** $\hat{c}_{i,j}^{Z} < 1$ **then**

Remove $X_j$ from $\text{TPC}_i$.

Exit `for` loop.

**end if**

**end for**

**end while**

**Algorithm 3** HITON-EZK

**Input:** $N, N_1, N_2$

**Output:** CPN $G$ for $(\Omega, \mathcal{B}, P)$

**for all** $X_i$ in $\mathbb{X}$ **do**

Calculate $\text{TPC}_i$. ▷ Calling Algorithm 2

**end for**

**for all** $i, j$ such that $X_j \in \text{TPC}_i$ **do**

**if** $X_i \notin \text{TPC}_j$ **then**

Remove $X_j$ from $\text{TPC}_i$.

**end if**

**end for**

While the documentation gives guidance on the purpose and use of the package, we note that in order to compute the EZK measure one must be able to sample system states with certain variables conditioned and the others drawn from random distributions (in particular, this is how draws are made from the $\mu_i\left(\underline{x}_j^{(k)}\right)$ distributions). Therefore, the user must define a `distributions` value in the input parameters, defining how each variable will be sampled. This is used in the `eliminate` method of the package with an array $\{x_k\}_{k\in\mathcal{K}}$ of conditional values (to determine the values one fixes the $\{X_k\}_{k\in\mathcal{K}}$ at) to create input system states. That is, each variable at the start time (potential parents of the target variable, $X_i$) are either perturbed according to `distributions` or conditioned if they belong to $\mathcal{K}$. These starting values are then used to generate an output value of the variable of interest, $X_i$, in the provided `simulator` function.

This approach is particular to working with dynamical systems; this is expanded upon below in Subsection 2.1.2.

# 2 Theory II: Markov Blankets and System Mapping

Once one has the CPN $G$ for a system $(\Omega, \mathcal{B}, P)$ in hand, it remains to

- describe the functional form of the causal relationships,
- generalize these functions across scales, and
- use this description to determine how well the (source) system reflects the behavior of another (target) system at a given scale.

To move from the microscale dynamics of individuals to larger scales, we calculate the Markov blanket of the individual, which forms its interface with the rest of the system. This can have its Markov blanket computed in turn, and so on, to create a nested series of subnetworks of $G$. For each such subset, we will apply statistical learning tools to approximate the subset's dynamic with a policy function, and, finally, we calculate the likelihood of a target system's activity (e.g. empirical data from an open world system) according to that policy. This quantifies how well the closed world system that we study (source system, represented by $G$) reflects the target system at a given scale of interest.

In the remainder of this section we define Markov blankets and note their key characteristics, formalize the cross-scale approach, and describe in general terms the goal of policy learning. Finally, the formalism for system mapping is described.

## 2.1 The Markov Blanket

### 2.1.1 Preliminaries and definitions

Given a DAG $G = (\mathcal{V}, \mathcal{E})$, a **path** in $G$ is a sequence of distinct vertices $(v_{i_1}, v_{i_2}, \ldots, v_{i_J})$ such that successive pairs of vertices have an edge between them (either $v_{i_k} \to v_{i_{k+1}}$ or $v_{i_k} \leftarrow v_{i_{k+1}}$ for $k = 1, \ldots, J-1$). Given a path in $G$, a **collider** is a vertex $v_{i_k}$ such that both its edges are incoming: $v_{i_{k-1}} \to v_{i_k} \leftarrow v_{i_{k+1}}$.

**Definition 9 (d-separation)** *We say that a set $Z$ d-***separates** *a path in $G$ if there is either*

1. *a chain of vertices $v_1 \to v_2 \to v_3$ or a fork $v_1 \leftarrow v_2 \to v_3$ with the middle vertex $v_2$ in $Z$, or*
2. *a collider $v_1 \to v_2 \leftarrow v_3$ such that $v_2$ and all its descendents are* **not** *in $Z$.*

*The set $Z$ is said to d-separate $X$ and $Y$ if every path from $X$ to $Y$ is d-separated by $Z$.*

This definition arises in Pearl's treatment of causal systems [6, 7], and connects the graphical representation of the system to its dependence relationships by the following theorem, found in [11].

**Theorem 1** *The condition $I(X, Y|Z)$ is equivalent to the variables $X$ and $Y$ being d-separated by $Z$ in a faithful CPN of $(\Omega, \mathcal{B}, P)$.*

For simplicity in what follows we define $Sp(X)$, the **spouses** of the variable $X$, to be those variables that share a child with $X$:

$$Sp(X) = \{Y \in \mathbb{X} \backslash \{X\} \ : \ Ch(X) \cap Ch(Y) \neq \emptyset\}.$$

In terms of a faithful CPN $(G, P)$, $w$ is a spouse of $v$ if and only if there exists a vertex $u$ such that $v \rightarrow u \leftarrow w$.

**Definition 10 (Markov Blanket)** *The Markov blanket (MB) of $X \in \mathbb{X}$ is formed by the parents, children, and spouses of $X$:*

$$MB(X) := Pa(X) \cup Ch(X) \cup Sp(X) \tag{10}$$

*For a subset $H \subset \mathbb{X}$ the Markov blanket, written $MB(H)$, is the set*

$$\left( \bigcup_{X \in H} MB(X) \right) \Big\backslash H \tag{11}$$

*of causes, effects, and spouses of members of $H$, which do not already belong to $H$.*

By Theorem 1 and Definition 10, one sees that the Markov blanket $MB(H)$ is the minimal set of vertices that $d$-separates the vertices of $H$ from the rest of the CPN, $\mathbb{X} \backslash \{MB(H) \cup H\}$.

**Corollary 1** *Given a subset $H \subset \mathbb{X}$, the Markov blanket $MB(H)$ is the smallest set of variables such that for any $Y \notin MB(H) \cup H$ and any $X \in H$, $I(X, Y|MB(H))$ holds.*

In practice, this means that if one is interested in approximating the state of $H$ using information from elsewhere in $\mathbb{X}$, one only needs to consider the values of $X_i \in MB(H)$. This forms our approach to learning policies in subsection 2.3.

### 2.1.2 MBs for dynamical systems

The above definitions of $MB(X)$ and $PC(X)$ notwithstanding, the variables of a dynamical system $X_i$ are observed as time series $X_i = (\dots, X_{i,-1}, X_{i,0}, X_{i,1}, \dots) = (X_{i,t})_{t\in\mathbb{Z}}$ and this significantly simplifies the task of calculating $MB(X_{i,t})$. We make the assumption that causal relationships obey the arrow of time: if $X_{i,s} \to X_{j,t}$, then $X_i$ must occur strictly prior to $X_j$, i.e. $s < t$. For any given system it is helpful to have an estimate for the speed of information transfer ($\delta t$) so that one can limit the potential causes $X_{i,s}$ of $X_{j,t}$ to those variables observed at time $s < t - \delta t$.

In physical systems, for example, we know (by Newton's third law, say) that two variables may effect change in one another simultaneously: $X_{i,t} \longleftrightarrow X_{j,t}$. However, in simulations time evolves in discrete steps, $t_1, t_2, \dots$, so two billiard balls' states, $X_1$ and $X_2$, may be influenced only by their own previous kinematic state

$$X_{i,t_{k-1}} \to X_{i,t_k} \text{ and } X_{j,t_{k-1}} \not\to X_{i,t_k} \text{ for } i = 1, 2 \text{ and } j \neq i,$$

until a collision event, occurring at time $t^\star \in (t_{k-1}, t_k]$, in which case

$$X_{i,t_{k-1}} \to X_{i,t_k} \leftarrow X_{j,t_{k-1}} \text{ for } i = 1, 2 \text{ and } j \neq i.$$

That is, the influence of $X_i$ on $X_j$ obeys some inborn speed limit. For this reason we consider it impossible for concurrent variables $X_{i,t}, X_{j,t}$ to be causes of one another. Furthermore, we assume that the rate of observation, $t_k - t_{k-1}$, is large enough that some variables $X_{j,t_{k-1}}$ from the previous time step $t_{k-1}$ will be causes of some variables $X_{i,t_k}$ of the present time step $t_k$.

Under these assumptions we have the following theorem.

**Theorem 2** *For a dynamical system $\mathbb{X}$ evolving in discrete time steps $t_1, \dots, t_k, \dots$, observed only within the window $[t_1, t_k]$, and obeying*

$$X_{i,s} \to X_{j,t} \implies s < t, \tag{12}$$

*we have*

$$MB(X_{i,t_k}) = Pa(X_{i,t_k}). \tag{13}$$

This is an immediate consequence of the fact that at time $t_k$ the variables $X_{i,t_k}$ cannot be a cause of the concurrent variables $X_{j,t_k}$, nor variables of the past. Hence, $X_{i,t_k}$ has no descendants at time $t_k$, precluding both children and spouses: $Ch(X_{i,t_k}) = \emptyset = Sp(X_{i,t_k})$ for all $i$.

In studying dynamical systems below we will be mainly interested in calculating $I(X_{i,t_k}, X_{j,t_{k-1}}|Z)$ for different families $Z \subset \mathbb{X}_{t_{k-1}} = \{X_{m,t_{k-1}}\}_m$, i.e., determining the causal influence of time step $t_{k-1}$ on the

following time step, $t_k$, though some systems may reward searching for direct causes from earlier system states, $\mathbb{X}_{t_j}$, $j < k-1$.

## 2.2 Causal Clusters

In this section we discuss the construction of the causal cluster hierarchy (CCH) which is used to organize $(G = (\mathcal{V}, \mathcal{E}), P)$ into subnetworks spanning sizes from a single individual to the entire network .

Let us assume that the joint distribution $P$ of the system is **causally connected**, meaning that it has a faithful CPN $G$ that is weakly connected: given any two vertices $v_1$ and $v_2$ of $G$ there is a path from $v_1$ to $v_2$. In particular, we note that the path does not need to adhere to the directionality of the graph's edges. Consider the set

$$MC(X) := MB(X) \cup X, \tag{14}$$

called the **Markov cluster** of $X$; we iteratively calculate Markov clusters of a vertex, written

$$MC^{k+1}(X) := MC(MC^k(X)) \text{ for } k = 0, 1, 2, \ldots \tag{15}$$

We know that as long as $MC^k(X) \neq \mathcal{V}$, the successive Markov cluster $MC^{k+1}(X)$ is strictly larger, since the Markov cluster of a set $H$ is a superset of the nearest neighbors of $H$. It should likewise be clear that because $|\mathcal{V}| < \infty$ there exists an integer $r(X) \geq 0$ such that

$$X \subsetneq MC(X) \subsetneq MC^2(X) \subsetneq \cdots \subsetneq MC^{r(X)}(X) = \mathcal{V}. \tag{16}$$

In this way the successive clusters of any random variable $X$ exhaust the entire space $\mathbb{X}$ of observed variables as long as the system is causally connected. We will refer to sets of the form $MC^k(X)$ for $k \geq 0$, $X \in \mathbb{X}$ as **causal clusters** (CCs).

Since for any distinct r.v.s $X_1, X_2$ we have

$$MC^0(X_1) = X_1 \neq X_2 = MC^0(X_2), \tag{17}$$

but also

$$MC^{r(X_1)}(X_1) = \mathbb{X} = MC^{r(X_2)}(X_2), \tag{18}$$

there is some minimal pair $s(X_1), s(X_2) \in \mathbb{N}$ such that $MC^{s(X_1)}(X_1) = MC^{s(X_2)}(X_2)$. In summary, the collection of sets in (16) for each $X \in \mathbb{X}$ forms a partially ordered set and each pair of sequences eventually coincides.

As a result, the set of all causal clusters within $G$ forms a hierarchy of nested Markov clusters, which we can express as a branching tree $\mathcal{T}$ with the entire observed system $\mathbb{X}$ at its root and individual variables at

its leaves. We call the system of nested sets a **causal cluster hierarchy**, or CCH, and refer to the associated triple $\langle G, P, \mathcal{T} \rangle$ as a CCH, or CCH tree.

## 2.3 Policies

We begin in this section to consider the observed system $\mathbb{X}$ not as a tuple of disjoint, one-dimensional random variables, but as a network of interacting agents, as is natural in complex adaptive systems. This means that the $i^{\text{th}}$ agent $\xi_i$ may consist of several random variables $X_{i_1}, X_{i_2}, \ldots$ describing its state. (Not every group of variables needs to represent an individual in the system - it may be an element of the environment that reacts to agents' actions, such as a soccer ball - but we will still refer to such a collection as an "agent".) We then write $\mathbb{X} = (\xi_1, \xi_2, \ldots, \xi_n)$ for the collection of $n$ agents making up the system, and write $\xi_i \to \xi_j$ when one or more of the variables $X_{i_k}$ of $\xi_i$ is a direct cause of $X_{j_\ell}$, one of the components of $\xi_j$.

An agent's **true policy** is their decision making apparatus, or their method of deciding what behavior to engage in given their internal state and what they observe of the world around them. Outside of very simple physical systems or models whose internal workings are known, our ability to describe a true policy is limited. Indeed, in many systems the elementary act of enumerating the possible actions that an agent might take can be daunting. In practice, we simplify the problem by reducing our definition of the system's state to a relatively few coordinates, $\mathbb{X}$, that capture most of the interesting parameters, but this projection loses information that might, in some number of cases, be the factor on which the actor's choice hinges. Likewise, we seek a parsimonious **action space** $\mathcal{A}$ to describe the agents' potential decisions, though some granularity of the agents' behavior is often lost.

In this way, an agent occupying the same state (in our chosen coordinates) on separate occasions might choose different actions, so that even deterministic systems can appear to exhibit randomness. Moreover, many of the cases of interest involve biological agents (esp. humans) any one of which may, under the same circumstances, reach very different decisions upon repeated trials. So it is natural to describe the $i^{\text{th}}$ agent's processing function, $\phi$, as a map from its observed state $MC(\xi_i)$ to a probability distribution on actions $a \in \mathcal{A}$:

**Definition 11** *Given the system* $(\Omega, \mathcal{B}, P)$ *with observables* $\mathbb{X}$ *and action space* $\mathcal{A}$*, a* **policy** *is a map*

$$\phi : \mathbb{X} \to P_{\mathcal{A}}. \tag{19}$$

If we express the action of agent $i$ at time $t$ as $a_{i,t}$ then $\phi_{i,t}(\mathbb{X}_t)(a_{i,t}) = P(\xi_i \text{ takes action } a_{i,t} \text{ at time } t)$.

### 2.3.1 Policies of individuals

Take the system $\mathbb{X} = (\mathbb{X}_{t_k})_{k\in\mathbb{N}}$ with $\mathbb{X}_{t_k} = (\xi_{i,t_k})_{i\in\mathcal{I}}$ and let $\phi_{i,t}$ represent the policy of agent $\xi_i$ at time $t$. Due to feedbacks, nonlinearities, and behavioral complexity, even in simulation environments it may be difficult to express the policy of a given agent. Instead, we restrict ourselves to a family $\mathcal{F}$ of functions, such as neural networks or polynomials, and attempt to locate an effective proxy within that family:

**Definition 12** *Let $\mathbb{X} = (\xi_1, \ldots, \xi_n)$ be a collection of agents. The* **trained policy** *of agent $\xi_i$ with respect to the family of functions $\mathcal{F}$ given the training data $(\xi_{i,t_1}, \xi_{i,t_2}, \ldots, \xi_{i,t_K})$ is*

$$\phi_i := \min_{\phi\in\mathcal{F}} \mathcal{L}(\phi|\xi_i)$$

*where $\mathcal{L}$ represents a* **likelihood measure** *applied to the observations of $\xi_i$:*

$$\mathcal{L}\left(\phi|\xi_i\right) := \left\langle \phi(\xi_{i,t_k})(a_{i,t_k})\right\rangle_k = \frac{1}{K}\sum_{k=1}^{K} \phi(\xi_{i,t_k})(a_{i,t_k}). \tag{20}$$

In order to compare closed and open systems, we will need to formulate a policy for each agent in the system. In homogeneous systems the policy may be constant through time and shared among the system's agents ($\phi_{i,t} \equiv \phi$ for all $i$ and $t$). In other cases, each individual may obey a specific set of rules, requiring their policies to be trained separately, and potentially varying over time.

## 2.4 System mapping

One fundamental goal of this research is to quantify in a new way how well the dynamics of interest in one system (called the **source** system, where we have access to plenty of data) reflect those in another (the **target** system). This is done through a method we call system mapping, which is essentially choosing a hierarchical level of interest, extracting policies at that level, then seeing how well those policies capture the observed dynamics of the target system.

Consider the CCH of the system of agents $\Xi^{(A)} = (\xi_1, \ldots, \xi_n)$ governed by $(\Omega, \mathcal{B}, P)$, and suppose we are invested in subsystems of a given size, between $S_1$ and $S_2$, say. Choosing $S_i$, $i = 1, 2$, to be $O(1)$ (individuals), $O(n^{1/2})$ (local subgroups), or $O(n)$ large submodules), we define the cluster size of interest. We collect from the full CCH $\mathcal{T} = \{MC^k(\xi_j) : k = 0, \ldots, r(\xi_j), j \in \mathcal{I}\}$ those causal clusters of a given size

$$\mathcal{C}_{S_1,S_2} := \{C \in \mathcal{T} : S_1 \leq |C| \leq S_2\}. \tag{21}$$

If this source system behaves similarly to the target system, $\Xi^{(B)} = (\eta_1, \ldots, \eta_m)$ at the scale defined by $S_i$, then the policies trained on the clusters in $\mathcal{C}_{S_1,S_2}$ should reflect what we observe in the target system. For

simplicity we will assume that the system $\Xi^{(A)}$ is homogeneous, so that policies are constant and agents share the same policy.

**Definition 13** *Let $\{(a_{i,t})_{i\in ??}\}$ be the empirical data gathered from the target system $\Xi^{(B)}$, and let $\{\phi_{C_j}\}$ be the collection of policies for the clusters $C_j$ in $\mathcal{C}_{S_1,S_2}$. We define the* **system affinity** *of $\Xi^{(A)}$ and $\Xi^{(B)}$ at scale $(S_1, S_2)$ by*

$$\Phi^{(A,B)}_{S_1,S_2} := \frac{1}{|\mathcal{C}_{S_1,S_2}|}\sum_j \mathcal{L}\left(\phi_{C_j}|\{\eta_{i,t}\}_1^m\right) \tag{22}$$

This measure is large when the policies $\{\phi_{C_j}\}$ predict well how $\Xi^{(B)}$ changes through time. In practice, many of the causal clusters within $\mathcal{C}$ may have a significant proportion of shared agents, or may be rapidly changing over time. To better extract typical cluster behavior, one may want to filter the set $\mathcal{C}_{S_1,S_2}$ to mollify the effect of overemphasizing certain agents' activity or remove low-fidelity policies.

# 3 Application I: Flocking Models

All of the foregoing theory was first tested in an entirely simulated environment where agents $\{\xi_1, \ldots, \xi_n\}$ inhabiting a torus $\mathbb{T}$ form one or more flocks as they travel through the space. Using simple flocking models we compare distinct emergent group phenomena in silica, and show that in a simple, easily-visualized setting the tools of CPN inference, policy learning, and system mapping can be coordinated to establish a model-to-model metric of similarity.

Working with simulated data, we create multiple systems with different emergent behaviors (whole group flocking vs. avoidant subgroup flocking). For each such model we infer the underlying CPN using the HITON-EZK algorithm, describe the policies at different scales using simple linear regression models, and then demonstrate that given "empirical" data, positional data generated by an unknown underlying model, system mapping can accurately determine the model which most closely resembles the generative model.

As a final exercise we compare policies of a large-scale CC, representing a coherent flock, to those of individual agents. We see that the policies of the large groups are essentially the same as those of individuals: if two flocks are mutually avoidant, their policies upon collision reflect those of two individual avoidant agents. So we see that the emergent behavior of flocks acting as superorganisms is captured by the similarity of policy functions over the two scales. This is a first example of how comparing policies can be used to draw an analogy between different subsystems.

## 3.1 Models and data

We begin with a classical model of flocking behavior, commonly referred to as the Toner and Tu model [15], in which the agents (called boids) mimic the direction of travel of the agents close to them while traveling within $\mathbb{T}^2$, a 2-torus of side length $L$ (throughout we take $L = 1$). By shifting the radius of interaction, $r \in \mathbb{R}_{\geq 0}$ (that is, what counts as 'close' to the focal agent) the system will transition from a disorganized state where the flock as a whole has little to no collective velocity, to an organized state where many of the agents are aligned and the flock has a distinguished direction of travel (the mean velocity is clearly nonzero).

### 3.1.1 The model of Toner and Tu

Each agent $\xi_i$ is defined by their position $(x_i, y_i) := (\xi_i(1), \xi_i(2)) \in \mathbb{T}^2$ and heading $\theta_i = \xi_i(3) \in S^1$. The model has parameters for agent speed, $v$, interaction radius, $r$, and the magnitude $\varepsilon$ of the driving noise for

the agents. The injected randomness takes the form of a perturbation of the $i^{\text{th}}$ boid's heading at time $t$ by $\varepsilon z_{i,t}$, where the $\{z_{i,t}\}$ are independent standard normal random variables. At time $t = 0$ we initialize all agents' positions and headings by sampling from $\mathbb{T}^2 \times S^1$ uniformly at random, and for successive time steps the agents' states are updated according to the following set of equations:

Positions are incremented faithful to the current heading $\theta_{\cdot,t}$

$$x_{i,t+1} = x_{i,t} + v\cos(\theta_{i,t}),$$
$$y_{i,t+1} = y_{i,t} + v\sin(\theta_{i,t}),$$

while the heading is determined by summing the influences of all nearby agents, who are indexed by the set

$$B_{i,t}(r) = \{j \neq i : d_{\mathbb{T}^2}\big((x_{i,t}, y_{i,t}), (x_{j,t}, y_{j,t})\big) < r\},$$

i.e., those agents contained in the ball of radius $r$ (with respect to $d_{\mathbb{T}^2}$, the standard metric on the torus) about the $i^{\text{th}}$ agent at time $t$. The influence of agent $j$ on agent $i$, written $f(\xi_i, \xi_j)$, is a velocity vector, and for the classic Toner and Tu model it is simply the velocity of the $j^{\text{th}}$ agent:

$$f(\xi_i, \xi_j) = \langle \cos(\theta_{j,t}), \sin(\theta_{j,t})\rangle.$$

Finally, up to a noise term $\varepsilon z_{i,t}$, the heading $\theta_{i,t+1}$ is chosen to align with the vector sum of all influences:

$$\Delta_{i,t} = \langle \cos(\theta_{i,t}), \sin(\theta_{i,t})\rangle + \sum_{\xi_j \in B_{i,t}(r)} f(\xi_i, \xi_j), \qquad \text{(the desired direction)}$$
$$\theta_{i,t+1} = \tan^{-1}\left(\frac{\Delta_{i,t}(2)}{\Delta_{i,t}(1)}\right) + \varepsilon z_{i,t}.$$

For these models, the output of the simulations are time series of $n$ agents' positions and headings in a three-dimensional ambient space, that is, trajectories taking values in $\left(\mathbb{T}^2 \times S^1\right)^n$ corresponding to the spatial variables $(x, y)$ and the heading, $\theta$.

### 3.1.2 New models

To create new systems exhibiting different emergent behaviors, we alter the classical model (where all agents are interchangeable and homophilic) so that the boids have a 'species' and they interact with other boids based on their species. The species are constant through time in our models, so rather than treat them as a fourth coordinate for each boid, we consider the boid-species assignment a new parameter determining the model.

The modes of interaction we allow are friendly ($F$), avoidant ($A$), or neutral ($N$). This interspecies behavior is summarized by an **interaction matrix** $I$; the $(a, b)$-entry of $I$ takes its value in $\{F, A, N\}$ and defines the way nearby agents of species $b$ influence those of species $a$. In particular,

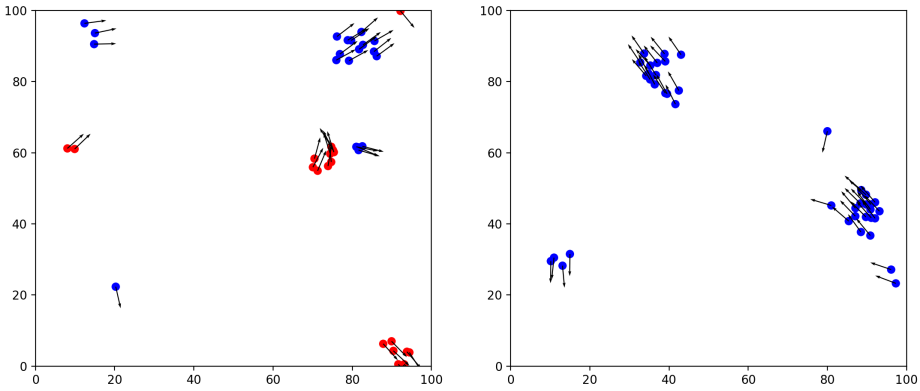


Figure 5: (A) A flocking model with two species of boid which seek to avoid one another. (B) The classical Toner and Tu model, executed with the same parameters.

- if $I_{a,b} = F$, then agents of species $a$ attempt to align their heading with those of species $b$, $f(\xi_i, \xi_j) = \langle \cos(\theta_j), \sin(\theta_j) \rangle$.
- if $I_{a,b} = A$, then agents of species $a$ avoid members of species $b$, moving in the opposite direction: $f(\xi_i, \xi_j) = \langle x_i - x_j, y_i - y_j \rangle$.
- if $I_{a,b} = N$ agents of species $a$ will ignore those of species $b$, $f(\xi_i, \xi_j) = \langle 0, 0 \rangle$.

Note that agents within a given species may avoid one another, or ignore one another, rather than attempt to flock together. See Figure 5 for examples of a modified model with two species, $I = \begin{bmatrix} F & A \\ A & F \end{bmatrix}$ and the original Toner and Tu model, $I = \begin{bmatrix} F \end{bmatrix}$, in their steady states, when the radius $r$ is large enough for flocking behavior to emerge.

## 3.2 Methods

Let $\mathcal{N}_d(m, \sigma)$ indicate a normal probability distribution on $\mathbb{R}^d$ with mean $m$ and standard deviation $\sigma$, and define the embedding

$$\rho : \mathbb{R}^3 \to \mathbb{T}^2 \times S^1,$$
$$(x, y, \theta) \mapsto (x \bmod L,\ y \bmod L,\ \theta \bmod 2\pi),$$

along with the partial inverse

$$\rho^{-1} : \mathbb{T}^2 \times S^1 \to \mathbb{R}^3,$$
$$(x, y, \theta) \mapsto (x, y, \theta),$$

### 3.2.1 CPN inference

We begin with the task of estimating a faithful CPN for the system. This is done by executing a simulation of a flocking model – we initialize the system with chosen boid types (see Table 1 for the flock membership data), and their positions and headings sampled uniformly at random from $\mathbb{T}^2 \times S^1$ – for $T$ time steps, then saving the system's history of states $\Xi_t = (\xi_{k,t})_{k=1}^n$, $t \in \{0, 1, \ldots, T\}$, to a data file. Then for those times $t$ that we are interested in computing the CPN, we pass the recorded state into the HITON-EZK algorithm, as well as parameters $\sigma_1, \sigma_2 > 0$ which define the distributions that the states $\xi_{k,t}$ will be resampled from.

In order to sample from $\mu_{i,j}^K$ in step 6 of Algorithm 1, we form the perturbed agent states

$$\xi'_{k,t} := \rho\left(\rho^{-1}\left(\xi_{k,t}\right) + \varepsilon_k\right) \text{ where } \varepsilon_k \sim \begin{cases} \delta_{(0,0,0)} & \text{if } k \in K, \\ \mathcal{N}_2(0, \sigma_1) \times \mathcal{N}_1(0, \sigma_2) & \text{otherwise.} \end{cases}$$

Here $\delta_x$ represents the Dirac delta function that takes the value 1 at $x$ and 0 elsewhere. This means that $\xi'_{k,t}$ is unchanged if $k$ lies in the conditioning set $K$, while for all other agents their positions and headings are perturbed by some normal random variable. Write $\Xi'_t = (\xi'_{k,t})_{k=1}^N$ for the full set of perturbed system variables

In perturbing this system we want to sample states that are different enough from the current state that the agents' environments will be affected (a change is registered), but we don't want to sample in such a way that the new state $\Xi'_t$ has nothing to do with the original state $\Xi_t$. This is because any agent in the system may affect any other given the right conditions, so creating a new sample from whole cloth can introduce a bevy of new causal connections, some of which will obfuscate the changing distribution of $\xi_i$ under changes in $\xi_j$. By making relatively moderate changes in agents' positions and headings we don't fundamentally alter the state of the system, preserving most of the behavior of interest.

This is essentially an assumption that the action of an agent is (roughly) a continuous response to the state of the system, so by keeping perturbations reasonably minor we can explore the variable response without introducing too many new confounding variables; these may average out over many samples, but if we avoid introducing them in the first place we can deduce causal influence with fewer samples. However, it is worth noting that if an agent is influenced by a large number of other agents (which can happen with the

| Model | Flock membership | Parameters $(r, v, \varepsilon, \sigma_1, \sigma_2)$ | Simulation time $t'$ | Interaction matrix |
|---|---|---|---|---|
| A | (10) | (0.2, 0.5, 0.1) | 1 | $[F]$ |
| B | (30) | (0.2, 0.5, 0.1) | 1 | $[F]$ |
| C | (20, 20) | (0.1, 0.5, 0.1) | 1 | $\begin{bmatrix} F & N \\ N & F \end{bmatrix}$ |
| D | (20, 20) | (0.1, 0.5, 0.1) | 1 | $\begin{bmatrix} F & A \\ A & F \end{bmatrix}$ |
| E | (1, 1) | (0.1, 0.5, 0.1) | 1 | $\begin{bmatrix} F & A \\ A & F \end{bmatrix}$ |

Table 1: Parameters determining the flocking models used in validating system mapping.

models of this section after large flocks have formed) one may need larger perturbations to lower the number of unconditioned agents influencing the target agent $\xi_i$. In other words, the threshold for a "minor" perturbation increases as the degree of the CPN increases.

Acknowledging this, we choose the variances $\sigma_1$, $\sigma_2$ so that nearby agents may pass in or out of each other's interaction radius ($\sigma_1 \approx r/2$) and their heading will vary over a significant portion of the interval $[0, 2\pi)$ ($\sigma_2 \approx \pi/6$).

With a CPN for the system $\Xi$ at time $t$ in hand, one can immediately compute the associated CCH and construct the CCH tree. We remark that we have the ground-truth CPN available to us, wherein $\xi_1 \in MB(\xi_2)$ and $\xi_2 \in MB(\xi_1)$ if and only if the two agents lie within a distance of $r$ of each other. Thus we can construct the true CPN and CCH tree for comparison with those inferred via HITON-EZK.

#### 3.2.2 Policy learning

Policy learning for causal cluster $MC^k(\xi_{i,t})$ was performed by a simple linear regression among the neighboring boids.

#### 3.2.3 System mapping

With the policies in hand, we calculate the system affinity $\Phi_{1,1}^{(\cdot,\cdot)}$ among the models. We want to see how well the micro-scale behavior is captured among models with very different behaviors (e.g. comparing the

| Test number | Generative model | Source models | CC size | Time scale |
|---|---|---|---|---|
| 1 | A | B, C, D, E, F | 1 | 1 |
| 2 | A | A | 1 | 1 |
| .... | | | | |

Table 2: Tests run to judge the efficacy of system mapping. The generative model is the model from Table 1 that was used to create the faux-empirical data; the source models are those whose policies are mapped onto the empirical data to test for agreement. We operate at different scales across tests, cosidering CCs of a given cardinality (one, for individual policies; $O(n)$ for whole flock behaviors).

interactions of model A, which are always friendly, to a model featuring avoidant behavior such as D), but we also want to see the evolution of causal clusters and their policies as the agents naturally form collectives. A natural hypothesis we might form is that both

$$\begin{bmatrix} F & N \\ N & F \end{bmatrix} \text{ and } \begin{bmatrix} F & A \\ A & F \end{bmatrix}$$

interaction types will lead to two macro-scale groups over time, so that their long-term, large-scale structures might be very similar, while their micro-scale interactions are quite different.

We then perform several system mapping experiments to test these hypotheses; Table **??** summarizes the models that are compared to one another and at what scale. In particular, we measure the scale of interest by the number of agents belonging to the CC whose policy is being tested against.

### 3.3 Results

For each flocking model type we executed a 1000 step simulation, and selected a subset of 5 time steps featuring a range of flocking behaviors (e.g. disorganized, many small groups, large flocks, or collisions of subgroups), and calculated CPNs and policies for each.

For each of the models in Table 1 we test their agent-level policies against novel simulation data from each separate model.

Last, we compare the functional form of the regression for a set of colliding flocks of different species to that of colliding individuals of different species. One can see by observing the state of the flock over time that its state changes occur over slightly longer time scales, as the change in heading takes a short time to propagate through the flock.

# 4 Application II: Google Research Football and Soccer Play

As an application of a complex social-spatial system we are pursuing the same line of research as detailed in the previous section, but applied to Google Research Football (GRF) data. The GRF program is an open-source project recently developed and employed by Google to investigate the competitiveness of soccer teams composed of artificial agents (ML models) trained under various reinforcement learning protocols. Included with the package (hosted at https://github.com/google-research/football) are stock agent behaviors, hard-coded by the developers, which can be used immediately to generate gameplay.

The authors have created a GitHub repository [13] holding our fork of the GRF codebase.

## 4.1 Codebase structure and modifications

While there is documentation included with the GitHub project, we describe briefly here the major changes required to achieve these changes to facilitate continued development and extensions of this work.

We have modified this codebase extensively to

1. expose more variables of the players' states to the python environment,
2. record basic variables (the data described in Table 3) periodically during simulation, and to
3. run multiple simulations in parallel.

The final item is critical for generating data efficiently, and can be done by writing a wrapper for the core GRF simulation function. In contrast, the first two items required intrusive changes to the codebase, and a deeper understanding of the software's internal composition.

This is due to the fundamental structure of the project. The soccer gameplay is simulated and rendered within a C++ package (largely contained in the `third_party/gfootball_engine/src/` folder of the GRF project). This package is then built and compiled with certain objects exposed to a Python environment using the Boost C++ package. In particular, a GameEnv_Python object allowing Python scripts to initialize and execute game instances and a SharedInfo struct exposing ball and player positions are available. For (1) above, we need to modify the PlayerInfo class of `src/defines.hpp` header file to hold the player's action (an element of the enum e_FunctionType). This can then be accessed once we update the PlayerInfo class defined in `ai.cpp`, which delineates the structure of the C++ objects when exposed to the Python environment through the BOOST_PYTHON_MODULE of the boost library. With the

| Field | Data structure | Frequency |
|---|---|---|
| Player position | 2-vector | Per player |
| Player direction | 2-vector | Per player |
| Ball position | 2-vector | Unique |
| Ball direction | 3-vector | Unique |
| Ball rotation | 3-vector | Unique |
| Player team | $\pm 1$ | Per player |
| Player possession | Boolean | Per player |
| Player skill level | float in $[0, 1]$ | Per player |
| Player fatigue level | float in $[0, 1]$ | Per player |
| Current player action | Categorical | Per player |
| Score | 2-vector | Unique |

Table 3: Data output from GRF simulations.

variables of interest available in a SharedInfo object during game execution, we can arrange to have them recorded in a `.yml` file fairly easily (item (2)). This occurs in `gfootball/env/football_env_core.py`, in the step method, for each time step that is a multiple of a fixed number of steps. (Each step represents a tenth of a second, so a typical choice is to record the summary game state every fifth step, or half second.)

This information is enough to observe the evolution of a soccer match over time and create observational time series of player states and actions. But if we want to perform any interventions we need to have the ability to revisit the game in a particular state, adjust some variables, and continue execution from the new state to observe the effect of the perturbation. This is particularly difficult, as it requires manipulating the objects defined and held in C++ memory and execution (NB: initializing a game using the built-in functions resets player motion, interrupting play). To do this, we make use of the boost::serialization library (this requires updating `third_party/gfootball_engine/CMakeLists.txt` to locate and link the serialization library).

The additional goals we have to employ the HITON-EZK algorithm and complete the system mapping tests require further edits to

- allow loading and saving entire game states from binaries, and
- perturb loaded states among select variables.

Our research team has succeeded in creating binary files to hold the ball state and a `SpatialState` struct for each player, which contains their position, direction, etc. However, the process of injecting this data back into the game has proven difficult, as initializing the players may involve the simulator's graphics engine, as well as `humanoidSourceNode` objects, which hold the arrangement of a player's limbs, head, and torso. It remains to develop the functionality to recreate the players in action, for the purpose of repeatedly sampling their behavior under perturbations.

## 4.2 Methods

In order to talk about the state of the soccer pitch and players we make the following definitions. Given a player $\xi_i$, we define their own team to be $T_A(\xi_i)$ or just $T_A$, the opposing team as $T_B$, and the set of all players to be $T = T_A \cup T_B$. The team in possession of the ball is denoted $T_O$, the team on defense, $T_D$. The line connecting two points $p_1$ and $p_2$ is expressed as $\overline{p_1 p_2}$, and if $d$ is the usual euclidean distance, defined between two sets by

$$d(A, B) = \inf_{x \in A, y \in B} \|x - y\|.$$

(Note we write $d(p_1, p_2)$ instead of $d(\{p_1\}, \{p_2\})$. ) Thus, the distance from a point $p_3$ to a line between two other points is $d(p_3, \overline{p_1 p_2})$, and finally, the field boundary is denoted $\partial F$.

We define the following secondary variables in order to transform the raw positional data into terms that are more likely to correspond to features influencing a player's choices:

1. distance to nearest opponent,
$$\min_{y \in T_O} d_x(\xi_i, \xi_j),$$
2. width of passing lanes,
$$\min_{p_D \in T_D} d(p_D, \overline{p_1 p_2}), \;\; p_1, p_2 \in T_O,$$
3. distance to the field's boundary,
$$d(p, \partial F),$$
4. distance to the nearest opponent passing lane,
$$\min_{p_{B1}, p_{B2} \in T_B} d(p_A, \overline{p_{B1} p_{B2}}).$$

In inferring players' policies we would employ the above processed variables to ease the machine learning task. Due to the current limitations of the GRF package, as well as the scarcity of positional soccer data,

the task of training such policies and performing system mapping from the simulation environment to live game data remains.

### 4.2.1 Empirical data

A group of researchers in Tromsø, Norway used specialized sensors attached to soccer players from the Tromsø IL team to record their motion and estimate their energy expenditure [8]. There are thus several soccer games with player position data available for use online (see the paper), however, in none of the games provided did the opposing team (Anzhi Makhachkala, Tottenham Hotspurs, or Stromsgødset IF) also wear these sensors and consent to their data being collected and distributed. Moreover, the location of the ball on the pitch is also unknown.

In order for policies from simulated systems to be compared to real-world data, the two must share a common set of coordinates. In this instance then, we would need to train our policies from GRF data limited to include only a player's own team, and not the ball itself. The policy would have to infer whether a player was in possession of the ball as well, which is perhaps the most important factor in determining what action a player might take. This is a significant challenge to performing the desired system comparison; for instance, items (1), (2), and (4) are no longer calculable in the list of secondary variables above.

The team came across one other potential soccer dataset named Magglingen2013 [10]. This repository had a record of the position and direction of all players and the ball during two professional soccer matches, recorded at a frequency of 10Hz. The data is no longer hosted, but through personal communications the research team was able to obtain a short snippet of one game, consisting of about 60 seconds of gameplay. This data is now available in a JSON file located at `worlds/football/magglingen2013/emp_play.json` within the GitHub repository `github.com/thepredictionlabllc/worlds.git`.

## Case-study 3: Information Spread and Conflict on Twitter

Online social networks represent an interesting and important counterpart to the other example systems we have chosen to study so far, from their structure as dynamical systems to their broad societal relevance and importance to national security. From a modeling perspective, social networks have a discrete, rather than continuous state and action space. Furthermore, agents in social networks typically have their own identities, with each agent occupying a unique position in a heterogeneous network and exerting different influence from the other agents in the network. The slowly evolving structure of social networks and the unique agent identities gives social networks a *Lagrangian* character, distinguishing them from *Eulerian* systems, like flocking or Google Research Football, where agents regularly switch positions and may have nearly identical policies[4]. The uniqueness of agents presents a tremendous challenge for identifying policies, which we hope can be mitigated by the availability of large amounts of data and the existence of prior knowledge that constrains the local causal networks, for example the prior information from follower networks.

We selected social interactions on Twitter as an example system for policy discovery. Users on Twitter write 280 character posts, called tweets, which can include media content or links in addition to text. Users interact by liking, retweeting, or commenting on the tweets that appear in their timeline, and use short phrases called hashtags to identify the context of a tweet or as a short way to signal a message. Each user's timeline consists of tweets from users in their follower network. Activity on twitter exhibits a wide range of interesting dynamics: tweets and hashtags can propagate well beyond the followers of the user that originates them, in some cases reaching the entire network[9]. Thus Twitter has tremendous potential for information dissemination[2], for political debate and discussion[16], or for the spread of disinformation[5]. The Twitter network contains numerous sub-networks which center on topics, such as political campaigns, scientific issues such as climate-change, and current events such as the SARS-CoV2 pandemic. These sub-networks are highly connected and contain numerous examples of repeated, sometimes adversarial interactions, which can range in complexity from simple, two-party political races to nuanced arguments over the best way to suppress covid or decarbonize the electricity grid. By focusing on these sub-networks, we can substantially reduce the computational difficulty of policy identification while taking advantage of the persistent, complex interactions that take place there.

A policy for a twitter user consists of a probability distribution on the set of liking, retweeting, or commenting on items in their timeline. Twitter users have information on how other users interact with they content they see, and are likely to take similar actions to agents they agree with and take opposite actions to agents that they disagree with. User policies thus may reflect the allegiances of twitter users. We

propose to use a handful of simple models of contagion to learn the policies of twitter users in selected sub-networks, enabling us to apply our open and closed worlds framework for model-comparison. We selected models of social contagion on heterogeneous networks, with varying probabilities of behavior transmission at different points in the network, and a range of transmission rules ranging from simple to complex contangions[18, 9]. Concretely, our system has $N$ agents, each of which has a state $\mathbf{s}_{i,t}$, defined as the state of the $i$th user at time $t$. The values of $\mathbf{s}$ belong to the action space $S = \{\text{Null}, \text{Retweet}, \text{Comment}\}$. We then consider the propagation of each tweet beginning from an initial condition where a single focal agent has state $\mathbf{s}_{f,0} = \text{retweet}$ and all other agents have state Null. At each time step, each neighbor of an active agent randomly selects a neighbor from the other agents with probability of agent $i$ choosing agent $j$ defined as $p_{0,ij}$, where $p_{ij} = 0$ if $i$ does not follow agent $j$. Then, agent $i$ updates their state by sampling from their policy distribution. If agent $j$ is inactive, agent $i$ remains inactive, and if agent $j$ is active, agent $i$ will choose the same state as agent $j$ with probability $p_{\text{copy},ij}$, and the opposite state with probability $p_{\text{opp},ij}$. With probability $p_{\text{null},ij} = 1 - p_{\text{copy},ij} - p_{\text{opp},ij}$, agent $i$ keeps the same state. Each active agent has a probability of becoming inactive each time step of $p_{\text{inactive}}$, though this agent may become active again responding to other agents later.

The policies described above represent a simple contagion process, where each exposure of an agent to an action leads to an independent probability of action. Although this may be a good way to model the policies of agents in a social network, other processes, in particular complex contagions[18], have also been proposed to describe how agents react to information. In a complex contagion the chance of transmission of a behavior is a non-linear function of the states of the neighbors of a given agent. For an agent with active neighbors we can write:

$$p_{\text{retweet},i} = G\left(\sum_{N_i} I_{\text{retweet},ij}(\mathbf{s}_{jt})\right) \tag{23}$$

$$p_{\text{comment},i} = H\left(\sum_{N_i} I_{\text{retweet},ij}(\mathbf{s}_{jt})\right) \tag{24}$$

$$p_{\text{null},i} = 1 - p_{\text{retweet},i} - p_{\text{comment},i}, \tag{25}$$

where the functions $G$ and $H$ saturate for large values of their arguments, are non-negative, and satisfy the constraint $G + H \leq 1$.

Simple and complex contagion give us several multiple frameworks to compare via system mapping. We have begun scraping twitter for data from several active sub-networks, beginning with the US Congressional election in NJ-02, and soon to potentially include sub-networks based on SARS-CoV2 and climate change mitigation. In each case, we begin by identifying a set of key Twitter accounts by hand and build a sub-network by examining the accounts which they follow, their followers, and accounts which

interact strongly with them. We repeat this process to find a highly cliquish sub-network with interactions focused on a single topic. Then we scrape Twitter and download the record of tweets, comments, retweets, and hashtags used by these accounts. We consider each original tweet as a potential contagion event, and use the spread or lack-thereof within the network to parameterize models of simple or complex contagion based on each sub-network, within a Bayesian statistical framework.

**From Collective Behavior of Animals to Social Behavior in Humans**

Complex Systems comprise agents interacting at small scales with their own policies, leading to emergent behavior at large-scales[**?**, **?**]. Social systems belong to the class of complex systems[**?**] and posses a variety of societally relevant emergent phenomena. Herding behavior, where agents adopt the same actions as their neighbors, lead to some of the most dramatic events in social and economic systems, including bubbles and market crashes in economics[**?**, **?**], the spread of rumors and misinformation, the changing of social norms[**?**], even to stampedes at crowded events[**?**]. Clustering during group formation or opinion dynamics can lead to the separation of agents into a small number of groups with distinct states, leading to polarization. Small-scale changes in the system can lead to abrupt shifts on the largest scale, critical transitions, which can describe political revolutions and other sudden societal shifts. Collective behavior can also reveal information about the underlying system[**?**], which is sometimes called the *Wisdom of the Crowds*[**?**], enabling agents with incomplete information to collectively combine their signals to predict prices, future events[16], or even act as detectors of illicit behavior[**?**].

In the past decade, we have made substantial progress modeling collective behavior in animal groups, a revolution driven by improvements in data collection and modeling. In particular, there has been a transition from simple models to learning the actual policies that animals use to make decisions, and this has led to improved prediction and understanding of macroscopic phenomena[**?**]. Detailed measurements of the sensory environments of animals, enabled using computer vision, catalyzed accurate modeling of their local causal neighborhoods and the policies they use to make decisions.

Human social behaviors show greater complexity than animals, but we have many analogies we can leverage and many of the same theoretical tools apply. Behaviors like herding, collective cognition, and simple heuristics for decision making exist in humans as well as in animals. Data exists on many aspects of human behavior, from out motions on the soccer field, to our movements through our interactions in cyberspace, and this high resolution data equals or exceeds the data available on animals. These changes have begun to transform the social sciences from a qualitative to a quantitative field, and we believe that the next step will allow the application of the same sophisticated tools which allow us to understand animal collectives to solve applied problems in the computational social sciences.

There exist huge potential payoffs for understanding human collective behavior, and one route to getting there is taking cues from successes studying animal groups. We increasingly live in a highly connected society, and collective behavior regularly has global impacts. These impacts can happen quickly, as rumors, fake news, and disinformation go viral online. On slightly longer time-scales, networks can grow around misinformation, leading to increased polarization and vulnerability to its spread. Collective spread of

misinformation and the resulting polarization has been weaponized by our adversaries and now represents a grave national security threat. The SARS-CoV2 pandemic rapidly spread around the globe, dominating our lives throughout 2020, and changes in individual behavior in response to the pandemic shaped its spread and represent one of the key tools for fighting it. To meet these challenges we need to build a comprehensive understanding of collective behavior, from the bottom up by understanding the policies by which humans make decisions, to the local causal neighborhoods which propagate their influence, and the largest macroscopic scale on which collective behavior emerges.

# References


[1] Constantin F Aliferis, Alexander Statnikov, Ioannis Tsamardinos, Subramani Mani, and Xenofon D Koutsoukos. Local causal and markov blanket induction for causal discovery and feature selection for classification part i: algorithms and empirical evaluation. *Journal of Machine Learning Research*, 11(1), 2010.

[2] Eytan Bakshy, Jake M Hofman, Winter A Mason, and Duncan J Watts. Everyone's an influencer: quantifying influence on twitter. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 65–74, 2011.

[3] Jalal Etesami, Kun Zhang, and Negar Kiyavash. A new measure of conditional dependence. *arXiv preprint arXiv:1704.00607*, 2017.

[4] G Flierl, D Grünbaum, S Levins, and Donald Olson. From individuals to aggregations: the interplay between behavior and physics. *Journal of Theoretical biology*, 196(4):397–454, 1999.

[5] Srijan Kumar and Neil Shah. False information on web and social media: A survey. *arXiv preprint arXiv:1804.08559*, 2018.

[6] Judea Pearl. Probabilist reasoning in intelligent systems. *Morgan Kauf-mann, San Mateo, California*, 1988.

[7] Judea Pearl. *Causality*. Cambridge university press, 2009.

[8] Svein Arne Pettersen, Dag Johansen, Håvard Johansen, Vegard Berg-Johansen, Vamsidhar Reddy Gaddam, Asgeir Mortensen, Ragnar Langseth, Carsten Griwodz, Håkon Kvale Stensland, and Pål Halvorsen. Soccer video and player position dataset. In *Proceedings of the 5th ACM Multimedia Systems Conference*, MMSys '14, page 18?23, New York, NY, USA, 2014. Association for Computing Machinery.

[9] Daniel M Romero, Brendan Meeder, and Jon Kleinberg. Differences in the mechanics of information diffusion across topics: idioms, political hashtags, and complex contagion on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 695–704, 2011.

[10] Martin Rumo. Magglingen2013, 2013.

[11] Peter Spirtes, Clark N Glymour, Richard Scheines, and David Heckerman. *Causation, prediction, and search*. MIT press, 2000.

[12] Bharath K Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, and Gert RG Lanckriet. Non-parametric estimation of integral probability metrics. In *2010 IEEE International Symposium on Information Theory*, pages 1428–1432. IEEE, 2010.

[13] LLC The Prediction Lab. Fork of https://github.com/google-research/football, 2021.

[14] Mathew Titus. hiton_ezk github repository, 2021.

[15] John Toner and Yuhai Tu. Flocks, herds, and schools: A quantitative theory of flocking. *Phys. Rev. E*, 58:4828–4858, Oct 1998.

[16] Andranik Tumasjan, Timm O Sprenger, Philipp G Sandner, and Isabell M Welpe. Predicting elections with twitter: What 140 characters reveal about political sentiment. In *Fourth international AAAI conference on weblogs and social media*. Citeseer, 2010.

[17] Cédric Villani. *Topics in optimal transportation*. Number 58. American Mathematical Soc., 2003.

[18] Duncan J Watts. A simple model of global cascades on random networks. *Proceedings of the National Academy of Sciences*, 99(9):5766–5771, 2002.