EpiHiper - A High Performance Computational Modeling Framework to Support Epidemic Science

Jiangzhuo Chen[1,4], Stefan Hoops[1,4], Henning S.Mortveit[1,2,4*], Bryan L. Lewis[1], Dustin Machi[1], Parantapa Bhattacharya[1], Srini Venkatramanan[1], Mandy L. Wilson[1], Chris L. Barrett[1,3] and Madhav V. Marathe[1,3]

[1*]Network System Science and Advanced Computing Division, Biocomplexity Institute, University of Virginia, United States.
[2]Department of Engineering Systems and Environment, University of Virginia, United States.
[3]Department of Computer Science, University of Virginia, United States.
[4]Joint first authors.

*Corresponding author(s). E-mail(s): henning.mortveit@virginia.edu;
Contributing authors: jc3qw@virginia.edu; sh9cq@virginia.edu; bl4zc@virginia.edu; dmachi@virginia.edu; pb5gj@virginia.edu; sv8nv@virginia.edu; alw4ey@virginia.edu; clb5xe@virginia.edu; mvm7hz@virginia.edu;

Abstract

This paper describes EpiHiper, a state-of-the art high performance computational modeling framework supporting epidemic science. The EpiHiper modeling framework permits custom disease models and can simulate epidemics over dynamic, large-scale networks, while supporting modulation of the epidemic evolution through a set of user-programmable interventions. The nodes and edges of the social-contact network have customizable sets of static and dynamic attributes which allow the user to specify intervention target sets at a very fine-grained level; these also permit the network to be updated in response to non-pharmaceutical interventions such as school closures. The execution of interventions is governed by trigger conditions, Boolean expressions formed using any of EpiHiper's primitives and sets. Rich expressiveness, extensibility and high performance computing performance were central design goals to ensure that the framework could effectively target realistic scenarios at the scale and detail required to support state and federal public health policymakers in their efforts to plan and respond in the event of epidemics. The modeling framework is currently being used to support the CDC scenario modeling hub for COVID-19 response. EpiHiper was a part of a hybrid high-performance cloud system that was nominated as a finalist for the 2021 ACM Gordon Bell Special Prize for HPC-based COVID-19 Research.

Keywords: Computational epidemiology, high performance computing, social-contact networks, Agent-based models, Programmable pharmaceutical and non-pharmaceutical interventions, public policies, scenario modeling

1 Introduction

Despite significant progress in medical and public health sciences, epidemics caused by infectious diseases continue to have a global impact. The ongoing COVID-19 pandemic serves as a grim reminder of the significant social economic and health impacts of pandemics. Society, businesses, and health systems worldwide are struggling in their response to contain the COVID-19 pandemic. Computational and mathematical models have proven useful in planning and responding to epidemics in the past. Their use has steadily increased in the last two decades as policy makers and epidemiologists seek to study a range of questions, including, studying what-if scenarios, forecasting, assessing diverse intervention strategies and supporting logistics. COVID-19 pandemics saw extensive development and use of computational models to support policy makers and epidemiologists. The models developed range from descriptive, for example, static estimates of correlations within large databases, to generative, for example, computing the spread of different kinds of contagions via person-to-person interactions through a large population—these include the spread of a disease, as well as (mis)information and fear about the disease.

Networked Epidemiology. Compartmental mass action models have been used to study epidemic dynamics and complex interventions for over a century. They have been the cornerstone of epidemic science and have been used due to their simplicity, underlying theory and computational tractability. An alternative approach to represent epidemic processes and interventions is to use networks to explicitly represent the underlying social contact network between individual agents. Such a representation allows us to capture the underlying heterogeneity of the contact structure, and also allows us to formally and succinctly capture individual behaviors and interventions. Network-based models have become increasingly popular over the last fifteen years as policy makers seek to develop targeted policies and response strategies. Nevertheless such models are computationally intensive and thus naturally motivate the use of high performance computing for implementing these models.

EpiHiper. In this Resource, we present EpiHiper which is a high performance computational modeling framework supporting epidemic science and that addresses many of the challenges listed in [1]. This framework builds on our earlier work, first published in [2] and subsequently [3–8]. The design of EpiHiper integrates four major components as illustrated in Figure 1: (*i*) a labeled, time-varying social *contact network* over which contagions spread, (*ii*) fully *customizable disease models* capturing disease transmission between hosts as well as within-host disease progression; (*iii*) *user-programmable interventions* covering both pharmaceutical and non-pharmaceutical ones; and (*iv*) the discrete time, parallel simulator architected to take full advantage of modern high performance computing (HPC) hardware. This modeling framework has been used extensively throughout the ongoing COVID-19 pandemic by directly supporting planning and response efforts by our group to support state, local, and federal authorities.

The use of EpiHiper has been demonstrated during the last year in our work with the COVID-19 Scenario Modeling Hub[1] where it has been used for six rounds, where each round presented several challenges. The modeled scenarios range from limited vaccine supply [9], reduced non-pharmaceutical interventions (NPIs), emerging new variants in early 2021, vaccine hesitancy, waning immunity, and childhood vaccination in late 2021 [10]. Some of this work formed the basis for our submission to the 2021 ACM Gordon Bell Prize – the paper was selected as a finalist in the category of *High Performance Computing-Based COVID-19 Research*. Recently EpiHiper was used to support the White House US-UK challenge to develop privacy enhancing technologies [11, 12].

Do we need yet another modeling framework for computational epidemics? Our group has been developing data-driven, computational modeling frameworks in support of networked epidemiology for close to two decades [2]. Subsequent efforts by us include [3–8], that of others [13–15], and recent work related to COVID-19 planning and response including [16–26]. In

[1] https://covid19scenariomodelinghub.org/

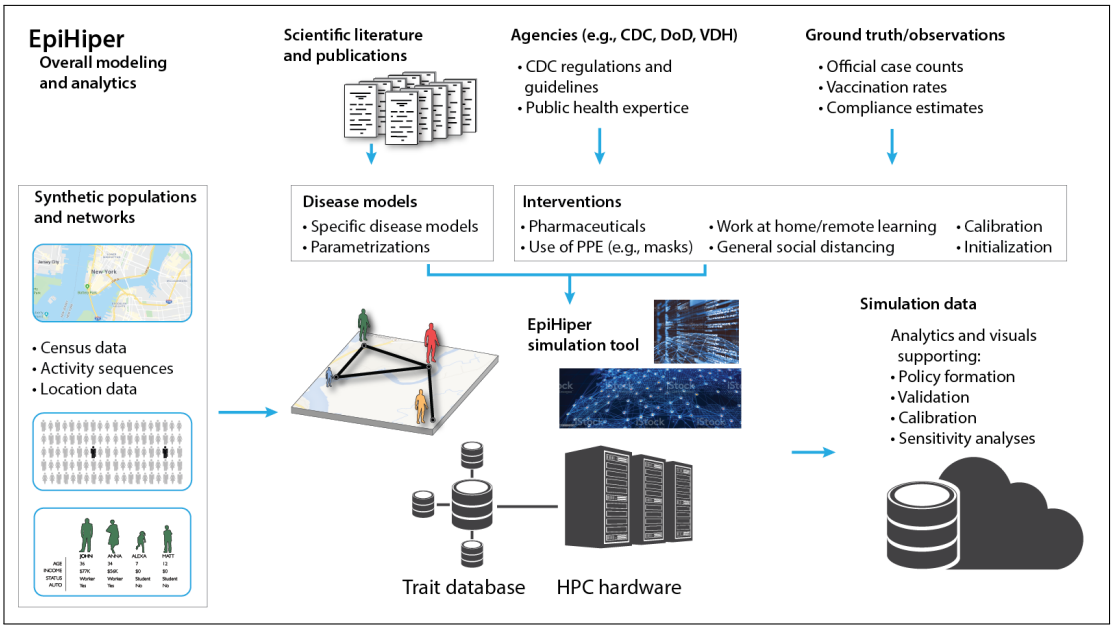


Fig. 1 The EpiHiper framework integrates disease models with parameters from peer-reviewed literature, highly detailed interventions matching CDC and public health guidelines, and detailed populations and contact networks to generate simulation outcomes and analytics for virtually any scenario involving COVID and influenza-like diseases.

this body of work, smaller regions have been analyzed using detailed models, and larger regions using aggregate models. With the exception of the work reported in [21], we are not aware of any epidemic models for COVID-19 that can handle complex interventions, disease progression models and national-scale networks. The work in [21] is largely focused on the United Kingdom (UK), although basic results were provided for the US as well.

Despite there being many other frameworks for computational epidemiology, there are also many shortcomings which the EpiHiper project directly addressed through model design and architecture. Some of the key aspects are: *scalability:* very few existing systems scale to the network sizes considered which are of the order 100 million nodes. Systems that claim scalability (but also other systems) often have limited *capabilities and expressiveness* in terms of disease modeling and interventions. Typically, disease models and/or interventions are hard-coded, effectively making it rather challenging for others to add or extend any of these elements as this will require detailed knowledge of the model and its implementation. While such designs may have been the result of having to support policy decisions for COVID-19 in near real-time, it renders frameworks inflexible, frequently requiring extensions to the code base whenever a new scenario must be addressed. A framework greatly benefits by having external specifications of both disease models and their interventions. Not only does this approach support more transparent peer-review of models and interventions, it also avoids the inherent risk of introducing model errors as well as coding errors through frequent code changes. We would argue that having these model components specified externally will typically lead to a more flexible framework that can respond more rapidly and more reliably to the needs of public policy makers. One hope with this paper is that it will set the tone for researchers to precisely describe their modeling frameworks in sufficient details to ensure a basic level of reproducibility[2] along with the advantages of this in support of validation (e.g., [28, 29].)

[2]By reproducibility we mean the ability for other researchers to re-implement the model using knowledge obtained from the original paper, possibly in a different simulation tool or programming language than the one reported, and through simulation verify the results reported in the study [27].

2 Results

Summary of methods. In the following we describe the components of the EpiHiper framework as illustrated in Figure 1.

Social contact network. The EpiHiper modeling framework simulates epidemic spread over a *social contact network* of agents (nodes) where each node has labels (or attributes) encoding properties of that agent. The labels generally include social and demographic factors, but may include behavioral attributes as well as disease attributes (e.g., vaccination history). In the network, an edge captures the possibility of transmission between its end-points; edges are also labeled (e.g., contact duration, activity types of the two nodes.) While the synthesis of such contact networks is out of scope of this paper, a brief description is provided in the supplementary material.

Disease model. The disease model in the EpiHiper framework is fully programmable, and is split into *disease transmission* and within-host *disease progression*. The former captures the transmission aspect of the disease, while the latter captures all other aspects of the disease evolution over a set $\mathcal{X} = \{X_1, X_2, \ldots, X_m\}$ of custom *health states*. As the COVID-19 pandemic progressed, the within-host models became increasingly more complex: asymptomatic states were important at early stages, then followed the need to represent vaccine uptake which would impact for example the ability to transmit and the ability to become infected. Yet more states were necessary to capture immune waning and to account for multiple strains/variants, and their cross-correlations. The within-host model is specified externally as in input file (JSON), and is represented internally as a probabilistic timed transition system (PTTS). This allows the user to specify all transitions among health states, the probabilistic dwell time in health states, as well as weights for transitions in case there is more than one possibility.

For the transmission process, EpiHiper determines all *propensities* associated arising from contacts between infectious nodes/people and susceptible nodes/people. This formalized through the notion of *contact configurations* $T_{i,j,k} = T(X_i, X_j, X_k)$ where a susceptible person P in health state X_i may transition to an exposed state X_j when in contact with an infectious person P' in state X_k. The *propensity* of such a contact is determined through a range of factors including the disease's *state susceptibility* and *state infectivity* associated to the health states X_k and X_i, and the *infectivity scaling factor* of P' and the *susceptibility scaling factor* of P. The latter two quantities are per-person, dynamic variables and are often modulated in response to pharmaceutical as well as non-pharmaceutical interventions as described below. The duration of contact is also factored into the propensity. To determine if an infection takes place for a susceptible person P, and also to whom we attribute the infection, we apply the Direct Gillespie Method to the collection propensities involving P and infectious people P' encountered within the iteration.

Interventions are central to the EpiHiper framework and represent the mechanisms that allow one to represent complex pharmaceutical as well as non-pharmaceutical interventions. Examples of interventions include city-scale lock-downs and closures of various facilities, vaccinations which may be temporally and spatially constrained based on production schedules, test-trace-isolate, and behavioral adaptation by individual citizens, including mask wearing, vaccine hesitancy, and social-distancing. Clearly, the diversity of possible interventions is vast which underscores the importance of having a generic and flexible approach for their representation [30]. Just as for the disease model, EpiHiper allows to user to specify highly custom and complex interventions as a file (JSON) which is provided as as part of input configuration. We believe that this is one of the unique features of our framework. As the pandemic progressed, this is one of main factors that allowed us to keep up with the continuous need for implementing increasingly complex interventions to support public policy.

Informally speaking, in EpiHiper an intervention consist of (i) a *trigger*, (ii) a *target*, and (iii) a set of operations organized into procedural control structures similar to those one would expect to find in a programming language. The trigger determines the time step(s) for which an intervention is executed, and the target is the set of nodes or edges upon which the operations of the intervention are applied. A person receiving a vaccine, for example, can be captured by setting that person's susceptibility scaling factor to a

suitable value depending on their attributes (e.g., age, NAICS code, or health state). The intervention language permits an operation to take place with a *delay*, can be assigned a *priority* to break ties with other operations, and may additionally be given a condition that must evaluate to true at the time of execution lest it be dismissed.

The EpiHiper *discrete time parallel simulator.* In order to support the required scale with networks with 100 million nodes or more, EpiHiper was designed and implemented to take full advantage of modern HPC hardware. It can distribute an epidemic simulation across as many computational nodes as is made available, and within each node it can fully use all available cores. A significant effort has gone into *calibration* and there are several supported approaches. This includes through parameters such as the intrinsic transmissibility and any disease parameter and intervention parameter such as compliance and efficacy. In [31] EpiHiper was calibrated using a Gaussian, process-based, Bayesian framework to fit time series data based on confirmed cases, and in [32] EpiHiper was calibrated to fit an estimated effective reproduction number using most recent surveillance data.

EpiHiper Use Cases. EpiHiper was developed partly in response to the ongoing COVID-19 pandemic and for the past two years, EpiHiper has been used extensively to help inform analysts at the US Department of Defense, the Virginia Department of Health (VDH), and Centers of Disease control. Recently, we have begun exploring its use to support the European Centers for Disease control.[3]

In late 2020, EpiHiper was used to study the risks of workplace outbreaks after office reopening, the impact on neighboring communities, and to evaluate testing, tracing, and isolation strategies. The results were briefed to the Defense Threat Reduction Agency. In early 2021, through our collaboration with the VDH, EpiHiper was used to evaluate the effectiveness of contact tracing and quarantine measures in Virginia. Key findings were that contact tracing helped to reduce cases, hospitalizations, and deaths, and to reach the targeted level of case numbers much faster [32].

Finally, over the last year, we have worked closely with the COVID-19 Scenario Modeling Hub[4]. EpiHiper has been used for 6 rounds now; each round presenting its own sets of challenges. A recent paper [33] summarizes the ensemble results from round 7. A paper based on Round 9 is in submission. The modeled scenarios range from limited vaccine supply [9], reduced non-pharmaceutical interventions (NPIs), emerging new variants in early 2021, vaccine hesitancy, waning immunity, and childhood vaccination in late 2021 [10]. Some of this work formed the basis for our submission to the 2021 ACM Gordon Bell Prize – the paper was selected as a finalist in the category of *High Performance Computing-Based COVID-19 Research* and will be published soon [10]. In each round, four scenarios are defined across two axes, chosen to study emerging issues and topics. Each of these axes usually specifies two different settings of the parameters involved, including both pessimistic and optimistic values. Such issues range from vaccine availability and hesitancy, NPIs, new variants, waning immunity, immune escape, childhood vaccination, and boosters. In round 12, we focus on the impact of the Omicron variant, with one axis on immune escape (80% and 50%), and another on the reduction in severity of Omicron infections relative to Delta (70% and 30%).

1. Capability in disease modeling. We have implemented a PTTS to represent complex infection and immune states of a single individual pertaining to COVID-19 outbreak. This includes asymptomatic state and detailed disease outcome states (hospitalization, ventilation, and death). We have implemented a tool that allows us to expand the base states by age groups, and assign age-dependent dwell time distributions to the states and age-dependent transition probabilities between the states. Along the SMH rounds, we have augmented the base disease model with various vaccinated states: one-dose Pfizer/Moderna, two-dose Pfizer/Moderna, Johnson & Johnson, boosters, as well as a partially susceptible state for nodes with waning of immunity obtained by natural infection or vaccination. We have implemented age-dependent protections against infection, transmission, and severe disease for nodes with immunity (with vaccination, infection, or

[3] Information about our ongoing efforts to support these agencies can be found at: https://nssac.github.io/covid-19/index.

[4] https://covid19scenariomodelinghub.org/

waning). All these implementations are accomplished by specifying the health states and transitions between them without changing EpiHiper code. In round 12, the changes to our disease model are minimal since we have already implemented various vaccinated states, waning immunity, the Omicron variant, and infection of Delta-immune people by Omicron by round 11. We only need to update the parameters to the most recent values.

2. Capability in initialization. As the COVID-19 pandemic comes close to its third year, we choose to start the simulations from December 2021. EpiHiper allows us to sample a given number of people from age-stratified distributions at the county level, and set them to *recovered* states for prior infections, *vaccinated* states for prior vaccinations, *partially susceptible* states for people with waned immunity, and *infected* states for active infections in December 2021. This capability allows us to start the simulation from a given snapshot with the set states for all nodes and the network. Related is a feature of EpiHiper to save a snapshot of nodes and network states during the simulation and to allow resumption from that snapshot.

3. Capability in intervention modeling. In round 12, we have implemented both non-pharmaceutical interventions (NPIs) and pharmaceutical interventions (PIs). The NPIs include (i) individual-level preventive behavior (e.g. mask wearing) with 25% compliance; (ii) individual-level social distancing (reducing daily non-essential activities) with 15% compliance; (iii) universal school closure due to winter break; and (iv) household-level voluntary home isolation of symptomatic infections with 75% compliance. For NPI (i), we change the infectivity parameter of compliant nodes. For NPI (ii), we change the activities of compliant nodes and remove their contacts with other people during the reduced activities. For NPI (iii), we temporarily disable all contacts in school. For NPI (iv), we temporarily disable contacts of isolated nodes with people outside their households. The PIs include various vaccinations, which mainly move nodes to appropriate vaccinated states with reduced susceptibility, infectivity, and reduced probability to transition to severe outcomes. The interventions are triggered by time (e.g. school closure) or the global/local state, targeting a defined set of nodes or network edges, and can have a compliance, a delay in action, and the efficacies of the changes on node attributes or edge weights.

2.1 Scenario modeling hub (SMH) round 12

Here we describe how EpiHiper was used to support round 12 of the CDC COVID-19 Scenario Modeling Hub. More details of the round can be found at https://covid19scenariomodelinghub.org/archive.html. The main purpose of this round, conducted at the beginning of 2022, was to evaluate the impact of the Omicron wave, on the disease outcomes in the first three months of 2022. The design has two axes: (*i*) the immune escape axis has a higher level which assumes that nodes with prior immunity (natural or vaccinal) of pre-Omicron variants are 80% more likely to be infected by Omicron than by any pre-Omicron variant, and a lower level which assumes 50% for this difference; (*ii*) the severity axis has a pessimistic level which assumes that the risk of severe outcomes, including hospitalization and death, of a node infected by Omicron is 70% of that if the node had been infected by Delta, and an optimistic level which assumes 30% for the same ratio. Figure 2 shows the 2x2 design with two axes and four scenarios in Round 12.

Round 12	**Higher immune escape:** 80% of previously immune are susceptible to infection	**Lower immune escape:** 50% of previously immune are susceptible to infection
Lower severity: 70% reduction in severity of Omicron infection, relative to Delta (all-age risk of hospitalization and death times 0.3) among all immune classes.	Scenario A Optimistic severity, High immune escape	Scenario B Optimistic severity, Low immune escape
Higher severity: 30% reduction in severity of Omicron infection, relative to Delta (all-age risk of hospitalization and death times 0.7) among all immune classes.	Scenario C Pessimistic severity, High immune escape	Scenario D Pessimistic severity, Low immune escape

Fig. 2 Design of scenarios in CDC COVID-19 Scenario Modeling Hub Round 12 [34].

Our disease model includes susceptible, vaccinated, exposed, symptomatic, asymptomatic, hospitalized, recovered, and decreased states,

with age-stratification. It models two COVID-19 strains: one represents all pre-Omicron variants, the other corresponds to Omicron. We model immunity waning explicitly in the disease model with a partially susceptible state. A node in either recovered state (natural immunity) or vaccinated state (vaccinal immunity) is moved to the partially susceptible state after a random dwell time, sampled from an exponential distribution with mean of 6 months. Protection on nodes in the partially susceptible state against infection, comparing with nodes in the fully susceptible state, is 60% (or 40%) for those under 65 years old (or 65+ years old); protection against severe disease is 90% (or 80%) for under 65 (or 65+). Figure 3 shows a simplified illustration of our disease model. Note that it is expanded for age stratification as well as for two virus strained in the actually implemented model.

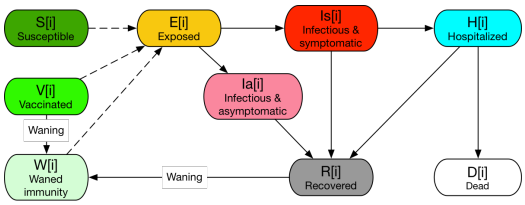


Fig. 3 Disease model used for EpiHiper simulations in CDC COVID-19 Scenario Modeling Hub Round 12. Note that the figure shows the states of a node of age group i; all age groups have the same states and transitions but the parameters, including susceptibility, infectivity, dwell time, and transition probability may be different between different age groups.

We modeled the same nonpharmaceutical interventions (NPIs) in all scenarios and across all states: (i) A fraction (15%) of the population chooses to reduce non-essential (shopping, religion, and other) activities. (ii) All K-12 schools are closed from late-December 2021 to the beginning of 2022; and face masks are used in school while schools are open. (iii) A fraction (75%) of symptomatic people choose to self-isolate themselves at home. Note that the compliance rates to (i) and (ii) are assumed to remain the same during the projection period, but EpiHiper can be configured to model time varying NPIs if supporting data is available.

The simulations are initialized by data-driven assignment of nodes' initial health states. Based on county level data of prior confirmed cases, age specific case ascertainment rate, state level data of prior vaccinations, and waning of immunity (natural and vaccinal), we initialize each individual to one of naively susceptible, vaccinated, partially susceptible (with waned immunity), and non-susceptible (currently or recently infected) states, depending on whether and when the individual is/was infected and/or vaccinated.

For each state, we calibrate the transmissibility parameter in our disease model targeting the state level estimated effective reproduction number from the most recent confirmed cases. We run EpiHiper simulations for each state separately, and combine the outputs to get results of the whole US. The simulations produce daily infections, hospitalizations, and deaths; and each simulation runs for multiple replicates. We aggregate daily data to get weekly data and compute distribution of projection for each target from the multiple replicates.

Our projections show that at national level confirmed cases peak in late January (best scenario) to early February (worst scenario), followed by a quick drop to a level that will last for a while. Among the four scenarios, even the best one (optSev_lowIE: optimistic severity, low immune escape) might reach 3000 cases per 100K in the peak week, possibly stressing the testing capacity. Similar trends are projected for hospitalizations, with a less dramatic gap between current and previous peaks comparing with cases, due to less severe outcomes of Omicron infections and protection provided by vaccines. These are shown in Figure 4. We also project significant heterogeneities between states regarding peak size and timing in cases and hospitalizations.

Performance assessments and comparisons. There are several other frameworks for computational epidemiology, and we have summarized key features of some that we consider most related to EpiHiper in Table 1. Additional details are provided in the supplementary material for these and other systems.

In Table 1, a model or intervention entry marked as 'fixed' means that this feature is hard-coded. While a user of the software may add additional models/interventions, this would require adequate coding skills in the programming language that was used, and sufficient insight into

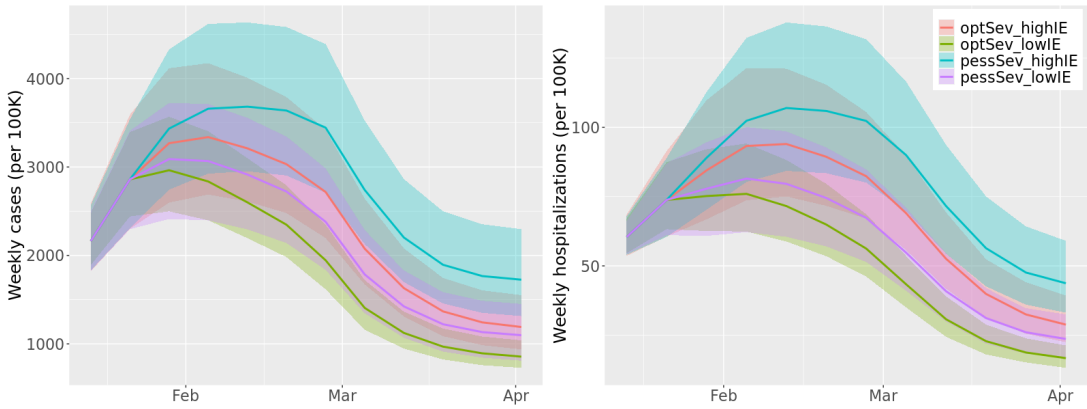


Fig. 4 CDC COVID-19 Scenario Modeling Hub Round 12 projection by EpiHiper model for cases and hospitalizations under four hypothetical scenarios in the first three months of 2022. The curves show median of projections and the ribbons show 95% projection interval.

Framework	Epidemic model	Interventions	Networks	Language	Open	License	Scaling/ Parallelization
Covasim [18]	Single, fixed, parameterized	Several, hard-coded interventions included	SynthPop + random graphs	Python	Yes	CC BY-SA 4.0	Single node/single core
Covid-Sim [21]	Multiple, fixed	Multiple, fixed	Layered networks	C++	Yes	GPLv3	Single node/-multiple core
FRED [15][5]	Multiple, fixed, un-coupled	Multiple, fixed	Location-based with uniform mixing per location	C++	Yes	Unclear (open version)	Single node/-multiple cores
EpiHiper	User-programmable, externally specified	User-programmable, externally specified	General, directed, labeled networks	C++	No	Apache ver. 2.0	Multiple node and multiple core

Table 1 An overview of key features for related epidemic simulation models and frameworks.

the implementation. A network model is 'layered' if it is constructed using layers (e.g., household layer, school layer, work layer) where nodes are joined within layers to satisfy, e.g., data from a mixing matrix (e.g., [35–37]); it may be done according to demographics (e.g., age bins). While SynthPop [38], which uses the layered network synthesis approach, offers networks that can be explicitly examined, many epidemiological simulation models construct such networks on the fly, thus making it hard to assess the networks' properties or adequacy. An epidemic model is *location based* if it tracks peoples' visits to locations, constructs the contacts at each location, and runs the epidemic process over the induced sub-networks (e.g., FRED [15], EpiSimdemics [4]).

Here we focus on **Covasim** [18] and **Covid-Sim** [21]. Since **FRED** [15] is location-based, and also since the open version of FRED is rather dated, we omit FRED from comparisons. All these, and other system such as the **Google and Open ABM Project**, **COVID-19 ABM** [39, 40] are described in more detail in the supplementary material.

By software design, the current version of Covasim only supports epidemic runs on a single core. However, the framework can run multiple, independent model instances distributed across the cores of a single computing node (using the Python sciris library). We note that Covasim supports dynamic rescaling where an agent in the simulation represents N (the scaling factor) individuals, and may thus permit additional scaling for scenarios where this is appropriate, see also [29]. Covid-Sim is a threaded application and may therefore use all the cores on a single computing node. Its software design thus provides inherently better computational scaling than that of Covasim. However, for both Covasim and Covid-Sim, being limited to a single computational node,there is currently no meaningful way of comparing their scaling properties (e.g., with respect network size) to those of EpiHiper. This becomes even more evident for the

complex classes of interventions typically needed when supporting the scenarios posed by policy makers. Such interventions frequently require a large set of static and dynamic attributes for each person in the simulation model. This imposes a large memory footprint, severely limiting single node architectures.

EpiHiper scaling. Of more practical interest than a standard scaling study, is an analysis of how the choice of disease model, interventions, and their complexities impact computation time along with their scaling of memory requirements. In the supplementary material (Section D) we have provided a detailed analysis of all these aspects for a computational experiment run on the synthetic population of Virginia (US). In summary, the eight computational experiments (labeled I through VIII) used the disease models as specified in Supplementary Material Table 4 where the intervention overview is described in Table 5, and the correspondence between experiments and sets of interventions are provided in Table 6. For reference, each experiment was conducted on compute nodes with dual CPUs having 20 cores each and 375 GB total memory for its specified collection of interventions using 15 replicates.

As shown in Figure 5, the impact of intervention complexity on running time is very clear. In the Supplementary Material Section D, we have also provided Figure 20 which gives a scaled version analogous to Figure 5 but in this case taken across all US states using a scenario (different than the above) that used to support the Scenario Modeling Hub.

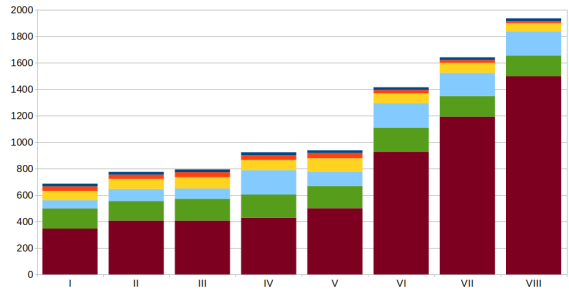


Fig. 5 Impact of intervention complexity on computation time for experiments I–VII (see text) factored by the main simulation tasks of intervention, transmission, update, synchronization, output, and initialization. The unit is seconds.

In Figure 6 we have provided a scaling analysis using the synthetic populations and networks of the U.S. states for one of our vaccine studies [9]. As we see, the running time scales more or less

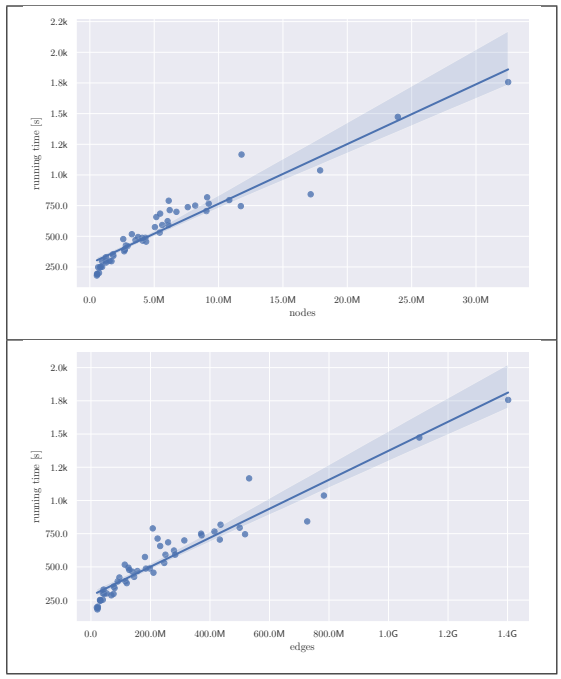


Fig. 6 EpiHiper running time as a function of network size for the U.S. states. Top (Bottom): running time as a function of the number of nodes (edges).

linearly with node- and edge counts.

Performance scaling and computational resources. We performed code profiling to determine whether the chosen partitioning strategy is feasible for different problem sizes. Figure 20 indicates that increasing the number of cores by a factor of 60 (from 2 for small states (AK, DC, MT, ND, SD, VT, WY) to 120 for CA) only increases the synchronization by a factor of 8 (3.23%–25.2%), indicating that inter-processes communication scales approximately with $\sqrt{N}$ where N is the number of cores supporting the chosen partitioning strategy. The memory and run time requirements for different problem sizes scale linearly with the problem size. In fact we found that the memory requirements are changing only minimally with varying disease model and intervention complexity. The scaling of the run time though always linear is very much dependent on the complexity of both.

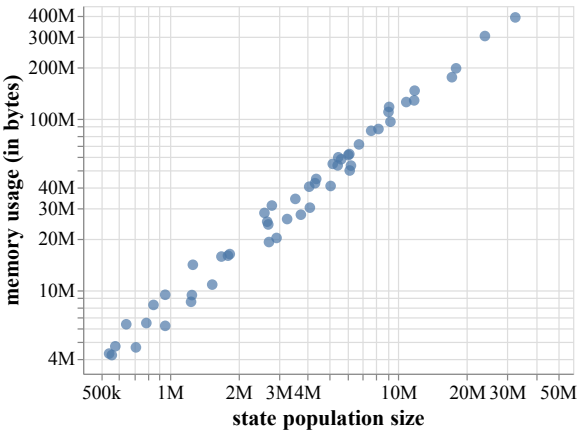


Fig. 7 Scaling of state size vs memory requirements for one of our vaccine studies [9].

3 Discussion

We believe a key factor in ensuring that the broader scientific community trusts epidemic models is to base the software on a *formal mathematical model* that is published in full. This serves two essential purposes. First, it gives the necessary separation of concern between the model and the code, thus helping avoid the situation where the code becomes the *de facto* formal model. The latter situation poses serious challenges for scientific reproducibility, peer review, as well as model validation and code verification. (Is the model wrong? Or are there code errors?) Second, it allows us to take a step towards producing reproducible epidemic modeling environments, see [41–44] for further discussion. Computer Scientists have advocated this approach for a long time as they have developed complex software systems.

Design principles. The design of the EpiHiper modeling framework and its software architecture adhered strictly to the above principle, and was guided by a careful consideration balancing (*i*) the expressive power required to flexibly capture most person-to-person transmissible diseases and the interventions needed to support actual public health policy formation, and (*ii*) computational complexity and running times of the software to ensure adequate scaling, and (*iii*) the time required to construct, test and validate the implementation. As a result, the intervention language used by EpiHiper (see the Methods section) does not offer all the functionality and constructs that one will find in programming languages like Python or C++, but have so far been able to fully support all requirements in our work with federal and state agencies. An additional design decision that we consider essential is that EpiHiper's disease models and interventions are specified *externally*. Thus user can provide their own models and provide them to EpiHiper as part of the input data. Not only does this design eliminate the need for recompiling any code as new disease models or interventions are introduced, but it also helps support scientific reproducibility as well as peer review. As can be seen in Table 1, this is not a typical design.

Scaling. EpiHiper was designed with performance as one of the key goals in a concerted effort seeking a precise and rich mathematical model as well as an implementation able to take full advantage of current software architectures. As illustrated in the Results section, EpiHiper can handle a very rich set of epidemic models along with advanced intervention capabilities capable of accommodating what is needed to inform public policy experts and their scenarios. Disease models and interventions are cleanly specified, and can easily be shared or reused in a manner that directly supports peer review.

As detailed in Section 2, we have not attempted to assess how large networks EpiHiper can handle or push computational boundaries. While that certainly has value, the major driver of computational complexity is generally governed by the collection of interventions that are applied. EpiHiper has routinely been used for simulation over our California networks which have more than 30 million nodes and close to 700 million edges. While we may eventually benchmark EpiHiper for network scaling using some standard set of interventions, such results may not be of much practical value to policy formation. The linear scaling in terms of network size (node- and edge counts) of Figure 6 and computer memory (Supplementary Material Figure 7) leaves us confident that we can handle networks of the scale of the entire U.S. population and beyond.

Technology choices. The EpiHiper simulation model can run on any modern computer ranging from laptops to super-computers. Naturally, larger networks will require matching hardware.

The software design uses standard, open technologies, major examples being MPI [45] and OpenMP [46] for computing, Postgres for the person trait database, and Frictionless [47] and JSON for specification of input data. This ensure that it can be deployed on virtually anywhere.

Data dependencies. EpiHiper relies on fairly advanced input data: constructing realistic populations and networks for a region is generally a large effort in itself.

4 Methods

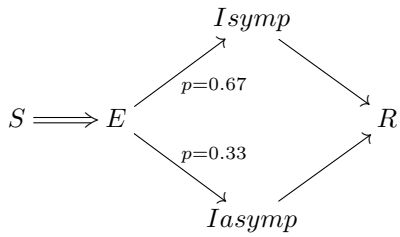
Here we describe the EpiHiper disease modeling capability, the structure of the programmable interventions, and the implementation of the epidemic simulator. A basic example is given alongside the description to help illustrate the concepts. We remark that the EpiHiper framework is "abstract", in the sense that it can capture general, intervention-adjusted, contagion processes propagating over a collection of entities connected by a network structure, not just people and COVID-like diseases.

The EpiHiper Disease model is fully programmable, and starts from a set $\mathcal{X} = \{X_1, X_2, \ldots, X_m\}$ of *health states*. The *disease progression*, which captures the disease evolution once a person becomes infected, is represented using a probabilistic timed transition system (PTTS) over $\mathcal{X}$. These are an extension of finite state machines with the additional features that state transitions are probabilistic and timed.

As an illustration we consider a hypothetical case of a classic influenza (or COVID-like) outbreak in Albemarle County, Virginia. Here we use the set

$$\mathcal{X} = \{S, E, Isymp, Iasymp, R\}$$

to encode the five *health states* susceptible (S), exposed (E), infectious and symptomatic ($Isymp$), infectious and asymptomatic ($Iasymp$), and recovered (R) with the combined transmission and progression diagram as follows:



Apart from the double arrow/edge $S \Longrightarrow E$ which specifies transmission, each edge corresponds to a transition in the disease progression model. We note that disease progression from health state E to $Isymp$ is twice as likely as progression from E to $Iasymp$ (or 0.67/0.33 to be precise). The dwell time distribution for both transitions out of the state E, which we denote by edges $(E, Isymp)$ and $(E, Iasymp)$, are

$$\{0 : 0.1;\ 1 : 0.2;\ 2 : 0.6;\ 3 : 0.1\}\ ,$$

meaning that, e.g., the probability of a dwell time of duration 2 (days) is 0.6. Similarly, the dwell time distributions for the transitions $(Isymp, R)$ and $(Iasymp, R)$ out of states $Isymp$ and $Iasymp$ are both

$$\{3 : 0.3;\ 4 : 0.4;\ 5 : 0.2;\ 6 : 0.1\}\ .$$

Note that a dwell time distribution is associated with an edge (health state transition) and that the unit of time is one iteration, which in this example equals one day.

To describe the *disease transmission model*, we refer to Figure 8, which shows a network where

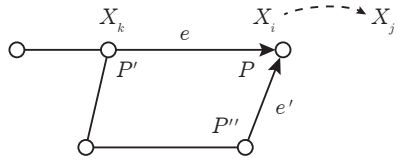


Fig. 8 An example network for EpiHiper with agents/nodes P, P' and P''. Here, the infectious person P' may infect the susceptible person P, who as a result may transition from health state X to an exposed state X'.

a susceptible person P is in contact with infectious persons P' and P''. Focusing on the the

pair (P', P), we combine the *state susceptibility* and *state infectivity* of their respective health states X_k and X_i with the *infectivity scaling factor* of P' and the *susceptibility scaling factor* of P to form the *propensity* associated with the *contact configuration* $T_{i,j,k} = T(X_i, X_j, X_k)$ for the potential transition of the health state of person P to X_j as:

$$\rho(P, P', T_{i,j,k}, e) = \left[T \cdot \tau\right] \times w_e \times \alpha_e \times \left[\beta_s(P) \cdot \sigma(X_i)\right] \times \left[\beta_i(P') \cdot \iota(X_k)\right] \times \omega(T_{i,j,k}) \quad (1)$$

Here, T is the duration of contact for the edge $e = (P', P, w, \alpha, T)$, w is an edge weight, and α is a Boolean value indicating whether or not the edge is active (e.g., not disabled because of an ongoing school closure).

In the example, there are two transmission configurations (1 susceptible state × 2 infectious states) are

$$T_s = T(S, E, Isymp) \text{ and } T_a = T(S, E, Iasymp)\ ,$$

both having the default weight of $\omega = 1.0$. Regarding infectivity, people in either of the states $Isymp$ and $Iasymp$ can transmit infections, with the asymptomatic reduction in infectivity being 60% and thus $\iota(Iasymp) = 0.40$ while the susceptibility and infectivity values of all other health states have the default value of 1.0

For each time step, and for each person P, the propensities ρ from Equation (1) are collected across all edges e and contact configurations T as the sequence $\rho_P = (\rho(P, P', T, e)_{P',T,e})$. To determine if P becomes infected is modeled using a Gillespie process [48, 49]; the person P' to whom one attributes P falling sick is also determined as part of this step. The full details are given in the supplementary material.

Disease model assumptions: it is assumed that (i) propensities for a person are independent across contact configurations, and (ii) that during any time step no person can change their health state. The first assumption is quite common and not unreasonable for the contact networks that are used. The second assumption can always be accommodated by reducing the size of the time step. Its real purpose is to ensure *order invariance* of contacts within a time step, thus providing the required guarantee for algorithm correctness.

Parameter	Description
P, P'	Persons/agents/nodes
X_i	Health state i
$\sigma(X_i)$	Susceptibility of health state X_i
$\iota(X_i)$	Infectivity of health state X_i
$\beta_\sigma(P)$	Susceptibility scaling factor for person P
$\beta_\iota(P)$	Infectivity scaling factor for person P
w_e	Weight of edge $e = (P, P')$
α_e	Flag indicating whether the edge e is active
$T(X_i, X_j, X_k)$	Contact configuration for a susceptible transition from X_i to X_j in the presence of state X_k
$\omega_{i,j,k}$	Transmission weight of contact configuration $T(X_i, X_j, X_k)$
τ	Transmissibility
$\rho(P, P', T_{i,j,k}, e)$	Contact propensity

Table 2 EpiHiper core model parameters.

EpiHiper interventions. An EpiHiper intervention consists of a *trigger condition* C, an *intervention target* T, and a collection of operations that are applied against the variables associated to the elements of the target, or against variables not attached to target entities (through the **once** construct). The trigger condition C is a Boolean expression formed using EpiHiper primitives (see Table 3) and sizes of sets; the intervention target is a set consisting of vertices or edges, and is formed using predicates over the same elements as for the triggers; the operations are organized into the control blocks specified in Listing 1.

Semantics of intervention blocks. The operations within the **once** block are executed whenever the trigger condition C holds, even if the target set is empty. It is used to set variables that are not attached elements of the intervention target (e.g., the number of available vaccines on a given day). All operations within the **foreach** block are applied to the matching variables of the target elements. Aspects such as compliance are handled through the **sampling** block: several sampling methods are supported where operations applied to the **sampled** and **nonsampled** sets. We note that recursive application of operation ensembles are supported in the **sampling** control structure.

Semantics of operations. The syntax of operations is provided in Listing 1. In an operation, a variable is assigned the value of an expression, the assignment being scheduled for execution using

Listing 1 The EpiHiper block structure for operations

```
operationEnsemble :=
  once
    <operationList>
  foreach
    <operationList>
  sampling <samplingSpecification>
    sampled
      <operationEnsemble>
    nonsampled
      <operationEnsemble>

samplingSpecification :=
(
  relativeSampling ( individual | \
    group ) <percentage> |
  absoluteSampling <integer>
)

operationList := <operation>+

operation := <variable>
   <operator> <expression> \
     delay(<integer>)       \
   [ priority(<integer>) ] \
   [ condition(<bool_expression>) ]

operator := ( = | *= | /= | += | -= )
```

a non-negative offset **delay** relative to the current time step. The assignment may optionally be assigned an integer **priority** (default value 0) and a **condition** (default value True) which is a Boolean expression that must hold at the scheduled execution time for the assignment to be carried out.

Operation execution. All operations enter a priority queue which is sorted first by scheduled execution time and second by priority. Within a time step, all operations scheduled then are processed in sorted order. Collections of operations of equal priority are processed in random order. Finally, an operation is executed if only if its condition is true at the time of processing.

A note on processing order of interventions. When designing interventions, one needs to pay careful attention to cases where order of operations may matter. It is the responsibility of the person constructing the (set of) interventions to ensure, through specification of priorities and conditions, that the order becomes as they intend.

Returning to the example, we showcase two interventions: school closures, and vaccination.

(a) *School closure.* Schools will be closed each day for which $\geq 10\%$ of the population are infectious (health states $Isymp$ or ($Iasymp$) at the beginning of the day). While one may argue that this is an unrealistic condition (unless there is a very vigorous testing mandate), it will serve the purpose of illustration.

Trigger condition. Using $X(p)$ to denote the health state of $p \in V$ and N for the population size, the trigger C_a can be expressed as:

$$\big|\{p \in V \mid X(p) \in \{Iasymp, Isymp\}\}\big|/N \geq 0.10$$

Target: for this intervention, the target set M_a is a subset of the edge set E. Writing a (directed) edge as $e = (v, v')$ and denoting by $A_e(v)$ and $A_e(v')$ the activity of v and v' at their time of contact encoded by the edge e, the target set can be expressed as:

$$M_a = \{e \in E \mid A_e(v) = \texttt{School} \text{ or } A_e(v') = \texttt{School}\}$$

Operation ensemble:

```
foreach // Note: "e in M" is implicit
  active = 0, delay(0), priority(0)
  active = 1, delay(1), priority(1)
```

The operation ensemble will deactivate any edge associated to a school activity, and will do so immediately (current day). It will also schedule a school re-opening on the following day by re-activating the edge with a delay of 1. Note, however, that if the trigger condition holds on the following day as well, this intervention will be executed again. In this case, the edge deactivation of the next day will take priority (technically, it will overwrite values) over the edge reactivation that is scheduled on the current day. This illustrates *an essential design point*: by allowing delays and priorities, we can avoid the the bookkeeping of tracking which entities and/or their variables were modified, as well as when they may need to be restored.

(b) Vaccination. In this case, we have a pharmaceutical intervention where people of age ≥ 60 are advised to accept a vaccine offered on day 5, with a compliance rate of 90%, the vaccine being 80% efficacious causing an 80% reduction in susceptibility and infectivity. The vaccine becomes effective two days following inoculation.

Trigger condition. In this case, we simply use the EpiHiper observable `time` to express the condition as

$$C_b : \texttt{time} = 5 \, .$$

Target set. In this case, the target set consists of vertices/people:

$$M_b = \{v \in V \mid \texttt{age}(v) \geq 60\}$$

Operation ensemble.

```
sampling relativeSampling individual 90.0
  sampled // sampled subset of M_b
    beta_i *= 0.20, delay(2)
    beta_s *= 0.20, delay(2)
  nonsampled
    {}
```

A complete examples would also need to include initialization (implemented as interventions triggered at the start of the simulations) and would typically be explored through an experimental design.

Synthetic populations and contact networks. EpiHiper uses the notion of a digital twin of the population and a social contact network of a region to describe the people (vertices) and their interactions (edges), see [50]. The synthesized population consists of a *base population* partitioned into *households* with demographic attributes (e.g., age, gender, worker status, NAICS), and household attributes such as household income. These attributes are provided to EpiHiper via its *person trait database*. Based on demographic attributes, each person is matched with a detailed activity sequence (e.g., from NHTS) covering their activities on a typical day, and each such activity is assigned a location using an algorithm outlined in the supplementary material.

Using the assignment of all people's activities to locations (which we refer to as the person-location graph G_{PL}) in combination with a contact model at each location, we construct a *social contact network* G_P with nodes V_P (people) and edges E_P denote proximity and depends on the specific disease under consideration. Each node $v \in V_P$ has a number of configurable attributes specific to the questions being studied; similarly, each edge $e \in E_P$ has associated a set of configurable attributes that minimally includes the target (resp. source) person ID, the target (resp. source) person activity type, and the duration of contact. The main components of the synthetic populations and its network are illustrated in Figure 9.

Software design and implementation. The EpiHiper software architecture is a hybrid MPI/OpenMP design, and is implemented in C++ for high performance. The contact networks can be represented either as text or binary files, with the option to perform pre-partitioning for the desired target combinations of compute nodes and cores. A custom Frictionless [47] header encodes optional custom fields for the network. These customizable demographic traits attached to each person are handled as a software Postgres database. This database, which is typically shared among multiple computational experiments, has been finely tuned to handle a large number of simultaneous queries, particularly as they occur at the initialization stage of large EpiHiper compute jobs. Details regarding the EpiHiper algorithm as well as correctness are provided in the supplementary material.

5 Code availability

The EpiHiper code base can be accessed at https://github.com/NSSAC/EpiHiper, and is made available under the MIT license: https://mit-license.org/

References

[1] Giabbanelli, P.J., Badham, J., Castellani, B., Kavak, H., Mago, V., Negahban, A., Swarup, S.: Opportunities and challenges in developing covid-19 simulation models: Lessons from six funded projects. In: 2021 Annual Modeling and Simulation Conference (ANNSIM), pp. 1–12 (2021). https://doi.org/10.23919/ANNSIM52504.2021.9552089

[2] Eubank, S., Guclu, H., Kumar, V.A., Marathe, M.V., Srinivasan, A., Toroczkai, Z., Wang, N.: Modelling disease outbreaks in realistic urban social networks. Nature **429**(6988), 180–184 (2004)

[3] Bisset, K.R., Chen, J., Feng, X., Kumar, V.A., Marathe, M.V.: Epifast: a fast algorithm for large scale realistic epidemic simulations on distributed memory systems. In:

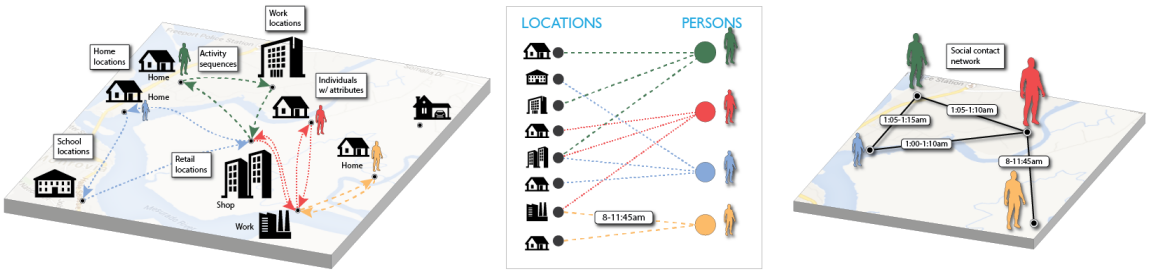


Fig. 9 An illustration of the population and network components used in EpiHiper. Location assignment is illustrated on the left, and captured formally as the people-location network G_{PL} (middle), which, in turn, gives rise to a social contact network G_P (right).

Proceedings of the 23rd International Conference on Supercomputing, pp. 430–439 (2009)

[4] Barrett, C.L., Bisset, K.R., Eubank, S.G., Feng, X., Marathe, M.V.: Episimdemics: An efficient algorithm for simulating the spread of infectious disease over large realistic social networks. In: SC '08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, pp. 1–12 (2008). https://doi.org/10.1109/SC.2008.5214892

[5] Bisset, K.R., Chen, J., Deodhar, S., Feng, X., Ma, Y., Marathe, M.V.: Indemics: An interactive high-performance computing framework for data-intensive epidemic modeling. ACM Trans. Model. Comput. Simul. **24**(1) (2014). https://doi.org/10.1145/2501602

[6] Bisset, K.R., Aji, A.M., Bohm, E., Kale, L.V., Kamal, T., Marathe, M.V., Yeom, J.-S.: Simulating the spread of infectious disease over large realistic social networks using charm++. In: 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops PhD Forum, pp. 507–518 (2012). https://doi.org/10.1109/IPDPSW.2012.65

[7] Bhatele, A., Yeom, J.-S., Jain, N., Kuhlman, C.J., Livnat, Y., Bisset, K.R., Kale, L.V., Marathe, M.V.: Massively parallel simulations of spread of infectious diseases over realistic social networks. In: 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), pp. 689–694 (2017). IEEE

[8] Yeom, J.-S., Bhatele, A., Bisset, K., Bohm, E., Gupta, A., Kale, L.V., Marathe, M., Nikolopoulos, D.S., Schulz, M., Wesolowski, L.: Overcoming the scalability challenges of epidemic simulations on blue waters. In: 2014 IEEE 28th International Parallel and Distributed Processing Symposium, pp. 755–764 (2014). IEEE

[9] Bhattacharya, P., Machi, D., Chen, J., Hoops, S., Lewis, B., Mortveit, H., Venkatramanan, S., Wilson, M.L., Marathe, A., Porebski, P., Klahn, B., Outten, J., Vullikanti, A., Xie, D., Adiga, A., Brown, S., Barrett, C., Marathe, M.: Ai-driven agent-based models to study the role of vaccine acceptance in controlling covid-19 spread in the us. In: 2021 IEEE International Conference on Big Data (Big Data), pp. 1566–1574 (2021). https://doi.org/10.1109/BigData52589.2021.9671811

[10] Bhattacharya, P., Chen, J., Hoops, S., Machi, D., Lewis, B., Venkatramanan, S., Wilson, M.L., Klahn, B., Adiga, A., Hurt, B., Outten, J., Adiga, A., Warren, A., Baek, H., Porebski, P., Marathe, A., Xie, D., Swarup, S., Vullikanti, A., Mortveit, H., Barrett, C.L., Marathe, M.: Data-Driven Scalable Pipeline using National Agent-Based Models for Real-time Pandemic Response and Decision Support. The International Journal of High Performance Computing Applications (2022). https://doi.org/10.1177/10943420221127034. Gordon Bell Award finalist

[11] US-UK Innovation Prize Challenge. https://www.whitehouse.gov/ostp/news-updates/2022/07/20/u-s-and-u-k-launch-innovation-prize-challenges-in-priva

[12] PETs Prize Challenge. https://www.drivendata.org/competitions/98/nist-federated-learning-1/

[13] Perumalla, K.S., Seal, S.K.: Discrete event modeling and massively parallel execution of epidemic outbreak phenomena. SIMULATION **88**(7), 768–783 (2012) https://arxiv.org/abs/https://doi.org/10.1177/0037549711413001. https://doi.org/10.1177/0037549711413001

[14] Skvortsov, Connell, A.R., Dawson, P., Gailis, R.: Epidemic modelling : Validation of agent-based simulation by using simple mathematical models. (2007)

[15] Grefenstette, J.J., Brown, S.T., Rosenfeld, R., DePasse, J., Stone, N.T., Cooley, P.C., Wheaton, W.D., Fyshe, A., Galloway, D.D., Sriram, A., *et al.*: Fred (a framework for reconstructing epidemic dynamics): an open-source software system for modeling infectious diseases and control strategies using census-based populations. BMC public health **13**(1), 940 (2013)

[16] Google: Agent Based Epidemic Simulator. https://github.com/google-research/agent-based-epidemic-sim (2021)

[17] Kerr, C.C., Stuart, R.M., Mistry, D., Abeysuriya, R.G., Hart, G., Rosenfeld, K., Selvaraj, P., Núñez, R.C., Hagedorn, B., George, L., Izzo, A., Palmer, A., Delport, D., Bennette, C., Wagner, B., Chang, S., Cohen, J.A., Panovska-Griffiths, J., Jastrzębski, M., Oron, A.P., Wenger, E., Famulare, M., Klein, D.J.: Covasim: an agent-based model of COVID-19 dynamics and interventions. medRxiv (2020)

[18] Kerr, C.C., Stuart, R.M., Mistry, D., Abeysuriya, R.G., Rosenfeld, K., Hart, G.R., Núñez, R.C., Cohen, J.A., Selvaraj, P., Hagedorn, B., *et al.*: Covasim: an agent-based model of COVID-19 dynamics and interventions. PLOS Computational Biology **17**(7), 1009149 (2021)

[19] Fitzpatrick, M.C., Galvani, A.P.: Optimizing age-specific vaccination. Science **371**(6532), 890–891 (2021)

[20] Agrawal, S., Bhandari, S., Bhattacharjee, A., Deo, A., Dixit, N.M., Harsha, P., Juneja, S., Kesarwani, P., Swamy, A.K., Patil, P., Rathod, N., Saptharishi, R., Shriram, S., Srivastava, P., Sundaresan, R., Vaidhiyan, N.K., Yasodharan, S.: City-scale agent-based simulators for the study of non-pharmaceutical interventions in the context of the covid-19 epidemic. Journal of the Indian Institute of Science **100**(4), 809–847 (2020)

[21] Ferguson, N., Laydon, D., Nedjati Gilani, G., Imai, N., Ainslie, K., Baguelin, M., Bhatia, S., Boonyasiri, A., Cucunuba Perez, Z., Cuomo-Dannenburg, G., et al.: Report 9: Impact of non-pharmaceutical interventions (NPIs) to reduce COVID19 mortality and healthcare demand (2020)

[22] Verity, R., Okell, L.C., Dorigatti, I., Winskill, P., Whittaker, C., Imai, N., Cuomo-Dannenburg, G., Thompson, H., Walker, P.G., Fu, H., et al.: Estimates of the severity of coronavirus disease 2019: a model-based analysis. The Lancet infectious diseases (2020)

[23] Chinazzi, M., Davis, J.T., Ajelli, M., Gioannini, C., Litvinova, M., Merler, S., y Piontti, A.P., Mu, K., Rossi, L., Sun, K., et al.: The effect of travel restrictions on the spread of the 2019 novel coronavirus (covid-19) outbreak. Science (2020)

[24] Kraemer, M.U., Yang, C.-H., Gutierrez, B., Wu, C.-H., Klein, B., Pigott, D.M., du Plessis, L., Faria, N.R., Li, R., Hanage, W.P., et al.: The effect of human mobility and control measures on the covid-19 epidemic in china. Science (2020)

[25] Peng, L., Yang, W., Zhang, D., Zhuge, C., Hong, L.: Epidemic analysis of covid-19 in china by dynamical modeling. arXiv preprint arXiv:2002.06563 (2020)

[26] Roosa, K., Lee, Y., Luo, R., Kirpich, A., Rothenberg, R., Hyman, J., Yan, P., Chowell,

G.: Real-time forecasts of the covid-19 epidemic in china from february 5th to february 24th, 2020. Infectious Disease Modelling **5**, 256–263 (2020)

[27] Manninen, T., Aćimović, J., Havela, R., Teppola, H., Linne, M.-L.: Challenges in reproducibility, replicability, and comparability of computational models and tools for neuronal and glial networks, cells, and subcellular structures. Frontiers in neuroinformatics **12**, 20 (2018)

[28] Hunter, E., Kelleher, J.: A framework for validating and testing agent-based models: a case study from infectious disease modelling. In: 34th Annual European Simulation and Modelling Conference (2020). https://doi.org/10.21427/2xjb-cq79

[29] Hunter, E., Kelleher, J.D.: Validating and testing an agent-based model for the spread of covid-19 in ireland. Algorithms **15**(8) (2022). https://doi.org/10.3390/a15080270

[30] Li, J., Giabbanelli, P.J.: Identifying synergistic interventions to address COVID-19 using a large scale agent-based model. In: Paszynski, M., Kranzlmüller, D., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M.A. (eds.) Computational Science – ICCS 2021, pp. 655–662. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77967-2_54

[31] Machi, D., Bhattacharya, P., Hoops, S., Chen, J., Mortveit, H., Venkatramanan, S., Lewis, B., Wilson, M., Fadikar, A., Maiden, T., *et al.*: Scalable epidemiological workflows to support covid-19 planning and response. In: 2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 639–650 (2021). IEEE

[32] Hoops, S., Chen, J., Adiga, A., Lewis, B., Mortveit, H., Baek, H., Wilson, M., Xie, D., Swarup, S., Venkatramanan, S., *et al.*: High performance agent-based modeling to study realistic contact tracing protocols. In: Proceedings of the 2021 Winter Simulation Conference (WSC) (2021)

[33] Truelove, S., Smith, C.P., Qin, M., Mullany, L.C., Borchering, R.K., Lessler, J., Shea, K., Howerton, E., Contamin, L., Levander, J., *et al.*: Projected resurgence of covid-19 in the united states in july—december 2021 resulting from the increased transmissibility of the delta variant and faltering vaccination. Elife **11**, 73584 (2022)

[34] Lessler, J., Shea, K., Viboud, C., Truelove, S., Borchering, R., Smith, C., et al.: COVID-19 Scenario Modeling Hub. https://github.com/midas-network/covid19-scenario-modeling-hub (2022)

[35] Mossong, J., Hens, N., Jit, M., Beutels, P., Auranen, K., Mikolajczyk, R., Massari, M., Salmaso, S., Scalia Tomba, G., Wallinga, J., Heijne, J., Sadkowska-Todys, M., Rosinska, M., John, E.W.: Social contacts and mixing patterns relevant to the spread of infectious diseases. PLoS Medicine **5**, 1–1 (2008). https://doi.org/10.1371/journal.pmed.0050074

[36] Prem, K., Cook, A.R., Jit, M.: Projecting social contact matrices in 152 countries using contact surveys and demographic data. PLOS Computational Biology **13**, 1005697 (2017). https://doi.org/10.1371/journal.pcbi.1005697

[37] Cattuto, C., Van den Broeck, W., Barrat, A., Colizza, V., Pinton, J., Vespignani, A.: Dynamics of person-to-person interactions from distributed rfid sensor networks. PLOS ONE **5**(7), 11596 (2010). https://doi.org/10.1371/journal.pone.0011596

[38] SynthPops. Last accessed: 22 Jul 2022 (2022). https://docs.idmod.org/projects/synthpops/en/latest/

[39] Shoukat, A., Wells, C.R., Langley, J.M., Singer, B.H., Galvani, A.P., Moghadas, S.M.: Projecting demand for critical care beds during COVID-19 outbreaks in Canada. Canadian Medical Association Journal **19**, 489–496 (2020). https://doi.org/10.1503/cmaj.200457

[40] Moghadas, S.M., Fitzpatrick, M.C., Sah, P., Pandey, A., Shoukat, A., Singer, B.H., Galvani, A.P.: The implications of silent

transmission for the control of COVID-19 outbreaks. Proceedings of the National Academy of Sciences **117**(30), 17513–17515 (2020) https://arxiv.org/abs/https://www.pnas.org/doi/pdf/10.1073/pnas.2008373117. https://doi.org/10.1073/pnas.2008373117

[41] Chawla, D.S.: Critiqued coronavirus simulation gets thumbs up from code-checking efforts. Nature **582**(7812), 323–325 (2020)

[42] Adiga, A., Dubhashi, D., Lewis, B., Marathe, M., Venkatramanan, S., Vullikanti, A.: Mathematical models for covid-19 pandemic: a comparative analysis. Journal of the Indian Institute of Science **100**(4), 793–807 (2020)

[43] Lucas, T.C., Pollington, T.M., Davis, E.L., Hollingsworth, T.D.: Responsible modelling: unit testing for infectious disease epidemiology. Epidemics **33**, 100425 (2020)

[44] Ivie, P., Thain, D.: Reproducibility in scientific computing. ACM Computing Surveys (CSUR) **51**(3), 1–36 (2018)

[45] MPI. Last accessed: 12 May 2021. https://www.open-mpi.org/

[46] OpenMP. Last accessed: 12 May 2021. https://www.openmp.org/

[47] Frictionless Data. Last accessed: 14 November 2019. https://frictionlessdata.io/

[48] Gillespie, D.T.: A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. J. Comp. Phys. **22**, 403–434 (1976)

[49] Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. J. Phys. Chem. **81**(25), 2340–2361 (1977)

[50] Chen, J., Hoops, S., Lewis, B.L., Henning S. Mortveit and, S.V., Wilson, A.: EpiHiper: Modeling and Implementation. NSSAC Technical Report Series: No. 2019–003 (2019)

[51] Mortveit, H.S., Adiga, A., Barrett, C.L., Chen, J., Chungbaek, Y., Eubank, S., Kuhlman, C.J., Lewis, B., Swarup, S., Venkatramanan, S., Wilson, A., Xie, D., Marathe, M.V.: Synthetic populations and interaction networks for the U.S. Technical report, NSSAC, University of Virginia (2020). NSSAC Technical Report: #2019-025

[52] Adiga, A., Beckman, R., Bisset, K., Chen, J., Baek, Y.C., Eubank, S., Gupta, S., Khan, M., Kuhlman, C., Lofgren, E., Lewis, B., Marathe, A., Marathe, M., Mortveit, H., Nordberg, E., Rivers, C., Stretz, P., Swarup, S., Wilson, A., Xie, D.: Synthetic Populations for Epidemic Modeling. International conference on computational social sciences (ICCSS), June 8-11, Helsinki, Finland, 2015 (2015)

[53] United States Census: 2017 North American Industry Classification System (NAICS) Manual. Last accessed: 15 July 2020. https://www.census.gov/library/publications/2017/econ/2017-naics-manual.html

[54] U.S. Census: Public Use Microdata Sample (PUMS). Last accessed: 24 May 2021 (2021). https://www.census.gov/programs-surveys/acs/microdata.html

[55] Beckman, R.J., Baggerly, K.A., McKay, M.D.: Creating synthetic baseline populations. Transportation Research Part A: Policy and Practice **30**(6), 415–429 (1996). https://doi.org/10.1016/0965-8564(96)00004-3

[56] Lum, K., Chungbaek, Y., Eubank, S., Marathe, M.: A two-stage, fitted values approach to activity matching. International Journal of Transportation **4**, 41–56 (2016). https://doi.org/10.14257/ijt.2016.4.1.03

[57] Breiman, L.: Classification and Regression Trees. Wadsworth statistics/probability series. Wadsworth International Group, New York (1984). https://books.google.com/books?id=ZxPvAAAAMAAJ

[58] U.S. Department of Transportation, Federal Highway Administration: The National Household Travel Survey (NHTS). Last accessed: February 2020. URL:https://nhts.ornl.gov.

[59] The University of Oxford: The Multinational Time Use Study (MTUS). Last accessed: February 2020. https://www.timeuse.org/mtus

[60] United States Department of Labor, Bureau of Labor Statistics: The American Time Use Survey (ATUS). Last accessed: February 2020. https://www.bls.gov/tus/

[61] Microsoft: U.S. Building Footprints. https://github.com/Microsoft/USBuildingFootprints (2020)

[62] HERE. http://www.here.com, Accessed April 2020 (2020)

[63] BuildingFootprintUSA: BuildingFootprintUSA. Last accessed 1 April 2020 (2020). https://www.buildingfootprintusa.com/

[64] SLIPO: World-scale OpenStreetMap POIs in CSV. Last accessed: June 2021 (2021). http://www.slipo.eu/?p=1551

[65] National Center for Education Statistics. Last accessed: December 2021. http://nces.ed.gov

[66] U.S. Census: 2010 Census Urban and Rural Classification. Last accessed: June 2021. https://www.census.gov/programs-surveys/geography/guidance/geo-areas/urban-rural/2010-urban-rural.html

[67] United States Censuc Bureau: 2011-2015 5-Year ACS Commuting Flows. Last accessed: April 2020. https://www.census.gov/data/tables/2015/demo/metro-micro/commuting-flows-2015.html

[68] U.S. Census: Longitudinal Employer-Household Dynamics. Last accessed: 26 November 2020 (2020). https://lehd.ces.census.gov/data/

[69] Bureau of Transportation Statistics: Local Area Transportation Characteristics for Households Data. Last accessed: 31 Jan 2022. https://www.bts.gov/latch/latch-data

[70] Mistry, D., Litvinova, M., Pastore y Piontti, A., Chinazzi, M., Fumanelli, L., Gomes, M.F.C., Haque, S.A., Liu, Q.-H., Mu, K., Xiong, X., Halloran, M.E., Longini, I.M., Merler, S., Ajelli, M., Vespignani, A.: Inferring high-resolution human mixing patterns for disease modeling. Nature Communications **12**(1), 323 (2021). https://doi.org/10.1038/s41467-020-20544-y

6 Acknowledgements

We thank members of NSSAC for their suggestions and comments. This work was partially supported by the National Institutes of Health (NIH) Grant R01GM109718, VDH Grant PV-BII VDH COVID-19 Modeling Program VDH-21-501-0135, NSF Grant No.: OAC-1916805, NSF Expeditions in Computing Grant CCF-1918656, CCF-1917819, NSF RAPID CNS-2028004, NSF RAPID OAC-2027541, US Centers for Disease Control and Prevention 75D30119C05935, DTRA subcontract/ARA S-D00189-15-TO-01-UVA, NSF XSEDE TG-BIO210084. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies. This work used resources, services, and support from the COVID-19 HPC Consortium (https://covid19- hpc-consortium.org/), a private-public effort uniting government, industry, and academic leaders who are volunteering free compute time and resources in support of COVID-19 research.

7 Author contributions

HM, SH, JC, and MM worked on system design, validation, and modeling. SF led the software development. HM led the overall system design. JC led the effort on use of the models for real-world studies. JC, SH, MM worked on extending the models, testing, validation and extensions to SMH use cases. PB, MW and DM worked on system porting, preparing data sets, supporting real-world studies and doing analytics. BL and SV worked on modeling disease progression models, model validation and modeling requirements. All worked on preparing, modifying and completing the manuscript. All authors participated in writing and editing the paper.

8 Competing interests

The authors declare no competing interests.

9 Additional information

Supplementary information is currently provided as an appendix contained within this document.

Correspondence and requests for materials should be addressed to Henning S. Mortveit, Jiangzhuo Chen, and Stefan Hoops.

Supplementary Material

A The EpiHiper model

This section covers all aspects of the EpiHiper model starting from foundational concepts and going all the way through interventions. A running example of concepts is provided in the main part of the paper.

A.1 Core concepts

Time and Iterations. The model advances in discrete time steps t_0, t_1, t_2, ..., t_n starting at t_0 and where *iteration* k advances time from t_{k-1} to t_k. The duration of each iteration is constant and is denoted by T, which implies that $t_n = t_{n-1} + T$. Iteration 0 handles all initialization and anything that takes place up until the time t_0. In particular, this covers invocation of interventions, including vaccinations that have been administered to members of the population prior to t_0.

Health states. The disease model has a set of *health states* denoted by

$$\mathcal{X} = \{X_1, X_2, \ldots, X_m\} \,. \tag{2}$$

For a standard SIR-model, we have $\mathcal{X} = \{S, I, R\}$. Each person, represented as a network node, is assigned a time-varying *health state* that is updated at initialization and as the simulation progresses through the *transmission process* and the *natural disease progression*. Health states can also be updated during interventions.

Susceptibility and infectivity. Each health state $X \in \mathcal{X}$ has associated *susceptibility* and *infectivity* denoted by $\sigma(X)$ and $\iota(X)$. These are fixed disease parameters (and are thus independent of people). To model *susceptibility* and *infectivity* of *individuals*, which will generally depend on factors such as vaccination and use of personal protective equipment, we assign each person P an *infectivity scaling factor* and a *susceptibility scaling factor*, and denote these factors by $\beta_\iota(P)$ and $\beta_\sigma(P)$. These are time-varying states with all with default value 1.0. Following this, the *effective susceptibility and infectivity* of person P in health state X_i are modeled as (skipping the reference to time, i.e., t_n)

$$\sigma_P(X_i) = \beta_\sigma(P) \times \sigma(X_i) \,, \text{ and}$$
$$\iota_P(X_i) = \beta_\iota(P) \times \iota(X_i) \,.$$

Traits. EpiHiper supports the notion of *traits*. These are configurable, dynamic variables that can be attached to each person (node traits) and/or to each contact (edge traits). The collection of traits are part of the model specification, and are one of the foundations for interventions, which we present below.

An example of a node trait useful to scenarios involving vaccination is a Boolean flag `vaccinated` that (through interventions) can track a person's vaccination status. Similarly, an edge may have an associated Boolean edge trait `indoors` encoding if the contact takes place indoors. An intervention can then conditionally handle people of different vaccination statuses as well as indoor and outdoor contacts.

Variables. In addition to the per-person or per-contact traits described above, EpiHiper supports declaration of *variables*. A prime example of a variable could be the number of vaccine doses administered within a health district on any given day, thus ensuring that the model can handle constrained resources. Another example of a variable is one tracking whether school closures are in effect. We note that variables are updated as part of interventions, and also that EpiHiper tracks a collection of standard variables, such as the total and relative counts of people in each of the health states as defined by the disease model.

EpiHiper primitives. These are the basic quantities that EpiHiper makes available for operations, either as part of the right-hand-side expression of assignments or as the left-hand-side (often referred to as an `lValue`) in assignment operations. Table 3 provides a list.

Sets. EpiHiper supports the declaration of sets. Sets may contain nodes and edges, and are formed using standard combinations of logical predicates, variables and EpiHiper primitives. An example usage of sets is to pre-compute and cache frequently used population subsets as intervention targets. Sets are a special case of variables. The Methods Section of the main paper provides an example related to school closure and vaccinations.

node.beta_i	|rw|	# infectivity scaling factor
node.beta_s	|rw|	# susceptibility scaling factor
node.healthState	|rw|	# health state
node.nodeTrait[featureName]	|rw|	# nodeTrait featureName of node
edge.edgeTrait[featureName]	|rw|	# edgeTrait featureName of edge
edge.active	|rw|	# active flag of edge
edge.weight	|rw|	# weight of edge
node.id	|r |	# PID of node
edge.sourceActivity[featureName]	|r |	# activiy of source of edge [only activityType]
edge.targetActivity[featureName]	|r |	# activity of target of edge [only activityType]
edge.sourceID	|r |	# source vertex ID of edge
edge.targetID	|r |	# target vertex ID of edge
observable	|r |	# current time, total population, etc
transmissibility	|rw|	# global transmissibility
<SetName>	|r |	# any set referenced by name
<VariableName>	|rw|	# any variable referenced by name
<TriggerName>	|r |	# any trigger referenced by name

Table 3 EpiHiper 1.0 primitives; here `w` means writable and `r` means readable.

A.2 The EpiHiper disease model

The disease model is split into (i) a disease transmission and (ii) *disease progression*. The transmission process governs how individuals become infected while the disease progression captures the health state evolution once infected. While it may be convenient to combine these processes, it is important to note that they are structurally different: transmission requires the presence of one or more infectious people to infect a candidate, susceptible person.[6]

A.2.1 Disease Transmission.

Disease transmissions that arise from contacts between infectious and susceptible individuals are modeled as follows: first, contacts are captured as directed edges in the contact network (see Section B). The potential infections that may take place are determined through a person's incident edges and a list of *contact configurations* of health states. Specifically, an individual P in health state X_i (the *entry state*) may transition to state X_j (the *exit state*) when in contact with a person P' in state X_k (the *contact state*). We call this a *transmission configuration*, denote it by $T_{i,j,k} = T(X_i, X_j, X_k)$, and associate to it the *transmission weight* $\omega_{i,j,k} = \omega(T_{i,j,k})$. This weight represents the relative weight of this particular transition and is set to 1 by default. For modeling the infection process, one will specify all the possible transmission configurations. Note that transmission configurations are disease parameters and independent of people and their attributes. In accordance with standard terminology, we call any entry state a *susceptible state*, any exit state an *exposed state*, and any contact state an *infectious state*. An EpiHiper model may have multiple susceptible and infectious states.

For the infection (or transmission) process of a person P in susceptible state X_i, we need to consider all possible transitions from X_i to possible exit states X_j in the presence of persons P' in possible contact states X_k.

Next, a disease model will have a *transmissibility* that we denote by τ. It is a global parameter representing a rate proportional to the likelihood of becoming infected by being in contact with a single susceptible individual for one time unit; it may be used for calibration of, for example, $R_{\text{effective}}$.

Finally, each edge e in the contact network is of the form

$$e = (P', a(P'), P, a(P), w, \alpha, T, \text{edgeTrait})$$

[6] EpiHiper does not currently incorporate transmission through inanimate object.

where $a(P')$ is the *activity* of person P' at the time of contact, T is the contact duration (within the current iteration), w is an edge weight, and α is Boolean (edge) variable indicating whether or not the edge is active. Finally, edgeTrait is a configurable (but optional) set of variables (edge traits) that may have been encoded when constructing the network. Figure 10 provides an overview of all the parameters and concepts covered above and that are involved in the transmission process.

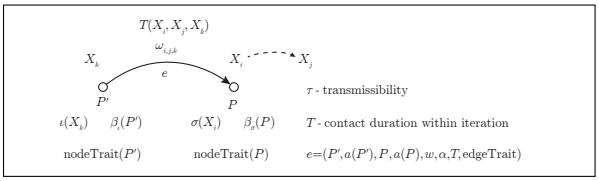


Fig. 10 Overview of disease model parameters governing disease transmission.

Under the assumption that transmissions across all edges and contact configurations are independent for each person P and their contacts P', we define the *propensity* ρ for the edge e and contact configuration as:

$$\rho(P, P', T_{i,j,k}, e) = \big[T \cdot \tau\big] \times w_e \times \alpha_e \times \big[\beta_s(P) \cdot \sigma(X_i)\big] \times \big[\beta_i(P') \cdot \iota(X_k)\big] \times \omega(T_{i,j,k}) \tag{3}$$

The algorithm to determine whether a disease transmission takes place in the current iteration (and also the decision of which contact to attribute the transmission) as a result of all candidate transmission configurations is modeled after the Direct Gillespie Method [48, 49].

For $(m \geq 1)$ transmission configuration candidates for person P within an iteration, we choose the actual contact configuration (i.e., person P' and contact configuration $T_{i,j,k}$ to whom we attribute the infection) by calculating

$$A(P) = \sum_{P',j,k} \rho(P, P', T_{i,j,k}) \ , \tag{4}$$

where the sum extends over all neighbors P' of P and indices j and k for which a transmission may occur. Without loss of generality, we assume that the index set of triples $K = \{(P', j, k)\}$ is well ordered. To determine *if* we have a transition we sample a random number

$$a = -\ln(\text{uniform}(0, 1))/A \ ,$$

and, if the inequality $a \leq T = \Delta t_n$ holds, we select the actual transition by sampling a uniform random number $\alpha \in [0, A]$ and determine the index $\kappa \in K$ for which

$$\sum_{\kappa-1} \rho(P, P', T_{i,j,k}) < \alpha \leq \sum_{\kappa} \rho(P, P', T_{i,j,k}) \ . \tag{5}$$

Finally, the selected health state transition is scheduled for time t_n.

A.2.2 Disease Progression

The disease progression process covers the health state transitions within an individual P that are independent of other people. For the EpiHiper model, a disease progression diagram (or specification) describes all the possible health state transitions that may take place within a person in the absence of transmission processes and interventions. The diagram has nodes the set of health states $\mathcal{X} = \{X_i\}$ and directed edges of the form $e = (X_i, X_j)$, each edge with an assigned probability $p_e = \text{prob}(X_i, X_j)$ and a *dwell time distribution* D_e. We require that, for each state X_i, the sum $\sum_j \text{prob}(X_i, X_j)$ over outgoing edges must equal 1. The dwell time distribution D_e for $e = (X_i, X_j)$ is the probability density for the dwell time in health state X_i given that the transition $X_i \longrightarrow X_j$ takes place.

Algorithmically, disease progression is modeled as follows: when an individual P enters a state X_i, the next state X_j is sampled according to the next state distribution induced by the probabilities p_e, the dwell time ΔT in state X_i is determined by sampling the dwell time distribution D_e, rounded to the nearest integer if necessary, and then bounded below by 0. Finally, the health state transition is scheduled to take place at iteration $t_n + \Delta T$. We remark that disease progression may be overridden by interventions and transmissions, should such events occur before the scheduled health state transition dictated by the disease progression.

A.3 Interventions

Disease transmission and disease progression governs the dynamics of health states in EpiHiper. However, the real power of EpiHiper lies in its rich *intervention model*. An intervention $\mathcal{I}$ is a triple $\mathcal{I} = (T, E, \mathfrak{O})$ where T is the *trigger condition*, a Boolean expression, the set E is the *intervention target*, a collection of vertices and edges, and $\mathfrak{O}$ is a set of operations that can be applied to the variables associated to the elements of target set E, as well as global variables. The trigger expression is a Boolean expression involving EpiHiper primitives and possible sizes of sets. An intervention is executed for every iteration for which its trigger evaluates to `true` at the start of the iteration.

While expressiveness of the set of operations $\mathfrak{O}$ of an intervention may not fully rival that of a general purpose programming language, the constructs that it permits are quite powerful. To describe it, we use the some new notions:

- `operation`: an assignment to a variable of the system;
- `operationList`: an ordered, possibly empty, list of `operations`;
- `operationEnsemble`: the permissible constructs used with EpiHiper.

The recursive, grammar-like definition of what EpiHiper accepts is shown in Listing 1 but is repeated here:

```
operationEnsemble :=
  once
    <operationList>
  foreach
    <operationList>
  sampling <samplingSpecification>
    sampled
      <operationEnsemble>
    nonsampled
      <operationEnsemble>
```

where

```
samplingSpecification :=
(
  relativeSampling ( individual | \
    group ) <percentage> |
  absoluteSampling <integer>
)
```

and

```
operationList := <operation>+

operation := <variable>
   <operator> <expression> \
     delay(<integer>)       \
   [ priority(<integer>) ] \
   [ condition(<bool_expression>) ]

operator := ( = | *= | /= | += | -= )
```

The blocks within the `actionEnsemble` have the semantics:

`once`: the `operationList` of the `once` block will be executed precisely once when the intervention is executed. Its purpose is to serve as a mechanism to invoke actions just once if the intervention is triggered to assign, e.g., global variables.

`foreach`: the `operationList` of the `foreach` block will be executed once against each element of the target set. If the target element is a node (resp. edge), only action statements that are valid for nodes (resp. edges) and variables will be executed while edges (resp. nodes) will be ignored.

`sampling`: the `sampling` block first partitions the target set into two sets called `sampledSet` and `nonSampledSet`, where `nonSampledSet` = `target\sampledSet`. The set `sampledSet` is determined as follows:

- `absoluteSampling N`: samples precisely min(`N`, {target″) elements of the target set;
- `relativeSampling individual percentage`: each element of the target set is sampled with probability `percentage/100` to form `sampledSet`;
- `relativeSampling group percentage`: precisely `percentage` percent of the target set is sampled at random to form `sampledSet`.

The target set of the `sampled` branch is `sampledSet`, and the target set of the `nonsampled` branch is `nonSampledSet`.

Requirement: at least one of the `once`, `foreach`, and `sampling` blocks must be present to form a valid `actionEnsemble`. There is no limit on the recursion depth made available through the sampling control structure, but, in practice, this will not be a concern.

B Populations and networks

EpiHiper relies on population and network data for the study region R. For this, we use our *digital twins* framework [51, 52] of the actual populations (also referred to as synthetic populations). These are statistically accurate representations of the actual population, and capture (i) the people and their household structures, (ii) activity schedules for each person modeling the activities they conduct (and when) during a typical day or week, (iii) the locations where they conduct their activities, and (iv) a contact network describing with whom they come in contact during this time; see Figure 9 for more details.

More specifically, a digital twin has a *base population*, which is a set P consisting of (digital) people with demographic attributes such as age, gender, race, and designation (derived from the NAICS classification [53] of the PUMS [54]). Household structure is central to many scenarios, in particular to epidemiology and the design of interventions. The base population is therefore partitioned into a set H of *households* through the iterative proportional fitting (IPF) [55] process and by using the data contained in the PUMS records. Households are augmented with data such as household income and number of workers.

Each person in the digital twin population is matched with an *activity sequence* using techniques such as CART and FVM [56, 57], with data collected mainly from the NHTS [58] in the case of digital twins for the United States, but also from sources such as Multinational Time Use Study (MTUS), American Time Use Survey (ATUS) [59, 60]. These activity sequences, upon standardization, contain typical activities such as work, school, college, shopping, religion, and other, as well as being at home, along with the start time and duration of each activity.

The *locations* where people may conduct their activities include residential dwelling units (residence locations) and locations where people conduct their non-home activities such as work, school, worship, or shopping (activity locations). Locations, which are geographically embedded, are constructed carefully through an ensemble of PostGIS and machine-learning based models fusing NSSAC's extension of the Microsoft Building Data [61] with point-of-interest (POI) data from HERE [62], BuildingFootprintUSA [63], and Scalable Linking and Integration of big POI data (SLIPO) [64]. This is augmented with data on school and college locations from the National Center of Education Statistics (NCES) [65], and classification steps based on, for example, land-use polygons and urban/rural classifications [66]. Locations, as explained below, represent the places where people interact, and are augmented with North American Industry Classification System (NAICS)-derived designations.

The population construction next performs a *location assignment* that maps people's activities to locations in a manner constrained by rules such as "school activities should happen at schools", "shopping activities must take place in locations supporting retail", and "a student's residence and school location should reside in the same county" (although there are clearly exceptions). Assignment of work locations uses the American Community Survey (ACS) commute flow data [67] and LEHD Origin-Destination Employment Statistics (LODES) [68] to accurately capture commuting patterns and long-distance travel, and to match average daily travel distances as reported by the Census [69].

The activity location assignment is succinctly captured by the *people-location network* G_{PL} shown in Figure 9 (middle). A contact model suited for the application scenario (e.g., epidemics in the current case) is applied at each location to construct a *social contact network* G_P. The network G_P has vertex set P, and the contact model is used to infer which simultaneous visits of people p and p' are deemed to result in a contact and thus be captured as an edge $e = (p, p')$ in G_P as illustrated on the right in Figure 9. From the network G_P we also construct *contact matrices* giving mixing rates among age groups of the form used in [35, 36, 70]. (i) a location assignment that, for each person and each of their activities, assigns a location to that activity based on a match of, e.g., designations (person designation vs. location designation: a school activity generally takes place at a school), American Community Survey (ACS) commute flow data, and National Center for Education Statistics (NCES) school district information, and (ii) a suitable contact model applied at each location to infer which simultaneous visits to a location should constitute an edge in the network. The contact model is

informed by the application scenario (e.g., spread of virus).

Mathematical model. Formally, a *social contact network* is a labeled network $G_P(V_P, E_P)$. The nodes V_P denote individual agents (people, animals, etc.). Edges in the set E_P denote proximity and depends on the specific disease under consideration. For air-borne disease an edge captures physical proximity within a few feet; for sexually transmitted disease, this would represent a sexual interaction. Each node $v \in V_P$ has a number of configurable attributes. The precise set of attributes depends on the questions being studied, but for the U.S. will minimally include person ID, household ID, age (in years), age group (as defined by the CDC), gender, the latitude and longitude of their residence and associated administrative IDs going four levels deep (i.e., state ID, county ID, census tract ID, block group ID). Similarly, each edge $e \in E_P$ has associated a set of configurable attributes that minimally includes the target (resp. source) person ID, the target (resp. source) person activity type, and the duration of contact.

The main components of the digital twin are illustrated in Figure 9 covering (i) the people, (ii) their residences, retail locations, schools and workplaces, and (iii) their mapping of activities to locations (left side). In the middle of Figure 9, the resulting *people-location network* G_{PL} is illustrated, while on the right, an example contact network G_P is shown. EpiHiper may pre-partition the contact network G_P, and will load the custom person- and edge attribute data into its *trait database*.

The population P and network data $G_P(V_P, E_P)$ are provided to EpiHiper in the form of the *person and edge trait database* (see Appendix Section C). This highly customizable database allows EpiHiper to efficiently access the digital twin data and to construct, for example, target sets for interventions, see Figure 1 of the main paper. In addition, this design also allows efficient augmentation of person traits by adding, for example, model-based inference of vaccination status and likelihoods of compliance.

C The EpiHiper Discrete time parallel simulator

Architecture description. EpiHiper is a software application designed to work in an HPC as well as in a desktop environment. It is implemented in C++ and uses MPI and/or OpenMP for parallel processing. The need for parallel processing arises mainly from the memory requirement to store the contact network (vertices and edges) which has time-evolving attributes, but it obviously also has an impact on the compute time. During initialization (see the algorithm in Listing 2), the network is distributed such that each process (i.e., compute node) is assigned a fixed set of vertices and all of their incoming edges while ensuring that the memory requirements are evenly distributed. For the operations described in Section A.3, only the owner process of vertices and edges is allowed to update their states. With this restriction, the execution of operations is done by each process using the queuing model described in the previous section. While processing interventions it can happen that variables of vertices or edges belonging to other processes must be altered. This is handled by scheduling these operations on the processes (i.e., compute nodes) that own those vertices or edges. In particular, we note that the disease process (transmission and progression) of vertices is fully controlled by their owner process.

As can be seen in the algorithms of Listings 2 and 3, we achieve unambiguous processing by cleanly separating the point of execution of operations from the point where operations are scheduled through disease transmission, disease progression, or by interventions.

The EpiHiper model and implementation support variables that are not attached to vertices or edges; we refer to such variables as *unattached variables*. For example, one may need pharmaceutical interventions addressing constrained resources such as a limited number of vaccine doses. A straightforward way to implement this is to introduce a variable (e.g., vaccineCounter) that tracks the count of remaining doses, and that is decremented whenever a dose of this vaccine is administered. Since different processes generally will operate at different computational speeds, it is clear that the presence of unattached

Listing 2 The EpiHiper initialization

```
Algorithm: InitializeEpiHiper
  if network not partitioned :
    PartitionNetwork()

  LoadAndCompile:
    network, diseaseModel, traits,
    interventions, initialization

  InitializePersonTraitDB()

  t := Tstart - 1
  Initialize:
    Output, StateCounts

  CreateDependencyGraph()
  UpdateAllDependencies()

  Scheduling:
    ProcessInitialization()
  Update:
    ExecuteActions()

  t += 1
  Output:
    healthStateChanges
    healthStateCounts
    network // optional

  SynchronizeChanges()
```

Listing 3 The EpiHiper main algorithm

```
InitializeEpiHiper()

while t < Tend :
  ResetVariable()
  UpdateAllDependencies()
  ProcessTriggers()

  Scheduling:
    InfectionProcess()
    ProcessInterventions()
  Update:
    ExecuteActions()

  t += 1
  Output:
    healthStateChanges
    healthStateCounts
    network // optional

  SynchronizeChanges()
```

variables may introduce scheduling dependencies. Based on the EpiHiper model description in Section A, the algorithm for execution of operations (Section A.3), and the above details regarding the EpiHiper architecture and algorithm in its implementation, we can now summarize all this as follows:

Theorem 1 (Correctness and invariance) *For a set of interventions $(\mathcal{I}_k)_k$ that does not contained unattached variables, the* EpiHiper *software architecture correctly implements the* EpiHiper *formal model. Modulo the equivalence on orders of operations of equal priority, for any fixed input specification,* EpiHiper *generates the same output.*

Regarding the invariance statement, it is understood that the input specification includes (a) the random seed used to initialize the random number generator, and (b) the process specification (i.e., the number of compute nodes).

A note on health state updates. Since it may not be obvious, we remark on how health state changes are implemented and executed: upon entry to a new health state X, the successor state X' is immediately determined along with the dwell time t_d in health state X'. The operation that sets the health state to X' is then scheduled with a delay of t_d and suitable priorities and conditions as needed.

C.1 Detailed algorithms

The following diagrams and figures detail the main algorithm and expansions thereof as per the following list:

- Main process: Figure 11
- Transmission process: Figure 12
- Trigger processing: Figure 13
- Intervention processing: Figure 14
- Action ensemble processing: Figure 15
- Action scheduling processing: Figure 16
- Action execution processing: Figure 17

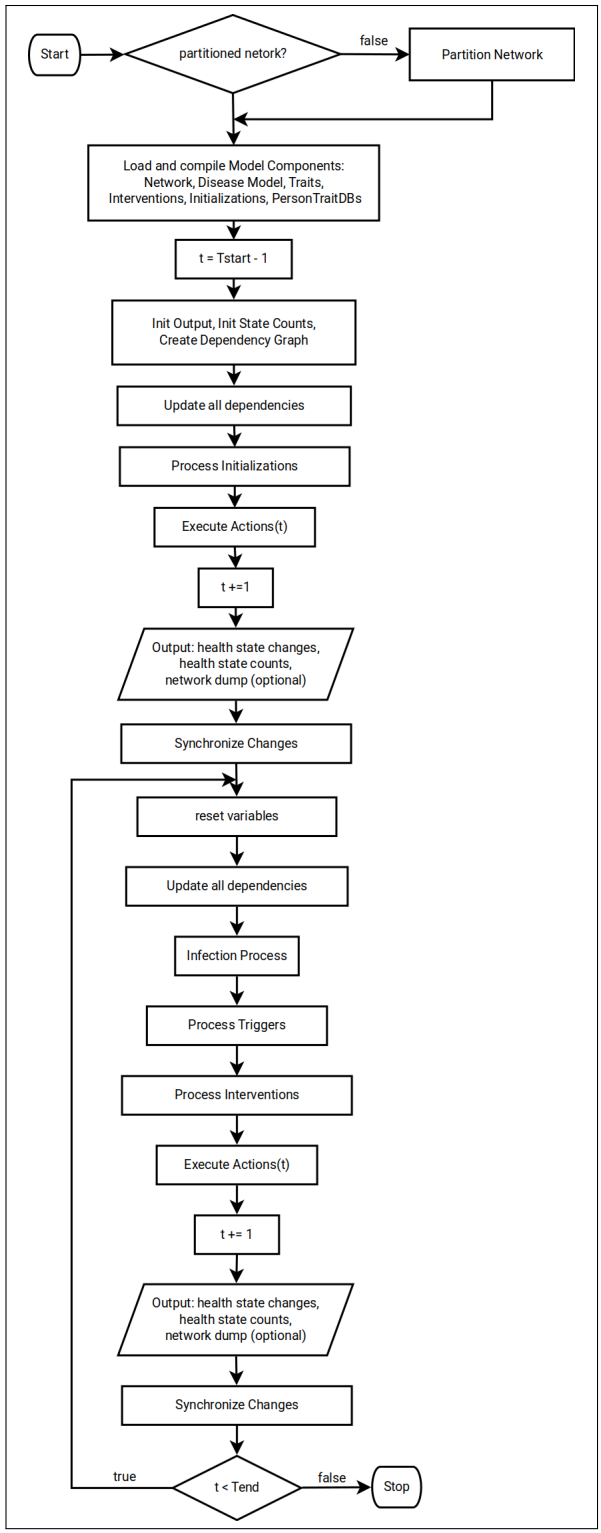


Fig. 11 EpiHiper main algorithm. This flow chart shows the overall implementation of the simulation process, including partitioning and initialization.

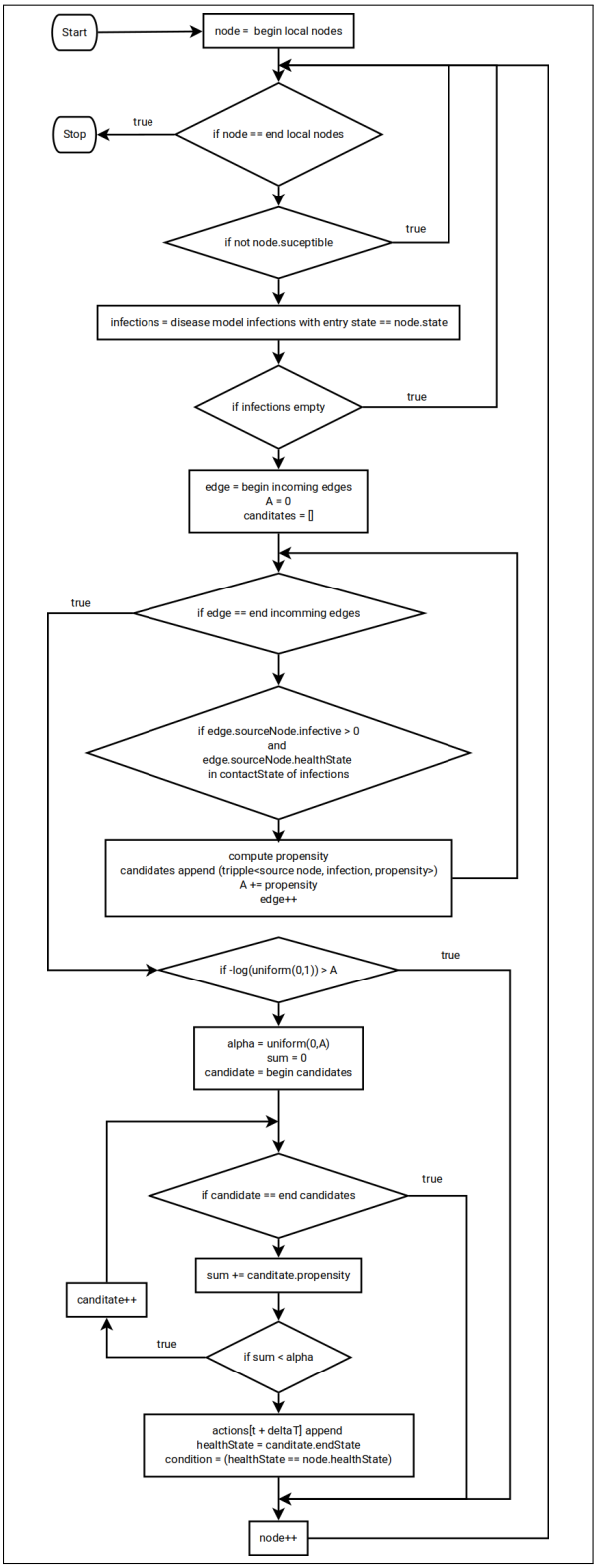


Fig. 12 EpiHiper Transmission processing algorithm.

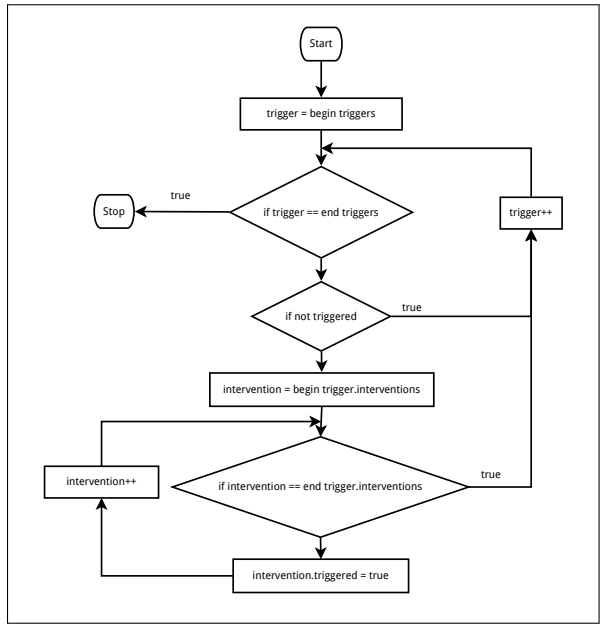


Fig. 13 EpiHiper trigger processing algorithm.

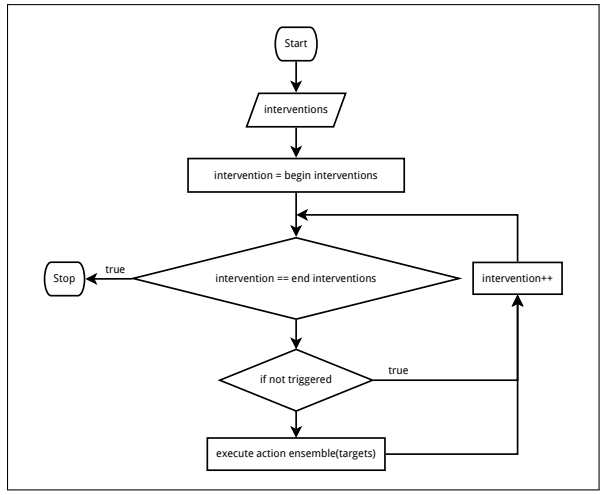


Fig. 14 EpiHiper intervention processing algorithm.

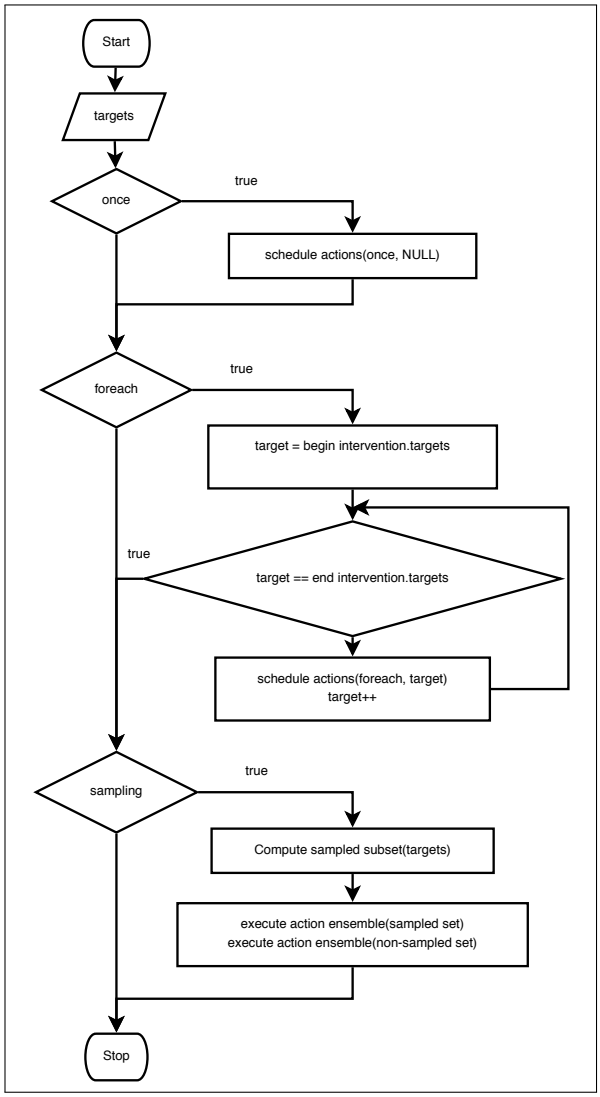


Fig. 15 EpiHiper action ensemble processing algorithm.

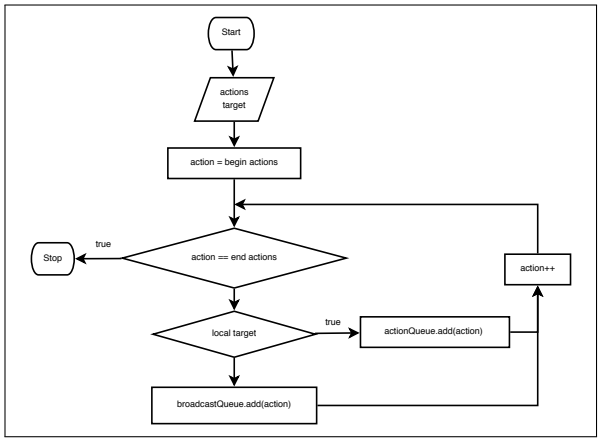


Fig. 16 EpiHiper action scheduling algorithm.

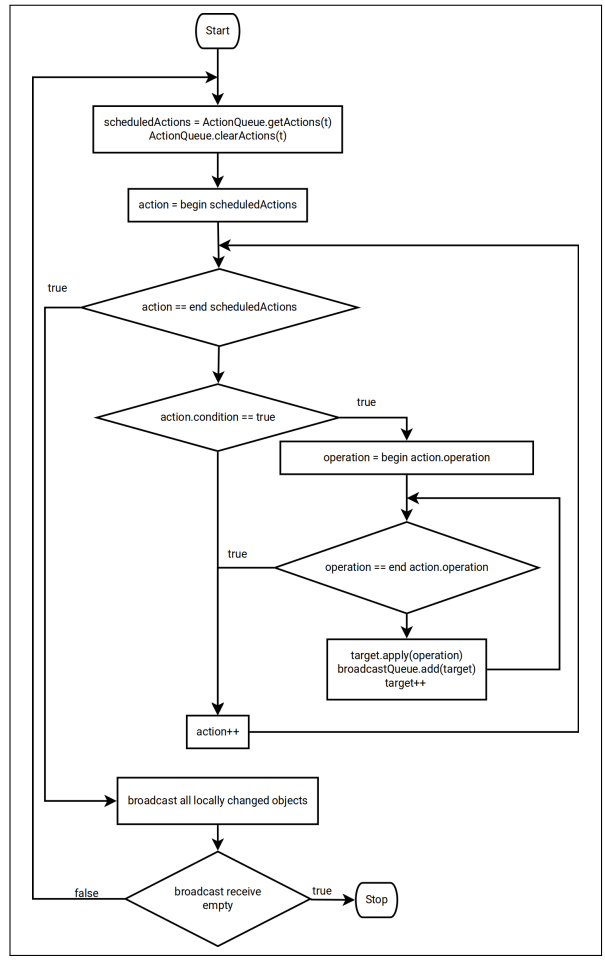


Fig. 17 EpiHiper action execution algorithm.

D EpiHiper scaling

Here we provide a detailed description of the scaling studies referenced in the Results section (2) of the main paper. First, computational experiments were conducted for eight disease models (I through VIII) across a set of interventions in the case of Virginia (US). Each experiment was conducted on compute nodes with dual CPUs having 20 cores each and 375 GB total memory across the eight disease replicates, each with its set of interventions, and each instance with 15 replicates. The details of each experiment is given by the following tables:

- Table 4: list of the eight diseases and their complexity elements;
- Table 5: the complete list of interventions used in the experiments along with characteristics impacting complexity;
- Table 6: the specific list of interventions used with experiments I through VIII.

The disease models listed in Table 4 are variations of the classical SEIR model with extra features. Their complexity can be represented by the number of states and the number of state transitions. Each model was calibrated so that we have a complete epidemic during the simulation for comparability.

Table 5 shows some common interventions implemented in EpiHiper for various studies. The column "Traits" refers to the usage of custom, time-varying attributes of nodes, whereas the column "Demographics" gives the number of fields accessed in the EpiHiper Postgres persontrait database. The demographic information is in these experiments only used during initialization, i.e, does not have any influence on the run time.

Running time as a function of disease and intervention complexity. The results (Figure 5 in the main paper and replicated here as Figure 18) show the results of the eight computational experiments (labeled I through VIII) relating intervention complexity and time complexity for combinations of interventions as listed in Table 5. Here the computation times are factored by the main simulation tasks (intervention, transmission, update, synchronization, output, and initialization), with the intervention details of each experiment as follows:

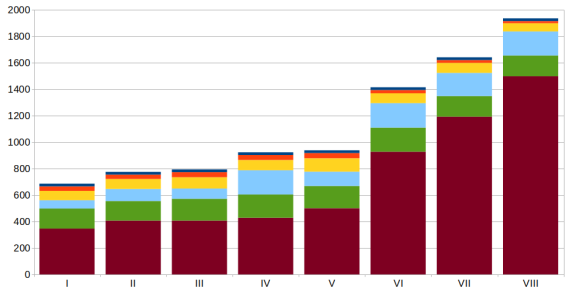


Fig. 18 Impact of intervention complexity on computation time for experiments I–VII (see text) factored by the main simulation tasks of intervention, transmission, update, synchronization, output, and initialization. The unit is seconds.

Running time as a function of network size (nodes/edges) Figure 6 of the main paper's Result section give the near linear scaling of running time as a function of the size of the network's node set and edge set in the case of one of our vaccine studies [9] covering all US states.

Memory requirement as a function of network size. In Figure 19 we have shown the near linear relation between the size of the network's vertex set and the corresponding memory usage for the vaccine study of [9].

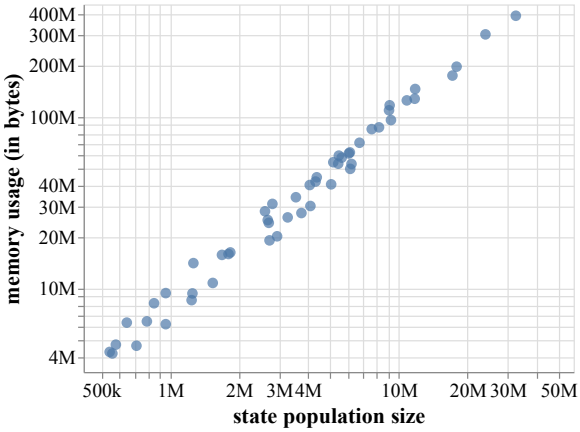


Fig. 19 Scaling of state size vs memory requirements for our vaccine study [9]

Performance and scaling with respect to computational resources. We performed code profiling to determine whether the chosen network partitioning strategy is feasible for different problem sizes. Figure 20 indicates that increasing the number of cores by a factor of 60 (from 2 for small

Table 4 EpiHiper disease models with different complexity beyond SEIR and their computation time.

No.	Disease	Features	Implementation	Complexity			time [s]
				States	Transmissions	Progressions	
I	Influenza	asymptomatic state	add state, transmission, and progression	5	2	4	140 ± 3
II	Ebola			8	3	8	142 ± 4
III	Measles	age stratified	states/transitions for each age group; transmissions across age groups	35	25	20	142 ± 1
IV	COVID v1	severe outcomes	add states and progressions	13	6	16	148 ± 5
V	COVID v2	v1 + age-dependent susceptibility and outcomes	states/transitions for each age group; transmissions across age groups	90	225	100	139 ± 3
VI	COVID v3	v2 + vaccines	vaccinated states with different transitions	105	300	120	146 ± 2
VII	COVID v4	v3 + multivariant	variant-specific infectious states	140	600	185	141 ± 5
VIII	COVID v5	v4 + immune waning/escape	transition from R to S; transmission across variants	170	975	250	140 ± 3

Table 5 EpiHiper interventions and factors influencing their computational complexity.

Intervention	Details	Node Sets	Edge Sets	Set Operations	Traits	Demographics
VHI	Voluntary home isolation	1	2	2	1	2
SC	School closure	0	1	0	1	2
SH, RO, PS	order, reverse, alternate stay at home order	1	1	1	1	2
TA	Test and isolation of asymptomatic cases	1	5	3	2	2
CTD1	Contact tracing distance 1	4	5	5	3	2
CTD2	Contact tracing distance 2	7	7	8	3	2

Table 6 The list of interventions used for each of the experiments I through VIII.

Experiment	Interventions
I	VHI, SC, SH
II	VHI, SC, SH, RO, TA
III	VHI, SC, SH, TA
IV	HI, SC, SH, RO, PS
V	VHI, SC, SH, RO
VI	VHI, SC, SH, RO, CTD1
VII	VHI, SC, SH, RO, CTD1, PS
VIII	VHI, SC, SH, RO, CTD2

US states (i.e., AK, DC, MT, ND, SD, VT, WY) to 120 for CA) only increases the synchronization by a factor of 8 (3.23%–25.2%), indicating that inter-processes communication scales approximately with $\sqrt{N}$ where N is the number of cores supporting the chosen partitioning strategy. The memory and run time requirements for different problem sizes scale linearly with the problem size. In fact we found that the memory requirements change minimally with varying disease model and intervention complexity. The scaling of the run time, even though always linear, is very much dependent on the complexity of both.

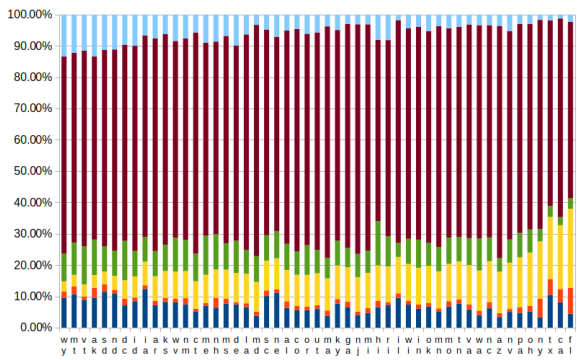


Fig. 20 Distribution of simulation tasks (■ update, ■ intervention, ■ transmission, ■ synchronization, ■ output, and ■ initialization) for Round 8 of the Scenario Modeling Hub scenario A (other scenarios lead to similar results). A large amount of time is spent on interventions, whereas the disease transmission uses less than 1/4 of it. The states are sorted by the time spent in synchronization.

E Additional Related Work

Covasim: The Covasim [18] project developed by Institute of Disease Management is perhaps the most comprehensive open source agent-based modeling environment that most closely relates to our work. The system is has been well documented[7] and provides an up to date information on extensions to the basic model, publications that use the model as well as information on the ongoing community wide effort. It is written in Python and largely meant for single CPU use. As a result issues related to parallelization are not a part of design consideration. It scales quite well for for populations of size up to a million nodes. This restricts the use of Covasim for US scale agent-based models. Various interventions are possible, but how they are generalized is not clear a priori. The disease models as described in the paper are somewhat limited but can presumably be extended.

Google and Open ABM Project. The Open ABM project[8] is an agent-based model to study COVID-19 pandemic. The model has a number of features but the documentation is relatively sparse. It appears to be developed for a single CPU with the core written in C. Various networks, disease models and interventions can be included like for Covasim, but how they are specified and how they can be generalized is not immediately obvious from the documentation. The Google ABM project is based on our Episimdemics project and Open ABM work. The development of the system seems to have stopped some time back.

COVID-19 ABM. This computational COVID-19 model by Galvani et al. [39] and used in for example [40], is an agent based simulator with code written for the Julia Language. Its scaling is somewhat limited focusing on populations of size $\leq 10,000$ and it uses a population structure similar to that of Covasim and OpenABM with network structures based contact data such as [35]. Their source is openly available[9].

FRED. The agent-based model of FRED [15] supports multiple, un-coupled (orthogonal) disease models. As for the models mentioned above, the basic transmission is done per place (location) using a uniform mixing model. The supported models appear to take the form of basic SEIR, SEIS, and SEIRS processes with opportunities for the user to specify dwell time distributions. As per their documentation there appears to be support for networks (page 48), but such data does not seem available and is not mentioned in [15]. Using their notion of places, the equivalent of a contact network is thus constructed by the epidemic simulator for each time step and for each location. The implementation of FRED as reported in [15] is a threaded C++ code using OpenMP. The simplifying assumption of uniform mixing within a location in the contact modeling, which may me adequate in many cases, allows one to omit explicit network representations, in turn permitting the reported scaling to populations of 30 million people. The open version of FRED is openly available[10], but we remark that the official and commercial version of FRED is now owned, maintained, and updated by the company Epistemix.

[7] Covasim: https://docs.idmod.org/projects/covasim/en/latest/index.html

[8] Open ABM: https://github.com/BDI-pathogens/OpenABM-Covid19

[9] COVID-19 ABM: https://github.com/affans/covid19abm.jl

[10] FRED: https://github.com/PublicHealthDynamicsLab/FRED

Covid-Sim. This is the simulation model used by Neil Ferguson and collaborators as reported in [21] with openly available source[11]. This is a threaded C++ software running on a single compute note with a collection of disease models and interventions hard-coded into the source.

[11] Covid-Sim: https://github.com/mrc-ide/covid-sim