



A framework for modeling human behavior in large-scale agent-based epidemic simulations

Jan de Mooij¹ , Parantapa Bhattacharya², Davide Dell'Anna¹ , Mehdi Dastani¹, Brian Logan¹ and Samarth Swarup²

Simulation: Transactions of the Society for Modeling and Simulation International
2023, Vol. 99(12) 1183–1211



© The Author(s) 2023

DOI: 10.1177/00375497231184898

journals.sagepub.com/home/sim

 Sage



Abstract

Agent-based modeling is increasingly being used in computational epidemiology to characterize important behavioral dimensions, such as the heterogeneity of the individual responses to interventions, when studying the spread of a disease. Existing agent-based simulation frameworks and platforms currently fall in one of two categories: those that can simulate millions of individuals with simple behaviors (e.g., based on simple state machines), and those that consider more complex and social behaviors (e.g., agents that act according to their own agenda and preferences, and deliberate about norm compliance) but, due to the computational complexity of reasoning involved, have limited scalability. In this paper, we present a novel framework that enables large-scale distributed epidemic simulations with complex behaving social agents whose decisions are based on a variety of concepts and internal attitudes such as sense, knowledge, preferences, norms, and plans. The proposed framework supports simulations with millions of such agents that can individually deliberate about their own knowledge, goals, and preferences, and can adapt their behavior based on other agents' behaviors and on their attitude toward complying with norms. We showcase the applicability and scalability of the proposed framework by developing a model of the spread of COVID-19 in the US state of Virginia. Results illustrate that the framework can be effectively employed to simulate disease spreading with millions of complex behaving agents and investigate behavioral interventions over a period of time of months.

Keywords

Agent-based modeling, social simulation, synthetic population, computational epidemiology, COVID-19, *PanSim*, *Sim-2APL*

1. Introduction

Computational epidemiology is a multidisciplinary field that investigates issues related to epidemiology, such as the spread of diseases and the potential effectiveness of public health interventions, by using computational tools such as computer-based simulations. In computational epidemiology, simulations traditionally rely on mathematical compartmental models (e.g., the susceptible–exposed–infectious–recovered—SEIR—model) that characterize the disease being studied (e.g., its infectivity and transmissibility) and where the (relative) sizes of the compartments change through differential equations. While these models have become quite sophisticated over the years and advanced techniques have been applied to optimize the SEIR parameters (e.g., via swarm optimization¹), they mostly rely on assumptions such as homogeneity of the individuals in a population and the distinction of agents solely based on their disease state, that limit the accuracy

of the predictions that can be made.^{2–5} In the real world, however, the spread of a disease is not only affected by the properties of the virus itself, but also by socio-psychological behavioral dynamics that heavily rely on aspects that are heterogeneous across individuals (including spatial and demographic heterogeneity⁶). Squazzoni et al.⁷ point out that, given the significant adverse effects that behavioral interventions can have on individuals, decisions “cannot be solely based on epidemiological knowledge, because the efficacy of implementation

¹Division of AI and Data Science, Department of Information and Computing Sciences, Utrecht University, The Netherlands

²Biocomplexity Institute, University of Virginia, USA

Corresponding author:

Jan de Mooij, Division of AI and Data Science, Department of Information and Computing Sciences, Utrecht University, Buys Ballotgebouw, Princetonplein 5, 3584 CC Utrecht, The Netherlands.
Email: A.J.deMooij@uu.nl

Table 1. An overview of agent-based epidemic and other social simulations. *Epid.* denotes an epidemic simulation other than COVID—indicates “not stated” or “not applicable.”

	Year	Sim.	Scalability			Type of compliance	Deliberative
			Nr. agents	Temporal resolution	Time period		
Ferguson et al. ⁴¹	2006	Epid.	300M	6 h	6 months	Stochastic	No
Barrett et al. ⁴²	2008		100M	Event-based	4 months	Trigger-based	
Bisset et al. ⁴³	2009		16M	Event-based	6 months	Stochastic	
Bhatele et al. ⁴⁴	2017		280M	Event-based	6 months	Trigger-based	
Macal et al. ⁴⁵	2018		3M	1 h	10 years	—	
Bissett et al. ⁴⁶	2021		300M	Event-based	6 months	Trigger-based	
Ferguson et al. ⁴⁷	2020	COVID	300M	6 h	8 months	Stochastic	No
Gaudou et al. ⁸	2020		10–20K	1 h	7 months	Stochastic	
Müller et al. ⁴⁸	2020		3M	< 1 h	3 months	Stochastic	
Churches and Jorm ⁴⁹	2020		100–300K	1 day	6 months	Stochastic	
Ozik et al. ⁵⁰	2021		2.7M	1 h	12 months	Full	
Dignum ⁵¹	2021		1K	6 h	12 months	Agent decision	
Koehler et al. ⁵²	2021		10K	12 h	8 months	Full	
Abadeer et al. ⁵³	2023		30K	Event-based	30 days	Stochastic	
Sierhuis et al. ⁵⁴	2007	Other	—	—	—	—	Yes
Sakellariou et al. ⁵⁵	2008		—	—	—	—	BDI
Bordini and Hübner ⁵⁶	2009		JVM threads	—	—	—	BDI
Zhang and Nuttall ⁵⁷	2011		25K	1 month	24 months	Full	TPB
Caballero et al. ⁵⁸	2011		JVM threads	—	—	—	BDI
Barrett et al. ⁵⁹	2013		700K	30 min	2 days	Agent decision	No
Bulumulla et al. ⁶⁰	2022		35K	1 min	120 min	Agent decision	BDI
ours	2023	COVID	8M	Event-based	4 months	Agent decision	Yes

BDI: belief–desire–intention; TPB: theory of planned behavior.

depends on people’s reactions, pre-existing social norms, and structural societal constraints.”

To overcome this limitation and introduce complexity and heterogeneity in the models of individuals, agent-based modeling and simulation (ABMS) has been proposed as a key approach for more realistic computational epidemiology.⁸ ABMS allows to simulate actions, decisions, and social interactions of individual agents, providing a powerful tool for studying and explaining complex social phenomena related to the spread of diseases.^{9–11} The importance of ABMS for studying the impact of behavioral health interventions has become particularly evident during the recent COVID-19 pandemic. As a consequence, in the last few years a large number of agent-based simulations (see Tables 1 and 2 for an overview) have been used to predict and study the effect of interventions on disease spread by taking into account aspects such as age, profiles of people and households, interaction patterns along with their evolution in time and space, and individual attitudes.

Despite the recent proliferation of epidemic agent-based models, however, the modeled dynamics of individual decision-making are still limited.^{12–14} Compliance of individuals with behavioral interventions is typically modeled via stochastic processes that uniformly apply a certain level of compliance across the population.¹⁵ In reality, compliance can be highly nonuniform as it is the result of

complex sociological dynamics and of individual decision-making that depend on a number of factors, including demographics, peer influence, political orientation, risk assessments, and beliefs about the efficacy of the behavior.^{16,17} To build realistic simulations that can accurately anticipate the acceptance or rejection—and thus the effectiveness—of behavioral interventions, agents in epidemic simulations should be capable of autonomous decision-making. This requires more complex and detailed models of agents, that, for the purpose of this work, we will call *deliberative agents* to distinguish them from those other more simplified models of agents.

Deliberative agents should be capable of deliberation that realistically characterizes the decision-making of humans. Depending on the research question at hand, this could include the capability to take into account the effects of such concepts as habits, fatigue, and political preferences. Moreover, they should be norm-aware, i.e., explicitly capable of taking into account existing norms such as behavioral interventions or laws when deciding their actions. They should further be socially aware, i.e., take into account the behavior and mentalities of other agents, in their decisions whether to comply with behavioral interventions.^{18,19} Finally, to ensure accurate reflection of real-world decision-making, the design of deliberative agents should be backed by theory and grounded in data.

Table 2. An overview of other COVID-19 simulations using individual agent models with agent-based S(E)(A)R-like infection mechanisms.

	Platform	Agent paradigm	Agent population	Compliance	Contact network	Locations	Scalability (#agents)	Temporal resolution	Calibration
EpiSimdemics ^{42,44,46}	Custom + MPI/Charmm++	NFA	Synth. Pop.	Trigger-based	Co-evolving	Real	US Pop. (280M)	Event-based	Sensitivity analysis
EpiFast ⁴³	Custom + MPI	Network node	Synth. Pop.	Stochastic	Co-evolving (precomputed)	Real	US Pop. (280M)	Event-based	Not reported
COMOKIT ⁸	GAMA + MPI	AOP	Distribution	Stochastic	Co-evolving	Real	~10–20K per laptop	1 h	No
CityCOVID ⁵⁰	Repast	Microsim.	Synth. Pop.	Not reported	Co-evolving	Real	Chicago Pop. (2.7M)	1 h	Disease model
Müller et al. ⁴⁸	Symphony + MPI MATSim	Microsim.	1:4	Stochastic	Static	Movement destination	750K	Not reported	No
ASSOCC ⁵¹	NetLogo	AOP*	Distribution	Agent choice	Emergent	Distribution	1000	6 h	No
Koehler et al. ⁵²	NetLogo	Network node	Distribution	100%	Network-degree	Distribution	10K	12 h	No
Ferguson et al. ^{41,47}	Custom + OpenMP	Microsim.	Distribution	Stochastic	Emergent	Distribution	US/GB Pop. (300M)	6 h	No
Abadeer et al. ⁵³	Vadere	Locomotion model	N/a	Stochastic	Emergent	Real	30K	Event-based	No
PanSim + Sim-2APL	Custom + 2APL + MPI	Deliberative	Synth. Pop.	Agent choice	Co-evolving	Real	Virginia Pop. (8M)	Event-based	Disease and behavior models

BEN: behavior with emotions and norms; MPI: message passing interface.

*GAMA supports BDI agents through the BEN architecture. However, to the best of our knowledge, this has not been used for epidemic simulations.

Various theories of deliberative decision-making have been proposed over the years, from rational decision theories²⁰ to psychologically based approaches such as the theory of planned behavior (TPB)²¹ and the self-determination theory.^{22,23} These theories conceptualize decision-making behavior in terms of motivational, informational, and deontic attitudes, together with decision rules for the agents to select actions.^{24–26} Based on these theories, various agent decision-making models have been proposed. The belief–desire–intention (BDI) model,²⁷ for example, attributes to agents’ mental states such as beliefs, desires, and intentions, and characterizes the agents’ deliberation and reasoning in terms of these mentalistic notions.^{28,29} Other examples include cognitive models such as Adaptive Control of Thought – Rational (ACT-R)³⁰ and SOAR.³¹

Various surveys on the use of decision-making agents in social simulations identify scalability as a key issue that limits their applicability.^{18,32–35} The difficulty lies in scaling these individual models to the population sizes required for meaningful studies due to their computational complexity. Indeed, the vast majority of solutions for epidemic agent-based simulation in the literature falls in one of two categories: frameworks or platforms that can simulate millions of individuals with simple behaviors (e.g., based on simple state machines “triggers”-based behavior), and frameworks or platforms that consider more complex behaviors and social dynamics (e.g., agents that act according to their own agenda and preferences, in a social context, and deliberate about norm compliance) but have limited scalability. As a result, the current literature lacks a unified solution for epidemic simulations that can both scale and support deliberative agents that can model realistic human behavior, for example their deliberate response to interventions.

In this paper, we address this issue by introducing a novel framework that enables large-scale distributed epidemic simulations with deliberative agents. The proposed framework supports simulations with millions of agents that can individually deliberate about their own knowledge, goals, and preferences, and can adapt their behavior during the simulation based on observations of other agents’ behaviors and on their (possibly evolving) attitudes and intentions to comply with norms. To implement deliberative agents, we propose a novel adaptation of A Practical Agent Programming Language (2APL)³⁶—a well-known Java-based programming language that is designed to support the implementation of autonomous agents—which we call *Sim-2APL*. In addition, we propose *PanSim*, a novel epidemic simulation platform that allows the execution of large-scale agent-based epidemic simulations. This platform distributes the execution of individual deliberative agents over multiple computing resources in a synchronized manner, and efficiently and distributedly calculates the disease transmission. The source code for *PanSim* is available at <https://github.com/parantapa/>

pansim and that for Sim-2APL at <https://github.com/A-Practical-Agent-Programming-Language/Sim-2APL>³⁷.

We demonstrate the applicability and scalability of the framework by modeling the spread of COVID-19 in the US state of Virginia with ~8 million deliberative agents. Our simulation takes a synthetic population with realistic demographics, weekly activity schedules, and activity locations drawn from real location data as its inputs. Each individual in the synthetic population is modeled as a software agent, which decides its intentions regarding compliance with nonpharmaceutical interventions (NPIs) such as mask wearing and physical and social distancing that were introduced in Virginia in the period March through June 2020. In building and evaluating such a simulation, we report for the first time experiments about the spread of COVID-19 resulting from an agent-based simulation that involves individual deliberative agents at this scale.

This paper significantly extends and integrates our previous work,^{38,39} where we presented a preliminary version of the framework and a COVID-19 simulation experiment, respectively. In this paper, we describe an improved version of the framework that does not suffer from communication overhead issues highlighted in our preliminary experiments,³⁸ and supports simulations with ~8 million agents. Moreover, we extend the simulation experiment³⁹ in two ways: (a) we provide a formal characterization of how normative reasoning takes place in the deliberation cycle of the Sim-2APL agents; (b) we refine and recalibrate the model of norm-reasoning agents, and present more realistic results quantifying the effects of normative interventions in the state of Virginia.

The rest of the paper is organized as follows. We start with a discussion of the state of the art, followed by a presentation of the simulation platform PanSim and the agent programming language Sim-2APL, respectively. Note that our platform is distinct from a similarly named platform that has recently been published.⁴⁰ Since our platform was presented earlier,^{38,39} we have not changed the name in this paper. We then explain how the proposed simulation framework is used to design, develop, and calibrate a simulation experiment for COVID-19 regulations, and report on the strong and weak scalability of our proposed framework as well as the results of the conducted simulation experiments. Finally, we conclude the paper with a discussion of some remaining issues and pointing out directions for future research.

2. State of the art

The past 20 years have seen the development of a number of agent-based (often referred to as individual-based) epidemic and social simulations. Table 1 gives an overview of some of the most influential work based on the three

aspects that are the most relevant to this paper: (a) the scalability of the approach; (b) the type of mechanism used to determine compliance with behavioral interventions; and (c) whether the simulation involves deliberative agents.

In the great majority of the epidemic simulations that we analyzed, the disease progression is modeled via S(E)I(A)R-like models at the level of individual agents. Each agent is characterized by its disease state (which determines the so-called *compartment* to which the agent belongs). An agent's disease state changes over time according to the disease progression model, based on collocation that determines the probability of being exposed to a virus carried by an infectious agent in the same location. The representation of the population varies from one simulation to another (see Table 2). In some works, the characteristics of individual agents are generated ad hoc, e.g., by fitting agent demographics to *distributions* that represent statistical data about the area of interest.^{8,47,51,52} In other works, a *synthetic population* of agents is used so that every agent characterizes a particular individual in the real population.^{43,44}

Several works have focused on the scalability of the simulation by utilizing a relatively simple model of the agents. For example, Ferguson et al.⁴⁷ adapted an earlier simulation platform created in the context of Influenza⁴¹ to study COVID-19. By leveraging OpenMP,⁶¹ an application programming interface that provides support for multiprocessor shared-memory multiprocessing programming, the authors were able to simulate approximately 300 million agents representing the entire US population over a period of 6 months. To achieve scalability, the agents were modeled as relatively simple entities, and the contact network (i.e., the network that determines or describes which agents come in contact with each other) was based on locations visited per 6-h time slot. Compliance of agents with the implemented interventions was determined based on a set of probabilities. Bhatele et al.⁴⁴ have also demonstrated epidemic simulation scaling up to the size of the US population using EpiSimdemics.^{42,46} The simulator was heavily optimized for the particular application and architecture, which allowed it to compute each simulated day in 57.8 ms on 655,360 cores of the Blue Waters supercomputer. The effect of NPIs was studied using a trigger-based approach, i.e., executing a predetermined function when certain conditions of a trigger were satisfied, e.g., to change the disease state or disease mitigation behavior of an agent. Similarly, EpiFast⁴³ is an epidemic simulator that leverages a realistic synthetic population of agents. To support the execution of large numbers of runs and provide statistically sound estimates about stochastic evolution of epidemics, changes in the contact network due to NPIs are computed in advance, based on a predefined and uniformly applied degree of compliance.

In the context of COVID, the CityCOVID simulator⁵⁰ was built on *ChiSIM*,⁴⁵ a general-purpose community model for use in studying various forms of social dynamics in Chicago, including the spread of infectious diseases. The model leverages the Repast ABM framework⁶² to create a distributed memory simulation executable on high performance computing (HPC) systems. Each simulated hour, agents select one activity to perform at some location from a set of options derived from their assigned activity schedules. Agents can further choose to modify their behavior by wearing a mask or practice physical distancing, or can stay home instead of selecting an activity as a proxy for quarantining. The *stay-home* activities of the agents and the parameters of the disease model were calibrated against hospital bed occupation and COVID-19-attributed death data. Churches and Jorm⁴⁹ have developed COVOID, an open source generic individual-agent-based simulation for COVID-19 that can be easily adapted to different settings (e.g., countries or locations). In their model, on each day, the contacts between agents are randomly determined based on compartment-dependent parameters for the *act rate*: a parameter specifying, per day and per agent in the compartment, the number of social contacts with potential for infection. In their work, they have been able to simulate 1 million agents in just 3 h on a single compute core, achieving near-linear scaling by running multiple simulations in parallel.

Abadeer et al.⁵³ have developed a flexible tool that simulates the physics of disease transmission—based on factors such as speech volume, ventilation, agent movement, and proximity—in which users can change simulation parameters at run time. However, their agent models are pedestrian locomotion models, extended with SEIR states, and agent decision-making is limited to collision avoidance.

Another strand of work has considered more complex models of agents with the intent to characterize additional information, such as age, economic characteristics, social relationships, personal attitudes, and political preferences that can influence the behavior of individuals, their potential response to interventions, and, therefore, the spread of a disease.

Koehler et al.⁵² developed a COVID model in NetLogo and simulated specific counties in the United States. In their model, agents visit a number of locations given by a precalculated contact network degree, alternating one location for 12 h with staying at home for the other 12 h of the day. The network degree was calculated from a contact network derived from data provided by the American Community Survey (ACS) 2020,⁶³ following the approach of Meyers et al.⁶⁴ The proximity of agents is based on agents co-locating at a $0.1 \times 0.1 \text{ km}^2$ “patch” in the NetLogo environment. To address the complexity of the simulation, the authors propose a strategy where the

population of a county is scaled down to 10,000 agents, and the size of the environment is changed to reflect population density.

COMOKIT⁸ is a COVID-19 disease simulation system based on the GAMA⁶⁵ multi-agent simulation environment which uses a synthetic population of agents (which can be generated from statistical data or created using *gen**). The activity performed by an agent at each hour in the day is determined either by being enrolled in an activity by other agents, or by selecting the activity corresponding to the current day and time. Compliance with NPIs (i.e., norms) is determined by a central authority that the agents query to determine whether they can perform an activity or not. Normative reasoning is, therefore, delegated to the central authority, and individual agents comply with the received directives.

In the work summarized above, agents’ decisions about their daily activities are affected by a number of largely external factors (e.g., NPI telling agents to exhibit behavior like mask wearing and physical distancing, closure of public locations, and the health status of the agent causing them to quarantine). These approaches do not consider agents that explicitly and individually reason about compliance with norms during the simulation. To the best of our knowledge, the only framework to explicitly consider what motivates individuals to comply with an intervention is Agent-based Social Simulation of the Coronavirus Crisis (ASSOCC).⁵¹ In the ASSOCC framework, the reasoning of the agents is grounded in behavioral theories from psychology and sociology, and agents make their decisions based on their needs, such as belonging, compliance, financial stability, health, and leisure. These needs are implemented in a so-called water tank model,⁶⁶ in which the satisfaction of needs slowly decays. Agents try to keep the water tanks for their needs filled, but the importance of the various needs differs from agent to agent, based on societal and individual-level values.^{67,68} However, in their work, only 1000 agents could be simulated.

Employing deliberative agents allows for more realistic modeling of human decision-making. Increased detail and heterogeneity in agent characteristics improve the quality of prediction and forecasting of those models. Even more so, frameworks that can explicitly model individual agent decisions allow for *explanation*.^{9–11,69} if such a model is able to reproduce observations, its implementation is a plausible explanation of events. This could help relevant parties to not just respond to an ongoing situation, but learn from previous situations and increase preparedness for future events. Despite these advantages, from Table 1 and the discussion above, it can be seen that none of the studied agent-based epidemic simulations makes use of such deliberative agents.

As has been widely observed,^{18,32–34} the absence of deliberative agents is largely due to the difficulty of scaling multi-agent simulations involving deliberative agents.

Further support for this observation can be found by considering the scale of nonepidemiological (social) simulations that do make use of various types of deliberative agents.

For example, both Bordini and Hübner⁵⁶ and Caballero et al.⁵⁸ developed social simulations with BDI agents programmed in Jason, a platform that supports the distributed execution of a multi-agent system. The performance of distributed Jason applications has been studied by Pérez-Carro et al.⁷⁰ and Fernández et al.⁷¹ However, Pérez-Carro et al. distributed just 256 agents over a network, testing the effect on communication speed of interagent messages, and Fernández et al.⁷¹ found that, even when interagent messaging speed was acceptable, the communication with the environment quickly forms a bottleneck, regardless of the execution mode. Sierhuis et al.⁵⁴ developed Brahms, a agent-based simulation platform based on activity theory for the detailed simulation of organizational processes. They do not explicitly discuss scalability; however, the example described in their paper has a small number of agents. Sakellariou et al.⁵⁵ proposed a simplified BDI reasoning library for the simulation platform NetLogo. The example described in their work has only 20 agents, and the scalability of their approach is limited by that of NetLogo. While Railsback et al.⁷² have shown how NetLogo code can be optimized, the scale and complexity that can be achieved still fall short of the scalable simulations discussed previously. Moreover, while the literature contains work on using NetLogo on HPC, much of this is aimed at using large numbers of computers to run multiple models in parallel^{73–76} rather than distributing a single simulation. Bulumulla et al.⁶⁰ developed a multi-level framework integrating the BDI programming language Jack with simulation components representing physical and social layers to study evacuation scenarios. In their simulations, however, only 35,000 agents were considered. Singh et al.⁷⁷ have proposed an approach for integrating BDI agents with simulation, using a master–slave architecture, in which the environment acts as the master, and control over compute resources alternates between the master environment and the BDI agent models. However, their approach does not provide support for canceling or aborting *planned* activities, which is an important aspect when studying behavior change due to behavioral interventions. Zhang and Nuttall⁵⁷ developed an agent-based model based on the TPB, which is grounded in psychological theory. Their model aimed at predicting the effectiveness of British government policies on promoting smart meters in an energy grid. Their model considered only a single agent behavior and involved only 25,000 agents.

From the brief review of the state of the art presented above, we observe that: (a) the literature on agent-based epidemic simulations is split between models that consider only individuals with simple behaviors (e.g., based on

simple finite state machines or drawn from distribution) but scale to hundreds of millions of individuals, and models that consider more complex behaviors (e.g., considering age matrices, social differences, and individual choices) but have limited scalability; (b) there is currently no framework for large-scale distributed epidemic simulation with deliberative agents, and one of the reasons is the difficulty of scaling such models to sizes that are adequate for providing meaningful answers in an epidemiological context; and (c) in the existing literature on agent-based epidemic simulations, models that scale lack mechanisms for explicit and adaptive normative reasoning in the decision-making of agents.

In this work, we address the above observations. In our framework, the epidemic simulation platform PanSim calculates the physical disease progression, and the Java-based agent programming language Sim-2APL deals with the individual deliberation of agents. Our framework can distribute epidemic simulations on HPC systems, and employ a realistic synthetic population and a contact network to characterize the population of agents. Moreover, Sim-2APL is designed to impose few restrictions on the simulation author, allowing the use of different constructs to model more complex types of decision-making not natively present in the core library. For example, we will show how Sim-2APL can be used to introduce explicit normative reasoning directly in the individual agents' decision-making regarding compliance with behavioral interventions, without requiring changes to the core library. Indeed, the experiments reported in this paper are the first large-scale simulations of COVID with deliberative norm-aware agents with adaptive behavior. Moreover, our experiments demonstrate that simulations with millions of deliberative agents are feasible, with a degree of scalability comparable to other approaches in the literature that consider much more simplified models of agents.

3. PanSim: a framework for large-scale distributed epidemic simulation

In this section, we introduce PanSim, a distributed agent-based epidemic simulator with support for complex agent behavior models. PanSim simulations leverage extensive existing work on developing realistic synthetic populations of US states.⁷⁸ In these synthetic populations, each agent is assigned demographic variables, such as age, sex, race, household income, and political orientation. The baseline behavior of these agents is characterized by weekly activity schedules. Activities in the agent activity schedules comprise of them visiting specific locations during specific times of the week. Section 5 describes the particular synthetic population used for our COVID-19 case study in more detail. In a PanSim simulation, the transmission of the disease happens probabilistically when two agents are

visiting the same location at the same time. In addition, agents are able to observe visible behaviors of other agents (such as coughing, mask wearing, and social distancing) who are at the same location as them at the same time. PanSim allows custom behavior models to be incorporated in the simulation that can modify an agent's baseline activity schedule based on the decision-making logic.

3.1. Design objectives

PanSim has been built to satisfy the following design objectives:

- *Ability to incorporate complex agent behavior models:* As we have shown, while in recent years many papers have been published describing agent-based epidemic models, few have incorporated realistic human behavior models, which are critical for capturing the complex social dynamics of individual decision-making. Thus, the primary objective of PanSim's design is to allow its users to incorporate realistic human-like behavior models that govern agent activity.
- *Ability to incorporate existing behavior modeling frameworks:* While it is definitely possible to rewrite the complex behavior models, from scratch, in fast low-level languages such as C or C++, it is expensive and error-prone to do so. Therefore, the second objective of PanSim's design is to allow its users to leverage existing libraries written in high-level languages, such as Python and Java, for writing the behavior models.
- *Fast implementation of the disease model and epidemic simulation:* While complex behavior models need the flexibility of high-level languages, Susceptible - Infected - Recovered (SIR)-like disease models and epidemic simulations involve much simpler logic but represent a significant fraction of the compute time in epidemic simulations. Thus, it is important to ensure that the epidemic simulation component of the framework (disease transmission and progression) is written in a fast low-level language. Therefore, the third design objective of PanSim is to seamlessly combine fast epidemic simulation code written in a low-level language (like C or C++) together with complex behavior models written in high-level languages.
- *Ability to exploit distributed memory HPC systems:* Large-scale simulations are compute and memory intensive. To run significant number of these large-scale simulations in reasonable time periods, it is, therefore, necessary to be able to exploit large distributed memory HPC clusters for their execution.

Thus, the fourth design objective of PanSim is to ensure that PanSim is easy to run on these HPC clusters.

3.2. Conceptual model

PanSim is a distributed discrete-time agent-based simulation framework. A simulation in PanSim progresses sequentially in discrete time steps, and within a given time step, the simulation progresses in multiple sequential phases. However, within any given phase, computations corresponding to different agents progress concurrently and in parallel. Thus, PanSim's design can be described as bulk synchronous parallel design.⁷⁹

PanSim simulations use realistic synthetic populations of US states⁷⁸ for agent data and baseline agent mobility information. In these synthetic populations, agents visit different locations at specified times. These visits can be modeled as a temporal bipartite agent–location graph $G = \{V, E\}$. Here, the vertex set $V = V_a \cup V_l$ can be partitioned into two disjoint sets V_a and V_l representing the agents and the locations, respectively. An edge in this bipartite agent–location graph $e = \{v_a, v_l, (t_s, t_e)\} \in E$ is directed from a vertex representing an agent $v_a \in V_a$ to a vertex representing a location $v_l \in V_l$ that the agent visits. In addition, each edge is also annotated with the start time t_s and end time t_e of the visit. In PanSim, agents interact with each other only at the locations they visit. Agents that are at the same location at the same time come into contact with each other. In this way, even though the simulation *globally* progresses in discrete time steps, agent interaction is in fact discrete event based. In PanSim, locations visited by a given agent can change from one time step to the next, depending on the baseline activity model of the synthetic population and the logic of the behavior model. The epidemic simulation in PanSim progresses over the dynamic agent–agent contact network which is the unipartite projection on the agent set V_a of the temporal bipartite agent–location graph G .

To facilitate distributed computation at the beginning of every PanSim simulation, the agent–location bipartite graph G is partitioned into roughly equal-sized components. These components are distributed across compute nodes. The exact partitioning algorithm used is described in more detail later in this section.

In a PanSim simulation, an agent's state is comprised of two parts, its disease state (corresponding to the disease model) and its behavioral state (corresponding to the behavior model). The agent behavior model is responsible for deciding the activities that an agent undertakes—which locations the agent visits and when—and how the agent behaves during those visits. Outward behavior exhibited by the agents during the location visits is categorized into

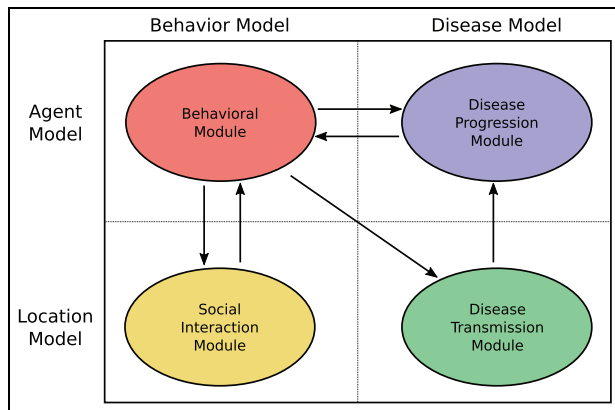


Figure 1. Structure of a PanSim simulation.

two classes: (a) disease modifier behaviors and (b) visible attribute behaviors. *Disease modifier behaviors*—such as wearing masks and social distancing—modify disease transmission properties, while *visible attribute behaviors*—such as displaying religious or political affiliations or symptoms of the disease—are used to indicate the agent’s stance and influence other agents.

In a PanSim simulation, during a given simulation time step, the following steps are executed. First, every agent in the system decides which locations to visit, and at what times, as well as how to “behave” during each of those visits. These behaviors include disease modifier behaviors and visible attribute behaviors. Second, when visiting a location, the agents come into contact with each other. During this step, disease transmission takes place probabilistically from infectious to susceptible agents. Also, while the agents interact, they observe each other’s visible attributes. *Finally*, for every agent, their disease state progresses and they update their behavioral state based on their current disease state as well as their observations of other encountered agents’ visible attributes.

3.3. Structure of a PanSim simulation

A PanSim simulation consists of four major modules: the behavior module, the social interaction module, the disease transmission module, and disease progression module. Figure 1 shows the overall organizations of the modules and their communication patterns.

The behavioral module and the social interaction module together represent the behavior model component of a PanSim simulation, while the disease transmission and progression modules together represent the disease model/epidemic component of the simulation. Another way of organizing the modules is to think of them from the perspective of the dynamic agent–location bipartite graph G that serves as the network on which both the epidemic progresses and on which information about social norms are exchanged. In this view, the behavioral and disease

progression models encapsulate the computation that happens on behalf of every agent/individual in the simulation, while the social interaction and disease transmission modules encapsulate the computation that happens on behalf of every location in the system.

To write a custom PanSim simulation, the simulation authors only need to provide the code for the behavior module. The rest of modules are provided by PanSim itself. PanSim provides a generic language-agnostic interface, written using Apache Arrow (<https://arrow.apache.org/>), that can be used to write the behavioral module in many high-level popular programming languages, including Java, Python, and R. The rest of the modules are implemented in PanSim itself in fast low-level languages. A formal description of PanSim is provided in Appendix A (Online Supplemental Material).

3.4. Disease model implementation

PanSim implements compartmental disease models,⁸⁰ also known as SIR-like disease models, for epidemic simulations. In these models, each agent can be in one of a finite set of disease states. The model also specifies the probabilities of transition between states—disease progression. In addition, when a “susceptible” agent comes in contact with an “infectious” agent, there is finite probability that the susceptible agent contracts the disease and moves to one of the “infected” states—disease transmission. The particular disease model used in the current work is a SEIAR model with five disease states: susceptible, exposed, infected symptomatic, infected asymptomatic, and recovered. In Section 5, we will show the specific parameters of the COVID-19 disease model. We refer readers to Brauer⁸⁰ for a more thorough discussion of compartmental models.

Since disease progression is time-dependent, we, in addition, annotate each state transition in the disease model with dwell time distributions. For each state transition, the disease model author also provides a distribution of transition times for the current state to the next state. During the disease progression step, once the specific next disease state has been selected, the transition time distribution is used to sample the duration after which the transition will occur. This addition makes the compartmental disease models used in PanSim a special case of Probabilistic Time Automata models.⁸¹

PanSim provides a custom domain-specific language to specify the disease model that it automatically parses and implements using a fast implementation written in a low-level language.

3.5. Distributed software implementation

PanSim is a distributed memory HPC application. For implementing PanSim we chose message passing interface

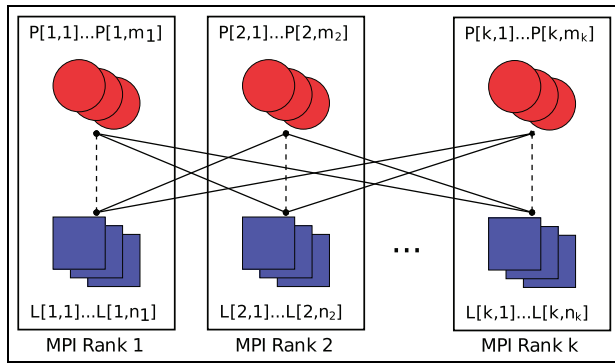


Figure 2. Partitioning of agents/individuals and locations for distributed processing on PanSim.

or MPI (<https://www.mpi-forum.org/>) as the distributed memory HPC messaging framework. PanSim makes heavy use of the nonblocking MPI communication primitives to overlap computation and communication phases. While many other distributed memory HPC messaging frameworks have been developed over the years (such as Charm ++,⁸² UPC ++,⁸³ and Legion and Regent⁸⁴), MPI remains the most widely supported HPC framework on HPC platforms.

PanSim is implemented in a mix of Python and C++. In PanSim, a C++ process (MPI rank) runs on each CPU core of each available compute node. The behavioral module, written in an arbitrary language, is run as a separate process and shares the CPU cores with the PanSim processes.

To ensure that the behavioral module processes and PanSim processes don't compete for CPU resources, we use MPI implementation-specific configuration to make PanSim processes sleep during the execution of the behavioral module. The PanSim and behavioral module processes on the same compute node communicate with each other using Apache Arrow tables using the Unix Interprocess Communication (IPC) mechanism.

To be able to utilize distributed computing hardware, the vertices in the agent–location bipartite graph G are statically partitioned across the MPI ranks at the beginning of the simulation. Figure 2 shows the overall partitioning strategy. To partition the graph evenly across the MPI ranks while keeping the cross-rank edges at a minimum, we use a two-step greedy process. In the first stage, the locations in the bipartite graph are sorted based on their maximum indegree. Next, the locations are assigned to the MPI ranks in a round-robin manner. Finally, the agents are assigned to the rank of the location that they are likely to visit the most frequently, which in most cases is their home location.

We have experimented with using Metis and ParMetis⁸⁵ for this partitioning. However, we found that our simple approach was much faster and produced adequately good partitions. Note that the partitioning method leverages the fact that agents in the simulation are modeled after real

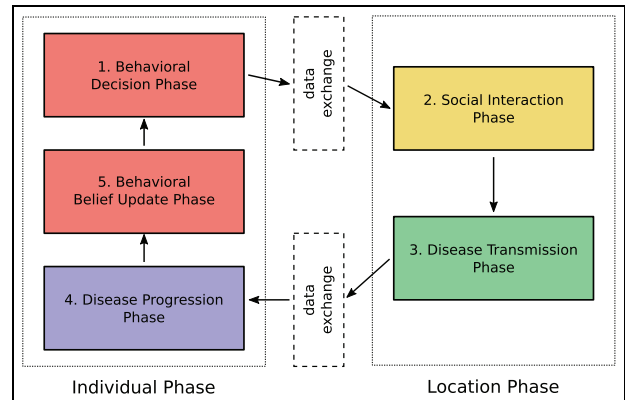


Figure 3. Different phases of computation in a single time step of a PanSim simulation.

people, who only frequently visit a small number of places. This partitioning, however, would significantly increase the overall runtime of the simulations if used in a scenario where each agent could visit any location uniformly at random.

PanSim uses a bulk synchronous parallel design.⁷⁹ A PanSim simulation progresses in discrete time steps. Within a time step, the execution progresses in five distinct phases, as described formally in Appendix A. The exploitation of parallel hardware in PanSim comes during each phase, during which computation related to all agents and locations can proceed in parallel.

Figure 3 shows the different phases of computation of a PanSim simulation for a single time step. First, in the behavioral decision phase, every agent decides the locations to visit, and how to behave during those visits. This is followed by data exchange among MPI ranks to transfer information to the rank corresponding to the location of the visits. Second, in the social interaction phase, the interactions of the individuals at every location are computed. Third, in the disease transmission phase, the probability of susceptible agents getting infected from visits is computed. After the third phase, data are again exchanged among the MPI ranks to send the social interaction and transmission updates back to the agents they correspond to. Fourth, in the disease transmission phase, the disease state of the agent is updated based on the transmission and progression models. Finally, in the behavioral belief update phase, the behavioral agent state is updated based on the social interaction and the updated disease state of the agent.

As shown in Figure 3, the first, fourth, and fifth phases of the simulation are collectively referred to as the individual phases. The computation of these phases can progress concurrently for every agent. Similarly, the second and third phases are location-specific and can be executed concurrently for every location.

3.6. Discussion

For the design of PanSim, we chose a discrete time architecture as opposed to the discrete-event architecture due to the relative simplicity of implementing efficient parallel discrete time models as opposed to parallel discrete-event models, which require additional event conflict detection and rollback logic (in case of an optimistic parallel discrete-event design) or event conflict avoidance/prevention logic (in case of a pessimistic parallel discrete-event design). In addition, due to the multilanguage nature of PanSim's design, a discrete-event design would incur high cost of going back and forth between the behavior model (implemented using an arbitrary language) and the disease and mobility model (implemented in C++).

PanSim uses static partitioning of the agent–location graph as opposed to dynamic partitioning.⁸⁶ While dynamic partitioning can potentially provide better load balancing, this comes at a cost of programming complexity as well as expensive migration and tracking costs. Migrating agents to different compute nodes involves serialization/deserialization of agent state data, destruction of agent objects on the source compute node, and reconstruction of the agent object at the destination compute node. Further additional dynamic tracking and routing mechanisms must be maintained to track the location of agents on compute nodes. In addition, since agent state in PanSim consists of two parts: disease model state (maintained by PanSim) and behavioral state (maintained by user code written by the simulation authors in higher level languages), this task is cumbersome. Thus, we trade off the potential benefits of dynamic load balancing for significant simplicity in implementation and ease of use for simulation authors.

4. Sim-2APL

We now discuss the behavior model of our framework, in which the individual software agents decide their visits based on their disease and behavior states. The agents implement the *Behavioral Module* in Figure 1. They perform the *Behavior Decision Phase* at the start of a PanSim simulation step and the *Behavioral Belief Update Phase* at the end of a PanSim time step (see Figure 3).

We introduce a novel agent programming language that we call *Sim-2APL*. Sim-2APL is an adaptation of 2APL,³⁶ a Java library for the implementation of autonomous software agents, i.e., software agents that sense their environment, decide their plans of actions, and execute those plans.

We first provide some necessary background on 2APL and discuss its limitations for simulation. Then, we present the key changes that make Sim-2APL. Finally, we illustrate how we model normative reasoning in the decision-making of Sim-2APL agents to support reasoning about compliance with NPIs during epidemic simulations.

4.1. Background: 2APL

2APL is a Java library for implementing autonomous software agents using object-oriented design patterns (see Dastani and Testerink³⁶ and Dastani⁸⁷ for technical details). A 2APL agent is programmed through four major components. First, the *belief base* is a set of data classes called *contexts*, whose fields' values store the agent-specific data or contextual information that the agent needs to determine its plan of action. Second, the *triggers/goal base* stores the objectives that an agent pursues or any triggers or events that the agent has to react to. Third, the set of *plan schemes* store conditional plans. An agent's plan has a goal/trigger condition and a belief condition such that the plan can be selected by the agent whenever its goal/trigger condition is satisfied by the agent's goal base (i.e., the goal/trigger is in the goal base) and its belief condition is satisfied by the agent's belief base. Finally, the *plan base* stores the plans that have been selected by the agent. These plans, which specify the behaviors of the agent, are Java programs that may change the state of the agent's environment, update the agent's belief and goal bases, or manage the communication with other agents.

The execution of a 2APL agent occurs through a cyclic process, called the *deliberation cycle*, which is shown in Figure 4. Each iteration of the deliberation cycle of a 2APL agent consists of two major steps: *plan selection*, where relevant plans are selected and inserted in the plan base, and *plan execution*, where all plans in the plan base are executed.

During the *plan selection* step of the agent's deliberation cycle, active *goals/triggers* (including received messages) are applied to the plan schemes for which the belief condition is satisfied by the belief base. Successful application of a goal/trigger to a plan scheme generates an instantiated plan that is subsequently added to the agent's plan base. Except for goals, each successfully applied trigger is immediately removed from the trigger base. Goals are only removed when achieved, i.e., when the `isAchieved` method, implemented by the programmer to test the agent's belief base, returns `true`. During the *plan execution* step, all plans from the agent's plan base are executed in order. If the plan is marked as *finished* at the end of its execution, it is removed from the plan base. Otherwise, it remains there to be executed again in the next iteration of the deliberation cycle. If a plan is instantiated through the successful application of a goal-type trigger, no new plans are instantiated for that trigger in subsequent deliberation cycles, unless that plan has been marked finished and removed from the agent's plan base.

In 2APL, the execution of agents and delivery of messages is handled by a so-called `Platform` that maintains a pool of threads on which the agent's deliberation cycles are scheduled for execution. By registering an agent with the platform, its deliberation cycle is automatically

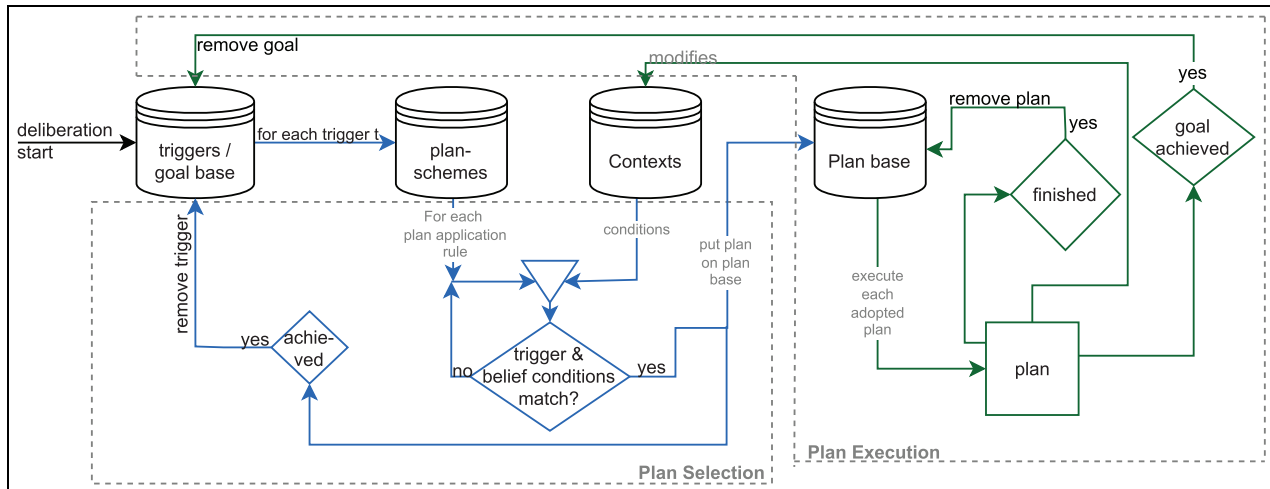


Figure 4. The 2APL deliberation cycle.

scheduled. As soon as an agent's deliberation cycle finishes, it is rescheduled automatically so its next cycle can be executed as soon as the thread pool queue has drained. One exception to rescheduling is when, after an iteration of the agent's deliberation cycle, its plan base and all its trigger bases are empty. In this case, the agent has no proactive behavior to pursue, and their deliberation cycle is not rescheduled to save CPU resources until a new external trigger or message is received.

4.1.1. Limitations of 2APL for simulation. As the agents can dynamically update their belief, goal, and plan bases, 2APL allows the implementation of many forms of complex proactive and reactive behavior.³⁶ However, 2APL has two main limitations that make it unsuitable for simulation.

4.1.1.1. Limitation 1 (scalability). 2APL is no exception to other agent programming languages with respect to limited scalability (as discussed in Section 2). More specifically, in 2APL, the amount of CPU resources that can efficiently be used in parallel is severely limited by the tight integration of agent execution and the execution of actions in the environment. This approach essentially makes 2APL a discrete-event-based system, where the events are any action performed by any agent in the environment. Because of the reasons outlined in the previous section, this poses an issue for efficiency if considered in a (distributed) simulation context.

4.1.1.2. Limitation 2 (lack of time-step synchronization). As discussed above, 2APL allows agents to directly change the environment by their plan execution. As a consequence, the agents are executed asynchronously both from each other and from the environment, and 2APL does not

contain an explicit notion of time. This means that agents may have different—possibly inconsistent—information about the environment's current state. Furthermore, because compute time of agents can differ due to the scheduling of the Java Virtual Machine (JVM), in a simulation context, some agents may be able to perform more deliberation cycles than others in the same (simulated) time, posing difficulties for the repeatability of a simulation.

4.2. Sim-2APL

Sim-2APL addresses the two limitations of 2APL outlined in the previous section. First, Sim-2APL separates the agent deliberation from the execution of actions by deferring the execution of actions until *after* the deliberation of all agents has completed. As a consequence, the behavior model implemented in Sim-2APL and the dynamics of the environment implemented in PanSim execute alternately. Second, an explicit notion of time steps is introduced to allow agents to execute exactly one deliberation cycle per time step. During its deliberation cycle, a Sim-2APL agent produces its actions, and these actions will be executed in the environment only at the end of the step. Specifically, each Sim-2APL time step consists of three consecutive stages: *Preparation*—in which belief update events from changes in the environment are sent to the agents, *Deliberation*—in which agents deliberate on the actions they will undertake in the current time step, and *Processing*—in which all agent actions are collected and processed in the environment (PanSim). We discuss these stages in detail in Appendix B (Online Supplemental Material).

Together, *action deferral* and *synchronized execution of agents* provide support for both scalability (*Limitation 1 of 2APL*) and time-step synchronization (*Limitation 2*). For users interested in code examples, or in creating new

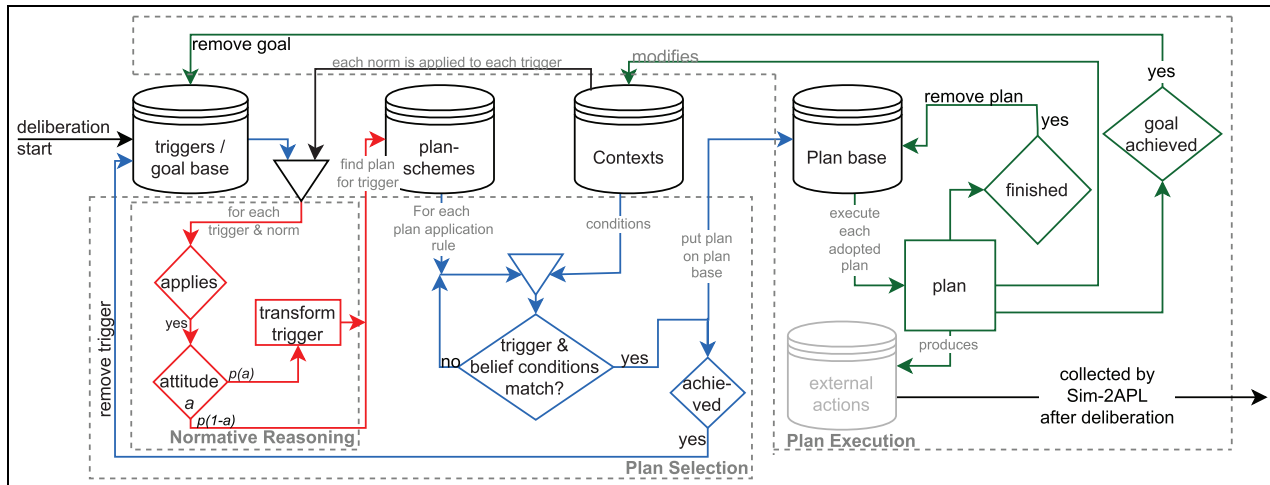


Figure 5. The normative deliberation cycle of a Sim-2APL agent starts with instantiating a plan for each trigger (transformed by the normative reasoning in the “Normative Reasoning” box) through the plan application rules in the plan schemes (“Plan Selection”) after which the instantiated plans are executed (“Plan Execution”) resulting in a ordered list of external actions.

simulations with Sim-2APL, a minimal demonstration simulation—without the interface with PanSim—is available at <https://github.com/A-Practical-Agent-Programming-Language/Sim-2APL-tutorial-and-demonstration>.

4.3. Normative reasoning in Sim-2APL

We now present the normative reasoning process (i.e., reasoning about *norms*) of (Sim-)2APL agents (while we discuss the normative reasoning here for ease of presentation, the process as described can also be used in non-simulation 2APL applications). A norm, in our framework, represents a behavioral intervention that is issued by a central authority (e.g., to wear mask or to close schools). We assume, therefore, that agents do not need to identify norms dynamically during the simulation (e.g., via imitation or learning⁸⁸). Instead, during the simulation, agents are notified about the enforcement of a new norm, by means of a norm event sent by the authority (in Sim-2APL’s *Preparation* stage). When a norm event is received by an agent for a particular norm n , the agent instantiates an internal representation of that particular norm, and decides whether to obey or violate the norm (via normative reasoning as described below) in the subsequent deliberation cycles. Therefore, Sim-2APL agents adapt their behavior to norms that are introduced dynamically during the course of a simulation.

We define a norm n instantiated by an agent as a tuple $\langle \text{condition}, \text{target}, \text{effect} \rangle$. The *condition* of n determines the context to which n applies, i.e., a set of beliefs of the agent. The *target* of n is the goal(s) of the agent to which the norm applies. The *effect* of n determines how that goal should change when the condition applies (i.e., the obligations, permissions, or prohibitions specified by the norm).

More specifically, the *effect* of a norm characterizes a *transformation* of the goals of the agent. Such transformation makes a goal *norm-aligned*. For example, a behavioral intervention to *close primary schools* may be represented by the norm $\langle \text{primary-school-kid}, \text{go-to-school}, \text{cancel} \rangle$, which specifies that if the agent believes it is of type *primary-school-kid*; the target goal *go-to-school* is *canceled*.

The normative reasoning process is implemented in the agents’ plan schemes, and its integration in the agent deliberation cycle is shown in Figure 5. During deliberation, before selecting a plan for some goal, the agent first finds all norms that *apply*, i.e., all norms for which the condition holds. For all the norms that apply, the agent determines its *attitude*, a , toward complying with the norm, i.e., a value in the range $[0, 1]$ representing the agent’s intention to comply with the norm. We interpret the attitude as the probability $p(a)$ that the agent complies with the norm (e.g., if $a = 0.75$, then there is 75% chance that the agent complies with the norm), but other evaluation strategies for the attitude (e.g., comparing against a threshold value) can be applied without changing the process outlined here. If the agent complies with a norm, it temporarily changes that goal for the current deliberation cycle to be norm-aligned (e.g., to cancel the activity in the case of school closure, or change the goal to include the wearing of a facial mask).

The agents’ attitudes are determined by particular beliefs they hold regarding their capabilities, the risks, and (observations of) the behavior of other agents. We call these beliefs *factors*. Each factor is a belief associated with a real value in the range $[0, 1]$ representing the evidence provided by the factor in support of compliance with a particular norm. Specifically, an agent’s attitude toward complying with a norm n is computed using equation (1):

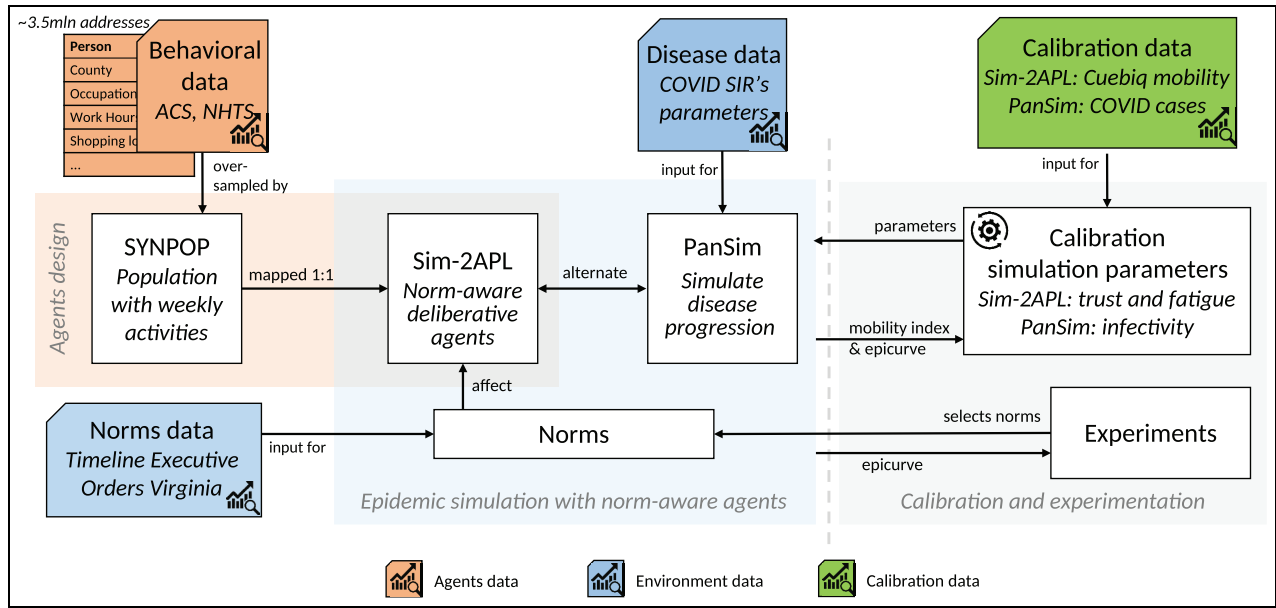


Figure 6. COVID-19 simulation setting.

$$a(n) = \frac{1}{1 + e^{(-k \cdot (\bar{f}(n) - t))}} \quad (1)$$

where t is the prior probability that the agent complies with the norm, $\bar{f}(n)$ is the average of the agent's values for the factors $\{f_1, \dots, f_m\}$ relevant for norm n , and k is a parameter representing the degree to which the agent is influenced by the factors (in this work we use $k = 10$). We interpret the prior, t , as the *trust* the agent has toward the institution enforcing norms. This design decision was influenced by previous work linking norm compliance with trust.⁸⁹ The higher the prior t , the more likely the agent is to comply with a norm, even if other factors (e.g., the observed compliance of other agents) provide low support for compliance. The general shape of this function for prior values $t \in \{0, 0.5, 1\}$ is shown in Appendix C (Online Supplemental Material). Since the value of the prior t and of the factors is different for different agents, some agents will be more influenced by their prior opinions, while others will be more influenced by the factors. If an agent has a very low or very high trust value (i.e., t is close to 0 or 1), the decision to comply (resp. not to comply) becomes more “resistant” to evidence supporting norm compliance (resp. noncompliance). We note that our approach to norm compliance is more flexible than many of the approaches to deliberating over norms in the literature. For example, a *norm-aware* variant of 2APL is presented in Alechina et al.,⁹⁰ in which agents can decide whether to comply with a conditional norm.⁹¹ However, in that work, compliance is determined by the relative utility of the target goal and the sanction the agent receives if the norm is violated. The agent's attitude toward the institution

enforcing the norm is not considered, nor is the agent's decision to comply affected by other factors, e.g., the compliance of other agents. Moreover, the approach in Alechina et al.⁹⁰ assumes that violations can be reliably detected, and the institution is able to effectively sanction violating agents; neither of these assumptions holds in our setting.

In the next section, we describe the norms and factors used in our epidemic simulation case study based on the NPIs implemented in Virginia, USA, against the spread of COVID-19, and how trust is computed.

5. A simulation of compliance with COVID-19 regulations

To illustrate and evaluate our framework, we describe the development and calibration of a simulation of behavioral responses to interventions during a pandemic in the context of COVID-19 (the full simulation model is available at <https://github.com/A-Practical-Agent-Programming-Language/Normative-COVID-19-Simulation>⁹²). The key components of this simulation are illustrated in Figure 6.

We start with a synthetic population of the US state of Virginia, where individuals are assigned realistic demographics, weekly activity schedules, and activity locations drawn from surveyed behavior and real location data. In our simulation, each individual in the synthetic population is represented by a Sim-2APL agent that reasons about whether to comply with the various NPIs that were implemented in the state of Virginia and that we model as *norms*. Sim-2APL agents interact through co-location resulting from their activity choices. Co-location is

calculated by and used for disease progression in PanSim as per Section 3.

In the following, we provide details about the data used to develop the simulation, and about the implementation of the agents and disease models.

5.1. Data sets used in the simulation

To model the agents and to calibrate and evaluate the simulation, we make use of the five data sets described below.

5.1.1. Synthetic population of Virginia, USA. Agents in our simulation are drawn from a synthetic population of the state of Virginia, USA. This synthetic population has been constructed from multiple data sources including the ACS, the National Household Travel Survey (NHTS), and various location and building data sets, as described in Adiga et al.⁷⁸ This gives us a very detailed representation of the region we are studying (multiple counties within Virginia). Agents are assigned demographic variables drawn from the ACS, such as age, sex, race, household income, or, optionally, a designation as an essential worker, e.g., medical or retail. The behavior of agents is characterized by weekly activity schedules, a set of typical daily activities the agents perform over the course of 1 week obtained by integrating data from the NHTS. The activity schedule defines the location, start time, and duration of the agent's activities as one of seven distinct high-level *activity types*: HOME, stay at or work from home; WORK, go to work or take a work-related trip; SHOP, buy goods (e.g., groceries, clothes, and appliances); SCHOOL, attend school as a student; COLLEGE, attend college as a student; RELIGIOUS, religious or other community activities; and OTHER, any other class of activities, including recreational activities, exercise, and dining at a restaurant. For example, one activity in an agent's schedule could state "SHOP at location l between 7 p.m. and 8 p.m." Appropriate locations are assigned to different activities using data from multiple sources, including HERE, the Microsoft Building Database, and the National Center for Education Statistics (for school locations).

5.1.2. Mobility data. To model the changes in agents' mobility due to various executive orders (EOs) implemented between March and July 2020, we use cellphone-based mobility data provided by Cuebiq. This data set contains location pings generated from the cellphones of a large number of anonymous and opted-in users throughout the United States. Cuebiq collected data with informed consent, anonymized all records, and further enhanced privacy by replacing pings corresponding to home and work locations with the centroids of the corresponding Census blockgroups. We aggregate the data to the county level as

follows. First, we calculate the average *radius of gyration* for cellphone users in the county. This is a metric that has extensively been used as a measure of changes in mobility and compliance with social distancing interventions during COVID-19.^{93–95} The radius of gyration is given by $r = \sum_l d(l, l_c)/k$, where l is the location (latitude and longitude) of the user, l_c is the centroid of all the locations visited by the user on that day, k is the number of locations visited by the user on that day, and d is the Haversine distance. We then calculate a *mobility index* as the percentage change in the average radius of gyration r over all users in a given region on a given day compared with the average for the same day of the week in the same region during January and February of 2020, i.e., before any EOs were issued. For example, the mobility index for a specific Monday in May 2020 is the percentage change in the average r on that day compared with the average over all Mondays in January and February 2020.

5.1.3. COVID-19 case data. We use county-level COVID-19 case data from US facts to calibrate the disease model in our simulation. A caveat is that the number of confirmed cases likely under-counted the number of actual cases substantially, especially early in the epidemic, due to limited testing. To account for this, in Section 5 we determine, via calibration of the simulation, a scale factor which we use to multiply the number of reported cases.

5.1.4. Disease model parameters. Table 3 shows the parameters with which we instantiate the disease model formalized in Section 3. The infectivity of symptomatically and asymptotically infected agents (isymp and iasymp resp) determine the probability of an infected agent infecting a susceptible agent if they spent one unit time (300 s here) in the same location, and are obtained through calibration, which is further explained in Section 5. After being exposed, the agent automatically transition to the next state after a *dwelt time*—drawn from a distribution—has passed. The state they transition to is specified by the *progression* probability. For example, an exposed agent will move to the *symptomatically infected* state with probability $p = 0.6$ after a number of days drawn from *dist1*, which for the current work is simply fixed to an incubation time of 6 days. The *behavior modifiers* determine how the infectivity probabilities (per unit time) are altered by behavior changes of the infected or susceptible individuals.

5.1.5. EOs in Virginia. We use a data set of EOs that were implemented in each state in the United States,⁹⁶ collected from the Johns Hopkins Coronavirus Resource Center.⁹⁷ From this, we extract the EOs that were implemented in Virginia in the period between 1 March and 30 June 2020. In our simulation, EOs are represented by one or more

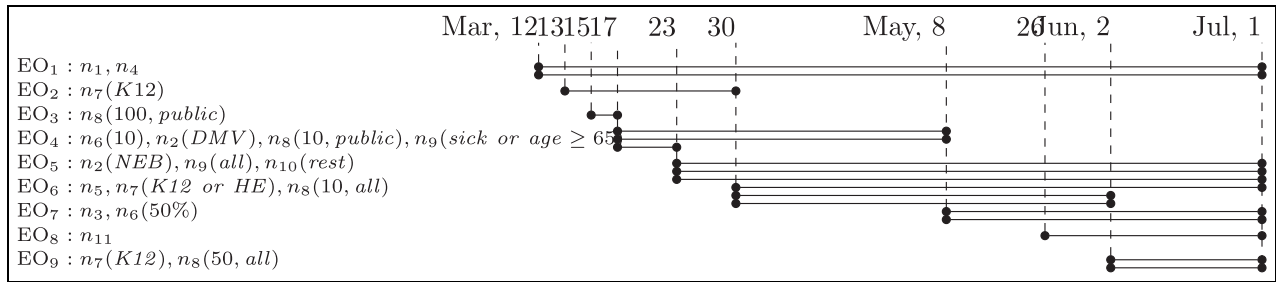


Figure 7. Timeline of the nine major EOs implemented in Virginia, USA, between March and June 2020. The column on the left indicates the norms belonging to the nine EOs. For each norm in an EO (e.g., n_1 in EO_1), a black line indicates the period during which the norm was enforced (e.g., the first two black lines indicate that both n_1 and n_4 from EO_1 were enforced from 12 March to 1 July).

Table 3. Simple contagion model (COVID-19 disease model).

Category	Parameter	Value
	unit_time	300.0
	states	[succ, expo, isymp, iasymp, recov]
	behaviors	[base, mask, sdist, mask_sdist]
	exposed_state	expo
susceptibility	succ	1
infectivity	isymp*	4.81e-05
	iasymp*	2.40e-05
progression	expo	{isymp = 0.6, iasymp = 0.4}
	isymp	{recov = 1.0}
	iasymp	{recov = 1.0}
dwelt time	expo	{isymp = dist1, iasymp = dist1}
	isymp	{recov = dist2}
	iasymp	{recov = dist2}
distribution	dist1	{dist = fixed, value = 6}
	dist2	{dist = fixed value = 14}
Behavior modifier	base	{base = 1.0, mask = 0.5, sdist = 0.5, mask_sdist = 0.25}
	mask	{base = 0.5, mask = 0.25, sdist = 0.25, mask_sdist = 0.15625}
	sdist	{base = 0.5, mask = 0.25, sdist = 0.25, mask_sdist = 0.15625}
	mask_sdist	{base = 0.25, mask = 0.15625, sdist = 0.15625, mask_sdist = 3.906e-3}

*Obtained through calibration.

norms that agents may obey or violate. We consider 11 norms representing a (simplified version of the) subset of EOs implemented in Virginia. In our simulation, the time when the norms are enforced corresponds to the time when

the corresponding EO was effectively put in place, as illustrated in Figure 7.

Table 4 provides the semantics of these norms. The parameters that are associated with the norms in the norm specify the applicability of a particular instance of the norm to particular types of activities or agents. For example, the *type* parameter of the *BusinessClosed* norm specifies the type of business to which the norm instance applies (e.g., an instance may specify that only nonessential business (*NEB*) should close). The *size* and *type* parameters of the *SmallGroups* norm specify the maximum size of groups permitted in a context of a particular type (e.g., no more than 10 people are allowed in a *public* space). Another example is the *type* parameter of *SchoolsClosed*, which specifies the grade levels that are closed: *K-12* specifies all K-12 level schools are closed, which means the norm applies only to activities of type *SCHOOL* when the agent performing the *SCHOOL* activity is attending K-12 level education.

5.2. Instantiating the behavior model

We instantiate the model such that each agent in the simulation represents one individual in the simulated counties of Virginia. These agents draw their demographic characteristics from the synthetic population and adopt (to-do) goals to perform activities from representative weekly activity schedules. The simulation progresses in discrete time steps that each represent 1 day. During the deliberation for a time step t , the agent selects a plan for each goal that represents an activity scheduled for the day of the week the active time step t corresponds to and then applying the normative reasoning process from Section 4. The instantiated plans selected by the agent can then produce an activity in the form of a *visit*, containing the visited location, the start time and duration of the visit, and the behaviors that the agent wishes to exhibit during that visit.

Table 4. A brief explanation of the norms enforced in our simulation and of their parameters.

Id	Interpretation	Parameters
n_1	Mask wearing is allowed and encouraged	—
n_2	Businesses of type <i>type</i> are closed	<i>type</i> \in { <i>NEB</i> }: the type of business, <i>NEB</i> = nonessential business
n_3	Employees working in retail must wear a mask during work activities	—
n_4	Telework is encouraged	—
n_5	Physical distance of 1.5 m should be maintained	—
n_6	Capacity of business should be reduced to <i>perc</i>	<i>perc</i> : percentage of business capacity
n_7	Schools of type <i>type</i> are closed	<i>type</i> \in { <i>K12</i> , <i>HE</i> , <i>K12 or HE</i> }: the type of school, <i>K12</i> = primary and secondary education <i>HE</i> = Higher Education
n_8	The maximum allowed size of groups of type <i>type</i> is <i>size</i>	<i>type</i> \in { <i>public</i> , <i>private</i> , <i>all</i> }: the target settings, either public, private, or both (<i>all</i>); <i>size</i> \in : maximum size of groups
n_9	Stay at home if belong to category <i>appl</i>	<i>appl</i> \in { <i>sick or age \geq 65</i> , <i>all</i> }: the group of agents to which the norm applies, either people sick or older than 65 (<i>sick or age \geq 65</i>), or everyone (<i>all</i>)
n_{10}	Only take away allowed for restaurants	—
n_{11}	A mask must be worn in public indoor settings	—

HE: higher education.

Table 5. Which activities are affected by regimented (R) and nonregimented (NR) norms, and the factors influencing the decision to comply with each norm.

Id	Norm	Activity type applicability (κ)						Agent beliefs factors					Transformation	
		WORK	SHOP	SCHOOL	COLLEGE	RELIGIOUS	OTHER	mask	distance	symptomatic	group	teleworking		
								f_1	f_2	f_3	f_4	f_5	T	
Nonregimented	n_1 MasksAllowed	✓*	✓	✓	✓	✓	✓	✓						T_1
	n_4 Telework	✓†								✓		✓		T_3
	n_5 Distance	✓	✓	✓	✓	✓	✓		✓					T_1
	n_8 SmallGroups #	✓†	✓	•	✓	•	•			✓		✓		T_3
	n_9 StayHome	✓†	✓	✓	✓	✓	✓							T_3
	n_{11} MandatoryMask		✓				✓	✓						T_1
Regimented	n_2 BusinessClosed		✓‡				✓‡							T_3
	n_3 EmployeesMask	✓§												T_1
	n_6 BusinessCapacity #		✓‡				✓‡							T_3
	n_7 SchoolsClosed			✓										T_3
	n_{10} TakeawayOnly						✓^							T_2, T_3

*Applies only if the parameter *type* includes *private*.

*If the agent has the essential designation *medical*, then always $a = 1$.

†Unless the agent as any essential designation.

‡Unless the activity location is marked *essential* or *residential*.

§If agent has the essential designation *retail*.

#As a proxy for agents communicating or being refused at the door, we stochastically apply this norm to the percentage of agents that results, on average, on the maximum allowed still continuing with the activity.

^This norm is applied only to visits with a duration of at least 20 min. Of those, we arbitrarily apply to only 10% because we have no data on what actual visits are restaurant visits. The transformation that is applied is t_2 with $p = 0.5$ and t_3 otherwise, as a proxy for some people canceling their plans of eating out, and others switching to takeaway.

In the following subsections, we describe the norms and the factors that we implemented in our COVID-19 simulation.

5.2.1. Norms and normative reasoning. We distinguish two types of norms:⁹¹ regimented norms (R)—which cannot be violated by the agents—and nonregimented (NR)—which the agents can autonomously decide whether to

comply with or not. In the case of R norms, the agents are regimented to always exhibit an *attitude* of 1. Note that regimentation in this context means the agents do not have choice. Some normative reasoning approaches apply sanctions⁹⁸ as a mechanism to enforce norms as an alternative to regimentation, meaning that, for example, business owners could choose to ignore the norm n_2 (business

closed) if they are willing to risk a fine. In our simulation, instead, we apply n_2 only to customers and clients of these businesses, and assume all businesses to which the norm applies (Table 5) are indeed closed. This means that, in our simulation, agents do not have choice regarding the norm n_2 , and we thus describe it as regimented.

We have identified which of the activity types each norm that we simulate applies to. These activity types are shown in Table 5 (activities of type HOME are always exempt from any norm, and thus not included in the table). These activity types determine the *target* of the norms instantiated by the agents. For example, the norm `Telework(n_4)` is only applicable to activities of type WORK. Furthermore, to determine the *condition* of applicability of the norms, we have also mapped each norm to belief conditions that every agent can assess individually. For example, the norm `StayHome(n_9)` is instantiated in `EO4` with parameter `sick or age ≥ 65` . We have modeled `sick` (as a consequence of the agent's disease state) and `age` as two distinct beliefs (the former Boolean and the latter numerical) that every agent has (and may update over the course of the simulation), so that the Java expression `sick || age ≥ 65` can be evaluated by the agent to assess whether the *condition* of the norm holds for them personally once the norm is enforced. For norm `EmployeesMask(n_3)`, we introduced a Boolean belief `works-in-retail` that indicates whether the agent works in retail.

As a special case, we mark the primary work location of each agent that is designated as an essential worker (i.e., with a belief indicating so), with the same designation. We also mark all locations that are used for the activity HOME as “residential.” The essential designations of agents or locations can form exceptions to the applicability of a norm. These exceptions are also indicated in Table 5.

For each norm that is not regimented and that applies to a goal of the agent, the agent calculates its attitude a based on their beliefs to decide whether they will comply as per equation (1). Norms can change the agents' default activities (goals) through one of the following three transformations:

- T_1 . The intention of a disease-modifying behavior (either wearing a mask or practicing physical distancing) is added to the goal.
- T_2 . The duration of the visit encoded by the goal is shortened.
- T_3 . The goal is replaced with a goal to stay home.

In the case of Transformation T_3 , the agent randomly decides to either change the visits' location of the affected goal to their home location or to not produce any visit at all (meaning it will drop the activity) with equal probability. The agent shifts all subsequent activities to start earlier so that there are no gaps in the schedule. The one exception here is if the next activity is of type WORK, which is

assumed not to be temporally flexible. In that case, the agent extends the duration of the activity right before the work activity to last until work starts. If the last activity in the schedule is of type HOME, the agent extends the duration so that it lasts until the end of the time step t . This ensures that, even when an increasing number of activities are canceled due to active NPIs, there continues to be some mixing in the population, despite the activities scheduled by default repeating weekly.

5.2.2. Trust. We assign each agent a value of prior *trust* t in the institution sampled from a beta distribution whose parameters are obtained through calibration of our model.

We assign each household either conservative with a probability equal to the fraction of Republican votes in the 2016 presidential elections in their county, and liberal otherwise. For each group, we create one beta distribution. For each agent, we sample their initial trust value at the start of the simulation from the beta distribution associated with the group of the household they belong to. The beta distributions are characterized by $Beta_v(\alpha_v, \beta_v)$ (with $v = C$ for conservative and $v = L$ for liberal), where $\alpha_v = \mu_v \cdot \kappa$, $\beta_v = (1 - \mu_v) \cdot \kappa$. The values of the means μ_C and μ_L are obtained through calibration (see Section 5); $\kappa = \alpha_v + \beta_v = 100$ characterizes the spread of the distribution, and, for simplicity, is fixed for both distributions.⁹⁹

The Cuebq mobility data show an increase in mobility after the initial reduction in the first few weeks when the first set of measures was instigated, without relaxations to these measures being applied. This suggests decreased compliance with norms over time. While the cause of this increase in mobility is speculation, we chose to model it as a form of *norm fatigue*. This norm fatigue is achieved by decreasing the prior trust attitudes of the agents with a constant f starting at t_f simulation steps (days) after the start of the simulation. Both f and t_f are subject to calibration but no distinction between voting preference is made for these two parameters.

5.2.3. Factors. We use five factors representing the beliefs that agents use as the evidence supporting norm compliance. All factors are real values in the range $[0, 1]$, with a higher value meaning the agent is more likely to comply.

- f_1 **mask.** If an agent observes more contacts wearing a facial mask, they are more likely to wear a mask themselves.
 $f_1 = \bar{\mu}_{\Delta d}^l$, where $\bar{\mu}_{\Delta d}^l$ is the *average* fraction of contacts at location l in the past d days who were wearing a facial mask.
- f_2 **distance.** If an agent observes more contacts following physical distancing guidelines, they are more likely to practice physical distancing themselves.

- $f_2 = \bar{p}_{\Delta d}^l$, where $\bar{p}_{\Delta d}^l$ is the *average* fraction of contacts at location l in the past d days who were practicing physical distancing.
- f_3 **symptomatic.** If an agent observes more contacts showing symptoms, they are more likely to comply with a norm as risk mitigation.
 $f_3 = \bar{s}_{\Delta d}^l$, where $\bar{s}_{\Delta d}^l$ is the *average* fraction of contacts at location l in the past d days who were symptomatic.
- f_4 **groups.** If an agent observes more contacts ($\bar{n}_{\Delta d}^l$) than are allowed by the maximum group size (m), their trust decreases, but they are more likely to comply with maximum group size interventions themselves. The effect increases exponentially the larger the observed difference ($\bar{n}_{\Delta d}^l - m$).
 $f_4 = 1 - \frac{1}{C \cdot (\bar{n}_{\Delta d}^l - m) + 1}$ where C is a constant indicating how quickly a larger-than-allowed number of contacts increases the agent's attitude toward compliance (in our model, $C = 0.4$). The shape of this function is shown in Appendix C (Online Supplemental Material).
- f_5 **teleworking.** $f_5 = 0.45$ indicating the probability an arbitrary agent can work from home.

In our model, $d = 14$ days for all factors. For each visit, an agent makes to a location l , PanSim reports n^l , μ^l , p^l , and s^l , i.e., the total number of agents, and the number of agents wearing a facial mask, practicing physical distancing, or showing symptoms, respectively. The average values over the past $d = 14$ days are calculated by the agents when they determine their attitude toward a norm.

In the current work, the value for f_5 is static and the same for all agents. This value has been taken from literature.¹⁰⁰ A more dynamic approach where different agents have their own probabilities to be accommodated to work from home based on their job and socio-economic status is left for future work.

5.3. Calibration

We now describe the calibration of the two components—the behavior model (as described in this section) and the disease model (Section 3)—of the simulation we have presented. We calibrate the key parameters of both models separately. The disease model is calibrated against the cumulative confirmed number of cases in Virginia to estimate the infectivity of symptomatically and asymptotically infected agents. The behavior model is calibrated against the mobility index, i.e., the percentage change in radius of gyration in each county.

Although both behavior and disease components are enabled in every simulation run that we perform for

calibration, we calibrate them independently in two separate processes, by fixing the component that is not the target of calibration for that process. The dynamics of one component affect the other, because agents base their decisions on observations of the behavior of other agents and in response to the disease progression, which in turn also depends on the behavior of agents. For this reason, we perform two rounds of calibration, where the first round serves to find initial estimates for both components, and the second round serves to obtain the final parameters.

Both calibration processes are performed by means of Nelder–Mead (NM) minimization.¹⁰¹ NM iteratively refines an initial configuration of parameters until it finds a local optimum that minimizes a given objective function, in this case the root mean square error (RMSE) between observations of each day in the simulated period in the real world and in the simulation initiated with the tested parameter configuration. The calibration simulations were performed using data from the counties of Charlottesville (41,119 unique agents in the synthetic population, 83.25% of which voted liberal, 16.75% conservative), Goochland (20,922 unique agents, 37.55% liberal, 62.45% conservative), Fluvanna (24,109 unique agents, 45.35% liberal, 54.65% conservative), and Louisa (32,937 unique agents, 37.60% liberal, 62.4% conservative). All in all, this means the calibration simulations were performed using a total of 119,087 agents, 54.92% liberal, 45.08% conservative. These counties have been selected for their proximity, number of agents, and variation in voting preference. For each set of parameters selected by NM, we run five different simulations to account for stochastic variation.

5.3.1. Agent parameters. We calibrate the four parameters of the agent model introduced in Section 10.2.2, i.e., the means μ_L and μ_C of the two beta distributions from which we sample the trust attitudes of liberal and conservative agents, respectively, the fatigue factor f , and the time step t_f in the simulation at which the fatigue becomes active. We calculate the RMSE between the mobility index in our simulation and in the real-world Cuebiq data (calculated as per Section 5). We apply a rolling average of 7 days to the mobility index before calculating the RMSE to normalize the intrinsic differences between the weekly repeating mobility trends between the synthetic population and the Cuebiq data.

We perform the first round of behavior calibration simulations with the parameters of the disease model fixed to $\langle inf_s = 4.5 \cdot 10^{-4}, inf_a = 3.375 \cdot 10^{-4} \rangle$. These parameters were chosen for being low enough to prevent the entire population to be infected too quickly, but high enough to sustain the disease progression in initial test

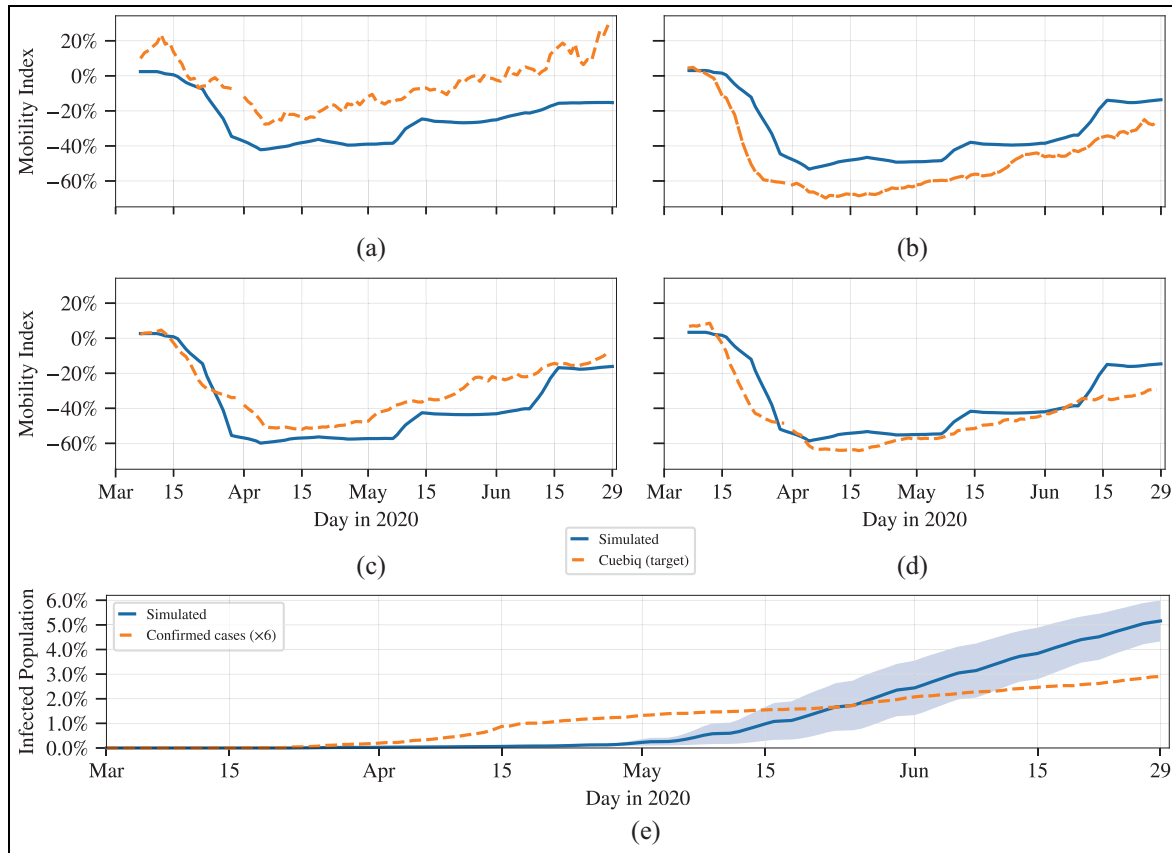


Figure 8. The mobility index observed in the simulation (solid lines) plotted against that recorded by Cuebiq (dashed lines) in each of the simulated counties of Charlottesville (a), Fluvanna (b), Goochland (c), and Louisa (d), and the cumulative percentage of the infected simulated population plotted against the number confirmed cases ($\times 6$) (e).

Table 6. Calibration results.

		Round 1	Round 2	Combined
Behavior	RMSE	15.74	15.73	15.77
	μ_L	$4.72 \cdot 10^{-3}$	$4.27 \cdot 10^{-4}$	$4.27 \cdot 10^{-4}$
	μ_C	0.712	0.662	0.662
	f	0.079	0.0342	0.0342
	t_f	111	110	110
Disease	RMSE	1150	1237	1405
	inf_s	$1.48 \cdot 10^{-5}$	$1.46 \cdot 10^{-5}$	$1.46 \cdot 10^{-5}$
	inf_a	$7.39 \cdot 10^{-6}$	$7.34 \cdot 10^{-7}$	$7.34 \cdot 10^{-7}$
	s	6	6	6

RMSE: root mean square error.

runs. In the second round of calibration, the parameters are fixed to those obtained from the first round of disease model calibration.

5.3.2. Disease model parameters. We calibrate two parameters of the disease model, the base level infectivity of symptomatic (inf_s) and asymptomatic (inf_a) agents, and a value s that we call the *scale factor*. We calculate the

RMSE by comparing the cumulative number of actually recorded infections in the four simulated counties with the total number of recovered agents in our simulation at each time step. To account for the uncertainty in testing in early 2020, we multiply the cumulative case count with the constant s before calculating the RMSE. In previous work,³⁹ we arbitrarily picked $s = 30$, while for this work, s is itself part of the calibration process. We performed the simulations for the first round of calibration with the parameters of the behavior model fixed to $\langle \mu_L = 0.5, \mu_C = 0.5, f = 0.0125, f_s = 60 \rangle$. In the second round of calibration, the parameters are fixed to those obtained from the first round of behavior model calibration.

5.3.3. Calibration results. Both models were calibrated until the objective function did not improve for 20 iterations. The final parameters obtained from the first two rounds of calibration are shown in Table 6. The last row (labeled “combined”) shows the average RMSE obtained for both components more than five simulations where both components were fixed to the parameters obtained from round 2. Figure 8 plots the mobility in each of the four simulated

counties separately, and the number of recovered agents (8e) in all of the four counties combined resulting from these parameters against the real data.

Table 6 shows that the best parameters for the disease model were obtained in the first round, and that the performance of both the behavior model and disease model was worse in the simulation where the final calibrated models were combined than what was obtained with the last calibration round. We believe this is due to minor changes in the starting configuration having a big impact on the final outcomes, so the calibration results in a precarious optimum that depends on the values with which the fixed component was instantiated. In the combined round, the fixed model had slightly different parameters than were used to obtain the calibrated values, but no further calibration took place, so a small negative impact on the results is to be expected. From the results shown in Table 6, it further appears the behavioral model is not very sensitive to the configuration of the disease model, with all parameters and the RMSE being very similar after both calibration rounds (although very different from the starting configuration). This also seems a valid conclusion from the small standard deviation (which is plotted, but hardly shows up in Figure 8) between the five simulations. The outcome of the disease model calibration does differ more starkly between simulation rounds, which we believe indicates a higher sensitivity to both the behavioral model and to the random perturbations resulting from the very low probabilities it concerns. In fact, looking only at the isolated RMSE of the disease component, the best fit appears to be obtained with the uncalibrated behavior model, and the worst fit with the behavior model fixed to the parameters obtained from the second round of calibration. However, both models improved on previous work, where the best RMSE obtained were 17.66 and 2052 for the behavior and disease models, respectively.³⁹

Comparing Figure 8(a)–(d), it can be seen that the recorded mobility changes varied widely from county to county. Despite our simulation using the same parameters for each county, it was still able to reproduce some of this variation, albeit less strongly than was recorded. The simulation was able to approximate the changes in mobility best in the county of Louisa (8d), closely followed by Goochland (8c). The observed mobility change in Charlottesville seems to be an outlier compared with the other simulated counties, with mobility *increasing* beyond the baseline in June. This increase is something that our simulation intrinsically cannot reproduce, as agents cannot choose to pick up activities not already in their activity schedules (on which the baseline is calculated). This is shown by a stronger simulated decrease in mobility than what was actually recorded, but this decrease is still less than for the other three counties.

From Figure 8(e), it can be seen that initially the outbreak in our simulation grows slower than what was

recorded, but that the simulated number of recovered agents overtakes the (scaled) number of recorded cases after about two-thirds of the simulation. The disease progression in the simulation is clearly exponential, while the recorded progression appears to be more linear (or even logarithmic). Whether this is (a) the result of under-testing in that period of time, (b) the result of randomness caused by the small numbers reported, (c) only showing the very start of a curve that is in actual fact exponential, or (d) an accurate reflection of the actual disease progression is unclear, but since the disease progression is exponential by necessity in our model, we do not expect we can obtain a much better fit with the current data.

5.4. Discussion

We have attempted to minimize the number of free parameters from which the model can be instantiated, to reduce the chance of over-fitting during calibration, but intercounty variations in mobility patterns suggested that mechanisms for decision-making should not be uniformly applied to all agents in the population. The heterogeneity of demographics that is spatially realistically assigned to agents should be able to account for some of this variation. However, on top of that, we looked for a mechanism specifically aimed at trust—which underlies all agents' attitudes—that we could use to distinguish different counties. Early in the pandemic, the topic of mask wearing has been considered highly polarizing in the United States. Although the media acknowledged many factors to be at play, partisanship was cited as highly influential in the context of mask wearing^{102,103} and social distancing,^{104,105} and correlations between the type of news consumption and inclination to wear masks have also been reported¹⁰⁶ (although it should also be noted that media polls suggest the effect of partisanship is grossly mis-estimated by opposing sides of the spectrum¹⁰⁷). We have not been able to find detailed statistics about types of news consumed in each county, so instead generalize to political voting data, which is available on a county-by-county level. However, the values for the means μ_L and μ_C of the beta distributions from which the agent's trust are sampled that were obtained through calibration (Table 6) do not reflect the correlations identified in the mentioned sources, suggesting the importance of partisanship has been overestimated, or at least is not significant in the simulation model we have proposed in this work.

To the best of our knowledge, our calibration approach is the first in the context of computational epidemiology that explicitly takes mobility data into account to calibrate the behavior of agents. Contrary to other data-driven models at scale, human decision-making is a key component of this model, and the parameters of that behavior should be grounded in data through calibration. Direct data on human decision-making in the context of NPIs are not

Table 7. The counties of the state of Virginia and the whole state used for the experiments, along with the number of persons, households, and weekly location visits in the synthetic population.

County	Persons	Households	Visits
Goochland	20,923	8240	680,571
Fluvanna	24,110	9776	779,337
Louisa	32,938	13,398	1,066,179
Charlottesville	41,120	18,377	1,335,596
Albemarle	93,570	39,920	3,047,807
Hanover	98,435	38,149	3,204,317
Henrico	298,354	125,782	9,700,944
Richmond	181,975	89,146	5,920,569
VA State	7,688,059	3,094,493	248,954,394

available at the scale required for calibration, but changes in mobility provide a suitable proxy,^{93–95} because a large number of the NPIs under consideration were aimed at reducing mobility.

6. Experiments

We now present two experiments to evaluate our framework, for which we use the calibrated simulation model presented in Section 5. *First*, we demonstrate the strong and weak scaling capabilities of our framework to show that it indeed supports the scaling up of epidemiological simulations that use realistic human-like models of deliberative agents. *Next*, we report on an experiment using a number of counterfactual simulations of the calibrated model to estimate the relative effectiveness of each of the nine EOs that were implemented in Virginia between March and June in 2020 to illustrate how such a model can be employed.

6.1. Scalability

For the purposes of the scaling experiments, we selected synthetic populations of eight counties in the state of Virginia, USA, with varying sizes, as well as the whole state of Virginia. Table 7 shows the number of persons, households, and their weekly activity schedule (location visits) in the synthetic populations of the selected counties, and of the entire synthetic population of Virginia. To understand the scalability of PanSim + Sim-2APL, we ran individual simulations for each of the eight counties, with each simulation simulating 1 week, representing the week starting from 2 March 2020, at which time the most interventions were active. For the eight counties, simulations were run with 40, 80, 160, and 320 CPU cores, and for the whole state of Virginia, the simulations were run on 320 and 640 CPU cores. All simulations were run on compute nodes that each have two Intel Xeon Gold 6148 CPUs with 20 CPU cores each. The compute nodes used to run the experiments also had 384 GB of DDR4 RAM

Memory and were connected to each other with Mellanox ConnectX-5 network adapters. Each simulation was run 10 times, and the running times were noted.

We study scaling in two ways. First, we study strong scaling by keeping the problem size fixed and increasing the number of CPU cores. This is done by running the simulation for each county with the four sets of core counts listed above, and for the whole state of VA with the two sets of core counts. The expectation is that the running time should decrease smoothly as the computational resources increase.

Second, to study weak scaling, we keep the computational resources fixed and increase the problem size. This is done by comparing the running times for simulations of increasingly larger counties, while keeping the number of CPU cores fixed. We carried out this experiment for all four sets of CPU core counts as well. The expectation is that the running time should not increase too sharply as the problem size increases.

In both cases, the resulting performance curves should ideally be linear. However, communication overheads can make the curves nonlinear. There is also inherent nonlinearity in the structure of the problem, as the disease spread computation is quadratic in the number of agents simultaneously present at a location. It is also expected that at some point, the overhead of communication between distributed parts of the simulation becomes higher than the efficiency gained by distributing the computation across multiple cores. For smaller problem sizes (i.e., smaller counties), this should become apparent with fewer cores.

6.1.1. Strong and weak scaling results. Figure 9 shows the variance in the runtime of the simulations when run with different numbers of CPU cores. We can see in Figure 9(a) that when the same simulation is run with increasing numbers of CPU cores (strong scaling), all eight counties show an almost linear decrease in run time up to 160 CPU cores on a log–log scale. In the case of the four smaller counties (the four lowest curves), increasing the number of CPU cores to 320 actually increases the runtime due the communication overhead becoming apparent, as discussed above. However, for a larger county like Henrico, the strong scaling results hold even with 320 CPU cores. For the simulation of the whole state of Virginia, increasing the number of CPU cores from 320 to 640 provides a speedup in runtime of 1.963.

A similar story can be seen for the weak scaling results shown in Figure 9(b), which shows the runtime of simulations with increasing compute load (number of individuals in the county simulated). We can see that for counties with more than 100K individuals, increasing the number of CPU cores to 320 shows definite benefits. However, for the rest of the counties simulated, the benefits of increasing CPU cores are observed only up to 160 CPU cores as

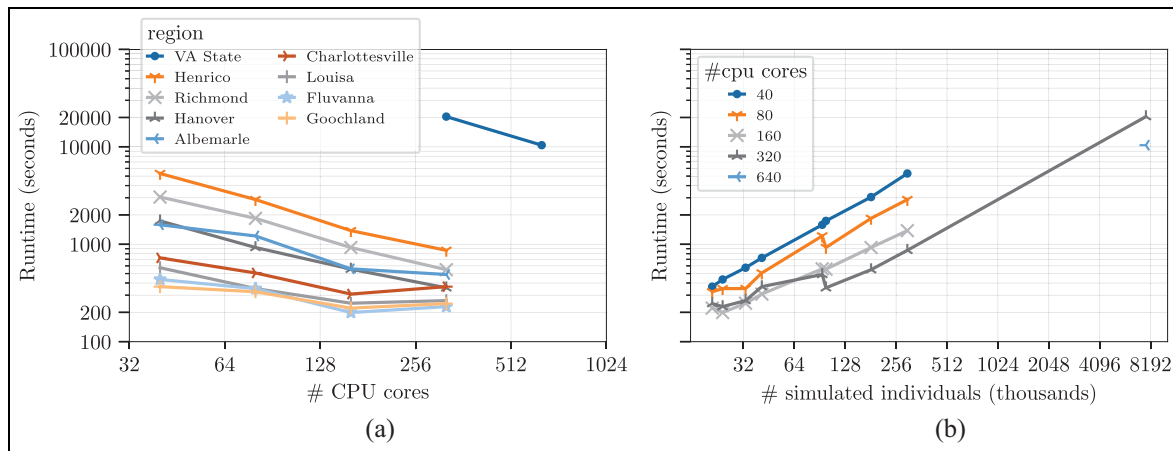


Figure 9. The variance in mean run time of PanSim + Sim-2APL simulations with respect to (a) the number of CPU cores used and (b) the number of simulated individuals, for eight counties in the state of Virginia and the whole state of Virginia.

there the compute load is insufficient for the benefits of distributing it to offset the additional overhead of MPI communication.

These results demonstrate that PanSim + Sim-2APL simulations integrate well, and can be used to simulate large populations by distributing the computational load.

The PanSim platform has been designed to efficiently scale complex multicontagion simulations. It assumes that the socio-psychological models that are being simulated using it are computationally heavy. In particular, for the current implementation the compute times—per agent, per time step—needs to be in the order of 1 ms for good scaling. The deliberation cycles of agents in Sim-2APL took in our experiments on average 500 ms. Our scaling studies have shown that our agent models were computationally too “light” to fully achieve the benefits of distributing the simulation, and the communication overhead dominated execution time. In our experiments we also tested with simpler and computationally faster models that took approximately 1 ms on average per deliberation cycle. In that case, overhead of the PanSim system started to dominate the runtime of the simulations for the smaller six of the eight counties tested with 160 CPU cores. Traditionally the community appears to exhibit some restraint in using individual agent-based models as opposed to stochastic models due to their computational demand. Our results, however, suggest that individual agent models are not only feasible, but that more computationally demanding agent models (i.e., many and complex reasoning behavior) provide more favorable conditions for scaling than more simple (individual agent) models.

6.2. Counterfactual simulations

In this subsection, we report experiments aimed at illustrating how the proposed framework can be applied for

policy study. In particular, we conduct a demonstrative policy study in which we quantify the efficacy of the various EOs in Virginia, taking into account the behavioral response of its residents to interventions, through counterfactual runs with our calibrated model described in the previous section. For this experiment, we employ the same 119,087 agents as were used for the calibration processes reported in Section 5.

Given the list of $n = 9$ EOs that were issued in Virginia as per Figure 7, we run 10 different experiments: in experiment E_i , for $0 \leq i \leq n$, we enact only the first i EOs. For example, in experiment E_0 , no norm is enforced, i.e., we simulate a scenario where no behavioral intervention takes place; in experiment E_1 , we enact only the first EO, i.e., norms $\{n_1, n_4\}$ starting from 12 March; in experiment E_2 we enact the first two EOs, i.e., $\{n_1, n_4\}$ starting from 12 March and also $\{n_t(K12)\}$ starting from 13 March, etc.

In each experiment, we compute the total number of agents that have been infected at the end of the simulation. We run each experiment 5 times to account for stochastic variation in the simulation.

Figure 10 shows the number of recovered agents at each time step in the simulations (SIR plots available in the code repository⁹²), with the standard deviation of the five runs shown as the confidence interval. E_0 shows that if no measures had been taken, the spread of COVID-19 would have been several times more rapid, with nearly 50% of the population infected within the first 4 months. The higher curves, associated with E_0 and E_1 , show the progression of the disease slowing down and nearly stopping near the simulation end. We believe this is caused by the simulation containing only 119,087 agents. After a sufficiently large portion of the population has been infected, already recovered agents form a buffer against the spread of the disease, as they cannot contract the virus again. This makes it relatively less likely for an infected agent to

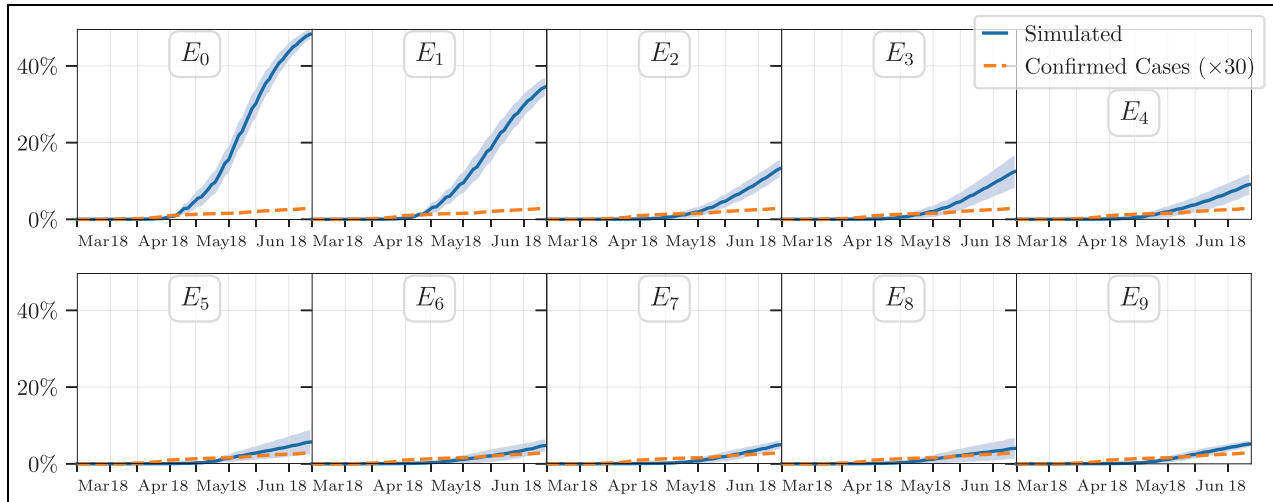


Figure 10. Cumulative cases in our simulation (solid line, with standard deviation plotted as confidence interval) and in the real-world ($\times 6$, dashed line) in each experiment $E_0 - 9$.

Table 8. The number of cumulative cases in each experiment.

E_i	Cumulative cases	Pct of pop.	Diff. w.r.t. E_{i-1}	Diff. w.r.t. E_0
0	57462.8 ± 1569.042	$48.25 \pm 1.32\%$		0.0%
1	41053.0 ± 2794.062	$34.47 \pm 2.35\%$	-28.56%	-28.56%
2	15725.5 ± 2703.217	$13.21 \pm 2.27\%$	-61.69%	-72.63%
3	14830.4 ± 4998.956	$12.45 \pm 4.20\%$	-5.69%	-74.19%
4	10809.4 ± 3226.403	$9.08 \pm 2.71\%$	-27.11%	-81.19%
5	6781.2 ± 3670.897	$5.69 \pm 3.08\%$	-37.27%	-88.2%
6	5666.0 ± 1887.58	$4.76 \pm 1.59\%$	-16.45%	-90.14%
7	5900.6 ± 1310.623	$4.95 \pm 1.10\%$	+4.14%	-89.73%
8	4793.5 ± 3193.588	$4.03 \pm 2.68\%$	-18.76%	-91.66%
9	6143.4 ± 991.514	$5.16 \pm 0.83\%$	+28.16%	-89.31%

encounter a susceptible agent. We expect this effect to be increasingly unlikely with larger population sizes, which would demonstrate the utility of larger-scale simulations.

Table 8 shows the total number of infected or recovered agents over the course of the simulation. The experiment E_9 corresponds to the scenario where all behavioral interventions were enforced the same way they have been in Virginia. Because the corresponding EO only contained relaxations of previous measures (n_7 : Higher Education reopens, and n_8 : maximum group size increased from 10 to 50 in all settings), the size of the outbreak is marginally larger than in E_8 , but the total number of agents that contracted the disease over the course of the simulation is almost 90% lower than if no behavioral interventions were put in place, showing the measures as a whole have been largely effective.

From the results, the single most effective EO appears to be the second (E_2) in which all K-12 level schools were closed. This EO resulted in a reduction of 61.69% in agents contracting the disease over the course of the

simulation compared with the previous intervention, and a reduction of 72.63% compared with no interventions at all. Interestingly, the first intervention which only provided suggestions rather than hard restrictions (work from home, n_4 , and the ban on face covering masks was lifted, n_1) already correlates with a significant decrease of 28.56% in infected agents compared with no interventions; the third most effective EO of all nine EOs.

The second most influential single EO appears to be the fifth, which extended the order to stay home when possible (n_9) to all groups, the closure of all NEBs (n_2) and closing restaurants except for takeaway (n_{10}).

Both EOs announcing relaxations of behavior interventions resulted in a slight increase in the number of infections. For the last EO, this increase was a difference of 28.16% compared with the previous EO, while these relaxations only came a single week after the previous restrictions, accounting for a very small difference in time between E_8 and E_9 . This may suggest the timing of this relaxation was ill-advised.

The simulations presented in this work have served as a proof of concept to demonstrate the applicability of the proposed framework. It should be noted that various simplifications have been applied to the motivations of the agents and to the actual norms enforced. Moreover, while we use an experiment where we study the effectiveness of the various EOs to demonstrate how our framework can be applied, in our simulation the time when the various EOs were issued (including relaxations) were fixed to the dates they were implemented in Virginia, while in reality they have been issued in response to the actual spread of COVID-19 at that time. Nevertheless, the results reported above show that behavioral responses of individual agents to normative interventions, and not just the effect of an a priori uniform assumption on the level of compliance, can be studied through our proposed simulation framework.

7. Conclusions

In this paper, we have introduced a novel epidemic simulation framework that allows for large-scale distributed simulations with deliberative norm-aware agents. First, in Section 3, we have introduced PanSim, a novel epidemic simulation platform that can (a) integrate realistic behavior models that can capture the complex social dynamics of individual decision-making, (b) incorporate existing agent behavior modeling frameworks written in high-level languages such as Python or Java, (c) seamlessly combine fast epidemic simulation code—written in a low-level language but easily specified in a simple declarative language—with these behavior models, and (d) exploit distributed memory HPC systems. Second, in Section 4, we have presented Sim-2APL, an agent programming language, based on 2APL, that supports the flexible implementation and execution of deliberative agents in discrete time distributed simulations executed with PanSim to implement a realistic human-like behavior model of agents' reasoning about compliance with nonpharmaceutical intervention norms. To demonstrate our framework combining PanSim + Sim-2APL, in Section 5, we have proposed a COVID-19 simulation with a population of agents instantiated from a representative synthetic population that explicitly models the normative reasoning of individuals about the NPIs issued in the state of Virginia in the United States in the period of March to June 2020. In contrast to similar simulations that uniformly apply compliance across the population, the agents in this work autonomously determine their attitude toward compliance with the NPIs based on a number of factors, most of which are dynamic and heterogeneous across the population. This attitude determines what locations agents visit which directly drives the disease progression. In this way, the (changing) behaviors of the population directly influence the progression of the studied disease. We have further

shown that a data-intensive calibration process is feasible with the proposed framework and model. In Section 6, we have (a) demonstrated both the strong and weak scalability of this framework through simulations with a population of 8 million individuals from the state of Virginia, as well as through several smaller simulations with population size ranging from 20,000 to 180,000 agents, and (b) shown how such a simulation can be used to study the efficacy of interventions under varying and heterogeneous compliance.

PanSim has been designed with the intention that the behavior module can be instantiated by arbitrary high-level behavior modeling frameworks written in, e.g., Python or Java. This allows simulation authors to use other, perhaps existing, models to take the place of Sim-2APL in this paper, or even use precomputed activities as its input. Moreover, many social phenomena can be modeled as contagion processes similar to the spread of disease, such as the spread of information¹⁰⁸ and misinformation,¹⁰⁹ the spread of technologies,^{110,111} fashions,¹¹² changes in language^{113,114} and more.^{115–118} PanSim allows, in its simple declarative language, the specification of “behaviors” that can also be used to represent, e.g., outwardly communicated ideas, information, and use of technologies. Moreover, the semantics of what we have called “locations” in this paper are given by the agents and in PanSim are just nodes in a graph. These locations might equally well be substituted for, e.g., newspapers or web pages (where “visits” become “reads”). All this allows PanSim to be used as a more generic environment and distribution platform than for just computational epidemiology. Sim-2APL supports similar generality. It can be easily connected to other existing simulation environments and platforms (that may or may not already contain complex environmental dynamics) different from PanSim.

In future work, we intend to synchronize the message passing between 2APL agents. In addition, we intend to make both our approach for normative reasoning and our proof-of-concept COVID-19 simulation more realistic, leveraging additional theories from the field of ABMS, psychology, and sociology. For example, while the effect of trust on compliance with NPIs is supported by the literature, the trends on mobility in Virginia show that trust is not static. In this work, we have taken *fatigue* as a proxy for decreasing trust, but have not considered the dynamics behind that fatigue. Moreover, the effect of political orientation reported in the media was not corroborated by our simulation. In future work, we intend to explore alternative mechanisms to political orientation for explaining the dynamics of trust evolution. In addition, other factors such as a fear factor, the influence of news or social media, (changes in) habits and routine, could be added to the decision mechanisms of the agents. With these additions, we hope to be able to provide more realistic epidemic simulations, which can help to identify important lessons from the COVID-19 pandemic and help policy makers

better understand the behavioral dynamics that affect the efficacy of NPIs. Finally, we are investigating methodology where our simulation framework can assist policy makers in determining which policies are most effective in bringing about an intended change in situations where the public's response to interventions is uncertain, rather than simply evaluating policies that have already been implemented.

Authors' Note

Brian Logan is now affiliated to School of Natural and Computing Sciences, University of Aberdeen, UK.


Acknowledgements


The authors thank Cuebiq; mobility data are provided by Cuebiq, a location intelligence and measurement platform. Through its Data for Good program, Cuebiq provides access to aggregated mobility data for academic research and humanitarian initiatives. This first-party data are collected from anonymized users who have opted-in to provide access to their location data anonymously, through a GDPR and CCPA compliant framework. To further preserve privacy, portions of the data are aggregated to the census-block group level. For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) license to any Author Accepted Manuscript version arising from this submission.

Funding

PB and SS were supported in part by NSF Expeditions in Computing Grant CCF-1918656.

ORCID iDs

Jan de Mooij  <https://orcid.org/0000-0003-4129-6074>

Davide Dell'Anna  <https://orcid.org/0000-0002-1162-8341>

Supplemental material

Supplemental material for this article is available online.

References

- Paggi M. Simulation of Covid-19 epidemic evolution: are compartmental models really predictive? 2020, <https://arxiv.org/abs/2004.08207>
- Chen J, Lewis B, Marathe A, et al. Chapter 12. Individual and collective behavior in public health epidemiology. In: Srinivasa Rao ASR, Pyne S and Rao CR (eds) *Disease modelling and public health, Part A*, vol. 36. Amsterdam: Elsevier, 2017, pp. 329–368.
- Funk S, Bansal S, Bauch CT, et al. Nine challenges in incorporating the dynamics of behaviour in infectious diseases models. *Epidemics* 2015; 10: 21–25.
- O'Neill PD. Introduction and snapshot review: relating infectious disease transmission models to data. *Stat Med* 2010; 29: 2069–2077.
- Alim M and Kesen SE. Investigating the effects of various control measures on economy and spread of Covid-19 in Turkey: a system dynamics approach. *Simulation* 2023; 99: 113–125.
- Davidson G, Fahlman A, Mereu E, et al. A methodological approach for modeling the spread of disease using geographical discrete-event spatial models. *Simulation*. Epub ahead of print 14 February 2023. DOI: 10.1177/00375497231152458.
- Squazzoni F, Polhill JG, Edmonds B, et al. Computational models that matter during a global pandemic outbreak: a call to action. *J Artif Soc S* 2020; 23: 10.
- Gaudou B, Huynh NQ, Philippon D, et al. COMOKIT: a modeling kit to understand, analyze, and compare the impacts of mitigation policies against the Covid-19 epidemic at the scale of a city. *Front Public Health* 2020; 8: 563247.
- Edmonds B. Computational modelling and social theory—the dangers of numerical representation. In: Mollona E (ed.) *Computational analysis of firm organization and strategic behaviour*. London: Routledge, 2010, pp. 36–68.
- Moss S and Edmonds B. Towards good social science. *J Artif Soc S* 2005; 8: 1–13.
- Hernán MA, Hsu J and Healy B. A second chance to get causal inference right: a classification of data science tasks. *CHANCE* 2019; 32: 42–49.
- Dignum F, Dignum V and Jonker CM. Towards agents for policy making. In: David N and Sichman JS (eds) *Multi-agent-based simulation IX*. Berlin; Heidelberg: Springer-Verlag, 2009, pp. 141–153.
- Silverman BG, Johns M, Cornwell J, et al. Human behavior models for agents in simulators and games: Part I: enabling science with PMFserv. *Presence: Teleop Virt* 2006; 15: 139–162.
- Gilbert N. When does social simulation need cognitive models? In: Sun R (ed.) *Cognition and multi-agent interaction: from cognitive modeling to social simulation*. Cambridge: Cambridge University Press, 2006, pp. 428–432.
- Verelst F, Willem L and Beutels P. Behavioural change models for infectious disease transmission: a systematic review. *J R Soc Interface* 2016; 13: 20160820.
- Becher M, Stegmueller D, Brouard S, et al. Comparative experimental evidence on compliance with social distancing during the COVID-19 pandemic. *medRxiv*, 2020, <https://www.medrxiv.org/content/medrxiv/early/2020/08/01/2020.07.29.20164806.full.pdf>
- Chan DKC, Zhang CQ and Weman-Josefsson K. Why people failed to adhere to COVID-19 preventive behaviors? Perspectives from an integrated behavior change model. *Infect Control Hosp Epidemiol* 2021; 42: 375–376.
- Adam C and Gaudou B. BDI agents in social simulations: a survey. *Knowl Eng Rev* 2016; 31: 207–238.
- Edmonds B and Moss S. From KISS to KIDS—an “anti-simplistic” modelling approach. In: Davidsson P, Logan B and Takadama K (eds) *Multi-agent and multi-agent-based simulation*. Berlin; Heidelberg: Springer, pp. 130–144.
- Dastani M, Hulstijn J and Van der Torre L. How to decide what to do? *Eur J Oper Res* 2005; 160: 762–784.
- Glanz K, Rimer BK and Viswanath K. *Health behavior and health education: theory, research, and practice*. Hoboken, NJ: John Wiley & Sons, 2008.

22. Deci EL and Ryan RM. The general causality orientations scale: self-determination in personality. *J Res Pers* 1985; 19: 109–134.
23. Ryan RM and Deci EL. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *Am Psychol* 2000; 55: 68–78.
24. Albarracín D, Kumkale GT and Johnson BT. Influences of social power and normative support on condom use decisions: a research synthesis. *AIDS Care* 2004; 16: 700–723.
25. Scalco A, Ceschi A, Shiboub I, et al. The implementation of the theory of planned behavior in an agent-based model for waste recycling: a review and a proposal. In: Alonso-Betanzos A, Sánchez-Marroño N, Fontenla-Romero O, et al. (eds) *Agent-based modeling of sustainable behaviors*. Cham: Springer, 2017, pp. 77–97.
26. Scalco A, Ceschi A and Sartori R. Application of psychological theories in agent-based modeling: the case of the theory of planned behavior. *Nonlinear Dynamics Psychol Life Sci* 2018; 22: 15–33.
27. Bratman ME. *Intention, plans, and practical reason*, vol. 10. Cambridge, MA: Harvard University Press, 1987.
28. Shoham Y. Agent-oriented programming. *Artif Intell* 1993; 60: 51–92.
29. Dennett DC. *The intentional stance*. Cambridge, MA: The MIT Press, 1989.
30. Anderson JR, Matessa M and Lebiere C. ACT-R: a theory of higher level cognition and its relation to visual attention. *Hum-Comput Interact* 1997; 12: 439–462.
31. Laird JE. *The Soar cognitive architecture*. Cambridge, MA: The MIT Press, 2019.
32. An L. Modeling human decisions in coupled human and natural systems: review of agent-based models. *Ecol Model* 2012; 229: 25–36.
33. An L, Grimm V and Turner BL II. Meeting grand challenges in agent-based models. *J Artif Soc S* 2020; 23: 13.
34. Balke T and Gilbert N. How do agents make decisions? A survey. *J Artif Soc S* 2014; 17: 13.
35. Swarup S, Eubank SG and Marathe MV. Computational epidemiology as a challenge domain for multiagent systems. In: Bazzan ALC, Huhns MN, Lomuscio A, et al. (eds) *International conference on autonomous agents and multi-agent systems (AAMAS'14)*, Paris, May 5–9, 2014. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2014, pp. 1173–1176.
36. Dastani M and Testerink B. Design patterns for multi-agent programming. *Int J Agent-Orient Softw Eng* 2016; 5: 167–202.
37. De Mooij J, Dell'Anna D, Bhattacharya P, et al. Sim—2APL—a practical agent programming language for simulation (v2.0.0), 2021, <https://doi.org/10.5281/zenodo.5761707>
38. Bhattacharya P, De Mooij J, Dell'Anna D, et al. PanSim + Sim-2APL: a framework for large-scale distributed simulation with complex agents. In: *Proceedings of the 9th international workshop on engineering multi-agent systems*, https://dl.acm.org/doi/10.1007/978-3-030-97457-2_1
39. De Mooij J, Dell'Anna D, Bhattacharya P, et al. Quantifying the effects of norms on COVID-19 cases using an agent-based simulation. In: *Proceedings of the 22nd international workshop on multi-agent-based simulation (MABS)*, <https://link.springer.com/book/10.1007/978-3-030-94548-0>
40. Reguly IZ, Cserecsik D, Juhász J, et al. Microsimulation based quantitative analysis of COVID-19 management strategies. *PLoS Comput Biol* 2022; 18: e1009693.
41. Ferguson NM, Cummings DA, Fraser C, et al. Strategies for mitigating an influenza pandemic. *Nature* 2006; 442: 448–452.
42. Barrett CL, Bissett KR, Eubank SG, et al. EpiSimdemics: an efficient algorithm for simulating the spread of infectious disease over large realistic social networks. In: *Proceedings of the 2008 ACM/IEEE conference on supercomputing*, Austin, TX, 15–21 November 2008, pp. 37:1–37:12. New York: IEEE.
43. Bissett KR, Chen J, Feng X, et al. EpiFast: a fast algorithm for large scale realistic epidemic simulations on distributed memory systems. In: *Proceedings of the 23rd international conference on supercomputing*, Yorktown Heights, NY, 8–12 June 2009.
44. Bhatele A, Yeom JS, Jain N, et al. Massively parallel simulations of spread of infectious diseases over realistic social networks. In: *Proceedings of the 17th IEEE/ACM international symposium on cluster, cloud and grid computing (CCGRID)*, Madrid, 14–17 May 2017. New York: IEEE.
45. Macal CM, Collier NT, Ozik J, et al. chiSIM: an agent-based simulation model of social interactions in a large urban area. In: *Proceedings of the 2018 winter simulation conference (WSC)*, Gothenburg, 9–12 December 2018, pp. 810–820. New York: IEEE.
46. Bissett KR, Cadena J, Khan M, et al. Agent-based computational epidemiological modeling. *J Indian I Sci* 2021; 101: 303–327.
47. Ferguson NM, Laydon D, Nedjati-Gilani G, et al. Impact of non-pharmaceutical interventions (NPIs) to reduce COVID-19 mortality and healthcare demand, 2020, <https://www.imperial.ac.uk/media/imperial-college/medicine/sph/ide/gida-fellowships/Imperial-College-COVID19-NPI-modelling-16-03-2020.pdf>
48. Müller SA, Balmer M, Neumann A, et al. Mobility traces and spreading of COVID-19. Technical report, Technische Universität Berlin, Berlin, 20 March 2020.
49. Churches T and Jorm L. Flexible, freely available stochastic individual contact model for exploring Covid-19 intervention and control strategies: development and simulation. *JMIR Public Health Surveill* 2020; 6: e18965.
50. Ozik J, Wozniak JM, Collier N, et al. A population data-driven workflow for COVID-19 modeling and learning. *Int J High Perform C* 2021; 35: 483–499.
51. Dignum F. *Social simulation for a crisis*. Cham: Springer, 2021.
52. Koehler M, Slater DM, Jacyna G, et al. Modeling COVID-19 for lifting non-pharmaceutical interventions. *J Artif Soc* 2021; 24: 9.
53. Abadeer M, Magharious S and Gorlatch S. Modeling and interactive simulation of measures against infection transmission. *Simulation* 2023; 99: 327–346.

54. Sierhuis M, Clancey WJ and Van Hoof RJ. Brahms: a multi-agent modelling environment for simulating work processes and practices. *Int J Simul Process Model* 2007; 3: 134–152.
55. Sakellariou I, Kefalas P and Stamatopoulou I. Enhancing NetLogo to simulate BDI communicating agents. In: Darzentas J, Vouros GA, Vosinakis S, et al. (eds) *Artificial intelligence: theories, models and applications (SETN 2008): Hellenic conference on artificial intelligence*. Berlin; Heidelberg: Springer, 2008, pp. 263–275.
56. Bordini RH and Hübner JF. Agent-based simulation using BDI programming in Jason. In: Uhrmacher AM and Weyns D (eds) *Multi-agent systems: simulation and applications*. Boca Raton, FL: CRC Press, 2009, pp. 451–471.
57. Zhang T and Nuttall WJ. Evaluating government's policies on promoting smart metering diffusion in retail electricity markets via agent-based simulation. *J Prod Innovat Manag* 2011; 28: 169–186.
58. Caballero A, Botía J and Gómez-Skarmeta A. Using cognitive agents in social simulations. *Eng Appl Artif Intel* 2011; 24: 1098–1109.
59. Barrett C, Bisset KR, Chandan S, et al. Planning and response in the aftermath of a large crisis: an agent-based informatics framework. In: *Proceedings of the 2013 winter simulation conference* (ed Pasupathy R, Kim SH, Tolk A, et al.), Washington, DC, 8–11 December 2013, pp. 1515–1526. Piscataway, NJ: IEEE Press.
60. Bulumulla C, Singh D, Padgham L, et al. Multi-level simulation of the physical, cognitive and social. *Comput Environ Urban* 2022; 93: 101756.
61. OpenMP Architecture Review Board. OpenMP application program interface (version 3.0), 2008, <http://www.openmp.org/mp-documents/spec30.pdf>
62. Collier NT and North MJ. Parallel agent-based simulation with repast for high performance computing. *Simulation* 2013; 89: 1215–1235.
63. US Census Bureau. About the American community survey, 2020, <https://www.census.gov/programs-surveys/acs/about.html>
64. Meyers LA, Pourbohloul B, Newman ME, et al. Network theory and SARS: predicting outbreak diversity. *J Theor Biol* 2005; 232: 71–81.
65. Taillandier P, Gaudou B, Grignard A, et al. Building, composing and experimenting complex spatial models with the GAMA platform. *GeoInformatica* 2019; 23: 299–322.
66. Vanhée L, Dignum F and Ferber J. Modeling culturally-influenced decisions. In: *Proceedings of the international workshop on multi-agent systems and agent-based simulation*, Paris, 5–6 May 2014, pp. 55–71. Cham: Springer.
67. Schwartz SH. An overview of the Schwartz theory of basic values. *Online Read Psychol Cult* 2012; 2: 11.
68. Mercuur R, Dignum V and Jonker C. The value of values and norms in social simulation. *J Artif Soc S* 2019; 22: 1–9.
69. Watts DJ, Beck ED, Bienenstock EJ, et al. Explanation, prediction, and causality: three sides of the same coin? 2018, <https://ideas.repec.org/p/osf/osfxxx/u6vz5.html>
70. Pérez-Carro P, Grimaldo F, Lozano M, et al. Characterization of the Jason multiagent platform on multi-core processors. *Sci Programming* 2014; 22: 21–35.
71. Fernández V, Grimaldo F, Lozano M, et al. Evaluating Jason for distributed crowd simulations. In: Filipe J, Fred ALN and Sharp B (eds) *Proceedings of the international conference on agents and artificial intelligence (ICAART)*, Valencia, 22–24 January 2010, vol. 2. INSTICC Press, 2010, pp. 206–211.
72. Railsback SF, Ayllón D, Berger U, et al. Improving execution speed of models implemented in NetLogo. *J Artif Soc S* 2017; 20: 3.
73. Thompson J, Zhao H, Seneviratne S, et al. Improving speed of models for improved real-world decision-making, 2021, <https://github.com/melbhz/netlogo-hpc>
74. Olsthoorn D and Nikolic I. Creating an application to run NetLogo on HPC clusters, 2018, <https://github.com/daveol/NetLogo-HPC>
75. Puig AJ. NetLogo-cluster, 2014, <https://github.com/jurnix/netlogo-cluster>
76. Tashakor G and Suppi R. HPCNetLogo, 2017, <https://github.com/hpcnetlogo/hpcnetlogo>
77. Singh D, Padgham L and Logan B. Integrating BDI agents with agent-based simulation platforms. *Auton Agent Multi-Ag* 2016; 30: 1050–1071.
78. Adiga A, Agashe A, Arifuzzaman S, et al. *Generating a synthetic population of the United States*. Technical report NDSSL 15-009, January 2015. Blacksburg, VA: Network Dynamics and Simulation Science Laboratory, Virginia Tech.
79. Gerbessiotis AV and Valiant LG. Direct bulk-synchronous parallel algorithms. *J Parallel Distr Com* 1994; 22: 251–267.
80. Brauer F. Compartmental models in epidemiology. In: Brauer F, Van den Driessche P and Wu J (eds) *Mathematical Epidemiology*. Berlin; Heidelberg: Springer, 2008, pp. 19–79.
81. Beauquier D. On probabilistic timed automata. *Theor Comput Sci* 2003; 292: 65–84.
82. Kale LV and Krishnan S. CHARM++: a portable concurrent object oriented system based on C++. In: *Proceedings of the 8th annual conference on Object-oriented programming systems, languages, and applications*, Washington, DC, 26 September–1 October 1993, pp. 91–108. New York: ACM.
83. Zheng Y, Kamil A, Driscoll MB, et al. UPC++: a PGAS extension for C++. In: *Proceedings of the IEEE 28th international parallel and distributed processing symposium*, Phoenix, AZ, 19–23 May 2014, pp. 1105–1114. New York: IEEE.
84. Slaughter E, Lee W, Treichler S, et al. Regent: a high-productivity programming language for HPC with logical regions. In: *Proceedings of the international conference for high performance computing, networking, storage and analysis*, Austin, TX, 15–20 November 2015, pp. 1–12. New York: IEEE.
85. Karypis G, Schloegel K and Kumar V. Parmetis: parallel graph partitioning and sparse matrix ordering library (version 2), 2003, <https://www3.cs.stonybrook.edu/~algorithm/Implement/ParMetis/distrib/manual.pdf>
86. Willebeek-LeMair MH and Reeves AP. Strategies for dynamic load balancing on highly parallel computers. *IEEE T Parall Distr* 1993; 4: 979–993.
87. Dastani M. 2APL: a practical agent programming language. *Auton Agent Multi-Ag* 2008; 16: 214–248.

88. Savarimuthu BTR and Cranefield S. Norm creation, spreading and emergence: a survey of simulation models of norms in multi-agent systems. *Multiagent Grid Syst* 2011; 7: 21–54.
89. Bollyky TJ, Hulland EN, Barber RM, et al. Pandemic preparedness and COVID-19: an exploratory analysis of infection and fatality rates, and contextual factors associated with preparedness in 177 countries, from Jan 1, 2020, to Sept 30, 2021. *Lancet* 2022; 399: 1489–1512.
90. Alechina N, Dastani M and Logan B. Programming norm-aware agents. In: *Proceedings of the 11th international conference on autonomous agents and multiagent systems*, Valencia, 4–8 June 2012, vol. 2, pp. 1057–1064. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS).
91. Dastani M, Grossi D, Meyer JJC, et al. Normative multi-agent programs and their logics. In: *Proceedings of the 1st international workshop on knowledge representation for agents and multi-agent systems*, Sydney, NSW, Australia, 17 September 2008, pp. 16–31. Berlin; Heidelberg: Springer.
92. De Mooij J, Dell'Anna D, Bhattacharya P, et al. Large-scale data-driven agent-based simulation of Covid-19 (v2.0.0), 2021, <https://doi.org/10.5281/zenodo.5761104>
93. Pepe E, Bajardi P, Gauvin L, et al. COVID-19 outbreak response, a dataset to assess mobility changes in Italy following national lockdown. *Sci Data* 2020; 7: 230.
94. Iio K, Guo X, Kong X, et al. COVID-19 and social distancing: disparities in mobility adaptation between income groups. *Transp Res Interdiscip* 2021; 10: 100333.
95. Hernando A, Mateo D, Bayer J, et al. Radius of Gyration as predictor of COVID-19 deaths trend with three-weeks offset. *medRxiv*. Epub ahead of print 2 February 2021. DOI: 10.1101/2021.01.30.21250708.
96. Northam RS. and Virginia governor Ralph S. Northam—executive actions, <https://www.governor.virginia.gov/executive-actions/> (accessed 7 October 2020).
97. Johns Hopkins Coronavirus Resource Center. Impact of opening and closing decisions in Virginia, new cases—Johns Hopkins, <https://coronavirus.jhu.edu/data/state-timeline/new-confirmed-cases/virginia/> (accessed 7 October 2020).
98. Dell'Anna D, Dastani M and Dalpiaz F. Runtime revision of sanctions in normative multi-agent systems. *Auton Agent Multi-Ag* 2020; 34: 43.
99. Kruschke JK. *Doing Bayesian data analysis: a tutorial with R and BUGS*. Burlington, MA: Academic Press, 2011.
100. Dey M, Frazis H, Loewenstein MA, et al. Ability to work from home. *Mon Labor Rev* 2020. Epub ahead of print 2020. DOI: 10.21916/mlr.2020.14.
101. Nelder JA and Mead R. A simplex method for function minimization. *Comput J* 1965; 7: 308–313.
102. Health Editorial Team. Why do some people refuse to wear a face mask in public? 2020, <https://www.health.com/condition/infectious-diseases/coronavirus/face-mask-refuse-to-wear-one-but-why> (accessed 20 September 2020).
103. Wong B. The psychology behind why some people refuse to wear face masks, 2020, https://www.huffpost.com/entry/psychology-why-people-refuse-wear-face-masks_1_5efb723cc5b6ca970915bc53 (accessed 20 September 2020).
104. Painter MO and Qiu T. Political beliefs affect compliance with Covid-19 social distancing orders. *Covid Econ* 2020; 4: 103–123.
105. Gollwitzer A, Martel C, Brady WJ, et al. Partisan differences in physical distancing are linked to health outcomes during the COVID-19 pandemic. *Nat Hum Behav* 2020; 4: 1186–1197.
106. Hamilton LC and Safford TG. Conservative media consumers less likely to wear masks and less worried about COVID-19, 2020, <https://scholars.unh.edu/cgi/viewcontent.cgi?article=1416&context=carsey>
107. Edwards-Levy A. Wearing a mask is more popular — and a little less partisan — than you might expect, 2020, https://www.huffpost.com/entry/face-mask-poll-support-partisan_n_5f1cc69ac5b69fd4730d52be (accessed 20 September 2020).
108. Apolloni A, Channakeshava K, Durbeck L, et al. A study of information diffusion over a realistic social network model. In: *Proceedings of the 2009 international conference on computational science and engineering*, Vancouver, BC, Canada, 29–31 August 2009.
109. Castillo C, Mendoza M and Poblete B. Information credibility on Twitter. In: *Proceedings of the 20th international conference on world wide web (WWW)*, Hyderabad, India, 28 March–1 April 2011.
110. Zhang H, Vorobeychik Y, Letchford J, et al. Data-driven agent-based modeling, with application to rooftop solar adoption. *Auton Agent Multi-Ag* 2016; 30: 1023–1049.
111. Thorve S, Hu Z, Lakkaraju K, et al. An active learning method for the comparison of agent-based models. In: *Proceedings of the 19th international conference on autonomous agents and multi-agent systems (AAMAS)*, Auckland, New Zealand, 9–13 May 2020.
112. Budak C, Agrawal D and El Abbadi A. Where the blogs tip: connectors, mavens, salesmen and translators of the blogosphere. In: *Proceedings of the 1st workshop on social media analytics (SOMA)*, Washington, DC, 25–28 July 2010.
113. Fagyal Z, Swarup S, Escobar AM, et al. Centers and peripheries: network roles in language change. *Lingua* 2010; 120: 2061–2079.
114. Swarup S, Apolloni A and Fagyal Z. A model of norm emergence and innovation in language change. In: *Proceedings of the 10th international conference on autonomous agents and multiagent systems (AAMAS)*, Taipei, Taiwan, 2–6 May 2011.
115. Kramer ADI, Guillory JE and Hancock JT. Experimental evidence of massive-scale emotional contagion through social networks. *Proc Natl Acad Sci U S A* 2014; 111: 8788–8790.
116. Blansky D, Kavanaugh C, Boothroyd C, et al. Spread of academic success in a high school social network. *PLoS One* 2013; 8: e55944.
117. Cabrales A, Gottardi P and Vega-Redondo F. Risk sharing and contagion in networks. *Rev Financ Stud* 2017; 30: 3086–3127.
118. Bauer M, Cahlíková J, Chytilová J, et al. Social contagion of ethnic hostility. *Proc Natl Acad Sci U S A* 2018; 115: 4881–4886.

Author biographies

Jan de Mooij holds a master's degree in the field of Artificial Intelligence and is currently pursuing his Ph.D. at the Department of Information and Computing Sciences of Utrecht University.

Parantapa Bhattacharya is a research scientist in the Network Systems Science and Advanced Computing division. He obtained his Ph.D. at the Indian Institute of Technology Kharagpur in 2017.

Davide Dell'Anna is a postdoctoral researcher in the Department of Information and Computing Sciences of Utrecht University and a member of the Hybrid Intelligence Centre. He obtained his Ph.D. at Utrecht University in 2021.

Mehdi Dastani is Professor and chair of the Intelligent Systems group of the department of Information and Computing Sciences at Utrecht University and program leader of the master program Artificial Intelligence.

Brian Logan is Professor of Computing Science at the University of Aberdeen and Associate Professor at Utrecht University. He also holds an honorary position (Special Professor), at the University of Nottingham.

Samarth Swarup is a research associate professor in the Network Systems Science and Advanced Computing division. He earned his Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign.

