Planning Epidemic Interventions with EpiPolicy

Zain Tariq
zain.tariq@nyu.edu
New York University Abu Dhabi
UAE

Miro Mannino
miro.mannino@nyu.edu
New York University Abu Dhabi
UAE

Mai Le Xuan Anh
alm818@nyu.edu
New York University Abu Dhabi
UAE

Whitney Bagge
bagge@ecohealthalliance.org
EcoHealth Alliance
USA

Azza Abouzied
azza@nyu.edu
New York University Abu Dhabi
UAE

Dennis Shasha
shasha@nyu.edu
New York University
USA

ABSTRACT

Model-driven policymaking for epidemic control is a challenging collaborative process. It begins when a team of public-health officials, epidemiologists, and economists construct a reasonably predictive disease model representative of the team's region of interest as a function of its unique socio-economic and demographic characteristics. As the team considers possible interventions such as school closures, social distancing, vaccination drives, etc., they need to simultaneously model each intervention's effect on disease spread and economic cost. The team then engages in an extensive *what-if* analysis process to determine a cost-effective policy: a schedule of when, where and how extensively each intervention should be applied. This policymaking process is often an iterative and laborious programming-intensive effort where parameters are introduced and refined, model and intervention behaviors are modified, and schedules changed. We have designed and developed EpiPolicy to support this effort.

EpiPolicy is a policy aid and epidemic simulation tool that supports the mathematical specification and simulation of disease and population models, the programmatic specification of interventions and the declarative construction of schedules. EpiPolicy's design supports a separation of concerns in the modeling process and enables capabilities such as the iterative and automatic exploration of intervention plans with Monte Carlo simulations to find a cost-effective one. We report expert feedback on EpiPolicy. In general, experts found EpiPolicy's capabilities powerful and transformative, when compared with their current practice.

CCS CONCEPTS

• **Computing methodologies → Modeling and simulation**; • **Human-centered computing → Interactive systems and tools**; **Visualization systems and tools**; **Activity centered design**.

KEYWORDS

Interactive Modeling, Epidemic Simulation, Policy Exploration

UIST '21, October 10–14, 2021, Virtual Event, USA
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8635-7/21/10…$15.00
https://doi.org/10.1145/3472749.3474794

ACM Reference Format:
Zain Tariq, Miro Mannino, Mai Le Xuan Anh, Whitney Bagge, Azza Abouzied, and Dennis Shasha. 2021. Planning Epidemic Interventions with EpiPolicy. In *The 34th Annual ACM Symposium on User Interface Software and Technology (UIST '21), October 10–14, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3472749.3474794

1 INTRODUCTION

"Tis the times' plague, when madmen lead the blind."
William Shakespeare

The COVID-19 pandemic has challenged policymakers across the world. Given an arsenal of interventions ranging from enforcing mask-wearing, remote learning, telecommuting, and social distancing to vaccination drives, policymakers have to design and communicate intervention plans that reduce disease burden while maintaining a healthy economy. Understanding the effects of these plans on disease spread and economic costs can be a complex modeling process often requiring different stakeholders such as public health officials, epidemiologists and economists working together to construct plans that impact millions of lives.

Current epidemic simulators make it difficult to engage in the exploratory what-if analysis required to determine the cost-benefit tradeoff of different possible intervention schedules. The reason for the difficulty is that the disease model, the intervention policy and the many parameters characterizing both are often entangled in code implementations that grow in complexity as an epidemic evolves. Eventually, minor changes in policy or in the parameters describing a disease become difficult to implement and test. Moreover, new diseases require new models, which are often constructed with little code transfer from previous models.

EpiPolicy is a novel epidemic-control policy planning and optimization tool. It provides high-level abstractions to support the specification of a disease model, a population's demographic and mobility patterns, a set of possible interventions, and schedules of interventions that embody different policies. EpiPolicy allows stakeholders to engage in what-if analysis where modifications to parameters or schedules do not require labor-intensive and error-prone code rewrites. It provides a Monte Carlo simulation-based optimizer that explores thousands of possible intervention schedules to suggest ones that reduce the overall health and economic burden.

This paper describes our design process and the design of EpiPolicy. We analyze the key tasks that a policymaking team engages in when mitigating an epidemic from disease modeling to intervention planning to what-if analysis and policy optimization (Section 2.1). We have designed EpiPolicy with an understanding of the core challenges of model-driven policymaking (Section 2.2). EpiPolicy supports the iterative and evolving process of disease modeling and policymaking. We report the feedback of experts on EpiPolicy for a variety of different use-cases and in a more focused COVID-19 case study (Section 4). Policymaking is complex and driven by many factors such as socio-political ones that go beyond what any tool can realistically model. The main contribution of EpiPolicy is to support the policymaking process, rather than directly dictating policy, by enabling effective intervention planning discussions that are not hampered by code complexity, hidden parameters, or slow turnaround times when exploring alternatives.

2 MOTIVATION & DESIGN PROCESS

The motivation for EpiPolicy came from our conversations with epidemiologists and public health policy-makers, including one of the authors. As these experts worked to contain epidemics like COVID-19, they often explored several existing disease simulation models, customizing them for the unique socio-economic characteristics of their administrative locales. They continuously updated their models and engaged in a time-consuming and laborious programming effort to integrate the effects of different interventions under consideration. As their code monoliths became complex, understanding and isolating the impact of different parameter choices, changes to interventions schedules, etc., were not trivial. Moreover, as groups of experts (epidemiologists, statisticians and modelers, health economists, etc.) collaborated in policymaking, it became difficult to fully understand how the models or interventions behaved without accessing multiple different email threads, spreadsheets and shared documents that kept track of information such as how certain parameters were derived, why a specific compartmental model was chosen, what exactly is the cumulative cost of an intervention, etc. As these details were often contained deep within code, it was difficult to surface such information without relying on secondary channels.

2.1 Task Analysis

Given the apparent need for a tool that supports policymaking, we worked with different experts to identify the main tasks that our tool should support.

- t1. model Specify a reasonably predictive model (i.e., compartment model) of how a disease spreads.
- t2. population Specify relevant baseline population characteristics and dynamics such as relevant demographic data (e.g. percentage of seniors, adults and children, blue-collar vs. white-collar workers, etc.), movement across administrative regions and interaction within different facilities (e.g., homes, workplaces, schools).
- t3. intervention Specify individual and *parameterizable* interventions in terms of cumulative *effects* on disease or population parameters as well as setup and running *costs*.
- t4. schedule Specify multiple intervention plans or *schedules*: where, when and how different interventions are applied.
- t5. what-if Evaluate and compare different schedules in terms of overall societal, health and economic costs (number of infections, deaths, hospitalizations, etc.) over time. This task underlies all forms of *what-if analysis* that policy-makers undertake when choosing which intervention schedule to implement.
- t6. optimize Search for alternative intervention schedules that are more cost-effective than current ones.
- t7. refine Refine models, population dynamics, interventions, and schedules as an epidemic evolves or as more accurate representations of the disease or interventions are found.
- t8. audit Audit models and policies: Policy-makers and epidemiologists need to trace model and intervention parameter settings to their sources, and to analyze the sensitivity of simulations on these parameters.
- t9. communicate Communicate model and intervention plan choices and simulation findings to different team members to aid collaborative policymaking.

2.2 Challenges

In a rapid-prototyping design phase, we initially experimented with existing tools such as the Spatiotemporal Epidemiological Modeler [18], and had conversations with epidemiologists and policy-planning teams to better understand the limitations of existing tools and to understand the challenges they face.

c1. mismatch **“All models are wrong,** but some are useful" is an aphorism that both haunts and inspires epidemiologists and policy-makers. George Box, to whom this aphorism is attributed to, explains “for a model there is no need to ask the question ‘Is the model true?’. If ‘truth’ is to be the ‘whole truth’ the answer must be ‘No’. The only question of interest is ‘Is the model illuminating and useful?’" [7] And so, with epidemic modeling for policy-planning, the concern is always how well a given model can inform a policy. Does it *parsimoniously* model the relevant phenomena?

There are largely two options for epidemic modeling. Deterministic models, which are described by well-behaved *ordinary differential equations* (ODEs), express the transition rate of a population through different disease compartments. For example, the textbook "SEIR" model uses four compartments with transitions: susceptible (S) to exposed (E) to infected (I) to recovered (R). More detailed models may introduce compartments such as vaccinated, quarantined, etc. to model how the population moves through these compartments at every time step of a simulation. Figure 2 illustrates a deterministic model for COVID-19 that reflects various severities of infection.

By contrast, stochastic agent-based models describe the behavior of multiple agents in a closed world with interactions between infected and susceptible agents carrying a certain probability of disease transmission. These stochastic models can be extremely expressive, but are difficult to scale (simulations are often limited small populations of 100,000 agents or fewer) [40], and may behave inconsistently requiring multiple simulation runs to derive robust predictions. Critically, at large population scales (1 million individuals or more), the model outputs of deterministic and stochastic models are often similar [3]. For these reasons, deterministic models are clear winners for modeling disease spread over large populations.

However, there is a **mismatch between how deterministic equations model a disease and how interventions are specified**. Consider an intervention such as social distancing. In an agent-based model, this is expressed easily by agents limiting their interactions with other agents not in their immediate household. To model this in a compartment model such as SEIR, we must explicitly model different "facilities" (households, schools, congregant settings and so on).

c2. parameters **Difficulty of parameterizing models & interventions.** With a novel disease, many key parameters are simply unknown. For example, the exact case fatality rate of COVID-19 is still unclear with some countries reporting rates lower than 1% and others reporting rates higher than 9% (Mexico) [12]. COVID-19's case fatality rate is the source of charged political and scientific debate [9, 23]. Overall, in policy-planning, one can have 10s-100s of parameters: parameters that describe the disease progression and spread, the population's demographic (e.g. % of seniors, % of essential workers, etc.), the population's mobility, or parameters that control an intervention (e.g. available treatment doses, hospital capacity), or describe a schedule (e.g. start and end dates of a lockdown). In our observations, a modeling team can spend hour-long meetings discussing parameter settings even for interventions where data is available. For example, how do we estimate hospital capacity? Do we include day-surgery wards? Private clinics?

So how are parameters set? "Where possible ... parameters are estimated based on experimental or observational data, and where parameter values have gone unestimated, they are often set to plausible values or ranges based on analogous systems, statistical inference or expert opinion" [52].

Many tools remain oblivious to the complex process of setting parameters. This is manifested as (a) parameters being set as constants within code [39] or (b) parameters being described in secondary spreadsheets and shared documents. A computationally expensive post-hoc *sensitivity analysis* is then used to characterize the response of a model to variations in parameter settings. This analysis serves to identify parameters that can be eliminated to yield simpler models, or to determine the range of possible outputs for an uncertain parameter, or most importantly to determine the *robustness* of a model's *qualitative* conclusions [38].

c3. no MMS **Lack of an Interactive Model Management System (MMS).** Policy-planning for epidemic control is an active and continuous process, where models, parameters and plans are constantly refined. Moreover, as policy-makers engage in extensive what-if analysis — *what if we build a field hospital? what if we increase the rate of vaccination? what if mask-wearing compliance drops? what if we ease border restrictions?* —, or explore different disease models — *what if the disease is more infectious than the current assumed rate, what if young adults are less likely to social distance?* — they generate multiple models, intervention plans and simulation results that they need to track and compare.

Often a policymaking team will work on multiple independent tools such as (i) a modeling tool, (ii) a spreadsheet tool to track parameter values and intervention plans as well as to perform some preliminary data analysis, (iii) a visualization dashboard, and (iv) a presentation tool to communicate results. The absence of a central repository that maintains the results of all simulations, and allows the easy refinement of existing models hinders the effectiveness of such teams: simple simulation look-ups require searching across multiple data sources and tools.

These delays have consequences. Liu and Heer observed that even a minuscule delay of 500ms in interactive data exploration incurred "significant costs, decreasing user activity and data set coverage" and that analysts "not only perform knowledge discovery at a higher rate under low latency conditions, their explorations are arguably more dynamic, engaging in sensemaking loops of observing, generalizing and hypothesizing [31]". In our setting, depending on a team's content management practices, looking up data such as an existing model's outputs or simulating a new scenario may take minutes to hours. Such long turnaround times hinder a team's ability to pose what-if questions or to qualitatively explore and evaluate the results of several simulated policy and disease scenarios.

3 THE DESIGN OF EPIPOLICY

We now describe the design of EpiPolicy that helps users with the tasks listed in Section 2.1, while tackling some of the challenges described in Section 2.2. We will first begin by describing a core set of design principles that influenced EpiPolicy's overall design and then demonstrate EpiPolicy's functionality with the help of a use-case scenario.

3.1 Design Principles

p1. abstraction **Providing high-level abstractions** In our discussion of challenge c1. mismatch we described how the implementation of disease interventions is at odds with how disease models are often described with standard, deterministic ODEs. Recent Lagrangian approaches aim to better capture population heterogeneity and mobility within deterministic models. We base our models on the recent multi-patch, multi-group epidemic modeling framework by Bichara and Iggdir [6] and *we surface three high-level abstract concepts* to simplify the process of defining a disease model: groups, locales and facilities.

(1) A **group** is used to model unique characteristics of a subpopulation. For example, one can use groups to define disease-specific parameters for super-spreaders or seniors or front-line workers.
(2) A **facility** is used to model unique environments that affect disease dynamics and how often and how long different population groups reside in these environments. For example, one can describe how disease spreads between individuals of a group in malls, places of worship, shared dormitories, or schools.
(3) A **locale** is used to model distinct administrative geographic regions that may have different policies and population distributions. For example, agricultural towns are distinct from urban centers; the former may have an older, less densely packed population than the latter.

EpiPolicy connects groups, facilities and locales by *mobility matrices* that describe how population groups move between facilities and locales.

These concepts help policy-makers express interventions as they define groups and environments on which different interventions directly apply. By defining a facility for schools, for example, the effect of an intervention such as school closure is easily specified as limiting the movement of children and adults to the school facilities. By defining groups such as pregnant women, one can model diseases such as Zika that affect specific population groups differently. These high-level concepts support our users with tasks t1. model, t2. population and t3. intervention ,while directly addressing challenge c1. mismatch .

p2. separation **Separation of Concerns.** A policymaking team has many tasks and concerns to tackle more or less simultaneously. To allow users to focus on one specific task and concern at a time, EpiPolicy provides separate pages for each task. For example, a disease model page allows users to focus only on describing the disease dynamics (the compartmental model) without worrying about population heterogeneity. A groups page allows the user to define different population groups and a parameters page allows users to define group-specific disease dynamics.

We extend the separation of concerns to how we specify interventions and plans. Each intervention has an `effect()` and a `cost()` function applied at each time step of a simulation when implemented. Interventions either increase or decrease capacities (e.g. building a field hospital) or influence model parameters (e.g. masks reduce transmission rates). All interventions that affect parameters do so through multiplicative factors. This allows users to specify interventions independently.

One can think of interventions as single moves in a game that a planner can choose to combine and deploy at different locations and times by constructing a *schedule*. This ***declarative*** nature of interventions is intimately intertwined with the separation of concerns design principle. First, by separating planning from intervention specification, we enable EpiPolicy to automatically search for cost-effective plans using Monte Carlo Tree Search (MCTS) [15]: users describe what they wish to achieve — low disease burden and low economic costs — and EpiPolicy determines how best to achieve the desired outcome. Second, we enable the division of labor or task delegation, which is common in collaborative teams [49]. For example, epidemiologists and health economists can focus on specifying the disease model and the inner-workings of different interventions, while policymaking officials can focus on constructing a politically acceptable schedule of interventions.

This design principle supports our users with tasks t1-t7 and addresses challenge c3. no MMS .

p3. API **Sufficient and Minimal APIs.** Our observations of epidemiologists showed a high degree of programming literacy and sophistication. EpiPolicy is not meant to be a drag-and-drop UI that hides all complexity. On the contrary, EpiPolicy is a framework that surfaces as cleanly as possible (i) complex disease modeling (ii) intervention design and (iii) planning choices. Expert teams can thereby clearly communicate their decisions, refine them and study their impact.

Specifically, EpiPolicy allows users to write in Python code how an intervention behaves through the `effect()` and `cost()` functions. Each intervention has a clearly separated set of control parameters that can be set as constants or as ranges that can be set by Monte Carlo simulation when searching for improved schedules or when conducting sensitivity analysis. For example, a workplace closure intervention can have the degree of closure as a control parameter with range 0 (no closure) to 1 (full closure) and EpiPolicy can choose the appropriate degree of closure over time to contain the epidemic. EpiPolicy's simulator calls the effect and cost functions at every simulation time-step. These functions may need to access or modify the values of parameters (the infection rate in a specific locale for a specific population group), the internal state of the simulation (e.g. the current number of infected or vaccinated individuals), or update running aggregates such as costs. We enforce a hierarchical naming convention for all parameter- and state-values and a high-level regular-expression language enables the access and appropriate modification of these values within each function (See Section 3.2.3).

Not everyone in an expert team may be capable of coding interventions. Our goal is to allow the users with programming expertise (e.g. computational epidemiologists) to construct these interventions in a fashion where they can (i) easily expose their parameters for other members of the policymaking team, (ii) separate their effects from their costs and (iii) methodically access and modify the internal state of the system (e.g. the disease transmission rate, the time spent by a group within certain facilities) at every simulation time step through a *minimal* simulator API of only four well-defined methods. These methods aim to improve code-readability, ensure correctness (i.e. ODEs remain continuous) and enable the concurrent execution of multiple interventions (See Section 3.2.3).

This principle supports users with tasks t3. intervention and t4. schedule and addresses challenge c2. parameters .

In addition to the three main principles discussed above, EpiPolicy ***visualizes*** the disease model, mobility patterns and intervention schedules and the results of one or more different intervention plans. These visualizations allow users to validate their specifications by eye and to interactively assess different policies.

EpiPolicy uses an open JSON specification format to interface with external data sources or third-party tools. For example, we developed plug-ins that allow EpiPolicy to use population data from NASA's gridded population of the world (GPW) data set [13], to utilize standard mobility matrix generating scripts, and to extract parameters from an agent-based population simulator that models human interaction patterns in schools and workplaces [53]. EpiPolicy also exports simulation results in JSON to allow users to integrate results in custom visualization dashboards as well or to enable further analysis beyond the current scope of EpiPolicy.

Finally and quite importantly, EpiPolicy provides *basic provenance*: every configurable object in EpiPolicy such as a parameter, an intervention or a schedule can be associated with a research paper, a script, a URL, or a free-form text description that describes how the object was configured.

These capabilities collectively make EpiPolicy a more complete model management system, hence addressing challenge `c3. no MMS`.

3.2 EpiPolicy in Action

Consider a team of epidemiologists and public-health officials that are working to curb the COVID-19 epidemic in United Provinces (UP)— a fictitious[1], economically developed country with three administrative regions, Hills, Beaches, and Pastures, and a population of 2.2 million. The team has to conduct several of the tasks listed in Section 2.1. EpiPolicy separates out each of the steps (sub-tasks) involved in policymaking into separate pages. Figure 1 illustrates the UI of EpiPolicy: the left menu lists a page for each sub-task. An accompanying video demonstrates the features of each page. Amira and Ben are members of the expert team who will illustrate the different features of EpiPolicy.

3.2.1 Compartmental Model. Amira first needs to specify her disease model in the Model page (Task `t1. model`). EpiPolicy provides a set of standard models obtained from existing literature (e.g. SEIR and SIR), which can be used as-is or modified as needed (Figure 3.1). Users can also create their own models from scratch by specifying compartments, equations, and parameters. Amira constructs an SEIR COVID-19 model with additional compartments modeling disease severity, hospitalization, vaccination and quarantine. She also introduces parameters for reinfection and vaccine failure. EpiPolicy graphically visualizes the model (Figure 2). For provenance (`t8. audit`), Amira can add a reference to the source (e.g. citation, link or generating script) of any parameter value in any page (Figure 3.1).

3.2.2 Population Modeling: Locales, Groups, and Facilities. We will now describe the sequence of pages that will help Amira model the population (Task `t2. population`). Amira moves to the Locales page to describe administrative locales relevant to the team's public-health policies. Amira can either provide custom shapefiles and regional population data in a `JSON` format or utilize a plug-in that loads this data directly from NASA's GPW data for a specific country (Figure 3.2). A country can have multiple administrate levels: `0` for the entire country, `1` for states or provinces within a country, `2` for sub-regions or counties within level `1` locales and so on. Here, Amira selects UP as the country, `2020` as the year, and chooses `1` as the administration level, as she wants to model the epidemic at the provincial-level.

Amira creates the following three groups in the Groups page: children (ages 0-19), adults (ages 20-49), and seniors (ages over 50). Amira can define any group if she knows its proportion of the population. A plug-in allows Amira to utilize demographic data available from NASA's GPW dataset to define age- and sex- based groups.

In the Facilities page, Amira defines four facilities, f. Members of different groups, g, move to these facilities throughout the day, reside within them for a certain proportion of their time, and interact with members of other groups (Figure 3.4). For example, children spend roughly a 1/3 of their time in schools interacting mostly with other children (> 90%) of the time. Amira provides two matrices (per locale if the distributions vary regionally): time spent by each group in each facility (fg) and for each facility the percentage of time spent interacting with other group members (fgg). EpiPolicy also provides a plug-in to extract this data from the agent-based human-interaction simulator: SynthPop [54].

Mobility Between Locales. On the Mobility page, Amira describes mobility patterns across land borders with mobility matrices (Figure 3.5). EpiPolicy provides several mobility algorithms to automatically construct these matrices from each locale's border perimeter and population size (e.g. the impedance algorithm [37]).

Tailoring Parameters to Subpopulations. In the Parameters page, Amira refines specific parameters in her model (Task `t7. refine`). Because older patients suffer more severe outcomes with COVID-19, she filters for disease severity and hospitalization parameters and increases their rates for seniors across all locales (Figure 3.3).

3.2.3 Specifying Interventions and Costs. In EpiPolicy, an intervention is described by two functions: `effect()` and `cost()`. Before we describe how these functions are specified, it is important to understand how EpiPolicy maintains information about (i) the current state and (ii) the parameters that modify the ordinary differential equations that model disease spread across the different population groups as they reside within facilities and move across locales.

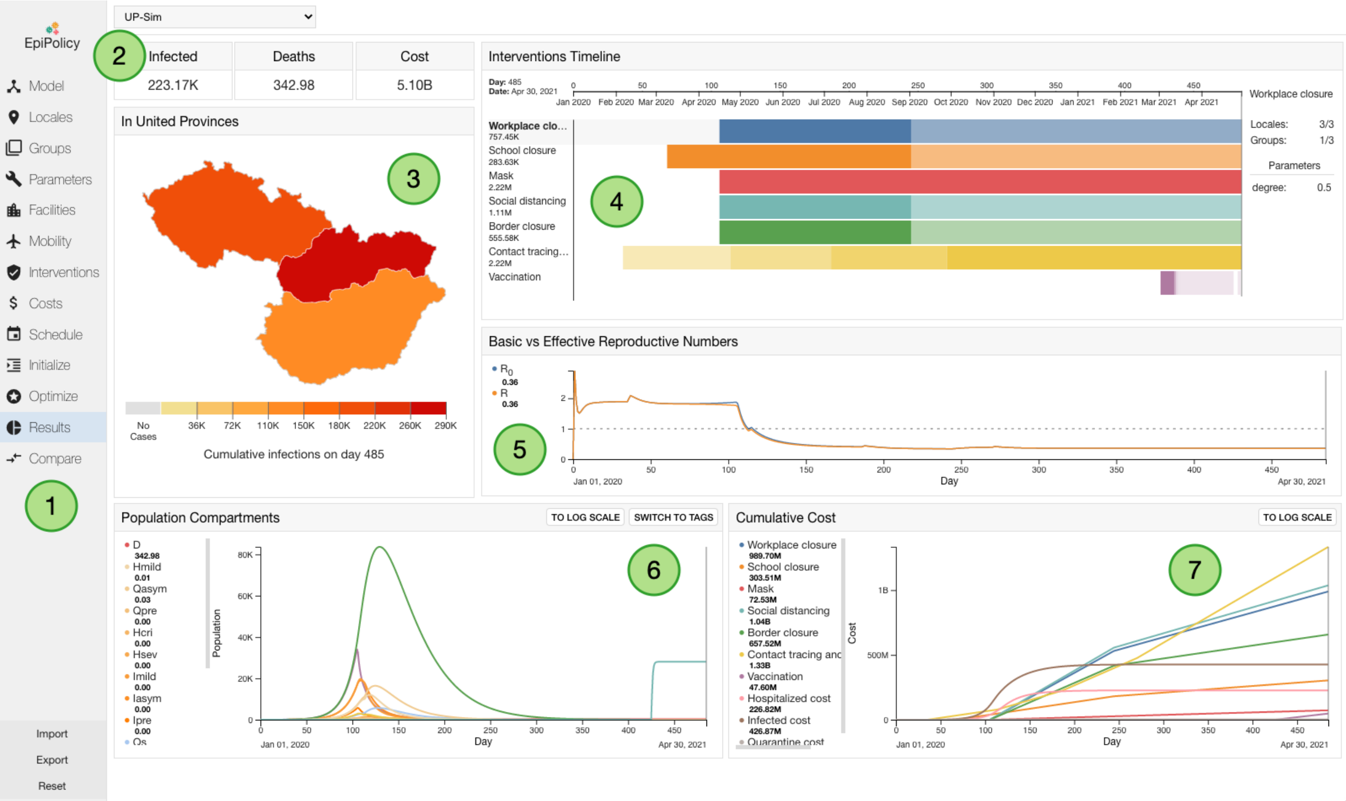
EpiPolicy maintains the following five matrices:

(1) A *state matrix* ($S[c, l, g]$) maintains the number of individuals within compartment c, locale l and group g.
(2) A *mobility matrix* ($M[g, l_1, l_2]$) describes the percentage of time an individual of population group g from locale l_1 spends in locale l_2 due to mobility.
(3) A *facility matrix* ($F[l, f, g]$) describes the percentage of time an individual of group g spends within a facility f in locale l.
(4) A *contact matrix* ($C[l, f, g_1, g_2]$) describes the percentage of time an individual of group g_1 interacts with an individual of group g_2 within a facility f in a locale l in such a way to allow disease transmission.
(5) A *parameter matrix* ($P[p, l, f, g]$) describes the disease parameter values (e.g. incubation rate, recovery rate) for a certain population group g within a specific locale l and facility f.

These matrices are used by the simulator to update the state matrix at every time step (by default, a single day) of the simulation. For example, to update the number of exposed individuals ($c = \mathtt{E}$) of a group g at locale l for the disease model described in Figure 3, the simulator executes the ODE $d\mathtt{E}/dt = \mathtt{beta} \times \mathtt{I} \times \mathtt{S} \ / \ \mathtt{N} - \mathtt{sigma} \times \mathtt{E}$ with the (i) group-locale-specific parameter values of `beta` (transmission rate) and `sigma` (incubation rate) extracted from the parameter matrix P, (ii) and an estimate of the infectious population in contact with the group (`I`) computed from the mobility M, facility F, and contact C matrix.

Different pages of the UI (Figure 3) expose different views of these matrices to allow users to adjust their base values. EpiPolicy also provides a programmatic API, **a selector**, to access and modify the values in these matrices: `sim.select(S)`. The selector can also

[1] The goal of this paper is to motivate and demonstrate the use of EpiPolicy, not to present or analyze the specific modeling choices or measures of any government in handling an epidemic.



UP-Sim

Infected	Deaths	Cost
223.17K	342.98	5.10B

Workplace closure

Locales: 3/3

Groups: 1/3

Parameters

degree: 0.5

Day: 485
Date: Apr 30, 2021

Figure 1: User interface of EpiPolicy. On the left is (1) a menu panel to navigate to different pages — each page supports a specific task. The Results **page visualizes the results of a simulation to help users better understand and communicate the impact of a specific policy. The page shows: (2) a table of total infections, deaths, and total economic cost, (3) a color-coded map illustrating cumulative infections per locale , (4) a schedule of interventions, (5) a chart of reproduction numbers over time, (6) standard compartment time-series, and (7) cumulative cost line charts.**

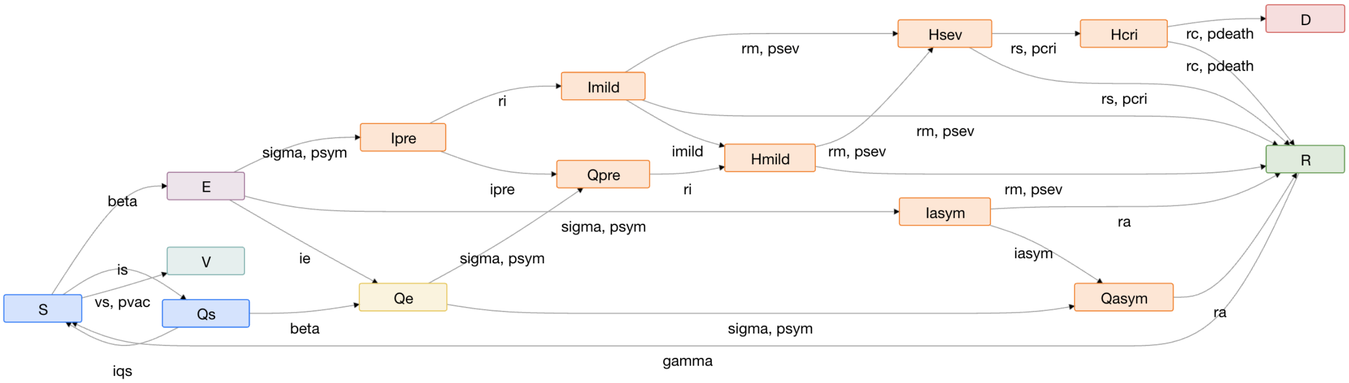
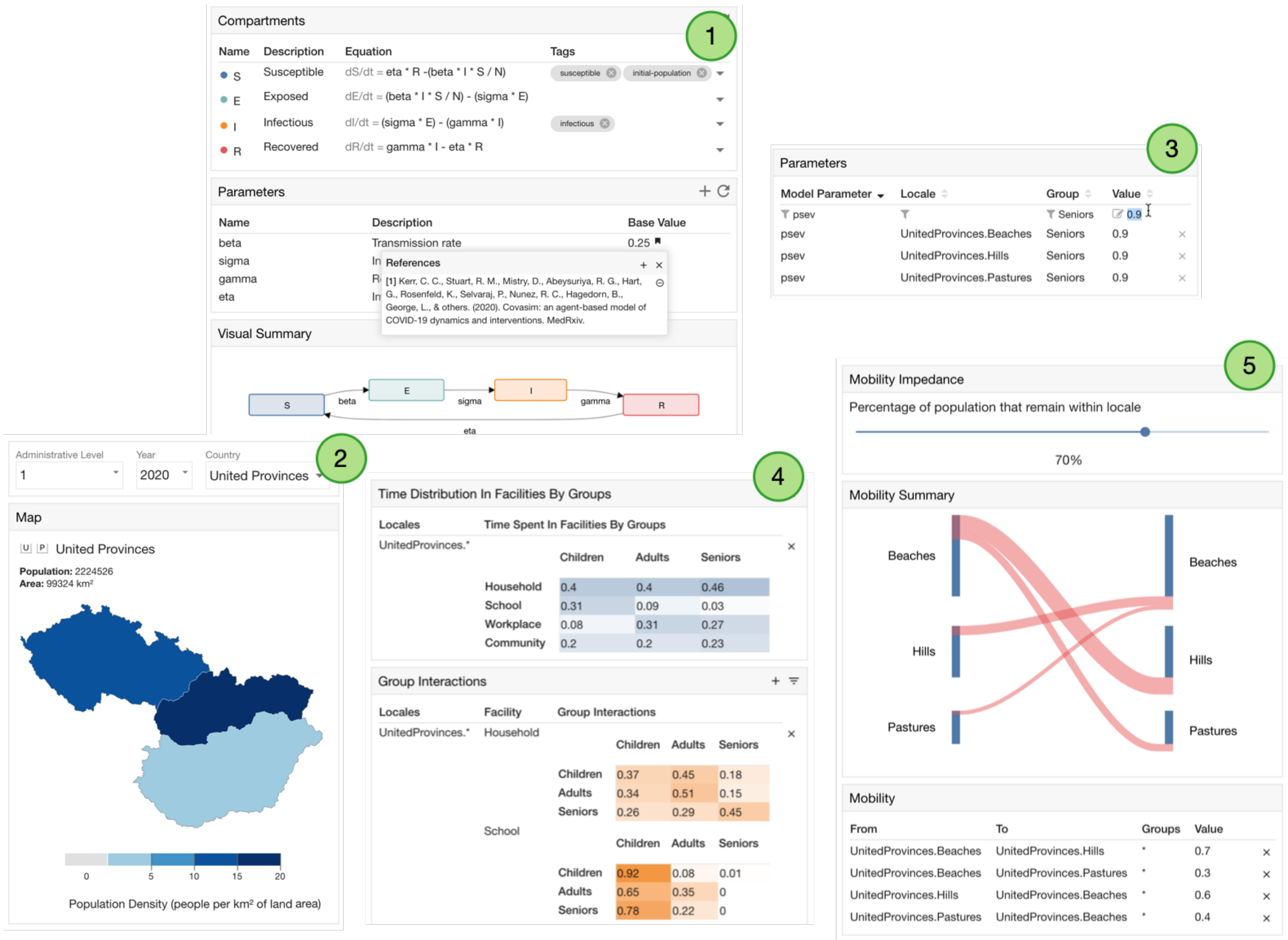


Figure 2: A COVID-19 compartmental model with nodes representing compartments and edges annotated with parameters influencing the transition ODEs. EpiPolicy automatically generates this visualization from the ODEs that describe each compartment's dynamics.

access other objects such as the state of a schedule at a given time point, or an intervention's running cost using simple dictionaries and regular-expressions. The design of the selector is influenced by tabular query languages such as SQL and Python pandas. This is a natural API design choice, because the matrices are stored as multi-dimensional tables and a selector simply returns a view from one of these tables.



Compartments

Name	Description	Equation	Tags
● S	Susceptible	dS/dt = eta * R -(beta * I * S / N)	susceptible ⊗ initial-population ⊗
● E	Exposed	dE/dt = (beta * I * S / N) - (sigma * E)	
● I	Infectious	dI/dt = (sigma * E) - (gamma * I)	infectious ⊗
● R	Recovered	dR/dt = gamma * I - eta * R	

Parameters

Name	Description	Base Value
beta	Transmission rate	0.25
sigma	In	
gamma	R	
eta	In	

References

[1] Kerr, C. C., Stuart, R. M., Mistry, D., Abeysuriya, R. G., Hart, G., Rosenfeld, K., Selvaraj, P., Nunez, R. C., Hagedorn, B., George, L., & others. (2020). Covasim: an agent-based model of COVID-19 dynamics and interventions. MedRxiv.

Visual Summary

Administrative Level: 1
Year: 2020
Country: United Provinces

Map

United Provinces

Population: 2224526
Area: 99324 km²

Population Density (people per km² of land area)

Parameters

Model Parameter	Locale	Group	Value
psev		Seniors	0.9
psev	UnitedProvinces.Beaches	Seniors	0.9
psev	UnitedProvinces.Hills	Seniors	0.9
psev	UnitedProvinces.Pastures	Seniors	0.9

Time Distribution In Facilities By Groups

Locales: UnitedProvinces.*

Time Spent In Facilities By Groups

	Children	Adults	Seniors
Household	0.4	0.4	0.46
School	0.31	0.09	0.03
Workplace	0.08	0.31	0.27
Community	0.2	0.2	0.23

Group Interactions

Locales: UnitedProvinces.*

Facility: Household

	Children	Adults	Seniors
Children	0.37	0.45	0.18
Adults	0.34	0.51	0.15
Seniors	0.26	0.29	0.45

Facility: School

	Children	Adults	Seniors
Children	0.92	0.08	0.01
Adults	0.65	0.35	0
Seniors	0.78	0.22	0

Mobility Impedance

Percentage of population that remain within locale

70%

Mobility Summary

Mobility

From	To	Groups	Value
UnitedProvinces.Beaches	UnitedProvinces.Hills	*	0.7
UnitedProvinces.Beaches	UnitedProvinces.Pastures	*	0.3
UnitedProvinces.Hills	UnitedProvinces.Beaches	*	0.6
UnitedProvinces.Pastures	UnitedProvinces.Beaches	*	0.4

Figure 3: UI snippets from EpiPolicy. (1) Specifying the disease model with compartments and ODEs in the Model **page. (2) Loading population data and administrative locales in the** Locales **page. *Note United Provinces (UP) is a fictitious country used only for illustrative purposes.* (3) Filtering and modifying disease parameters for a specific group within some locales in the** Parameters **page. (4) Specifying time spent by groups in different facilities and interacting with other groups in the** Facilities **page. (5) Using the impedance algorithm to generate border mobility patterns for UP in the** Mobility **page.**

For example, to select the transmission rate (`'beta'`) for children in the Hills locale within the school facility, we simply make the following call to EpiPolicy's simulator API:
`sim.select({'parameter':'beta', 'locale': 'UP.Hills', 'facility':'School', 'group':'Child'})`

To select the number of infected individuals across the different groups and locales, we make the following call:
`sim.select({'compartment':'I', 'locale': '*', 'group': '*'})`
This call returns a *table* — a pandas data frame — with the number of individuals for each group and each locale. Regular expressions provide richer expressive power when constructing selectors. For example, we can select compartments outside a specific locale using the following regular expression: `'locale':'~UP.Hills'`. Syntactic shortcuts allow users to construct more concise expressions: e.g. `sim.select({'compartment':'I'})` is equivalent to the expression above. The use of data frames allows users to formulate more complex operations over the resulting tables. For example, `sim.select({'compartment':'I'})['Value'].sum()` finds the total number of infected individuals.

We now describe how to specify the effect and cost of an intervention by using selectors and selection-expressions.

Intervention Effect. An intervention ultimately affects how a population group transitions through the different compartments. To do so, an intervention's `effect()` function manipulates some of the values in one of EpiPolicy's matrices using one of the following two methods:

- `apply(S, f)` *multiplies* a given factor `f` onto the value selected by the selection expression `S`. For example, in Listing 1, `sim.apply({'parameter':'beta'}, 1-f*c)` reduces the transmission rate by a factor of `f*c`.

- `move(S1, S2, n)` *transfers* `n` individuals from a source compartment `S1` to a destination one `S2`, each defined by a selection expression while ensuring the ODEs remain continuous. For example, in Listing 2, `sim.move({'compartment': 'S', 'group':'Adult'}, {'compartment': 'V', 'group':'Adult'}, n)` moves `n` adult individuals from the susceptible to the vaccinated compartment *proportionally* across all locales.

These two methods are restrictive but powerful. With `apply(S, f)`, we allow users to change parameter values using multiplicative factors. This enables the straightforward integration of multiple intervention effects that target the same parameter. For example, if mask-wearing reduces transmission rates by a factor of 0.8 and increasing hand-disinfection stations independently reduces transmission rates by a factor of 0.5, then collectively, they reduce transmission rates by a factor of 0.5×0.8. With `move(S1, S2, n)`, we capture a variety of interventions that have *hard capacity limits*. For example, vaccination interventions where only a fixed number of doses can be distributed daily, or constructing field hospitals that have a fixed number of beds. These interventions directly move individuals from one population compartment to another without modifying disease characteristics, mobility or group interaction patterns.

EpiPolicy separates out the control parameters of an intervention (e.g. the degree of compliance, the efficacy of a vaccine, number of available doses, percentage of people allowed in workplaces or schools, etc.) to allow users (i) to easily modify them without changing the code when constructing schedules (See Section 3.2.4), or (ii) to provide discretized ranges for each parameter to enable Monte Carlo Tree Search to search for optimal settings (See Section 3.2.6) or to conduct sensitivity analysis.

Intervention Cost & Other Costs. EpiPolicy's programmatic API also provides the following method to enable the specification of intervention costs:

- `add(S, c)` *adds* a given amount `c` to a running cost aggregate defined by a selection-expression `S`. For example, in Listing 2, `sim.add({'intervention' : 'vaccination'}, c*doses)` adds the daily cost of administering a fixed number of doses to the running costs of the vaccination intervention.

We allow users to specify costs with the help of a Python function rather than through simple constants to capture more complex but realistic cost models. For example, an intervention that increases hospitalization capacity by constructing new field hospitals has (i) an initial construction cost, (ii) a daily running cost and (iii) a tear-down cost. Certain intervention costs may also depend on the internal state of the simulation such as the number of individuals across different compartments (e.g. Listing 3).

In listings 1, 2 and 3, we see how Ben describes the effect and costs of a mask-wearing intervention and vaccinations as well as the costs of quarantining individuals using EpiPolicy's API. Ben also describes several other interventions including school and workplace closures, social distancing, border closure, and contact tracing and testing (Task t3. intervention). He also describes other disease burden costs such as the costs from the loss of life and the economic toll from reduced work productivity due to sick days or work absences.

Why a Python API?. A programmatic API empowers users to describe any intervention that can modify the simulation state beyond those that one can create from a set of programming-free, form-based widgets. A Python API is a natural choice for EpiPolicy as its simulator and MCTS optimizer are developed in Python. In our discussions with computational epidemiologists we asked about the programming languages they were comfortable with and while R was the most commonly used programming language for epidemic modelling, they also expressed a familiarity with Python and its syntax.

3.2.4 Intervention planning with schedules. In the Schedule page, Ben can specify *when*, *where* and *how* to apply an intervention (Task t4. schedule). To specify *when*, Ben can either provide multiple exact starting and stopping dates or a periodic schedule. Alternatively, Ben can provide a conditional start and stop with the help of triggers: predicate functions that start or end an intervention if certain conditions are met (e.g. infections rise above or drop below a given threshold). To specify *where*, Ben can describe for each intervention time interval, which locales to selectively apply the intervention to with the help of selection-expressions (e.g. `{'locale' : 'UP.*'}` applies an intervention over all the provinces in UP). To specify *how*, Ben can provide explicit values for each intervention's control parameters. As Ben constructs the schedule, EpiPolicy visualizes it with a Gantt chart that color codes each intervention and uses opacity to encode the intensity with which it is applied (Figure 4).

3.2.5 Viewing results. Amira wishes to view the results of Ben's schedule and to compare it to one that she had previously constructed (Tasks t5. what-if, t7. refine). In the Initialize page, she defines starting simulation conditions such as the number of infected individuals at different locales. After executing the simulations, she can view the outcomes of each schedule independently in the Results page. The page shows information that experts may require to assess the efficacy of the policy under consideration (Task t9. communicate): (a) total infections, deaths, and cost provide an overview of the cumulative impact of the epidemic, (b) a map color-coded by cumulative infections per locale illustrates the spread of infection across the country, allowing users to easily identify hard-hit areas, (c) a schedule of interventions provides a quick overview of the policy, (d) a chart of reproduction numbers (basic R_0 and effective R) provides a rough indication on whether the outbreak will die out ($R_0 < 1$) or persist ($R_0 > 1$) or whether herd immunity has been attained ($R < 1$), (e) standard compartment time-series (e.g. daily total number of infected individuals) and (f) cumulative 'dollar' cost graphs for the disease and the policy including the cost of each intervention over time. All these charts are linked: selecting a locale in the map focuses the results to only that locale, moving a guideline across the schedules or any of the other line charts provides details specific to that day in all the other charts.

Amira can also use the Compare page to compare two or more simulations (Tasks t5. what-if, t9. communicate). This page contains fewer visualizations than the more detailed Results page. We selected the following visualizations for easier policy comparisons:

```
def effect(control_parameters):
    c = control_parameters['compliance']
    f = control_parameters['factor_reduction']
    sim.apply({'parameter':'beta'}, 1-f*c)
def cost(control_parameters):
    price = control_parameters['daily_mask_pp_cost']
    c = control_parameters['compliance']
    n =  sim.select({'compartment':'~D', 'locale':'UP.*', 'group':'*'})['Value'].sum() #the total living population size
    sim.add({'intervention':'mask'}, price*c*n)
```

Listing 1: A mask wearing intervention applied to all provinces in UP. `effect()` **reduces the global beta infection transmission rate by a multiplicative factor.** `cost()` **estimates the total cost of wearing masks per day when only a fraction of the total population complies.**

```
def effect(control_parameters):
    doses = control_parameters['daily_doses_administered']
    eff = control_parameters['efficacy']
    ratio = control_parameters['seniors_to_adults_ratio']
    sim.move(
        {'compartment': 'S', 'group': 'Adult'}, {'compartment': 'V', 'group': 'Adult'}, doses * eff * (1 - ratio))
    sim.move(
        {'compartment': 'S', 'group': 'Senior'}, {'compartment': 'V', 'group': 'Senior'}, doses * eff * ratio)
def cost(control_parameters):
    doses = control_parameters['daily_doses_administered']
    c = control_parameters['vaccine_cost_pd']
    sim.add({'intervention' : 'vaccination'}, c*doses)
```

Listing 2: A vaccination intervention. `effect()` **moves a fixed number of individuals (equal to the number of administered doses) from the** `susceptible` **compartment to the** `vaccinated` **compartment. Only seniors and adults can be vaccinated. The '**`seniors_to_adults_ratio`**' parameter controls the distribution of the available doses across adults and seniors.** `cost()` **adds the cost of the total number of doses administered per day to the running cost of the intervention.**

```
def cost(control_parameters):
    c = control_parameters['daily_quarantine_cost_pp']
    q = sim.select({'compartment': '^Q'})['Value'].sum()
    sim.add({'running-costs' : 'quarantine'}, c * q)
```

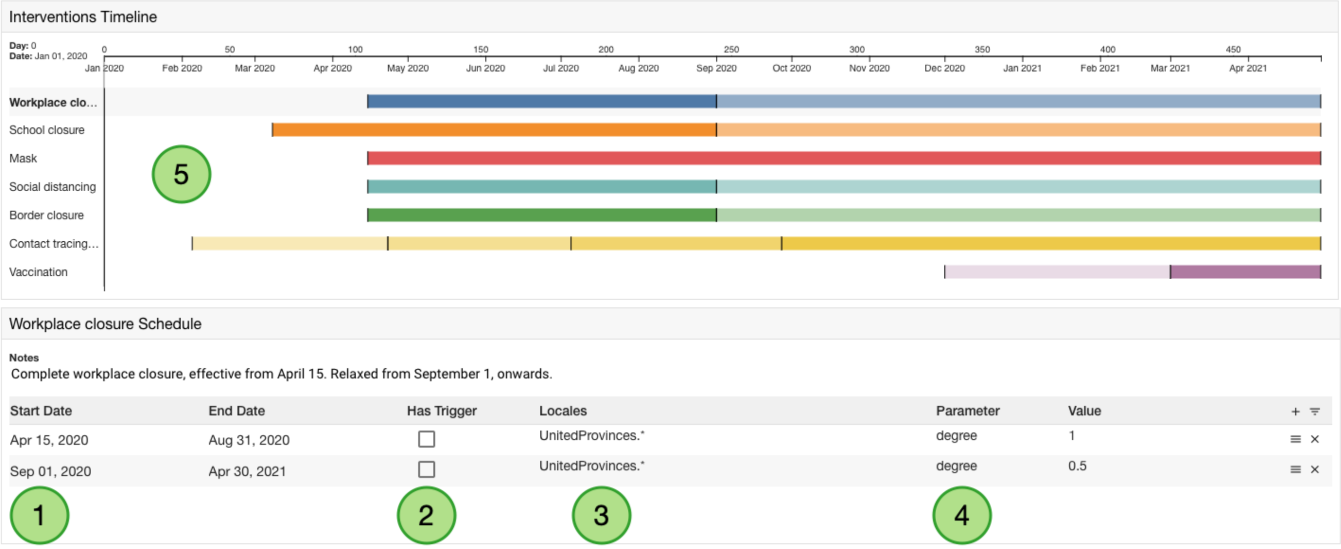
Listing 3: The daily `cost()` **of quarantining individuals during an outbreak. All compartments that start with 'Q' are quarantine compartments. Calling** `select` **with a regular expression compartment selector** `^Q` **returns the total number of quarantined individuals.**

(a) table of total infections, deaths and cumulative costs per policy, (b) a Gantt chart for each schedule that visually illustrates key policy differences, and (c) a line chart of each policy's reproduction numbers that can help determine which policies are more effective at containing the outbreak in a shorter time-frame.

3.2.6 *Automatic Exploration of Policies.*

Specifying intervention schedules is often a trial-and-error process as users must adjust multiple intervention parameters (when, where and how) in order to find an intervention plan that minimizes overall disease burden and economic cost. To illustrate the complexity of this problem: with n interventions, l locales, and a planning timeline of t days: there are n^{l^t} possible schedules to consider! This is excluding the possible options for interventions with free control parameters (e.g. the degree of workplace closures). In the Optimize page, Ben can rely on EpiPolicy's Monte Carlo Tree Search (MCTS) algorithm to search for a cost-effective policy (Task t6. optimize). MCTS will generate several thousands of plans, simulate each one, and suggest the most cost-effective one it has found.

For a primer on MCTS and its effectiveness in solving planning problems, we refer the reader to references [14, 15]. In EpiPolicy, each MCTS execution recommends an *action*: a set of interventions and their parameterizations for a single day given the day's *state*.



Workplace closure Schedule

Notes
Complete workplace closure, effective from April 15. Relaxed from September 1, onwards.

Start Date	End Date	Has Trigger	Locales	Parameter	Value
Apr 15, 2020	Aug 31, 2020	□	UnitedProvinces.*	degree	1
Sep 01, 2020	Apr 30, 2021	□	UnitedProvinces.*	degree	0.5

Figure 4: Schedule Specification. Users can specify *when* an intervention is applied with (1) fixed start and end dates, or (2) trigger functions that start or end an intervention when certain conditions are satisfied; (3) *where* it is applied; and (3) *how* by providing explicit values for its control parameters. Here, full work place closures (degree of closure = 1) are enforced from mid April to end of August 2020 in all provinces, and relaxed (degree of closure = 0.5) in all provinces from Sept 2020 to end of April 2021.

The state includes the distribution of the different population groups across locales and the different disease compartments. To construct a plan for an entire year, MCTS is executed 365 times, once for each day, where the state of day t is determined by applying the action or the interventions selected by the previous MCTS execution on day $t-1$.

Each MCTS execution consists of several thousand iterations of four phases that construct a *tree*: selection, expansion, simulation and backup. At the root of the tree is the node representing the current day's state. Each iteration begins at the root and traverses the tree by selecting a child node (the next day's state) as a consequence of applying an *action* — here a specific set of interventions along with their parameterizations — or expands the tree by exploring a new action. The choice between exploiting a promising past action or exploring a new action is determined by the standard Upper Confidence Bound (UCT) algorithm [26]. EpiPolicy's MCTS tree is only 30 nodes deep. On reaching a leaf node, a random playout of actions for each of the remaining 335 days is generated. The cumulative cost at the end of the 365-day time-horizon, which includes both the economic costs of the interventions as well as the overall costs of the disease, is back-propagated up the tree from the leaf node. Each node averages the costs of the trajectories initiating at its children to estimate the *expected return cost* for each action. After 10,000 or a configurable number of iterations, MCTS chooses the action from the root node with the lowest cost normalized by how often that action was traversed in the MCTS execution; this normalization avoids choosing outliers. One can configure MCTS to run more or fewer iterations: the greater the number of iterations, the larger the space of plans that MCTS visits leading to the potential discovery of a better plan. Given the computationally intensive nature of MCTS, we perform several optimizations to ensure reasonable performance including parallelization, coarse-discretizations of the parameter space, heuristically pruning out infeasible plans, and using just-in-time compilation to ensure the Python simulator can run a full-year simulation in less than 3 seconds.

MCTS runs asynchronously and once complete, its chosen schedule can be loaded into the UI, and its results viewed or compared with other hand-crafted schedules. Despite its effectiveness in solving planning problems, MCTS can only explore a small fraction of possible plans and it requires coarse discretizations of the state and action space. It is also difficult to transfer *policies* — the probability distribituion of actions given states — learned from one MCTS execution to the next. Finally, the current implementation does not recommend a set of alternative and equally cost-effective plans, only the best one found so far. We are currently exploring alternative reinforcement learning techniques that avoid some of these limitations.

3.2.7 Further analysis. EpiPolicy's UI, a client, generates a JSON specification to describe an entire simulation: the disease and population models, the interventions, and optionally the schedule. This is fed to EpiPolicy's simulator, a server, which also returns its results as a JSON file that is visualized by the client's dashboards. The client acts as a model management system keeping track of all simulations and MCTS executions. The division of work across the client and server and the materialization of inputs and outputs as open-format JSON files allows the integration of many third-party tools and custom external scripts that perform a variety of more specialized analysis using EpiPolicy's simulator such as parameter sensitivity analysis and intervention impact analysis (Task t8. audit). In Section 4, we describe how one group conducted impact analysis with EpiPolicy.

4 QUALITATIVE FEEDBACK

We gathered feedback on EpiPolicy from experts in two ways: (a) informal interviews with public-health officials (after demonstrating EpiPolicy) to understand what diseases and interventions they worked with and how they modeled them; we then conducted small-scale feasibility studies where we modeled their use-cases on EpiPolicy, and (b) a hands-on multi-day collaboration with an expert team of public-health policy-makers, where EpiPolicy was used to analyze the impact of different policies in reducing infections and deaths from the COVID-19 pandemic.

Participants. Participants (30-70 years old) in our feasibility studies and focus group included several researchers at a university, a public-policy consultancy, EcoHealth Alliance, and six public-health professionals (2 females, 4 males) working within the health department of a government. All participants had higher-education degrees including MDs, PhDs or Masters degrees in Statistical Modeling, Epidemiology, Mathematics or Economics, or several years of professional experience in the public health sector and infectious disease control. Of the participants who have experience with the computational modeling of epidemics, all had programming experience with either R, Python or Matlab, and all were familiar with Python.

Feasibility Studies. From our discussions with different stakeholders implementing the following scenarios with EpiPolicy was straightforward with direct mappings between the concepts in the use-cases and in EpiPolicy.

(1) *Hospital Capacity Building.* With EpiPolicy, one can define interventions that move a discrete number of individuals from one compartment to another. This allows us to construct fixed capacity compartments for hospitalized patients. Non-hospitalized patients transition through other compartments with higher likelihood of worse outcomes. A build-hospital intervention increases capacity at some cost as well as incurs running cost of the increased capacity through time. Using MCTS, EpiPolicy suggests increasing hospital capacity by building field hospitals in anticipation of infection peaks and decreasing hospital capacity when infections drop to baseline hospital capacities to balance the economic costs of building hospitals with the disease costs associated with mortality and infection. We ran 10,000 iterations for each MCTS execution with a total of 8 hours of computation time on a 64-core machine for this scenario.

(2) *Vaccination Portfolio Selection.* One can define multiple interventions: one for each vaccine available and provide control parameters on its efficacy, availability and cost. Using MCTS, EpiPolicy can propose a schedule and distribution of vaccinations across groups that reduces overall disease burden and economic cost. We ran 20,000 iterations for each MCTS execution with a total of 16 hours of computation on a 64-core machine. For this scenario's disease, population demographics, administrative locales, and vaccination availability, MCTS produces a plan that immediately utilizes all available vaccinations and prioritizes high-risk seniors over adults. This plan reduces the overall mortality in seniors by 80% but also slightly increases cumulative infections in adults by 10% when compared to an indiscriminate vaccination plan. The overall costs of both plans are similar (within a 2% difference margin).

(3) *Modeling Distinct Work forces.* Different groups of a population face higher risks of disease transmission. In the US for example, COVID-19 spread rapidly among workers in the meat-packing industry [43]. In Singapore, blue-collar workers living in shared dormitories had greater risk of exposure [42]. Using groups and facilities, it is straightforward to model these distinct work forces and construct targeted interventions.

(4) *Modeling Hosts and Carriers.* Diseases like Rift Valley Fever affect cattle (hosts) but are transmitted by mosquitoes (carriers) at water bodies. With EpiPolicy, one can model hosts and carriers as separate groups that interact at water bodies (a facility). Interventions like draining pans (small pools that form after rain), or vaccinating a herd are straightforward to model in EpiPolicy, demonstrating its use beyond human-transmittable diseases.

The feasibility studies served mainly as a proof of principle that EpiPolicy can capture different disease models and intervention scenarios.

Focus Group Observations. In this multi-day collaboration, we created an alternative model to an existing one that was built and maintained by our partners over several months. While the models were slightly different, they were both deterministic compartmental models. The model included compartments for vaccinated and quarantined individuals and allowed for re-infection and death. It used NASA's GPW dataset to construct age-based population groups and used impedance to describe border mobility. It had four facilities: household, workplace, school and community places. We implemented the following interventions: workplace and school closures, border travel restrictions, mass-screening and contact tracing, surface-sterilization, social distancing, an intervention for every available vaccine within the administrative regions of this team, and lockdowns.

First, the team wanted to describe as closely as possible a schedule of interventions similar to one that had been implemented in their administrative regions. We observed the team as they collaborated on Zoom with EpiPolicy. The team members grasped with minimal effort the details of the model and the interventions. The different visualizations in EpiPolicy enabled this. Three members of the team — those with programming or computational epidemic modeling expertise — then participated in modifying the coded interventions over Zoom. They dictated code modifications and stated that the effect and cost functions were easy to construct and modify once they understood how interventions worked and examined a few code examples of interventions similar to those in listings 1 and 2. The team's ability to discuss and iteratively refine the interventions reflects the importance of our third design principle (p3. API)— keep the API of interventions as minimal as possible with only two functions, effect and cost, and provide a sufficient programmatic API that allows users to cleanly access the matrices that determine the progression of an epidemic and the aggregate (mobility or interaction) behavior of population groups within interventions. The members debated the exact schedule and how to better implement or parameterize certain interventions: *Does mask wearing really reduce transmission rate by this much? Are we exaggerating its effect here: children under 3 years old don't wear masks, people don't wear masks within households.* Each such discussion resulted in a redesign of an intervention or modifying its parameters or schedule (start or end date). These modifications were done during the meeting reflecting the ease with which EpiPolicy supports such refinements. Thus, EpiPolicy elevated the discussion from code specifics to aspects relevant to disease modeling and policymaking as was intended by our second design principle of separating concerns (p2. separation).

Second, the team wanted to conduct a post-hoc impact analysis of each intervention: *what-if we eliminated intervention X entirely from the schedule? How would this impact overall deaths and infections? and How would this impact deaths and infections over time?* As the team interacted with EpiPolicy, it was evident how easy turning on or off an intervention and executing a new simulation was. After viewing a few configurations, the team wondered how easy it would be to generate a powerset of schedules. This was not within the scope of EpiPolicy. However, since all model-, intervention- and schedule- specifications are described in JSON files, it was a few hours' effort to write an external Python script that generated the powerset of schedules, executed each combination on EpiPolicy's simulator and stored the JSON output results of each simulation for further analysis.

EpiPolicy's simulator can execute a one year simulation in 2-3s. This high-performance is crucial for the Monte Carlo Tree Search algorithm that can generate thousands of simulations to search for a cost-effective schedule. The team commented that conducting such an analysis with their current model was not feasible without a major code rewrite.

A further affirmation of the importance of separating concerns p2. separation) was a request by the team: They wanted EpiPolicy to support multiple views for different roles. A view for the 'modeler' role that has access to the pages for specifying disease models, writing intervention code, and the ability to directly modify the JSON; and another view for the 'policy-planner' role that supports only changes to intervention parameters, schedules and exploring simulation results and comparisons.

Limitations. In our observations of users working with EpiPolicy, we found that users still struggled with challenge c1. mismatch : the (groups, locales, facilities) concepts (p1. abstraction) helped connect the mental model of how an intervention should work with its mathematical representation but certain constructs were difficult to map. For example, epidemiologists often think of disease transmission as a result of contact: the more contacts an agent has the higher their likelihood of infection. Agent-based models naturally capture this intuition but transforming this intuition into appropriate transmission rates at different facilities is not trivial. A future extension of EpiPolicy might explore ways of automatically transforming (albeit approximately) an agent-based model specification into a deterministic compartmental one. One possible way is to run small-scale agent-based simulations to globally infer parameters of interest. For example, one can simulate the effect of limiting social gatherings in public spaces to x contacts in an agent-based model to estimate the global effect of this intervention on the infection transmission rate as x changes.

Users understood that EpiPolicy models interactions among interventions as multiplicative factors. For example, mask wearing and social distancing each reduce transmission by some fraction less than 1. In EpiPolicy, doing both simultaneously results in reducing transmission by the product of those fractions. Combining the effects of multiple interventions may sometimes be better modeled by a more complex function. EpiPolicy will allow that.

Finally, EpiPolicy surfaced many options for the custom parameterization of the disease and population models. The ability to easily see all possible parameters and modify their values was greatly appreciated. This appreciation was almost always followed by *"Now, set them for me!"* While MCTS sets intervention control parameters (e.g. the degree of mask compliance to enforce) and where and when to apply an intervention, it cannot set parameters that describe the disease (e.g fatality rate) or the population (e.g. the percentage of the population with chronic diseases, or the percentage of the population that travels regularly between locales). Our users have to make these choices if they are pertinent to their policies.

One can argue that finding the right values for parameters is one of the principle tasks of the epidemiological modeler. We feel, however, that with a sufficient collection of open models and interventions built over time on EpiPolicy, it would become easier to infer parameter settings from prior examples.

5 RELATED WORK

EpiPolicy draws inspiration from recent trends in the HCI community in the areas of public policymaking and interactive modeling. Here, we discuss these trends and also review related work in the broader epidemic modeling literature.

5.1 Policymaking and HCI

The intersection of policymaking and HCI research has largely focused on three directions: (i) how public policy influences HCI research funding, regulatory and ethical considerations, and practices, (ii) how HCI research findings should influence policies on accessibility, usability, ergonomics, civic participation, urban planning, etc., and (iii) how best to communicate relevant HCI findings to policymakers [16, 27–29, 41]. Like EpiPolicy, certain works have focused on opportunities for HCI research to influence policymaking within specific domains: e.g. environmental public policies [45] or city-planning [33]. A core theme through all these research works is that HCI and policymaking research can only have an impact if the relevant stakeholders are adequately engaged in the design process of policymaking support tools.

5.2 Epidemic Simulation

Epidemic modeling tools have and continue to be used by public-health officials to understand disease dynamics, to predict its scale and trajectory, and to inform policies including planning response

Table 1: A comparison of three state-of-the-art compartmental epidemic simulation systems with EpiPolicy. We break down each of the tasks described in Section 2.1 into more granular components and organize them by the challenges described in Section 2.2, to illustrate the differences between the tools and the core novel contributions of EpiPolicy.

		Task	Task Breakdown	EpiPolicy	CMS[32]	STEM[18]	GLEaMviz[47]
Challenges	c1. mismatch	t1. model	Exact ODEs	Yes	Yes	Yes	Yes
			Stochastic noise	Not yet	Yes	Yes	Yes
		t2. population	Describe mobility	Yes, no air travel yet	Yes - extensive library of algorithms for disease diffusion across regions.	Yes	Main emphasis on global air-travel mobility
			Describe population groups	Yes	Only through creating special compartments in the disease model	Supports structured and aging population groups	Only through creating special compartments in the disease model
			Describe interaction facilities	Yes	No	Some support for "Transformer" facilities that model Zoonotic disease spread in shared animal and human facilities	No
	c2. parameters	t3. intervention	Specify costs	Yes	No	No	No
			Specify effects	Yes	Yes - State and time based events that can only modify declared variables; Coded in a special-syntax model configuration file.	Yes - Define triggering conditions (e.g. day = 10) to start and end an intervention that modifies disease parameters	Some support by creating intervention compartments in the disease model. Little to no support for localized temporal interventions.
			Surface & modify parameters	Yes	Only through coding practices that surface intervention parameters	Intervention templates have a large number of parameters that can be set through different forms	No
		t4. schedule	When?	Yes	An intervention is coupled with its triggering conditions. This makes it difficult to generate multiple different schedules.		No
			Where?	Yes			No
			How?	Yes			No
	c3. no MMS	t5. what-if	Collaborative comparative analysis	Yes	Single view without a clear separation between model, interventions and schedules, makes it difficult to surface parameters for different team members to manipulate and compare different scenarios.		
		t7. refine	Modify, simulate and compare scenarios	Yes	No	Batch Experiments allows the execution of multiple scenarios with different settings for only pre-declared parameters.	A simulation history keeps track of past runs; Some comparative visualizations available.
		t8. audit	Provenance	Yes	No	Yes	No
		t9. communicate	Comparisons & Visualizations	Yes	No	Yes	Yes
		t6. optimize	Search for optimal intervention plans	Yes	No	No	No
			Unique features		Open source simulation tool with several sophisticated stochastic ODE solvers and diffusion algorithms. Models specified in CMS can be ported to EMOD - an agent-based modeling tool.	Large and active user base with many existing models for different diseases that users can easily build from.	Engaging visualizations of global disease spread; Efficient simulator implemented in C++: 1min runtime for 1 year simulation

strategies and allocating resources [2, 25, 30, 36, 44, 48]. Our observations of the challenges that public-health teams face when working with epidemic simulation tools are not unique. For example, a review of the impact of emergency modeling during the 2009 H1N1 pandemic on decision-making and the interactions between modelers and decision-makers echoes many of the challenges we describe in Section 2.2 including (i) difficulties in generating, sharing or disseminating modeling results to decision-makers (c3. no MMS), (ii) difficulties in understanding the structure and the assumptions made by the models (c1. mismatch), (iii) a lack of transparency or realism in implemented interventions (c2. parameters), and (iv) misaligned modeling expectations - predict the course of an epidemic vs. plan interventions [30].

One approach to epidemic modeling is to translate mathematical models directly into code in a general-purpose programming language such as `R` or `Python`. Modelers here use modeling libraries and frameworks such as [10, 11, 24, 50, 51]. While powerful, language-based approaches require additional time for debugging and error resolution, consuming almost half of the total programming time [8]. Professionals working with epidemic models are generally not trained in software development and thus overlook common programming best practices including error handling, unit testing, and documentation, resulting in erroneous results [1, 5, 21]. This is further highlighted by the widespread criticism levied at an early study modeling the COVID-19 pandemic [19] for its "unreliable" and "buggy" code which led to difficulties in reproducing its results [39].

Recently, considerable effort has been put into building tools with graphical and command-line user interfaces in order to make simulating epidemics accessible to people with limited programming expertise and curb programming-induced errors [22]. Much of this effort has concentrated on building tools that focus on a single class of epidemics, such as influenza. Among the tools that are generic enough to simulate any epidemic, GLEaMviz [47], CMS [32], and STEM [18] are the most commonly used, with active communities. STEM, in particular, has seen vast adoption among epidemiologists, academics, and policy experts [17]. While these tools handle the task of defining and simulating epidemics well enough, they lack support in important aspects such as scheduling interventions, estimating costs, comparing policies, and automatic generation of intervention schedules.

Table 1 summarizes the main differences between EpiPolicy and these three state-of-the-art epidemic simulation systems. Not evident in this table, however, is EpiPolicy's unique design choice (`p2. separation`) to support collaborative teams with different roles through the explicit separation of the policymaking workflow into distinct pages for different tasks such as disease and population modeling, intervention specification and planning. Single code-base or workflow tools like GLEaMviz, STEM and CMS may not incur a 'separation overhead,' allowing a computational epidemiologist to more easily set up an initial model but there is a trade-off: as the complexity of model, interventions, or schedules, and as the size of the policymaking team grow, the benefits of a single code-base or workflow breakdown as we describe in Section 2.

Agent-based stochastic models. As we discussed in Section 2.2, agent-based models are better suited at modeling individual disease-transmitting interactions between agents and can therefore better describe interventions targeted at modifying agent-behavior such as social distancing, isolation, etc. Some notable agent-based tools include EMOD [4], FRED [20], and CovidSim [19, 35]. These are command-line tools: they accept input configuration files that describe the model and simulation parameters and they generate results in the form of various output files. Users are able to define a wide-range of agent-specific features such as age, mortality, and vaccine eligibility. Stochastic agent-based models, while more expressive than deterministic compartmental models, are extremely computationally intensive, making it currently impractical to automatically explore different intervention plans with techniques like MCTS. EpiPolicy's abstraction design principle (`p1. abstraction`) aims to better connect the ODE compartmental model to the end-users' mental model through the introduction of facilities where population groups interact within. To a certain degree, it is successful in achieving this as demonstrated by some of the scenarios built for the feasibility studies and the focus group. Rethinking how we implement stochastic agent-based models to ensure scalability is a fundamental first step to supporting automatic plan exploration.

5.3 Interactive Modeling

Interactive modelling systems have been used to simulate the effects of various phenomena and processes such as human behaviours and governmental policies: *Gamette* utilizes a game-based interface approach to model human behaviour using an agent-based model by segmenting a simulation study into a number of game scenarios that involve human users to calibrate the underlying model and to evaluate the behaviour of the agents [34]. Tomlinson et al. utilized a simulation system based on an agent-based model to understand the impact of Accountable Capitalism Act [46].

EpiPolicy is a single design point in the rich space of model-driven policymaking support tools. While it targets the specialized area of epidemic control, the same framework could apply to other policymaking domains. For example, in a misinformation-mitigation setting, institutional mitigators may represent some information access and quality measures that misinformation campaigns negatively impact, and interventions such as providing information portals, fact-checking claims, etc. improve. As in EpiPolicy, there are different populations *groups* with different vulnerabilities to misinformation, media platforms resemble *facilities* where groups interact and spread (mis-)information and different *locales* may have different policies and laws. The design principles and many of EpiPolicy's features such as intervention-plan optimization can thus apply to other applications that can be described by state-based models and interventions.

6 CONCLUSION

This paper has described EpiPolicy: a tool to help formulate public health policies that curb disease outbreaks. EpiPolicy's user interface is divided into a series of task-oriented pages to help modelers through each step of the modeling process. EpiPolicy provides high-level abstractions that help connect how interventions work with the mathematical ordinary differential equations that describe disease spread. It visualizes the disease model, certain population demographics such as mobility patterns, the schedule of interventions and the results of a simulation. These visualizations help users understand, analyze and communicate different disease models and intervention policies. In EpiPolicy, the effects and costs of interventions are defined separately from their planning specifics: when, where, and how. This 'separation of concerns' allows users to easily construct, modify, and compare multiple policies. It further allows EpiPolicy to generate and simulate thousands of intervention schedules, with the help of Monte Carlo Tree Search, and select the most cost-effective ones.

Epidemiologists appreciated how EpiPolicy surfaced the many parameters that described the disease, the population and the intervention policy, and allowed users to easily modify these parameters. They also found EpiPolicy's support for the *interactive* exploration and evaluation of different policies powerful.

ACKNOWLEDGMENTS

This work was supported by the NYUAD Center for Interacting Urban Networks (CITIES), funded by Tamkeen under the NYUAD Research Institute Award CG001, by the Swiss Re Institute under the Quantum Cities™ initiative, and by the NYUAD COVID-19 Facilitator Research Fund.

REFERENCES

[1] 2016. Retraction [retraction of: Henson KE, Jagsi R, Cutter D, McGale P, Taylor C, Darby SC. J Clin Oncol. 2016 Mar 10;34(8):803-9]. *Journal of Clinical Oncology* 34, 27 (2016), 3358–3359. https://doi.org/10.1200/JCO.2016.69.0875 PMID: 27528722.

[2] David Adam. 2020. Special report: The simulations driving the world's response to COVID-19. *Nature* 580, 7802 (2020), 316–319.

[3] Marco Ajelli, Bruno Gonçalves, Duygu Balcan, Vittoria Colizza, Hao Hu, José J Ramasco, Stefano Merler, and Alessandro Vespignani. 2010. Comparing large-scale computational approaches to epidemic modeling: agent-based versus structured metapopulation models. *BMC infectious diseases* 10, 1 (2010), 1–13.

[4] Anna Bershteyn, Jaline Gerardin, Daniel Bridenbecker, Christopher W Lorton, Jonathan Bloedow, Robert S Baker, Guillaume Chabot-Couture, Ye Chen, Thomas Fischle, Kurt Frey, et al. 2018. Implementation and applications of EMOD, an individual-based multi-disease modeling platform. *Pathogens and disease* 76, 5 (2018), fty059.

[5] Jayanti Bhandari Neupane, Ram P Neupane, Yuheng Luo, Wesley Y Yoshida, Rui Sun, and Philip G Williams. 2019. Characterization of leptazolines A–D, polar oxazolines from the cyanobacterium Leptolyngbya sp., reveals a glitch with the "Willoughby–Hoye" scripts for calculating NMR chemical shifts. *Organic letters* 21, 20 (2019), 8449–8453.

[6] Derdei Bichara and Abderrahman Iggidr. 2018. Multi-patch and multi-group epidemic models: a new framework. *Journal of mathematical biology* 77, 1 (2018), 107–134.

[7] George EP Box. 1979. Robustness in the strategy of scientific model building. In *Robustness in statistics*. Elsevier, 201–236.

[8] Tom Britton, Lisa Jeng, Graham Carver, Paul Cheak, and Tomer Katzenellenbogen. 2013. Reversible debugging software. *Judge Bus. School, Univ. Cambridge, Cambridge, UK, Tech. Rep* (2013).

[9] Shannon Brownlee and Jeanne Lenzer. 2020. *The Ioannidis Affair: A Tale of Major Scientific Overreaction.* https://www.scientificamerican.com/article/the-ioannidis-affair-a-tale-of-major-scientific-overreaction/ Accessed 05-04-2021.

[10] Caitlin Rivers. 2015. Epipy: Python tools for epidemiology. https://cmrivers.github.io/epipy/

[11] Bob Carpenter, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus A Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. 2017. Stan: a probabilistic programming language. *Grantee Submission* 76, 1 (2017), 1–32.

[12] Coronavirus Resource Center. 2021. *MORTALITY ANALYSES.* https://coronavirus.jhu.edu/data/mortality Accessed 05-04-2021.

[13] Center for International Earth Science Information Network - CIESIN - Columbia University. 2016. *Gridded Population of the World, Version 4 (GPWv4): Population Density. Palisades, NY: NASA Socioeconomic Data and Applications Center (SEDAC).* http://dx.doi.org/10.7927/H4NP22DQ Accessed 13-07-2020.

[14] GUILLAUME M. J-B. CHASLOT, MARK H. M. WINANDS, H. JAAP VAN DEN HERIK, JOS W. H. M. UITERWIJK, and BRUNO BOUZY. 2008. PROGRESSIVE STRATEGIES FOR MONTE-CARLO TREE SEARCH. *New Mathematics and Natural Computation* 04, 03 (2008), 343–357. https://doi.org/10.1142/S1793005708001094 arXiv:https://doi.org/10.1142/S1793005708001094

[15] Rémi Coulom. 2006. Efficient selectivity and backup operators in Monte-Carlo tree search. In *In: Proceedings Computers and Games 2006*. Springer-Verlag.

[16] Janet Davis, Harry Hochheiser, Juan Pablo Hourcade, Jeff Johnson, Lisa Nathan, and Janice Tsai. 2012. Occupy CHI! engaging US policymakers. In *CHI'12 Extended Abstracts on Human Factors in Computing Systems*. 1139–1142.

[17] Judith V Douglas, Simone Bianco, Stefan Edlund, Tekla Engelhardt, Matthias Filter, Taras Günther, Kun Hu, Emily J Nixon, Nereyda L Sevilla, Ahmad Swaid, et al. 2019. STEM: an open source tool for disease modeling. *Health security* 17, 4 (2019), 291–306.

[18] Stefan B Edlund, Matthew A Davis, and James H Kaufman. 2010. The spatiotemporal epidemiological modeler. In *Proceedings of the 1st ACM International Health Informatics Symposium*. 817–820.

[19] Neil Ferguson, Daniel Laydon, Gemma Nedjati Gilani, Natsuko Imai, Kylie Ainslie, Marc Baguelin, Sangeeta Bhatia, Adhiratha Boonyasiri, ZULMA Cucunuba Perez, Gina Cuomo-Dannenburg, et al. 2020. Report 9: Impact of non-pharmaceutical interventions (NPIs) to reduce COVID19 mortality and healthcare demand. (2020).

[20] John J Grefenstette, Shawn T Brown, Roni Rosenfeld, Jay DePasse, Nathan TB Stone, Phillip C Cooley, William D Wheaton, Alona Fyshe, David D Galloway, Anuroop Sriram, et al. 2013. 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 (2013), 1–14.

[21] Thomas Herndon, Michael Ash, and Robert Pollin. 2014. Does high public debt consistently stifle economic growth? A critique of Reinhart and Rogoff. *Cambridge journal of economics* 38, 2 (2014), 257–279.

[22] David James Heslop, Abrar Ahmad Chughtai, Chau Minh Bui, and C Raina MacIntyre. 2017. Publicly available software tools for decision-makers during an emergent epidemic—Systematic evaluation of utility and usability. *Epidemics* 21 (2017), 1–12.

[23] John PA Ioannidis. 2020. A fiasco in the making? As the coronavirus pandemic takes hold, we are making decisions without reliable data. *Stat* 17 (2020).

[24] Samuel M Jenness, Steven M Goodreau, and Martina Morris. 2018. EpiModel: an R package for mathematical modeling of infectious disease over networks. *Journal of statistical software* 84 (2018).

[25] Michael A Johansson, Ann M Powers, Nicki Pesik, Nicole J Cohen, and J Erin Staples. 2014. Nowcasting the spread of chikungunya virus in the Americas. *PloS one* 9, 8 (2014), e104915.

[26] Levente Kocsis and Csaba Szepesvári. 2006. Bandit Based Monte-Carlo Planning. In *Machine Learning: ECML 2006*, Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 282–293.

[27] Jonathan Lazar. 2015. Public policy and HCI: making an impact in the future. *Interactions* 22, 5 (2015), 69–71.

[28] Jonathan Lazar, Julio Abascal, Simone Barbosa, Jeremy Barksdale, Batya Friedman, Jens Grossklags, Jan Gulliksen, Jeff Johnson, Tom McEwan, Loïc Martínez-Normand, et al. 2016. *Human–computer interaction and international public policymaking: a framework for understanding and taking future actions.* now publishers.

[29] Jonathan Lazar, Simone Barbosa, Jan Gulliksen, Tom McEwan, Loïc Martinez Normand, Philippe Palanque, Raquel Prates, Janice Tsai, Marco Winckler, and Volker Wulf. 2013. Workshop on engaging the human-computer interaction community with public policymaking internationally. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*. 3279–3282.

[30] BY Lee, LA Haidari, and MS Lee. 2013. Modelling during an emergency: the 2009 H1N1 influenza pandemic. *Clinical Microbiology and Infection* 19, 11 (2013), 1014–1022.

[31] Zhicheng Liu and Jeffrey Heer. 2014. The effects of interactive latency on exploratory visual analysis. *IEEE transactions on visualization and computer graphics* 20, 12 (2014), 2122–2131.

[32] Christopher W Lorton, Joshua L Proctor, Min K Roh, and Philip A Welkhoff. 2019. Compartmental Modeling Software: a fast, discrete stochastic framework for biochemical and epidemiological simulation. In *International Conference on Computational Methods in Systems Biology*. Springer, 308–314.

[33] Jennifer Manuel and Clara Crivellaro. 2020. Place-Based Policymaking and HCI: Opportunities and Challenges for Technology Design. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–16.

[34] Omid Mohaddesi, Yifan Sun, Rana Azghandi, Rozhin Doroudi, Sam Snodgrass, Ozlem Ergun, Jacqueline Griffin, David Kaeli, Stacy Marsella, and Casper Harteveld. 2020. Introducing Gamettes: A Playful Approach for Capturing Decision-Making for Informing Behavioral Models. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[35] MRC Centre for Global Infectious Disease Analysis. 2020. *COVID-19 CovidSim Model.* https://github.com/mrc-ide/covid-sim Accessed 13-07-2021.

[36] Lisa A Rosenfeld, Claude Earl Fox, Debora Kerr, Erin Marziale, Amy Cullum, Kanchan Lota, Jonathan Stewart, and Mary Zack Thompson. 2009. Use of computer modeling for emergency preparedness functions by local and state health officials: a needs assessment. *Journal of public health management and practice* 15, 2 (2009), 96–104.

[37] Kankoé Sallah, Roch Giorgi, Linus Bengtsson, Xin Lu, Erik Wetter, Paul Adrien, Stanislas Rebaudet, Renaud Piarroux, and Jean Gaudart. 2017. Mathematical models for predicting human mobility in the context of infectious disease spread: introducing the impedance model. *International journal of health geographics* 16, 1 (2017), 1–11.

[38] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. 2008. *Global sensitivity analysis: the primer.* John Wiley & Sons.

[39] Dalmeet Singh Chawla. 2020. Critiqued coronavirus simulation gets thumbs up from code-checking efforts. *Nature* (2020), 323–324.

[40] Alex Smajgl, Daniel G Brown, Diego Valbuena, and Marco GA Huigen. 2011. Empirical characterisation of agent behaviours in socio-ecological systems. *Environmental Modelling & Software* 26, 7 (2011), 837–844.

[41] Anne Spaa, Abigail Durrant, Chris Elsden, and John Vines. 2019. Understanding the Boundaries between Policymaking and HCI. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–15.

[42] Jia Bin Tan, Matthew James Cook, Prishanee Logan, Liudmila Rozanova, and Annelies Wilder-Smith. 2021. Singapore's Pandemic Preparedness: An Overview of the First Wave of COVID-19. *International journal of environmental research and public health* 18, 1 (2021), 252.

[43] Charles A Taylor, Christopher Boulos, and Douglas Almond. 2020. Livestock plants and COVID-19 transmission. *Proceedings of the National Academy of Sciences* 117, 50 (2020), 31706–31715.

[44] WHO Ebola Response Team. 2014. Ebola virus disease in West Africa—the first 9 months of the epidemic and forward projections. *New England Journal of Medicine* 371, 16 (2014), 1481–1495.

[45] Vanessa Thomas, Christian Remy, Mike Hazas, and Oliver Bates. 2017. HCI and environmental public policy: Opportunities for engagement. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 6986–6992.

[46] Bill Tomlinson, M Six Silberman, Andrew W Torrance, Kurt Squire, Paramdeep S Atwal, Ameya N Mandalik, Sahil Railkar, and Rebecca W Black. 2020. A Participatory Simulation of the Accountable Capitalism Act. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[47] Wouter Van den Broeck, Corrado Gioannini, Bruno Gonçalves, Marco Quaggiotto, Vittoria Colizza, and Alessandro Vespignani. 2011. The GLEaMviz computational tool, a publicly available software to explore realistic epidemic spreading scenarios at the global scale. *BMC infectious diseases* 11, 1 (2011), 1–14.

[48] Maria D Van Kerkhove and Neil M Ferguson. 2012. Epidemic and intervention modelling: a scientific rationale for policy decisions? Lessons from the 2009 influenza pandemic. *Bulletin of the World Health Organization* 90 (2012), 306–310.

[49] Danielle Varda, Jo Ann Shoup, and Sara Miller. 2012. A systematic review of collaboration and network research in the public affairs literature: implications for public health practice and research. *American journal of public health* 102, 3 (2012), 564–571.

[50] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature methods* 17, 3 (2020), 261–272.

[51] Hadley Wickham. 2017. The tidyverse. *R package ver* 1, 1 (2017), 1.

[52] Jianyong Wu, Radhika Dhingra, Manoj Gambhir, and Justin V Remais. 2013. Sensitivity analysis of infectious disease models: methods, advances and their application. *Journal of The Royal Society Interface* 10, 86 (2013), 20121018.

[53] Xin Ye, Karthik Konduri, Ram M Pendyala, Bhargava Sana, and Paul Waddell. 2009. A methodology to match distributions of both household and person attributes in the generation of synthetic populations. In *88th Annual Meeting of the Transportation Research Board, Washington, DC*.

[54] Ye, Xin, Karthik Konduri, Ram Pendyala, Bhargava Sana and Paul Waddell. 2009. *SynthPop.* https://github.com/UDST/synthpop Accessed 13-07-2020.