Knowledge and Information Systems
https://doi.org/10.1007/s10115-018-01326-x

REGULAR ARTICLE

Data-driven efficient network and surveillance-based immunization

Yao Zhang[1] · Arvind Ramanathan[3] · Anil Vullikanti[2] · Laura Pullum[3] · B. Aditya Prakash[1]

Received: 10 January 2018 / Revised: 23 December 2018 / Accepted: 28 December 2018

Abstract
Given a contact network and coarse-grained diagnostic information such as electronic Healthcare Reimbursement Claims (eHRC) data, can we develop efficient intervention policies from data to control an epidemic? Immunization is an important problem in multiple areas, especially epidemiology and public health. However, most existing studies rely on assuming prior epidemiological models to develop pre-emptive strategies, which may fail to adapt to the change in new epidemiological patterns and the availability of rich data such as eHRC. In practice, disease spread is usually complicated, hence assuming an underlying model may deviate from true spreading patterns, leading to possibly inaccurate interventions. Additionally, the abundance of health care surveillance data (such as eHRC) makes it possible to study data-driven strategies without too many restrictive assumptions. Hence, such a data-driven intervention approach can help public-health experts take more practical decisions. In this paper, we take into account propagation log and contact networks for controlling propagation. Different from previous model-based approaches, our solutions are solely data driven in a sense that we develop immunization strategies directly from the network and eHRC without assuming classical epidemiological models. In particular, we formulate the novel and challenging *data-driven immunization* problem. To solve it, we first propose an efficient sampling approach to align surveillance data with contact networks, then develop an efficient algorithm with the provably approximate guarantee for immunization. Finally, we show the effectiveness and scalability of our methods via extensive experiments on multiple datasets, and conduct case studies on nation-wide real medical surveillance data.

Keywords Graph mining · Social networks · Immunization · Diffusion

1 Introduction

Vaccination and social distancing are among the principle strategies for controlling the spread of infectious diseases [1,2]. CDC (Centers for Disease Control) guidelines for vaccine usage are typically based on age groups, e.g., for young children and seniors—these do not result

✉ Yao Zhang
yaozhang@cs.vt.edu

Extended author information available on the last page of the article

in optimal interventions, which minimize outcomes such as the total number of infections [1]. Additionally, most work on designing immunization algorithms from a data-mining viewpoint has focused on developing innovative strategies which assume knowledge of the underlying disease model [3,4] or make assumptions of very fine-grained individual-level surveillance data [5].

Recent trends have led to the increasing availability of electronic claims data and also capabilities in developing very realistic urban population contact networks. This motivates the following problem: given a contact network, and a *coarse-grained* propagation log such as electronic Health Reimbursement Claims (eHRC), can we learn an efficient and realistic intervention policy to control propagation (such as a flu outbreak)? Further, can we do it directly without assuming any epidemiological models? Influenza viruses change constantly, and hence designing interventions optimized for specific epidemic model parameters is likely to be suboptimal [6].

The diagnostic propagation log data provide us with a good sense of how diseases spread, while contact networks tell us how people interact with others. We take into account both for immunization and study the *data-driven immunization* problem. Some of the major challenges include: (1) the scale of these datasets (eHRC consists of billions of records and contact networks have millions of nodes), and (2) eHRC data is anonymized, and available only at a zipcode level. The main contributions of our paper are:

(a) Problem formulation We formulate the data-driven immunization problem given a contact network and the propagation log. We first sample the most-likely "social-contact" cascades from the propagation log to the contact network and then pose the immunization problem at a location level, and show it is NP-hard.

(b) Effective algorithms We present efficient algorithms to get the most-likely samples, and then provide a contribution-based greedy algorithm, ImmuConGreedy, with provably approximate solutions to allocate vaccines to locations.

(c) Experimental evaluation We present extensive experiments against several competitors, including graph-based and model-based baselines, and demonstrate that our algorithms outperform baselines by reducing upto 45% of the infection with limited budget. Furthermore, we conduct case studies on nation-wide real medical surveillance data with billions of records to show the effectiveness of our methods. To the best of our knowledge, we are the first to study realistic immunization policies on such large-scale datasets.

2 Preliminaries

We give a brief introduction of the propagation data eHRC and contact networks we used in this section.

2.1 Propagation data (eHRC)

The propagation data for this study were primarily based on IMS Health claims data, *electronic Healthcare Reimbursement Claims* (eHRC), which consists of over a billion claims for the period April 1, 2009–March 31, 2010. The claims data consist of reimbursement claims recorded electronically from healthcare practitioners received from all parts of the USA, including urban and rural areas. The dataset, its features, and its overall coverage/completeness are described in detail in [7,8]; for this study, we used daily flu reports,

based on ICD-9 codes 486XX and 488XX and individual locations (zipcode) recorded in the claims. Prior to our study, we obtained internal Institutional Review Board approval for analyzing the dataset.

2.2 Activity-based populations

We use city-scale activity-based populations as contact networks (see [9,10] for more details). These models are constructed by a "first-principles" approach and integrate over a dozen public and commercial datasets, including census, land use, activity surveys and transportation networks. The model includes detailed demographic attributes at an individual and household level, along with normative activities. These models have been used in a number of studies on epidemic spread and public-health policy planning, including response strategies for smallpox attacks [10] and the National strategy for pandemic flu [2].

3 Problem formulations

Table 1 lists the main notations used throughout the paper.

We use $G(V, E)$ to denote an undirected unweighted graph and $L = \{L_1, \ldots, L_n\}$ to denote a set of locations. $V_i \subseteq V$ denotes the set of nodes at location L_i; we assume there are no overlapping nodes between locations. Large medical surveillance data, such as eHRC, are usually anonymized due to privacy issues. Hence, in this paper, we assume that only the number of infections is given. Formally, the propagation log R is an infection matrix $\mathbf{N}$ ($(t_{\max} + 1) \times n$), where t_0 and $t_{\max}$ are the earliest and last timesteps. Each element $N(L_\ell, t)$ represents the number of patients in R at location L_ℓ at time t. Each row vector

Table 1 Terms and symbols

Symbol	Definition and description
$G(V, E)$	Graph G with the node set V and the edge set E
R	Propagation log
$\mathbf{N}$	Infection matrix for the propagation log R
$N(L_\ell, t_i)$	The number of patients at t_i in L_ℓ
t_0	The earliest timestep $t_0 = 0$
n	Number of locations
$L = \{L_1, \ldots, L_n\}$	Set of locations
m	Number of vaccines
$\mathbf{x}$	Vaccine allocation vector $[x_1, \ldots, x_n]'$
k	Number of samples in $\mathcal{M}$
$\mathcal{M}$	Set of sampled cascades $\{\mathbf{M}_1, \ldots, \mathbf{M}_k\}$
$\mathbf{M}$	A sampled cascade
$SI_{\mathbf{M}}$	The starting infected node set in $\mathbf{M}$
$\sigma_{G,\mathbf{M}}(\mathbf{x})$	The expected number of nodes $SI_{\mathbf{M}}$ can reach when $\mathbf{x}$ is given
$\rho_{G,\mathbf{M}_i}(\mathbf{x})$	$\sigma_{G,\mathbf{M}}(\mathbf{0}) - \sigma_{G,\mathbf{M}}(\mathbf{x})$
$\alpha_{\mathbf{M},\ell}$	Number of nodes that have at least one parent in $\mathbf{M}$ at location L_ℓ
S_ℓ	The initial starting node set at location L_ℓ, where $\lvert S_\ell \rvert = N(L_\ell, t_0)$

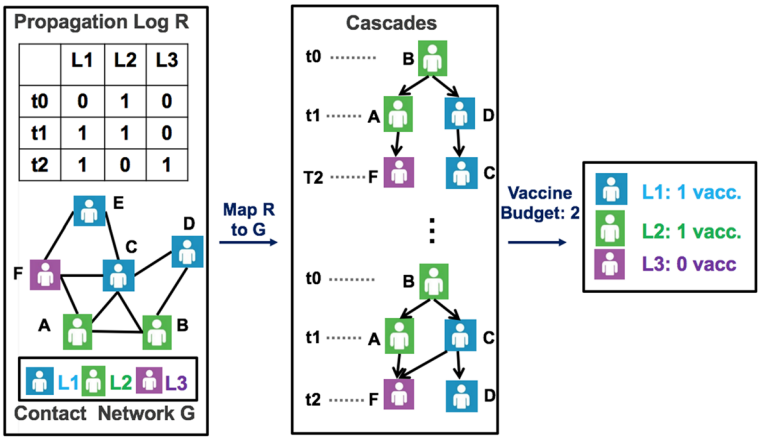


Fig. 1 Overview of our approach. We first generate a set of cascades, then allocate vaccine to different locations

$\mathbf{N}(t) = [N(L_1, t), \ldots, N(L_n, t)]$ represents the number of infections at time t, and each column vector $\mathbf{N}_{L_\ell} = [N(L_\ell, t_0), \ldots, N(L_\ell, t_{\max})]^T$ represents the number of infections at location L_ℓ.

3.1 Interactions and surveillance

A contact network G models people's interactions with others, which is a powerful tool to control epidemics. For example, Prakash et al. [11] showed that the first eigenvalue of the adjacency matrix of G is related to the epidemic threshold. An epidemic will be quickly extinguished given a small epidemic threshold. Several effective algorithms have been proposed to minimize the first eigenvalue to control epidemics [3,4,12]. However, all of them assume an underlying epidemiological model such as Susceptible-Infected-Recovered (SIR) [13]. In addition, they are strictly graph-based methods without looking into rich medical surveillance data. Although graph-based methods can provide us with good baseline strategies, they do not take into account particular patterns of a given virus. On the other hand, the disease propagation data R such as eHRC can give us a coarse-grained picture of infections. However, there is very little information on how an epidemic spreads via person-to-person contacts from R. Hence, we believe the disease propagation data R, along with a contact network G, can help us develop better and more implementable interventions to control an epidemic. For example, we can take the surveillance data of the past flu season to allocate vaccines for the current flu season.

3.2 Map *R* to nodes in *G*

The main challenge of integrating R and G is that R (such as eHRC) in practice is anonymized. Hence, we cannot associate each record in R with a node in G. In this paper, we tackle this challenge by mapping infections from R to nodes in G at the location level. The idea is that at each location L_ℓ and time t_i, we pick $N(L_\ell, t_i)$ nodes in G as infected nodes. Note that we can have multiple choices of mapping R to G. For example, in Fig. 1, $N(L_2, t_0) = 1$, and hence, we can pick either A or B as infected node at t_0. We denote these choices as $\mathcal{M}$, where $\mathcal{M}$ is a set of cascades. We define a *cascade* $\mathbf{M}$ as follows:

Definition 3.1 (*Cascade*) A cascade $\mathbf{M}$ is a directed acyclic graph (DAG) induced by R and G. Each node $u \in V_{\mathbf{M}}$ is associated with a location L_ℓ and a timestep t_i, where $u \in V_i$ and u are infected at t_i (denoted as $t(u) = t_i$). For node u and v in $\mathbf{M}$, if $e_{u,v} \in E$ and $t(u) = t(v) - 1$, there is a directed edge from u to v in $\mathbf{M}$. We denote $e(u, v) \in E_{\mathbf{M}}$.

We could select $N(L_\ell, t_i)$ nodes uniformly at random as infected nodes in G for each $\mathbf{M}$. However, it is not practical as infection distributions are not uniform. For example, if a node u has an infected neighbor, u can be infected by that node; in contrast, if u does not have any infected neighbor in R, it is unlikely to be infected. Hence, we propose to map R to G according to the SocialContact approach.

3.3 SocialContact

We say an infected node u gets infected by "social contact" in G, if u has a direct neighbor that is infected earlier than u. Otherwise, we say that a node is infected by external forces. In reality, infectious diseases (such as flu, mumps) usually spread via person-to-person contact. Hence, for a mapped cascade $\mathbf{M}$, we want to maximize the number of nodes caused by SocialContact. Formally, we define $\alpha_{\mathbf{M}} = |\{u|\exists v, e(v, u) \in E_{\mathbf{M}}\}|$, i.e., $\alpha_{\mathbf{M}}$ is the number of nodes that have at least one parent in $\mathbf{M}$. Then, maximizing the number of nodes infected by SocialContact is equivalent to maximizing $\alpha_{\mathbf{M}}$. Figure 1 shows two cascades with the best $\alpha_{\mathbf{M}} = 4$: as only the node that starts the infection does not have a parent. To get k cascades with SocialContact in $\mathcal{M}$, we formulate the mapping problem:

Problem 3.1 (*Mapping problem*) Given a contact network G, propagation log R, and number of cascades k, find $\mathcal{M}^* = \{\mathbf{M}_1^*, \ldots, \mathbf{M}_k^*\}$ where each node u in $\mathbf{M}$ is associated with a location L_ℓ and a time t_i:

$$\mathcal{M}^* = \arg\max_{\mathcal{M}} \sum_{\mathbf{M}_i \in \mathcal{M}} \alpha_{\mathbf{M}_i}, \text{ s.t. } |\mathcal{M}| = k \tag{1}$$

Remark 3.1 Since we do not specify any epidemiological model (such as SIR) for Problem 3.1, it is difficult to define any probability distribution for $\mathcal{M}$. Hence, the sample average approximation approach is not applicable for this problem.

3.4 Data-driven immunization

Once we generate $\mathcal{M}$, we want to study how to best allocate vaccines to minimize the infection shown in R. Recently, Zhang et al. [4] proposed a model-based group immunization problem, in which they allocate vaccines to nodes *within* groups uniformly-at-random—this mimics real-life distribution of vaccines by public-health authorities. We leverage their within-group allocation approach. Let us define $\mathbf{x} = [x_1, \ldots, x_n]'$ as a vaccine allocation vector, where x_i is the number of vaccines given to location L_i. If we give x_i vaccines to location L_i, x_i nodes will be uniformly randomly removed from V_i. The objective is to find an allocation that "breaks" the cascades most effectively. We define $SI_{\mathbf{M}}$ to be the starting "seed" infected nodes in $\mathbf{M}$, i.e., $SI_{\mathbf{M}} = \{u \in V_{\mathbf{M}} | t_u = t_0\}$, and $\sigma_{G,\mathbf{M}}(\mathbf{x})$ to be the expected number of nodes that $SI_{\mathbf{M}}$ can reach after $\mathbf{x}$ is allocated to locations in $\mathbf{M}$. Hence, we want to minimize $\sigma_{G,\mathbf{M}}(\mathbf{x})$ to limit the expected infection over any cascade $\mathbf{M} \in \mathcal{M}$. For example, in Fig. 1, once two vaccines are given to L_1 and L_2, we minimize the number of nodes that B can reach in the two cascades.

For ease of description, let us define $\rho_{G,\mathbf{M}}(\mathbf{x}) = \sigma_{G,\mathbf{M}}(\mathbf{0}) - \sigma_{G,\mathbf{M}}(\mathbf{x})$. $\rho_{G,\mathbf{M}}(\mathbf{x})$ can be thought as the number of nodes we can save if $\mathbf{x}$ is allocated. Since $\sigma_{G,\mathbf{M}_i}(\mathbf{0})$ is constant, minimizing $\sigma_{G,\mathbf{M}}(\mathbf{x})$ is equivalent to maximizing $\rho_{G,\mathbf{M}}(\mathbf{x})$. Formally, our data-driven immunization problem *given* $\mathcal{M}$ (from Problem 3.1) is:

Problem 3.2 (*Data-driven immunization*) Given a contact network G, a set of cascades $\mathcal{M}$, and budget m, find a vaccine allocation vector $\mathbf{x}^*$:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \frac{1}{|\mathcal{M}|} \sum_{\mathbf{M}_i \in \mathcal{M}} \rho_{G,\mathbf{M}_i}(\mathbf{x}), \text{ s.t. } |\mathbf{x}|_1 = m \tag{2}$$

3.5 Hardness

Both Problems 3.1 and 3.2 are NP-hard; this can be shown by reductions from the Max-K-Set Union problem [14] and the DAV problem [5], respectively.

4 Proposed method

In this section, we develop two efficient algorithms, MappingGeneration for Problem 3.1 and ImmuConGreedy for Problem 3.2.

4.1 Generating cascades from SocialContact

Main idea To tackle Problem 3.1, we first focus on a special case where $k = 1$ (find a single cascade $\mathbf{M}$), then extend it to multiple cascades. The challenge here is that even when $k = 1$, Problem 3.1 is still NP-hard. Our main idea to solve this is to first generate $SI_{\mathbf{M}}$ (the seed set), and then generate $\mathbf{M}$ from $SI_{\mathbf{M}}$. In principle, this can be done from checking $SI_{\mathbf{M}}$'s i-hop neighbors. Clearly, $SI_{\mathbf{M}}$'s quality will directly affect $\mathbf{M}$'s quality. However, it is still hard to find $SI_{\mathbf{M}}$ and generate $\mathbf{M}$ from $SI_{\mathbf{M}}$. Instead, we identify a necessary condition for the optimal $\mathbf{M}$, and propose a provable approximation algorithm to find $SI_{\mathbf{M}}$ that satisfies the condition. We make the algorithm faster by leveraging the Approximate Neighborhood Function (ANF) technique. Then, we generate the corresponding cascade $\mathbf{M}$ from $SI_{\mathbf{M}}$, and propose a fast algorithm MappingGeneration to extend it to k cascades for Problem 3.1.

Finding $SI_{\mathbf{M}}$ To find a high-quality $SI_{\mathbf{M}}$, we first examine what is the optimal $\mathbf{M}$. According to Eq. 1, the optimal $\mathbf{M}$ has the maximum value of $\alpha_{\mathbf{M}}$. Let us define $\alpha^*_{\mathbf{M}}$ as the maximum of $\alpha_{\mathbf{M}}$ ($\alpha_{\mathbf{M}} \le \alpha^*_{\mathbf{M}}$). Then, we have the following lemma:

Lemma 4.1 $\alpha^*_{\mathbf{M}} = \sum_{t=t_1}^{t_{\max}} |\mathbf{N}(t)|_1$, *i.e., the number of infections after the earliest time* t_0.

Proof When we map R to G, the *optimal* case for a cascade $\mathbf{M}$ is that every node u with $t(u) > t_0$ has at least one parent in $\mathbf{M}$, and the only nodes that do not have any parents are the ones infected at the earliest time t_0. Hence, $\alpha^*_{\mathbf{M}}$ is the number of nodes that are infected after t_0. □

Lemma 4.1 shows that the maximum $\alpha_{\mathbf{M}}$ is the number of infections after t_0. However, as shown in the next lemma, it is hard to find a $SI_{\mathbf{M}}$ with the optimal $\mathbf{M}$.

Lemma 4.2 *Finding a set* $SI_{\mathbf{M}}$ *for the cascade* $\mathbf{M}$ *with* $\alpha_{\mathbf{M}} = \alpha^*_{\mathbf{M}}$ *is NP-hard.*

Proof We can reduce it from a K-Set Union problem which tries to pick K sets that cover at least ρ elements. Consider an instance of the K-Set Union problem with n sets ($n > k$) where each set A_i contains m elements e_j, we can construct the following instance: assuming only one location L_0, we create any node set S that connects to A_i, and each set A_i has an edge to any $e_j \in A_i$. For the log R, we have $N(L_0, t_0) = |S|$, $N(L_0, t_1) = n$, and $N(L_0, t_2) = \rho$, then we want to find a $\mathbf{M}$ with $\alpha_{\mathbf{M}} = |K| + \rho$. If we can find such $\mathbf{M}$, the K-Set Union problem must be solvable. □

According to Lemma 4.2, it is intractable to examine the whole graph to get $SI_{\mathbf{M}}$ for large networks (such as `Houston` with 59 million edges in Sect. 5). Hence, instead we will look at each location independently to find $SI_{\mathbf{M}}$, and aggregate the result to generate $\mathbf{M}$.

Let us define $\alpha_{\mathbf{M},\ell}$ as the number of nodes that have at least one parent in $\mathbf{M}$ at location L_ℓ. Similar to $\alpha_{\mathbf{M}}$, we have $\alpha_{\mathbf{M},\ell} \leq \alpha^*_{\mathbf{M},\ell}$ where $\alpha^*_{\mathbf{M},\ell} = \sum_{i=1}^{t_{\max}} N(L_\ell, t_i)$. $\alpha^*_{\mathbf{M},\ell}$ is the number of patients after t_0 at location L_ℓ in R, and it is the optimal value for $\alpha_{\mathbf{M},\ell}$. Since we want to find a set of starting nodes, here we define S_ℓ as a node set at location L_ℓ: i.e., $S_\ell = \{v | v \in S \text{ and } v \in V_\ell\}$ where $|S_\ell| = N(L_\ell, t_0)$. For each location L_ℓ, we want to find a set S_ℓ as the starting infected node set, such that S_ℓ will yield a cascade $\mathbf{M}$ that minimizes $\alpha_{\mathbf{M},\ell}$. Our idea is to find S_ℓ that satisfies a necessary condition for the best $\alpha_{\mathbf{M},\ell}$. We denote $CF(S_\ell, t_i) = |\{u | u \in V_l, \exists v \in S_\ell, dist(v, u) \leq i\}|$, i.e., the number of nodes that S_ℓ can reach within distance i (i-hops) in L_ℓ in G. Similarly, we denote $CN(L_\ell, t_i) = \sum_{k=0}^{i} N(L_\ell, t_i)$ (the cumulative number of infections in L_ℓ in R until time t_i). The next lemma will show that for each location L_ℓ, when $\alpha_{\mathbf{M},\ell} = \alpha^*_{\mathbf{M},\ell}$, the constraint in Eq. 3 must be satisfied.

Lemma 4.3 (Necessary condition) *Given a cascade* $\mathbf{M}$ *generating from* S_ℓ, *if* $\alpha_{\mathbf{M},\ell} = \alpha^*_{\mathbf{M},\ell}$, *then for any timestep* $t_i \in [0, t_{\max}]$ *and all locations* L_ℓ, *we have*

$$CF(S_\ell, t_i) \geq CN(L_\ell, t_i) \tag{3}$$

Proof If $\alpha_{\mathbf{M},\ell} = \alpha^*_{\mathbf{M},\ell}$, every node that is infected after t_0 has a parent. For any node u that is infected at t_i, u must be within the ith hops of S_ℓ, which means the number of nodes within the i-hops of S_ℓ is greater than the number of nodes infected at t_i, i.e., $CF(S_\ell, t_i) \geq CN(L_\ell, t_i)$. □

Lemma 4.3 demonstrates a necessary condition (Eq. 3) for the maximum $\alpha_{\mathbf{M},\ell}$. Hence, we seek to develop an efficient algorithm that can produce accurate results for the necessary condition. Our idea is to construct a new objective function, which can get the necessary condition for the best $\mathbf{M}$ at location L_ℓ. To do so, we propose the following problem to find $SI_{\mathbf{M}}$:

Problem 4.1 Given graph G and infection matrix $\mathbf{N}$, find $S^* = \{S^*_1, \ldots, S^*_n\}$ s.t., $|S^*_\ell| = N(L_\ell, t_0)$ for any location L_ℓ, such that

$$S^*_\ell = \arg\min_{S_\ell} \theta(S_\ell) \quad \forall \text{ location } L_\ell,$$

where

$$\theta(S_\ell) = \sum_{i=0}^{t_{\max}} \mathbb{1}_{CF(S_\ell,t_i)<CN(L_\ell,t_i)}(CN(L_\ell, t_i) - CF(S_\ell, t_i)).$$

Here $\mathbb{1}_{CF(S_\ell,t_i)<CN(L_\ell,t_i)}$ is an indicator function: if $CF(S_\ell, t_i) < CN(L_\ell, t_i)$ then it is 1, otherwise 0.

Justification of Problem 4.1. Recall that $\alpha^*_{\mathbf{M},\ell}$ is the optimal value for $\alpha_{\mathbf{M},\ell}$, and $\theta(S_\ell)$ is nonnegative. We have the following lemma to connect $\theta(S_\ell) = 0$ with the optimal $\alpha_{\mathbf{M},\ell}$.

Lemma 4.4 *If $\alpha_{\mathbf{M},\ell}$ is optimal, then $\theta(S_\ell) = 0$.*

Proof In the appendix. □

Lemma 4.4 shows that if we minimize $\theta(S_\ell)$, we are able to get the necessary condition for the best $\mathbf{M}$ at location L_ℓ. Therefore, we propose Problem 4.1 to get $SI_{\mathbf{M}}$.
Hardness Problem 4.1 is NP-hard, as it can be reduced from the set cover problem [14].
Solving Problem 4.1 Let us define $g(S_\ell) = [\sum_{i=0}^{t_{\max}} CN(L_\ell, t_i)] - \theta(S_\ell)$. $\sum_{i=0}^{t_{\max}} CN(L_\ell, t_i)$ is constant, so minimizing $\theta(S_\ell)$ is equivalent to maximizing $g(S_\ell)$. The next lemma will show that $g(S_\ell)$ has interesting properties, which can help us get a near-optimal approximate solution for it.

Lemma 4.5 *$g(S_\ell)$ has the following properties: $g(\emptyset) = 0$; it is monotonic increasing and submodular.*

Proof In the appendix. □

Given the properties of $g(S_\ell)$ in Lemma 4.5, we can develop a natural greedy algorithm to solve Problem 4.1. with a provable guarantee (Lemma 4.6). We call it SAMPLENAIVEGREEDY: Each time it picks a node u^* such that

$$u^* = \arg\max_{u \in V_\ell} \; g(S_\ell \cup \{u\}) - g(S_\ell),$$

until $N(L_\ell, t_0)$ nodes have been selected to S_ℓ. We do it for all locations to get $SI_{\mathbf{M}}$.

Lemma 4.6 *For each location L_ℓ,* SAMPLENAIVEGREEDY *gives a $(1 - 1/e)$-approximate solution to $g(S_\ell)$.*

Proof Minimizing $\theta(S_\ell)$ is equivalent to maximizing $g(S_\ell) = (\sum_{i=1}^{t_{\max}} CN(L_\ell, t_i)) - \theta(S_\ell)$ as $\sum_{i=1}^{t_{\max}} CN(L_\ell, t_i)$ is constant. $g(S_\ell)$ has the following properties: (1) $g(\emptyset) = 0$; (2) it is monotonic increasing; (3) it is a submodular function. Hence, the greedy algorithm to maximize $g(S_\ell)$ gives a $(1 - 1/e)$-approximate solution [15]. □

SAMPLENAIVEGREEDY selects a node with the maximum marginal gain of $g(S_\ell)$ iteratively. It gives us a $(1 - 1/e)$ approximate solution; however, it takes $O(|V|(|V| + |E|))$ time if we run BFS to get each $CF(S_\ell, t_i)$ for each iteration. The time complexity to get all $|\mathbf{N}(t_0)|_1$ nodes as $SI_{\mathbf{M}}$ is $O(|\mathbf{N}(t_0)|_1|V|(|V| + |E|))$, which is not scalable to large networks. Hence, we need a faster algorithm.

Speeding up SAMPLENAIVEGREEDY In SAMPLENAIVEGREEDY, each time we recompute $CF(S_\ell \cup \{u\}, t_i)$ for all i, which takes $O(|E| + |V|)$ time. We can speed up this computation by leveraging the ANF (Approximate Neighborhood Function) algorithm [16], which uses a classical probabilistic counting algorithm, the Flajolet–Martin algorithm [17] to approximate the sizes of union-ed node sets using bit strings. Here, we refer to the bit string that approximates $CF(S_\ell, t_i)$ as $\mathbb{F}(S_\ell, i)$. To estimate $CF(S_\ell \cup \{u\}, t_i)$, we first do a bitwise-OR operation: $\mathbb{F}(S_\ell \cup \{u\}, i) = [\mathbb{F}(S_\ell, i) \,\text{OR}\, \mathbb{F}(\{u\}, i)]$, then convert it to $CF(S_\ell \cup \{u\}, t_i)$. According to the ANF algorithm, $CF(\cdot, t_i) = \phi(\mathbb{F}(\cdot)) = (2^b)/.77351$, where b is the average position of the leftmost zero bit of the bit string. Since the bitwise-OR operation takes constant time, we can reduce the running time of $CF(S_\ell \cup \{u\}, t_i)$ for all timesteps i from $O(|E| + |V|)$ to $O(t_{\max})$.

We propose SAMPLEGREEDY (Algorithm 1), a modified greedy algorithm with bitwise-OR operations for Problem 4.1. It first gets $\mathbb{F}(\{u\}, i)$ for all nodes at location L_ℓ over all timesteps using ANF [16] (Line 2), then follows SAMPLENAIVEGREEDY. However, we use bitwise-OR operations to speed up the computation of $CF(S_\ell \cup \{u\}, t_i)$ (Line 7–8).

Algorithm 1 SAMPLEGREEDY

Require: graph G, and propagation log matrix $\mathbf{N}$.
1: **for** each location L_ℓ **do**
2: Get $\mathbb{F}(\{u\}, i)$ for all timestep i, all $u \in V_\ell$ using ANF [16]
3: $y = N(L_\ell, t_0)$
4: $S_\ell = \emptyset$, and $\mathbb{F}(S_\ell, i) = 0$ for all timesteps i
5: **for** $i = 1$ to y **do**
6: **for** each node $u \in V_\ell - S_\ell$ **do**
7: $\mathbb{F}(S_\ell \cup \{u\}, i) = \mathbb{F}(S_\ell, i)$ OR $\mathbb{F}(\{u\}, i)$ for all t_i
8: $CF(S_\ell \cup \{u\}, t_i) = \phi(\mathbb{F}(S_\ell \cup \{u\}, i))$ for all t_i
9: **end for**
10: $u^* = \arg\max_{u \in V_\ell - S_\ell} g(S_\ell) - g(S_\ell \cup \{u\})$
11: $S_\ell = S_\ell \cup u^*$
12: **end for**
13: **end for**
14: **return** $SI_{\mathbf{M}} = \{S_1, \ldots, S_n\}$

Lemma 4.7 SAMPLEGREEDY *takes* $O((|V||\mathbf{N}(t_0)|_1 + |E|)t_{\max})$ *time.*

Proof Computing all $\mathbb{F}(\{u\}, i)$ over all locations takes $O((|V| + |E|)t_{\max})$ according to the ANF algorithm. Since bitwise-OR operation takes constant time, hence, Line 8 and 9 takes $O(t_{\max})$ time, and Line 7–10 takes $O(|V|t_{\max})$ time. Since $||\mathbf{N}(t_0)||_1$ is the total number of starting infected nodes, it takes $||\mathbf{N}(t_0)||_1|V|t_{\max}$ time to pick total $||\mathbf{N}(t_0)||_1$ nodes. Hence, the overall running time is $O((|V|||\mathbf{N}(t_0)||_1 + |E|)t_{\max})$. □

Generating cascades from $SI_{\mathbf{M}}$ Once we obtain $SI_{\mathbf{M}}$ from Algorithm 1, we can generate $\mathbf{M}$ from $SI_{\mathbf{M}}$. Similar to the result of Lemma 4.2, generating $\mathbf{M}$ from $SI_{\mathbf{M}}$ is also hard. Here we propose a heuristic, the CASCADEGENERATION algorithm (Algorithm 2) for $\mathbf{M}$. Let us define $D_i^\ell = \{u | u \in V_\ell, \exists v \in SI_{\mathbf{M}}, dist(v, u) = i\}$, i.e., a set of nodes in location L_ℓ that $SI_{\mathbf{M}}$ can reach at distance i. We first add $SI_{\mathbf{M}}$ to the cascade $\mathbf{M}$, and compute D_i^ℓ for all time t_i and location L_ℓ by running a BFS starting from $SI_{\mathbf{M}}$ (Line 2). Then, we select nodes into $\mathbf{M}$ by running another BFS from $SI_{\mathbf{M}}$ as well: at each distance i from $SI_{\mathbf{M}}$, for each location L_ℓ we pick $N(L_\ell, t_i)$ nodes uniformly at random to $\mathbf{M}$, and add corresponding edges (Line 4–18). Note that we do it by permutating the set D_i^ℓ. $N(L_\ell, t_i)$ nodes are selected as follows: (1) if $|\text{CANDIDATEQUEUE}_l| \geq N(L_\ell, t_i)$ (the constraint in Eq. 3 follows), we pick $N(L_\ell, t_i)$ nodes uniformly at random, and add them to $\mathbf{M}$ from CANDIDATEQUEUE (Line 8–10); (2) otherwise, we add all nodes in CANDIDATEQUEUE to $\mathbf{M}$, record the number of nodes left (Line 11–12), and finally randomly pick other nodes from V_ℓ, and add to $\mathbf{M}$ (Line 18).

Lemma 4.8 CASCADEGENERATION *takes* $O(|V| + |E|)$ *time.*

Proof Running BFS takes $O(|V| + |E|)$ (Line 2). For each timestep t at each nodes L_i, we check the nodes in $\hat{D}_i^\ell$; hence, overall we just need to traverse the nodes once, which takes linear time (Line 4–17). Hence, CASCADEGENERATION takes $O(|V| + |E|)$ time. □

Algorithm 2 CASCADEGENERATION

Require: Graph G, propagation log matrix $\mathbf{N}$, and node set $SI_{\mathbf{M}}$
1: Add all nodes in $SI_{\mathbf{M}}$ to the cascade $\mathbf{M}$
2: Compute D_i^ℓ for all time t_i (by running BFS from $SI_{\mathbf{M}}$)
3: PRESET = $SI_{\mathbf{M}}$, NUMLEFTNODE=0
4: **for** $i = 1$ to $t_{\max}$ **do**
5: **for** each location L_ℓ **do**
6: $\hat{D}_i^\ell = \texttt{Permutate}(D_i^\ell)$
7: Add $\hat{D}_i^\ell$ to the end of CANDIDATEQUEUE$_\ell$
8: **if** |CANDIDATEQUEUE$_\ell$| $\geq N(L_\ell, t_i)$ **then**
9: CURSET=pop $N(L_\ell, t_i)$ nodes from the top of CANDIDATEQUEUE$_\ell$
10: **else**
11: CURSET=pop all nodes in CANDIDATEQUEUE$_\ell$
12: NUMLEFTNODE+=($N(L_\ell, t_i)$− |CANDIDATEQUEUE|$_\ell$)
13: **end if**
14: Add CURSET to $\mathbf{M}$, and edges from PRESET to CURSET if $e(u, v) \in G$ for any $u \in$ PRESET and $v \in$ CURSET
15: **end for**
16: PRESET=CURSET
17: **end for**
18: Uniformly randomly pick NUMLEFTNODE nodes from V_ℓ to $\mathbf{M}$
19: **return** $\mathbf{M}$

Extend CASCADEGENERATION to k cascades We can simply extend Algorithm 2 to k cascades. Note that CASCADEGENERATION permutes the nodes in D_i^ℓ (Line 6); hence, for different permutations, we can generate different cascades. If the constraint in Eq. 3 holds, at time t_i, we add $N(L_\ell, t_i)$ nodes uniformly at random into $\mathbf{M}$ from $\sum_{j=1}^{i} |D_i^\ell| - \sum_{j=1}^{i-1} N(L_\ell, t_j)$ candidate nodes. If the constraint does not follow, we pick extra nodes from $V - V_{\mathbf{M}}$ uniformly at random, and add them to $\mathbf{M}$.

Remark 4.1 The above random process will generate $O(\prod_{L_\ell \in L} \prod_i |D_i^\ell|)$ cascades.

Remark 4.1 shows that we have a large number of cascades. In case if we need more, we can generate extra cascades by ranking the result of SAMPLEGREEDY: instead of picking the best S_ℓ, we pick the top sets (in Algorithm 1 Line 10–11). In practice, as shown in our experiments, we do not need to do this, as we have enough cascades. In addition, our cascades have high quality: The average value of $\alpha_{\mathbf{M}}$ is almost the same as the optimal solution (Table 3).

MAPPINGGENERATION Combining the above results, we propose the MAPPINGGENERATION algorithm (Algorithm 3) to solve Problem 3.1.

Claim 4.1 The time complexity of MAPPINGGENERATION (Algorithm 3) is $O((|V||\mathbf{N}(t_0)|_1 + |E|)t_{\max} + \hat{k}(|V| + |E|))$, where $\hat{k}$ is the number of runs for CASCADEGENERATION to get k cascades.

Algorithm 3 MAPPINGGENERATION

Require: graph G, propagation log R
1: Generate propagation log matrix $\mathbf{N}$
2: Run SAMPLEGREEDY (G, $\mathbf{N}$) (Algorithm 1) to get $SI_{\mathbf{M}}$
3: RUNCASCADEGENERATION (G, $\mathbf{N}$, $SI_{\mathbf{M}}$) (Algorithm 2) until k unique cascades are found for $\mathcal{M}$
4: **return** $\mathcal{M}$.

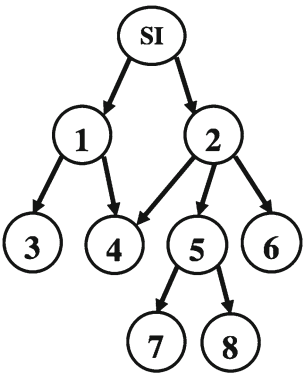


Fig. 2 Counter-example for the diminishing return property of $\rho_G(\mathbf{x}, \mathbf{M}_i)$

4.2 Data-driven immunization

Main idea In this section, we solve the data-driven immunization (Problem 3.2) assuming the samples are available. We first show that $\rho_{G,\mathbf{M}_i}(\mathbf{x})$ in Problem 3.2 is neither submodular nor supermodular. We then propose to optimize an alternative credit-based objective function, which is an upperbound of $\rho_{G,\mathbf{M}_i}(\mathbf{x})$ (Problem 4.2). We show that this function is nonnegative, increasing and has the diminishing return property. Based on these properties, we propose a greedy algorithm which gives a $(1 - 1/e)$-approximate solution.

Note that in Problem 3.2, $\rho_{G,\mathbf{M}_i}(\mathbf{x})$ is defined over an integer lattice, and is not a simple set function. If a function $h(\mathbf{x})$ has the diminishing return property over an integer lattice, then for any $\mathbf{x}' \geq \mathbf{x}$ and k, we have $h(\mathbf{x} + \mathbf{e}_k) - h(\mathbf{x}) \geq h(\mathbf{x}' + \mathbf{e}_k) - h(\mathbf{x}')$ ($\mathbf{e}_k$ be the vector with 1 at the kth index). According to [4], there exists a near-optimal algorithm to maximize $h(\mathbf{x})$. Unfortunately, $\rho_{G,\mathbf{M}_i}(\mathbf{x})$ does not follow the diminishing return property.

Remark 4.2 $\rho_G(\mathbf{x}, \mathbf{M}_i)$ does not have diminishing return property. Figure 2 shows a counter-example, where all nodes are in different locations. Suppose $\mathbf{x} = \mathbf{0}$, $\mathbf{x}' = \mathbf{e}_1$, then $\mathbf{x} \leq \mathbf{x}'$; however, $\rho_{G,\mathbf{M}_i}(\mathbf{x} + \mathbf{e}_2) - \rho_{G,\mathbf{M}_i}(\mathbf{x}) = 5$ and $\rho_{G,\mathbf{M}_i}(\mathbf{x}' + \mathbf{e}_2) - \rho_{G,\mathbf{M}_i}(\mathbf{x}') = 8 - 2 = 6$.

Instead, we develop a *contribution*-based approach. The idea is if we remove a node u in $\mathbf{M}_i$, the number of nodes u can save is related to u's children. Each child of u can contribute to the savings of removing u. First, let us denote $IN_{\mathbf{M}_i}(S)$ as the set of S's parents in $\mathbf{M}_i$, i.e., $IN_{\mathbf{M}_i}(S) = \{u | e(u, v) \in \mathbf{M}_i, v \in S\}$, and $OUT_{\mathbf{M}_i}(S)$ as the set of S's children in $\mathbf{M}_i$. We define the contribution $C_{G,\mathbf{M}_i}(S)$ recursively,

$$C_{G,\mathbf{M}_i}(S) = |S| + \sum_{v \in OUT_{\mathbf{M}_i}(S)} \frac{|IN_{\mathbf{M}_i}(\{v\}) \cap S|}{|IN_{\mathbf{M}_i}(\{v\})|} C_{G,\mathbf{M}_i}(\{v\}).$$

$\frac{|IN_{\mathbf{M}_i}(\{v\}) \cap S|}{|IN_{\mathbf{M}_i}(\{v\})|}$ is the fraction of savings v contributes to S, and $C_{G,\mathbf{M}_i}(\{v\}) = 1$. The intuition is that since we do not have any propagation models, it is reasonable to assume the infected v should be infected by any of its parents equally; hence, v contributes its savings *equally* to each of its parents. Now we define the contribution function over an integer lattice,

$$\zeta_{G,\mathbf{M}_i}(\mathbf{x}) = \sum_S \Pr(S) C_{G,\mathbf{M}_i}(S), \tag{4}$$

where S is a node set sampled from the random process of distributing $\mathbf{x}$ ($|S| = |\mathbf{x}|_1$). Lemma 4.9 shows that $\zeta_{G,\mathbf{M}_i}(\mathbf{x})$ is the upperbound of $\rho_{G,\mathbf{M}_i}(\mathbf{x})$.

Lemma 4.9 *Given a cascade* $\mathbf{M}_i$, $\rho_{G,\mathbf{M}_i}(\mathbf{x}) \leq \zeta_{G,\mathbf{M}_i}(\mathbf{x})$.

Proof In the appendix. □

We use $\zeta_{G,\mathbf{M}_i}(\mathbf{x})$ to estimate $\rho_{G,\mathbf{M}_i}(\mathbf{x})$. Hence, we formally define the following problem for Problem 3.2.

Problem 4.2 Given a contact network $G(V, E)$, a set of cascades $\mathcal{M}$, and budget m, find a vaccine allocation vector $\mathbf{x}^*$:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \frac{1}{|\mathcal{M}|} \sum_{\mathbf{M}_i \in \mathcal{M}} \zeta_{G,\mathbf{M}_i}(\mathbf{x}), \text{ s.t.} |\mathbf{x}|_1 = m. \tag{5}$$

$\zeta_{G,\mathbf{M}_i}(\mathbf{x})$ has interesting properties (as shown in the following Lemma 4.10), which can lead us to a near-optimal solution for Problem 4.2 (Lemma 4.11).

Lemma 4.10 $\zeta_{G,\mathbf{M}_i}(\mathbf{x})$ *has the following properties:*

(P_1) $\zeta_{G,\mathbf{M}_i}(\mathbf{x}) \geq 0$ *and* $\zeta_{G,\mathbf{M}_i}(\mathbf{0}) = 0$.
(P_2) *(Non-decreasing)* $\zeta_{G,\mathbf{M}_i}(\mathbf{x}) \leq \zeta_{G,\mathbf{M}_i}(\mathbf{x} + \mathbf{e}_i)$ *for* i.
(P_3) *(Diminishing returns) For any* $\mathbf{x}' \geq \mathbf{x}$, *we have* $\zeta_{G,\mathbf{M}_i}(\mathbf{x}+\mathbf{e}_i) - \zeta_{G,\mathbf{M}_i}(\mathbf{x}) \geq \zeta_{G,\mathbf{M}_i}(\mathbf{x}' + \mathbf{e}_i) - \zeta_{G,\mathbf{M}_i}(\mathbf{x}')$.

Proof (P_1) is trivially true because $C_{G,\mathbf{M}_i}(S) \geq 0$ and when $\mathbf{x} = \mathbf{0}$, $S = \emptyset$. For (P_2), when we add $\mathbf{e}_i$, it means that we add one more vaccine to $\mathbf{M}_i$. Note that when we add $\{u\}$ to a node set S, clearly $C_{G,\mathbf{M}_i}(S \cup \{u\}) \geq C_{G,\mathbf{M}_i}(S)$. Let us assume $C_{G,\mathbf{M}_i}(S \cup \{u\}) = C_{G,\mathbf{M}_i}(S) + \delta_u$ where $\delta_u \geq 0$. When the allocation is $\mathbf{x}+\mathbf{e}_i$, we can think the process as follows: we first give $\mathbf{x}$ vaccines, then we allocate the last vaccine $\mathbf{e}_i$. Hence, $\zeta_{G,\mathbf{M}_i}(\mathbf{x}+\mathbf{e}_i) = \sum_S \Pr(S)[C_{G,\mathbf{M}_i}(S) + \sum_{\{u\}} \Pr(\{u\})\delta_u) \geq \zeta_{G,\mathbf{M}_i}(\mathbf{x})$. For ($P_3$), it follows the proof of Lemma 1 in [4]. □

Given the properties of $\zeta_{G,\mathbf{M}_i}(\mathbf{x})$ in Lemma 4.10, we propose a greedy algorithm, ImmuNaiveGreedy for Problem 4.2: each time we give one vaccine to location L_{ℓ^*}, such that

$$\ell^* = \arg\max_{L_\ell} \sum_{\mathbf{M}_i \in \mathcal{M}} \zeta_{G,\mathbf{M}_i}(\mathbf{x} + \mathbf{e}_\ell) - \zeta_{G,\mathbf{M}_i}(\mathbf{x}),$$

until m vaccines are allocated.

Lemma 4.11 ImmuNaiveGreedy *gives a* $(1 - 1/e)$*-approximate solution to Problem 4.2.*

Proof In the appendix. □

In ImmuNaiveGreedy, since we distribute vaccines uniformly at random, we can apply the Sample Average Approximation (SAA) framework, i.e., $\zeta_{G,\mathbf{M}_i}(\mathbf{x}) \approx \frac{1}{|\mathcal{S}|} \sum_{S \in \mathcal{S}} C_{G,\mathbf{M}_i}(S)$, where $\mathcal{S}$ is a set of samples taken from the vaccine allocation process. This approach takes $O(|\mathcal{S}|(|V| + |E|))$ to estimate $\zeta_{G,\mathbf{M}_i}(\mathbf{x})$, and we need to look into $|\mathcal{M}|$ cascades to pick the best location L_{ℓ^*} for one iteration. We have $|L|$ locations and m vaccines. Hence, the total time complexity of ImmuNaiveGreedy is $O(m|L||\mathcal{M}||\mathcal{S}|(|V| + |E|))$, which is not practical for large networks. However, we can speed up this naive greedy algorithm.

Speeding up ImmuNaiveGreedy We propose a faster algorithm, ImmuConGreedy (*Contribution-based Greedy Immunization*) in Algorithm 4, which takes only $O(m|\mathcal{M}|$

$(|V| + |E|))$ time. The idea is that we can compute the contribution function efficiently when the budget $m = 1$, i.e., all values of $\zeta_{G,\mathbf{M}_i}(\mathbf{e}_\ell)$ in $\mathbf{M}_i$ can be obtained in $O(|V| + |E|)$ time. This is because $\zeta_{G,\mathbf{M}_i}(\mathbf{e}_\ell) = \sum_{u \in V_\ell} \frac{1}{|L_\ell|} C_{G,\mathbf{M}_i}(\{u\})$, and we can get $C_{G,\mathbf{M}_i}(\{u\})$ for all $u \in V$ by traversing $\mathbf{M}_i$ once. For simplicity, let $d_{in}(v) = |IN_{\mathbf{M}_i}(\{v\})|$. We have $C_{G,\mathbf{M}_i}(\{u\}) = 1 + \sum_{v \in OUT_{\mathbf{M}_i}(\{u\})} \frac{1}{d_{in}(v)} C_{G,\mathbf{M}_i}(\{v\})$. If u does not have any children ($OUT_{\mathbf{M}_i}(\{u\}) = \emptyset$), $C_{G,\mathbf{M}_i}(\{u\}) = 1$. Since $\mathbf{M}_i$ is a DAG, we can iteratively obtain $C_{G,\mathbf{M}_i}(\{u\})$ for all $u \in V$ from a reversed order of a topological sort, which takes $O(|V|+|E|)$ time.

In Algorithm 4, we compute contribution function $C_{G,\mathbf{M}_i}(\{u\})$ for all $\mathbf{M}_i$ (Line 4), which takes $O(|\mathcal{M}|(|V| + |E|))$ time. Then, we obtain $\sum_{\mathbf{M}_i \in \mathcal{M}} \zeta_{G,\mathbf{M}_i}(\mathbf{e}_\ell)$ for each location L_ℓ by summing up the contribution for each $u \in V_\ell$ (Line 5), which takes $O(|\mathcal{M}||V|)$ time. Once we allocate one vaccine to the best location L_{ℓ^*}, we update each $\mathbf{M}_i$ by uniformly at random removing one node in L_{ℓ^*} (Line 7). This way we can just compute $\sum_{\mathbf{M}_i \in \mathcal{M}} \zeta_{G,\mathbf{M}_i}(\mathbf{e}_\ell)$ instead of $\sum_{\mathbf{M}_i \in \mathcal{M}} \zeta_{G,\mathbf{M}_i}(\mathbf{x} + \mathbf{e}_\ell)$ after the next iteration.

Algorithm 4 ImmuConGreedy

Require: graph $G(V, E)$, propagation log R, and budget m
1: $\mathcal{M}$ =MappingGeneration (G, R) {Section 4.1}
2: $\mathbf{x} = 0$
3: **for** $j = 1$ to m **do**
4: $\forall \mathbf{M}_i \in \mathcal{M}$: compute $C_{G,\mathbf{M}_i}(\{u\})$ for each node u
5: $\forall$ location $L_\ell \in L$: compute $\sum_{\mathbf{M}_i \in \mathcal{M}} \zeta_{G,\mathbf{M}_i}(\mathbf{e}_\ell)$
6: $\ell^* = \arg\max_{L_\ell} \sum_{\mathbf{M}_i \in \mathcal{M}} \zeta_{G,\mathbf{M}_i}(\mathbf{e}_\ell)$
7: $\forall \mathbf{M}_i \in \mathcal{M}$: update $\mathbf{M}_i$ by uniformly at random removing one node at location L_{ℓ^*}
8: $\mathbf{x} = \mathbf{x} + \mathbf{e}_{\ell^*}$
9: **end for**
10: **return** $\mathbf{x}$

Lemma 4.12 ImmuConGreedy *takes* $O(m|\mathcal{M}|(|V| + |E|))$ *time.*

Proof First, for simplicity, let $d_{in}(v) = |IN_{\mathbf{M}_i}(\{v\})|$. Since $C_{G,\mathbf{M}_i}(\{u\}) = 1 + \sum_{v \in OUT_{\mathbf{M}_i}(\{u\})} \frac{1}{d_{in}(v)} C_{G,\mathbf{M}_i}(\{v\})$, if u does not have any children ($OUT_{\mathbf{M}_i}(\{u\}) = \emptyset$), clearly $C_{G,\mathbf{M}_i}(\{u\}) = 1$. Note that $\mathbf{M}_i$ is a DAG, we can iteratively obtain $C_{G,\mathbf{M}_i}(\{u\})$ from a reversed order of a topological sort, which takes $O(|V| + |E|)$ time. Hence, Line 4 takes $O(|\mathcal{M}|(|V| + |E|))$ time.

Second, $\zeta_{G,\mathbf{M}_i}(\mathbf{e}_l) = \sum_{\{v\} \in |L_\ell|} \Pr(\{v\}) C_{G,\mathbf{M}_i}(\{v\})$. Since we uniformly at random give vaccines to locations, $\Pr(\{v\}) = \frac{1}{|L_\ell|}$. Hence, $\zeta_{G,\mathbf{M}_i}(\mathbf{e}_l) = \sum_{v \in L_k} \frac{1}{|L_k|} C_{G,\mathbf{M}_i}(\{v\})$. Hence, Line 5 takes $O(\mathcal{M}|V|)$ time.

Third, Lines 6 and 7 take $|L|$ and $O(|\mathcal{M}|)$ time, respectively. Hence, the overall running time is $O(m|\mathcal{M}|(|V| + |E|))$. □

5 Experiments

We conducted the experiments using a 4 Xeon E7-4850 CPU with 512 GB of 1066 MHz main memory.[1]

[1] Code in Python: https://goo.gl/tsMueB.

Table 2 Network datasets

Dataset	Nodes	Edges	Locations
WorkPlace	92	757	5
HighSchool	182	2221	5
SBM	1000	5000	20
Portland	1.5 million	41 million	72
Miami	2.2 million	50 million	74
Houston	2.7 million	59 million	98

5.1 Experimental setup

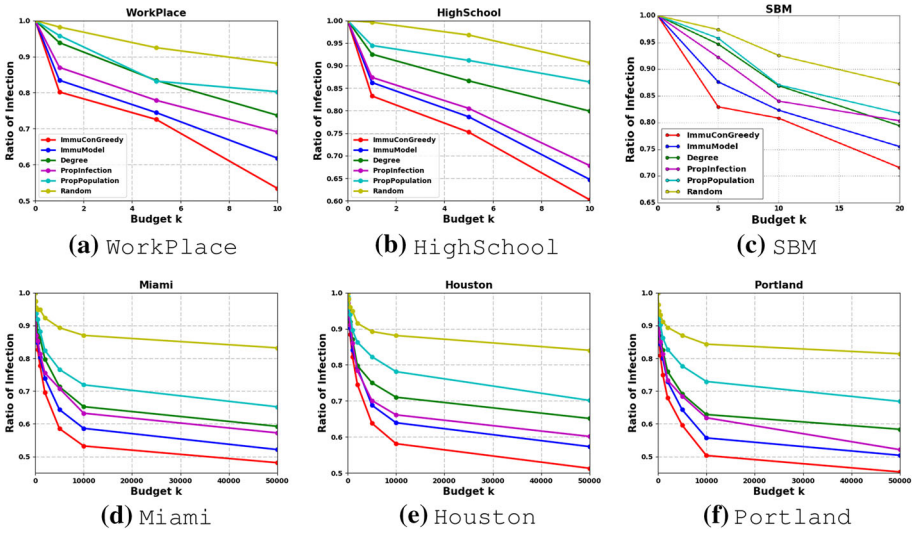
Networks We do experiments on multiple datasets (Table 2). Stochastic Block Model (SBM) [18] is a well-known graph model to generate synthetic graphs with groups. WorkPlace and HighSchool are social-contact networks.[2] Nodes in HighSchool are students from five different sections and edges represent two students who are in vicinity of each other. Nodes in WorkPlace are employees of a company with five departments and edges indicate two people are in proximity of each other. We treat each section/department as a location. Miami and Houston are million-node social-contact graphs from city-scale activity-based synthetic populations as described in Sect. 2. We divided people by their residential zipcodes.

Propagation logs We have the billion-record eHRC data (described in Sect. 2) as the propagation log R for Miami and Houston. The Miami and Houston have $118K$ and $132K$ patients, respectively. For SBM, HighSchool, and WorkPlace, we run the well-known SIR model (infection rate as 0.4, and recovery rate as 0.6) to generate the propagation log R: We first uniformly at random pick 5% nodes at each location as seeds at t_0, then run a SIR simulation to get other infected nodes.

Settings We set the number of samples $|\mathcal{M}| = 1000$ for MappingGeneration, and number of bitmasks as 32 for computing $\mathbb{F}(\cdot)$ in SampleGreedy (similar to the ANF algorithm [16]).

Baselines As we are not aware of any direct competitor tackling our problem, we use several baselines to better judge our performance. These baselines have been regularly used for immunization studies. However, none of them take into account both propagation log and contact networks.
(1) Random: uniformly randomly assign vaccines to locations.
(2) PropPopulation: a data-based approach: assign vaccines to locations in proportion to population in locations.
(3) PropInfection: a data-based approach: assign vaccines in proportion to total number of infections in locations.
(4) Degree: a graph-based approach: calculate the average degree d_{L_i} of each location L_i, and independently assign vaccines to L_i with probability $d_{L_i} / \sum_{L_k \in L} d_{L_k}$.
(5) ImmuModel: a model-based approach: apply the *model-driven group immunization* algorithm (the QP version) in [4]. ImmuModel aims to minimize the spectral radius of a contact graph. Spectral radius is the first eigenvalue of the graph, which has been proven to be the threshold of an epidemic in the graph [11]. We set edge weights to be 0.24 according to [8].

[2] http://www.sociopatterns.org.



(a) WorkPlace **(b)** HighSchool **(c)** SBM

(d) Miami **(e)** Houston **(f)** Portland

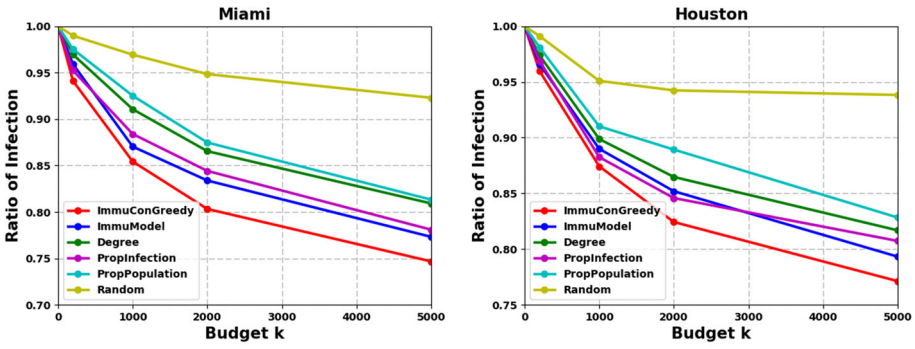
Fig. 3 Effectiveness of ImmuConGreedy on the whole R. Infection ratio r versus vaccine budget m. Infection ratio $r = \frac{\sum_{\mathbf{M}_i \in \mathcal{M}} \sigma_{G,\mathbf{M}_i}(\mathbf{x})}{\sum_{\mathbf{M}_i \in \mathcal{M}} \sigma_{G,\mathbf{M}_i}(\mathbf{0})}$. Lower is better. ImmuConGreedy consistently outperforms other baselines over all datasets

5.2 Results

In short, we demonstrate that our immunization algorithm ImmuConGreedy outperforms other baselines on all datasets. We also show our approach is robust as the size of the propagation log R varies. In addition, we show that our sampling algorithm SampleGreedy provides accurate results for generating cascade samples. Finally, we study the scalability of our approach.

Effectiveness of ImmuConGreedy Figure 3 shows results of minimizing the spread on cascades for the whole log R. In all datasets, ImmuConGreedy consistently outperforms others. WorkPlace and HighSchool have < 200 nodes; hence, we varied m till 10. SBM has $1K$ nodes, so we varied m till 20. However, even with the small budget, ImmuConGreedy can reduce 45% of the infection, which is about 10% better than the second best ImmuModel. For Miami, Houston and Portland with upto $2.7million$ nodes, ImmuConGreedy can reduce about 50% of the infection on the cascades generated by SocialContact with only $50K$ vaccines. Model-based ImmuModel and data-based PropInfection perform better than Random and Degree as they take into account either epidemic threshold in the contact graph or the eHRC data. However, ImmuConGreedy easily outperforms them, as it leverages both contact networks and the eHRC data.

We also study how to leverage the rich log data to develop vaccine interventions in the future. To do so, we split the eHRC data into training parts and testing parts: We get the vaccine allocations from the training parts (the fall regime of flu from August 2009 to October 2009), and apply the allocations to the testing parts (the winter regime of flu from November 2009 to February 2010) to examine how effective our approach ImmuConGreedy is. Figure 4 shows the results of infection reductions on the cascades generating from the testing data. ImmuConGreedy consistently outperforms others in both Miami and Houston: It can



(a) Miami **(b)** Houston

Fig. 4 Effectiveness of ImmuConGreedy for the testing data. Infection ratio r versus vaccine budget m. Lower is better. ImmuConGreedy consistently outperforms other baselines for both Miami and Houston

reduce about 25% of the infection with only $5K$ vaccines, compared to other baselines such as ImmuModel and PropInfection.

We use simulations of the SIR model to evaluate the performance of ImmuConGreedy on the activity-based urban social-contact networks (described in Sect. 2). These were first calibrated to get the same outbreak size as in the eHRC data for these cities. We then choose a random subset of individuals in each zipcode to be vaccinated, based on the allocation by ImmuConGreedy. We find the reduction in the number of infections is quite substantial in many cases. For instance, for Miami, for a budget of $50K$ vaccines, the ImmuConGreedy allocation leads to more than 50% reduction, compared to a random allocation. For Houston, when the budget is $50K$, ImmuConGreedy can lead to more than 38% reduction in the infection compared to ImmuModel and PropInfection.

Robustness of ImmuConGreedy We study how sensitive ImmuConGreedy is, as the size of the propagation log R varies next. To do so, we first generate synthetic propagation log R from the SIR model, then manually change the size of R as the input of our data. Finally, we compare ImmuConGreedy to the model-based approach ImmuModel. For each dataset, we generate R by running a SIR simulation (with the infection rate 0.4 and the recovery rate 0.6 for WorkPlace, HighSchool and SBM, and the infection rate 0.24 and timesteps to recovery 7 for Miami according to [8]). Once R is generated, we change the size of R by extracting a portion $[\mathbf{N}(t_0), \ldots, \mathbf{N}(t_{\max})]$ as the input ($p\%$ of R). For example, suppose $t_{\max} = 20$ and $p = 50$, we use $[\mathbf{N}(t_0), \ldots, \mathbf{N}(t_{10})]$ as the propagation log. Since we know all configurations come from the SIR model, we expect the model-based approach ImmuModel to do better than ImmuConGreedy. However, as p increases, as more data are used, ImmuConGreedy should approach ImmuModel. Figure 5 shows the results: as expected, for all datasets, clearly as p increases, ImmuConGreedy becomes better. Interestingly for smaller datasets such as WorkPlace, HighSchool, SBM, even with only 25% of data, we can get upto 85% of the performance. For large networks such as Miami, we need more data; however, when all the data are used, compared to ImmuModel, ImmuConGreedy can achieve 90% of the savings.

Number of cascades in ImmuConGreedy We study how ImmuConGreedy performs as the number of samples change. To do so, we first generate different number of cascades from the SIR model, then directly run ImmuConGreedy without MappingGeneration (Line 2–10 in Algorithm 4). Note that since the SIR simulation can generate true cascade, we skip

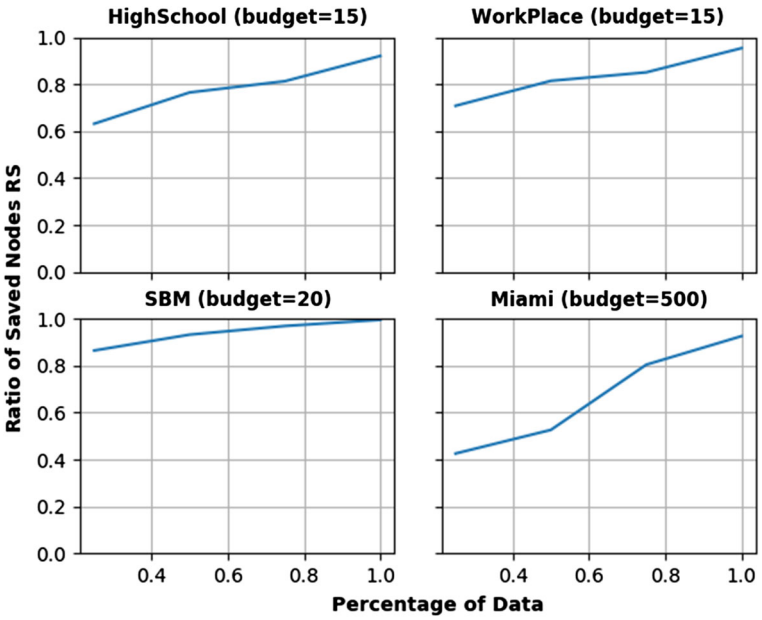


Fig. 5 Robustness of ImmuConGreedy as data size varies. Ratio of saved nodes RS versus percentage of used log data $p\%$. $RS = \frac{S_{\text{Data}}}{S_{\text{Model}}}$. S_{Data} (S_{Model}): the number of nodes we can save when vaccines are allocated according to ImmuConGreedy (ImmuModel). Percentage of used log data p: $[\mathbf{N}(t_0), \ldots, p\%\mathbf{N}(t_{\max})]$. Higher: ImmuConGreedy is closer to ImmuModel

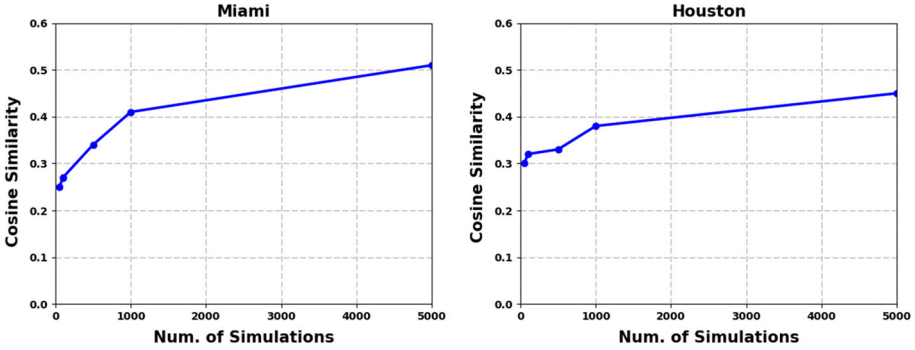
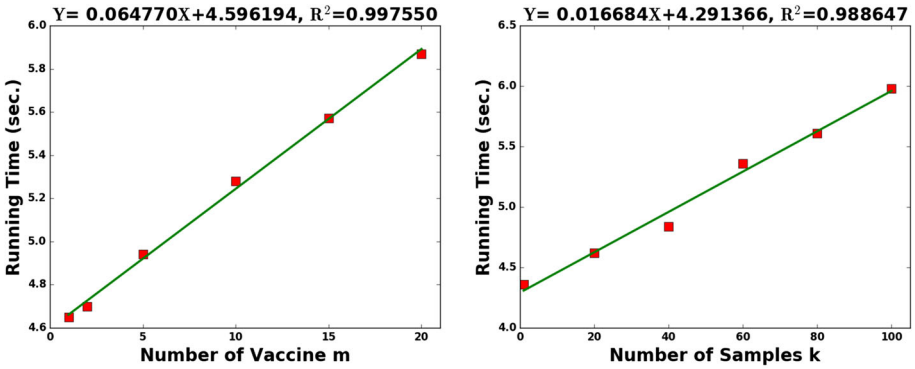


Fig. 6 Similarity of vaccine allocations as the number of simulations varies. The allocation vector of ImmuConGreedy is closer to ImmuModel as the simulation number increases

the process of generating cascades from the log R. Finally, we compare the cosine similarity of vaccine allocation vector to the one from the model-based approach ImmuModel. The intuition is that since our cascades are generated from simulations of the SIR model, as the number of cascades increases, vaccine allocated from ImmuConGreedy should be more similar to ImmuModel. Figure 6 shows the results for `Miami` and `Houston`: as expected, clearly the allocation vector from ImmuConGreedy is closer to the allocation from ImmuModel, as the number of the SIR simulations increases. When we generate $5K$ simulations, the cosine similarity is more than 50%.

Table 3 MAPPINGGENERATION. $\hat{\alpha}_{\mathcal{M}}$: average of $\alpha_{\mathbf{M}}$ over all $\mathbf{M} \in \mathcal{M}$; α^*: optimal value of $\alpha_{\mathbf{M}}$; $N = \sum_{t=t_1}^{t_{\max}} |\mathbf{N}(t)|_1$

Dataset	$\hat{\alpha}_{\mathcal{M}}$	α^*	N
WorkPlace	79.2	83.0	83
HighSchool	165.2	170.0	170
SBM	107.9	109.0	109



(a) Varying m **(b)** Varying k

Fig. 7 Scalability. **a** Total running time of MAPPINGGENERATION and IMMUCONGREEDY versus vaccine budget m; **b** total running time of MAPPINGGENERATION and IMMUCONGREEDY versus number of cascade samples k

Effectiveness of MAPPINGGENERATION We also study the performance of MAPPINGGENERATION by comparing $\alpha_{\mathbf{M}}$ to the optimal value α^* (Problem 3.1). We obtain α^* using the brute-force algorithm. See Table 3: $\hat{\alpha}_{\mathcal{M}}$, the average value of $\alpha_{\mathbf{M}}$ over all sampled cascades, is almost the same as α^* for all datasets. For example, in SBM, $\hat{\alpha}_{\mathcal{M}}$ is 107.9, a difference of only 1.1 from α^*. In addition, we found that α^* is exactly the same as the number of nodes that are infected after the first timestep t_0, which suggests the best scenario for SOCIALCONTACT is that only nodes which are infected at the earliest time are not caused by social contact.

Scalability Figure 7 shows the running time of MAPPINGGENERATION and IMMUCONGREEDY w.r.t. the vaccine budget m and the number of cascades k on SBM. For Fig. 7a, we set $k = 100$, while for Fig. 7b we set $m = 20$. We observe that as m increases and k increases, the running time scales linearly. (Figures also show the linear fit with R^2 values.) Consistent with the time complexity bounds for our algorithms in Sect. 4, large datasets need fairly extensive time. For example, Miami takes about 2 days to get $5K$ vaccines. This is still reasonable: importantly, note that we expect to run immunization algorithms for infectious epidemics, so the solution quality is much more critical than the fastest running time.

5.3 Case studies

We conduct case studies to analyze vaccine allocations per zipcode for both Houston and Miami. Figure 8 shows the total population, the total #patients in the eHRC data, the total

Fig. 8 Case studies for `Houston` and `Miami` per location. `Houston`: **a–e**; `Miami`: **f–j**; `Portland`: **k–o**. Heatmap of **a**, **f** and **k**: total population; **b**, **g** and **l**: patients in `eHRC`; **c**, **h** and **m**: number of vaccines actually taken in `eHRC`; **d**, **i** and **n**: vaccine allocations from ImmuModel; **e**, **j** and **o**: vaccine allocations from ImmuConGreedy

#vaccines taken in the `eHRC` data,[3] the total #vaccines from ImmuModel, and the total #vaccines from ImmuConGreedy, respectively.

Figure 8a–e shows the case study for `Houston`. First, the areas with zipcode 77030 and 77024 in Fig. 8b have the largest number of patients, and vaccine allocations from both `eHRC` (Fig. 8c), and ImmuConGreedy (Fig. 8e) also prefer these areas. Second, vaccines taken in the `eHRC` data do not follow the total population (Fig. 8a), but roughly follow the distribution of #patients in `eHRC`. This may suggest the immunization strategy in practice is to give vaccines based on the proportion of reported patients. Third, ImmuModel distributes 38% of vaccines to three areas (77002, 77008 and 77056), which are the center of Houston Metropolitan Area (such as downtown and uptown) with a large number of interactions in the contact network. However, both data-based and model-based approaches do not perform well (see Fig. 3). Our method, ImmuConGreedy, gives 43% of vaccines to the areas 77030, 77024 and 77002. The first two areas have the highest infections in `eHRC`, while the last one is essential for minimizing the epidemic threshold as ImmuModel suggests. Hence, ImmuConGreedy considers both `eHRC` and contact networks. It is interesting that the Texas Medical Center (one of the largest medical centers in the world) is in 77030, and Houston downtown is in 77002. Hence, ImmuConGreedy targets regions with high risk of influenza outbreak.

Figure 8f–j shows the case study for `Miami`. First, vaccines taken in `eHRC` (Fig. 8h) follow the distribution of #patients as well (Fig. 8g). Second, ImmuModel distributes 31% of vaccines in one area with zipcode 33165 (Fig. 8i). We believe this area with large number of households is critical to minimize the spectral radius of the contact network in `Miami`.

[3] We extract vaccine reports based on ICD-9 codes V04.81. These are actual vaccine allocations as given in the `eHRC` data.

However, both data-based and model-based approaches do not perform well in Miami as well (as shown in Fig. 3). Interestingly, as shown in Fig. 8j, our approach, ImmuConGreedy, gives most of the vaccines (29%, 18%) to areas with the largest number of patients (33140 and 33176, respectively). We observe that different from Houston, in Miami ImmuConGreedy tend to prefer data-based approaches. However, the areas adjacent to 33165, which ImmuModel targets, also get higher vaccine allocations than others—this means ImmuConGreedy also takes into account information in the contact network. In fact, the areas ImmuConGreedy targets indeed have high risk of an influenza outbreak: They are either tourist attractions (33140) or residential areas (33176). For example, 33140 belongs to Miami Beach, which is a famous place with large transient population.

Figure 8k–o shows the case study for Miami. First, similarly to Houston and Miami, the distribution of vaccine allocation in eHRC (Fig. 8m) is very similar to the distribution of patients (Fig. 8i). Second, ImmuModel gives 18% of vaccines to the area with the zipcode 97124. We believe this region has many households, which is important to the spectral radius of the contact network in Portland. However, only considering the contact network does not give us a better performance (as shown in Fig. 3). Our method, ImmuConGreedy in Fig. 8o, gives most vaccines to three areas with the zipcodes 97124, 97123 and 97216. Interestingly, different from Houston and Miami, in Portland, the vaccines distribution in Portland tends to slightly prefer network-based approaches. The area 97124 is given the most number of vaccines. However, this area also has a relatively high number of patients according to Fig. 8l.

6 Translation to practice

Our approach shows that combining partial incidence data with a detailed activity-based population model can help in developing more effective interventions for controlling the spread of an outbreak, compared to current baselines. Incidence data can be obtained in different ways: (1) eHRC data, as we assume here, (2) incidence from *previous year's* outbreaks, provided it is at a high spatiotemporal resolution, or (3) other proxies, such as surveys, and even non-traditional sources such as tweets and online media, so long as they give some indication of the incidence at a geographic level. Therefore, we believe our approach can be operationalized by public-health agencies, such as the CDC.

7 Related work

We review closely related work next. Remotely, related work includes those on blogs and propagations [19], and viral marketing [20] (e.g., Goyal et al. [21] studied the influence maximization problem using a data-based approach).

Epidemiology The early canonical textbooks and surveys include [13,22], which describe the fundamental epidemiological models such as the so-called SIS and SIR models. Epidemic thresholds (minimum virulence of a virus that causes an epidemic) for various models have been extensively studied [11,23]. In practice, viruses are always changing, and hence assuming a prior model may be suboptimal.

Immunization There has been a lot of work on developing optimal strategies to control propagation over graphs. Cohen et al. [24] proposed the popular *acquaintance* immunization policy, while Aspnes et al. [25] developed inoculation policies for victims of viruses using

game theory. Tong et al. [3,12], Van Miegham et al. [26], and Prakash et al. [27] studied the problem of minimizing the spectral radius (epidemic threshold) of the graph for a variety of models. In addition, other immunization work in the literature has been proposed based on differential equation methods [1,28]. The most related work includes Zhang et al. [4] who studied the immunization at the group scale, while Zhang and Prakash [5] and Khalil et al. [29] developed several model-based efficient algorithms for immunization given partial information of infections. All past work proposed either model-based or graph-based approaches for immunization. Instead, we leverage rich surveillance healthcare data together with the network information for the problem of controlling disease spread.

eHRC. There has been a lot of work in using eHRC data for inferring patient conditions [30]. Previous studies have also pointed to the utility of eHRC data to identify trends in epidemic incidence across the USA [31,32]. Leveraging eHRC, the spatial and temporal patterns of flu incidence during 2009–2010 pandemic flu season have been discovered [7]. In addition, Malhotra et al. [33] used sequential pattern mining techniques to reveal common sequences of clinical procedures administered to patients for a variety of medical conditions from eHRC. In sum, none studied the immunization problem with the eHRC data.

8 Conclusions

This paper addresses the novel problem of controlling epidemics in the presence of coarse-grained health surveillance data and population contact networks. We formulate the data-driven immunization problem, which first aims to align the propagation log with contact networks, and then allocate vaccines to minimize spread in the data. We develop an efficient approach MappingGeneration to obtain high-quality cascades, and then give an approximation algorithm ImmuConGreedy with provable solutions for immunization on sampled cascades. We demonstrate the effectiveness of our method through extensive experiments on multiple datasets including nation-wide real electronic Health Reimbursement Claims data. Finally, case studies in Miami and Houston metropolitan regions show that our allocation strategies take both the network and surveillance data into account to effectively distribute vaccines.

Future work can include investigating other sampling strategies, incorporating more data sources, and studying vaccine allocations to other groups, such as demographics like age.

Acknowledgements This paper is based on work partially supported by the NSF (IIS-1353346, CAREER IIS-1750407), the NEH (HG-229283-15), ORNL, the Maryland Procurement Office (H98230-14-C-0127), and a Facebook faculty gift to BAP. AV is partially supported by the following grants: DTRA CNIMS Contract HDTRA1- 11-D-0016-0010, NSF BIG DATA Grant IIS-1633028 and NSF DIBBS Grant ACI-1443054, NSF EAGER SSDIM-1745207. Publication of this article was also funded by ORNL LDRD funding to AR. Oak Ridge National Laboratory (ORNL) (Grant No. Order 4000143330) is operated by UT-Battelle, LLC, for the US Department of Energy under contract DE-AC05-00OR22725. The US Government retains and the publisher, by accepting the article for publication, acknowledges that the US Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for US Government purposes.

Appendix

Proof of Lemma 4.4 When $\alpha_{\mathbf{M},\ell}$ is optimal, $\alpha_{\mathbf{M},\ell} = \alpha^*_{\mathbf{M},\ell}$.
Second, let β_{S_ℓ} be the number of nodes without any parents. Maximizing $\alpha_{\mathbf{M},\ell}$ for Problem 3.1

is equivalent to minimizing β_{S_ℓ} at location L_ℓ. Suppose $\beta^*_{S_\ell}$ is maximum number of nodes without any parents in the sample at location L_ℓ. It is obvious $\beta^*_{S_\ell} = CN(L_\ell, t_0) = |S_L|$. For each timestep t_i, if $CF_i(S_\ell) < CN(L_\ell, t_i)$, then $CN(L_\ell, t_i) - CF_i(S_\ell)$ is the number of nodes that cannot be mapped to the cascade generated by S_ℓ at timestep t_i. Hence, $\theta(S_\ell)$ is the number of nodes that cannot be mapped to the cascade generated by S_ℓ. If there exists any t_i that $CF_i(S_\ell) < CN(L_\ell, t_i)$, we can always generate a cascade by mapping all $CF_i(S_\ell)$ nodes into the cascade, then uniformly at random map other $\theta(S_\ell)$ nodes into cascade. This way, the number of nodes without any parents, $\beta_{S_\ell} \leq \beta^*_{L_\ell} + \theta(S_\ell)$ as $\theta(S_\ell)$ nodes can have connection within themselves. Since $\beta_{S_\ell} + \alpha_{S_\ell} = \sum_{t_i} N(L_i, t_i)$, then $\alpha_{\mathbf{M},\ell} \geq \alpha^*_{\mathbf{M},\ell} - \theta(S_\ell)$. Hence, $\alpha^*_{\mathbf{M},\ell} - \theta(S_\ell) \leq \alpha_{\mathbf{M},\ell} \leq \alpha^*_{\mathbf{M},\ell}$. When $\alpha_{\mathbf{M},\ell} = \alpha^*_{\mathbf{M},\ell}$, $\theta(S_\ell) = 0$. □

Proof of Lemma 4.5 First, it is clear that $g(\emptyset) = 0$.

Second, to prove $g(S)$ is monotonic increasing, we need to prove $\theta(S)$ is a monotonic decreasing function. To do that, we first show that $CF_i(S_\ell)$ is monotone non-decreasing and submodular functions for any i and L_ℓ. First, let us define $f_i(S_\ell)$ as the number of nodes in L_ℓ that S_ℓ can reach in i-hops; hence, $f_i(S_\ell) \leq f_i(S_k)$ when $S_\ell \subseteq S_k$. Second, given $S_\ell \subseteq S_k$ and a node u, $f_i(S_\ell \cup \{u\}) - f_i(S_\ell)$ is marginal gain of a set union. Since the function in the set union problem is submodular [14], $f_i(S_\ell)$ is also submodular. Since $f_i(S_\ell)$ is monotone non-decreasing and submodular, the cumulative function $CF_i(S_\ell)$ is also non-decreasing and submodular.

Let $X_i = [\mathbb{1}_{CF_i(A\cup B)<CN_i}(CN_i - CF_i(A \cup B))]$, $Y_i = [\mathbb{1}_{CF_i(A)<CN_i}(CN_i - CF_i(A))]$. For any set A and B,

$$\theta(A \cup B) - \theta(A) = \sum_{i=1}^{T}(X_i - Y_i) \tag{6}$$

For any i, let us consider the following two cases:

(1) If $\mathbb{1}_{CF_i(A)<CN_i} = 0$, it means $CF_i(A) \geq CN_i$, then $CF_i(A \cup B) \geq CN_i$; hence, $\mathbb{1}_{CF_i(A\cup B)<CN_i} = 0$. We have $X_i - Y_i = 0$.

(2) If $\mathbb{1}_{CF_i(A)<CN_i} = 1$, we have two cases:

(2a) $\mathbb{1}_{CF_i(A\cup B)<CN_i} = 0$, then $X_i - Y_i = -Y_i = -(CN_i - CF_i(A)) < 0$;

(2b) $\mathbb{1}_{CF_i(A\cup B)<CN_i} = 1$, then $X_i - Y_i = (CN_i - CF_i(A \cup B)) - (CN_i - CF_i(A))$ $= CF_i(A) - CF_i(A \cup B) \leq 0$ (using Claim 2).

Putting together, we have $\theta(A \cup B) \leq \theta(A)$. Hence, $\theta(S)$ is monotonic decreasing, and hence $g(S)$ is monotonic increasing.

Third, to prove $g(S)$ is submodular, For any location l, we need to prove that, given $S \subseteq T$, $g(S \cup \{a\}) - g(S) \geq g(T \cup \{a\}) - g(T)$, which is equivalent to $\theta(S) - \theta(S \cup \{a\}) \leq \theta(T) - \theta(T \cup \{a\})$ (supermodularity). Let us write

$\delta(S, a, i) = [\mathbb{1}_{CF_i(S\cup\{a\})<CN_i}(CN_i - CF_i(S \cup \{a\}))] - [\mathbb{1}_{CF_i(S)<CN_i}(CN_i - CF_i(S))]$, and

$\delta(T, a, i) = [\mathbb{1}_{CF_i(T\cup\{a\})<CN_i}(CN_i - CF_i(T \cup \{a\}))] - [\mathbb{1}_{CF_i(T)<CN_i}(CN_i - CF_i(T))]$, then,

$\theta(S) - \theta(S \cup \{a\}) = \sum_{i=1}^{t} \delta(S, a, i)$, and $\theta(T) - \theta(T \cup \{a\}) = \sum_{i=1}^{t} \delta(T, a, i)$.

For any i, let us consider the following two cases:

(1) If $\mathbb{1}_{CF_i(S)<CN_i} = 0$, then $\mathbb{1}_{CF_i(S\cup\{a\})<CN_i} = \mathbb{1}_{CF_i(T)<CN_i} = \mathbb{1}_{CF_i(T\cup\{a\})<CN_i} = 0$. Hence, $\delta(S, a, i) = \delta(T, a, i) = 0$.

(2) If $\mathbb{1}_{CF_i(S)<CN_i} = 1$, we have the following cases:

(2a) If $\mathbb{1}_{CF_i(T)<CN_i} = 0$, then we have $\mathbb{1}_{CF_i(T\cup\{a\})<CN_i} = 0$. Let us consider the value of $\mathbb{1}_{CF_i(S\cup\{a\})<CN_i}$:

If $\mathbb{1}_{CF_i(S\cup\{a\})<CN_i} = 0$, then $\delta(S,a,i) = (CN_i - CF_i(S\cup\{a\})) < 0 = \delta(T,a,i)$.

If $\mathbb{1}_{CF_i(S\cup\{a\})<CN_i} = 1$, then $\delta(S,a,i) = CF_i(S) - CF_i(S\cup\{a\}) < 0 = \delta(T,a,i)$.

(2b) If $\mathbb{1}_{CF_i(T)<CN_i} = 1$, let us consider the value of $\mathbb{1}_{CF_i(S\cup\{a\})<CN_i}$:

If $\mathbb{1}_{CF_i(S\cup\{a\})<CN_i} = 0$, then $\mathbb{1}_{CF_i(T\cup\{a\})<CN_i} = 0$, and then $\delta(S,a,i) = -(CN_i - CF_i(S)) \leq -(CN_i - CF_i(T)) = \delta(T,a,i)$ (using Claim 2).

If $\mathbb{1}_{CF_i(S\cup\{a\})<CN_i} = 1$, then for $\mathbb{1}_{CF_i(T\cup\{a\})<CN_i}$:

If $\mathbb{1}_{CF_i(T\cup\{a\})<CN_i} = 1$, then $\delta(S,a,i) = CF_i(S) - CF_i(S\cup\{a\})) \leq CF_i(T) - CF_i(T\cup\{a\})) = \delta(T,a,i)$ (using Claim 2 that $CF_i(S)$ is a submodular function).

Otherwise, $\mathbb{1}_{CF_i(T\cup\{a\})<CN_i} = 0$, and then since we have $CF_i(T\cup\{a\}) \geq CN_i$, $\delta(S,a,i) = CF_i(S) - CF_i(S\cup\{a\})) \leq CF_i(T) - CF_i(T\cup\{a\})) \leq CF_i(T) - CN_i = \delta(T,a,i)$ (using Claim 2).

Putting all cases together, we have $\theta(S) - \theta(S\cup\{a\}) \leq \theta(T) - \theta(T\cup\{a\})$. Hence, $g(S\cup\{a\}) - g(S) \geq g(T\cup\{a\}) - g(T)$.

$g(S)$ is a submodular function. □

Proof of Lemma 4.9 Since we uniformly randomly allocate $\mathbf{x}$, $\rho_{G,\mathbf{M}_i}(\mathbf{x})$ can be written as $\rho_{G,\mathbf{M}_i}(\mathbf{x}) = \sum_S \Pr(S) r_{G,\mathbf{M}_i}(S)$, where S is a node set sampled from the random process of distributing $\mathbf{x}$ ($|S| = ||\mathbf{x}||_1$), and $r_{G,\mathbf{M}_i}(S)$ is the number of nodes $SI_{\mathbf{M}_i}$ can reach after removing S.

Since $\zeta_{G,\mathbf{M}_i}(\mathbf{x}) = \sum_S \Pr(S) C_{G,\mathbf{M}_i}(S)$ and $\rho_{G,\mathbf{M}_i}(\mathbf{x}) = \sum_S \Pr(S) r_{G,\mathbf{M}_i}(S)$, we need to show that $r_{G,\mathbf{M}_i}(S) \leq C_{G,\mathbf{M}_i}(S)$. $r_{G,\mathbf{M}_i}(S)$ is the number of nodes S can save in $\mathbf{M}_i$, we can show that given any node u that $SI_{\mathbf{M}}$ can save, the credit u given to $SI_{\mathbf{M}}$ must be 1. This is because if we can save u, it means every path from $SI_{\mathbf{M}}$ to u has been removed when S is removed. Hence, all nodes within the paths from $SI_{\mathbf{M}}$ have been removed. These nodes are all nodes that propagate u's credit to $SI_{\mathbf{M}}$, so all credits of u can be contributed to $C_{G,\mathbf{M}_i}(S)$. Hence, $C_{G,\mathbf{M}_i}(S)$ is at least equal to $r_{G,\mathbf{M}_i}(S)$. On the other hand, other nodes that S cannot save also make contributions to the credit of $C_{G,\mathbf{M}_i}(S)$. Hence, $C_{G,\mathbf{M}_i}(S) \geq r_{G,\mathbf{M}_i}(S)$, which leads to $\rho_{G,\mathbf{M}_i}(\mathbf{x}) \leq \zeta_{G,\mathbf{M}_i}(\mathbf{x})$. □

Proof of Lemma 4.11 We use a similar technique as in [4] given the properties of P_1, P_2 and P_3 of $\zeta_{G,\mathbf{M}_i}(\mathbf{x})$. For brevity, we write $\zeta_{G,\mathbf{M}_i}(\mathbf{x})$ as $\zeta(\mathbf{x})$.

First, we show that if $\mathbf{y} = (y_i, \ldots, y_n)^T$ where $\sum_j y_j = m$, then $\zeta(\mathbf{x}+\mathbf{y}) - \zeta(\mathbf{x}) \leq \sum_j y_j(\zeta(\mathbf{x}+\mathbf{e}_j) - \zeta(\mathbf{x}))$.

Let $\mathbf{y}$ can be recursively obtained from a sequence $\mathbf{e}^{(1)}, \ldots, \mathbf{e}^{(m)}$ ($\mathbf{e}^{(i)} \in \{\mathbf{e}_1, \ldots, \mathbf{e}_n\}$) such that $\mathbf{y} = \mathbf{y}^{(m)} = \mathbf{y}^{(m-1)} + \mathbf{e}^{(m)}$, $\mathbf{y}^{(i)} = \mathbf{y}^{(i-1)} + \mathbf{e}^{(i)}$ ($i \leq m$) and $\mathbf{y}^0 = \mathbf{0}$.

Obviously, $\sum_{i=1}^m \mathbf{e}^{(i)} = \sum_j y_j \mathbf{e}_j = \mathbf{y}$. Then,

$$
\begin{aligned}
&\zeta(\mathbf{x}+\mathbf{y}) - \zeta(\mathbf{x}) \\
&= \sum_{i=1}^{m} \zeta(\mathbf{x}+\mathbf{y}^{(i)}) - \zeta(\mathbf{x}+\mathbf{y}^{(i-1)}) \\
&= \sum_{i=1}^{m} \zeta(\mathbf{x}+\mathbf{y}^{(i-1)}+\mathbf{e}^{(i)}) - \zeta(\mathbf{x}+\mathbf{y}^{(i-1)}) \\
&\leq \sum_{i=1}^{m} \zeta(\mathbf{x}+\mathbf{e}^{(i)}) - \zeta(\mathbf{x}) \ (\textbf{Diminishing Return}) \\
&= \sum_{j=1}^{n} y_j(\zeta(\mathbf{x}+\mathbf{e}_j) - \zeta(\mathbf{x}))
\end{aligned}
$$

Now, let us prove that ImmuNaiveGreedy gives a $(1-1/e)$-approximate solution. Suppose $\mathbf{x}$ is the solution from ImmuNaiveGreedy, and $\mathbf{x}^*$ is the optimal solution. Clearly, we have $\sum_j x_j = \sum_j x_j^* = m$. Let us define $\mathbf{x}^{(i)}$ as the solution got from the ith iteration of the greedy algorithm; hence, $\mathbf{x} = \mathbf{x}^{(m)}$. And $\mathbf{x}^*$ can be represented as $\sum_j x_j^* \mathbf{e}_j$. We have

$$\begin{aligned}
\zeta(\mathbf{x}^*) &\leq \zeta(\mathbf{x}^* + \mathbf{x}^{(i)}) \\
&= \zeta(\mathbf{x}^{(i)}) + (\zeta(\mathbf{x}^* + \mathbf{x}^{(i)}) - \zeta(\mathbf{x}^{(i)})) \\
&\leq \zeta(\mathbf{x}^{(i)}) + \sum_j x_j^* (\zeta(\mathbf{x}^{(i)} + \mathbf{e}_j) - \zeta(\mathbf{x}^{(i)})) \\
&\leq \zeta(\mathbf{x}^{(i)}) + \sum_j x_j^* (\zeta(\mathbf{x}^{(i+1)}) - \zeta(\mathbf{x}^{(i)})) \\
&= \zeta(\mathbf{x}^{(i)}) + m(\zeta(\mathbf{x}^{(i+1)}) - \zeta(\mathbf{x}^{(i)}))
\end{aligned}$$

Hence, $\zeta(\mathbf{x}^{(i+1)}) \geq (1 - \frac{1}{m})\zeta(\mathbf{x}^{(i)}) + \frac{1}{m}\zeta(\mathbf{x}^*)$. Recursively, we can get $\zeta(\mathbf{x}^{(i)}) \geq (1 - (1 - \frac{1}{m})^i)\zeta(\mathbf{x}^*)$. Therefore, $\zeta(\mathbf{x}) = \zeta(\mathbf{x}^{(m)}) \geq (1 - (1 - \frac{1}{m})^m)\zeta(\mathbf{x}^*) \geq (1 - 1/e)\zeta(\mathbf{x}^*)$. □

References

1. Medlock J, Galvani AP (2009) Optimizing influenza vaccine distribution. Science 325:1705–1708
2. Halloran ME, Ferguson NM, Eubank S, Longini IM, Cummings DAT, Lewis B, Xu S, Fraser C, Vullikanti A, Germann TC, Wagener D, Beckman R, Kadau K, Barrett C, Macken CA, Burke DS, Cooley P (2008) Modeling targeted layered containment of an influenza pandemic in the United States. In: Proceedings of the National Academy of Sciences (PNAS), March 10 2008, pp 4639–4644
3. Tong H, Prakash BA, Tsourakakis CE, Eliassi-Rad T, Faloutsos C, Chau DH (2010) On the vulnerability of large graphs. In: ICDM
4. Zhang Y, Adiga A, Vullikanti A, Prakash BA (2015) Controlling propagation at group scale on networks. In: 2015 IEEE international conference on data mining (ICDM). IEEE, pp 619–628
5. Zhang Y, Prakash BA (2014) Dava: distributing vaccines over networks under prior information. In: Proceedings of the SIAM data mining conference, ser. SDM'14
6. Pellis L, Ball F, Bansal S, Eames K, House T, Isham V, Trapman P (2015) Eight challenges for network epidemic models. Epidemics 10:58–62
7. Ramanathan A, Pullum LL, Hobson TC, Steed CA, Quinn SP, Chennubhotla CS, Valkova S (2015) Orbit: Oak Ridge biosurveillance toolkit for public health dynamics. BMC Bioinform 16(17):S4
8. Ozmen O, Pullum LL, Ramanathan A, Nutaro JJ (2016) Augmenting epidemiological models with point-of-care diagnostics data. PLoS ONE 11(4):1–13 04
9. Barrett CL, Beckman RJ, Khan M, Anil Kumar VS, Marathe MV, Stretz PE, Dutta T, Lewis B (2009) Generation and analysis of large synthetic social contact networks. In: Winter simulation conference, pp 1003–1014
10. Eubank S, Guclu H, Anil Kumar VS, Marathe MV, Srinivasan A, Toroczkai Z, Wang N (2004) Modelling disease outbreaks in realistic urban social networks. Nature 429(6988):180–184
11. Prakash BA, Chakrabarti D, Faloutsos M, Valler N, Faloutsos C (2012) Threshold conditions for arbitrary cascade models on arbitrary networks. Knowl Inf Syst 33:549–575
12. Tong H, Prakash BA, Eliassi-Rad T, Faloutsos M, Faloutsos C (2012) Gelling, and melting, large graphs by edge manipulation. In: Proceedings of CIKM
13. Anderson RM, May RM (1991) Infectious diseases of humans. Oxford University Press, Oxford
14. Karp RM (1972) Reducibility among combinatorial problems. In: Complexity of computer computations. Springer, pp 85–103
15. Nemhauser GL, Wolsey LA, Fisher ML (1978) An analysis of approximations for maximizing submodular set functions—I. Math Program 14(1):265–294
16. Palmer CR, Gibbons PB, Faloutsos C (2002) Anf: a fast and scalable tool for data mining in massive graphs. Ser. KDD '02. ACM, New York, NY, USA, pp 81–90

17. Flajolet P, Martin GN (1985) Probabilistic counting algorithms for data base applications. J Comput Syst Sci 31(2):182–209
18. McDaid AF, Murphy B, Friel N, Hurley N (2012) Clustering in networks with the collapsed stochastic block model. arXiv preprint arXiv:1203.3083
19. Kumar R, Novak J, Raghavan P, Tomkins A (2003) On the bursty evolution of blogspace. In: WWW'03, pp 568–576
20. Kempe D, Kleinberg J, Tardos E (2003) Maximizing the spread of influence through a social network. In: KDD'03
21. Goyal A, Bonchi F, Lakshmanan LV (2011) A data-based approach to social influence maximization. Proc VLDB Endow 5(1):73–84
22. Hethcote HW (2000) The mathematics of infectious diseases. SIAM Rev 42:599–653
23. Ganesh A, Massoulie L, Towsley D (2005) The effect of network topology on the spread of epidemics. In: Proceedings of INFOCOM
24. Cohen R, Havlin S, Ben Avraham D (2003) Efficient immunization strategies for computer networks and populations. Phys Rev Lett 91(24):247901
25. Aspnes J, Chang K, Yampolskiy A (2005) Inoculation strategies for victims of viruses and the sum-of-squares partition problem. In: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms, series SODA'05, pp 43–52
26. Van Mieghem P, Stevanović D, Kuipers F, Li C, Van De Bovenkamp R, Liu D, Wang H (2011) Decreasing the spectral radius of a graph by link removals. Phys Rev E 84(1):016101
27. Prakash BA, Adamic LA, Iwashyna TJ, Tong H, Faloutsos C (2013) Fractional immunization in networks. In: Proceedings of SDM, pp 659–667
28. Shim E (2013) Optimal strategies of social distancing and vaccination against seasonal influenza. Math Biosci Eng 10(5):1615–1634
29. Khalil EB, Dilkina B, Song L (2014) Scalable diffusion-aware optimization of network topology. In: KDD 2014. ACM, pp 1226–1235
30. Saha B, Gupta S, Phung D, Venkatesh S (2017) Effective sparse imputation of patient conditions in electronic medical records for emergency risk predictions. Knowl Inf Syst 53(1):179–206. https://doi.org/10.1007/s10115-017-1038-0
31. Patwardhan A, Bilkovski R (2012) Comparison: flu prescription sales data from a retail pharmacy in the US with google flu trends and US ilinet (cdc) data as flu activity indicator. PloS ONE 7(8):e43611
32. Gog JR, Ballesteros S, Viboud C, Simonsen L, Bjornstad ON, Shaman J, Chao DL, Khan F, Grenfell BT (2014) Spatial transmission of 2009 pandemic influenza in the us. PLoS Comput Biol 10(6):e1003635
33. Malhotra K, Hobson TC, Valkova S, Pullum LL, Ramanathan A (2015) Sequential pattern mining of electronic healthcare reimbursement claims: experiences and challenges in uncovering how patients are treated by physicians. In: 2015 IEEE international conference on big data (big data). IEEE, pp 2670–2679

Yao Zhang obtained his PhD degree from the Department of Computer Science at Virginia Tech and his bachelors and masters degrees in computer science from Nanjing University, China. His current research interests are graph mining and social network analysis with focus on understanding and managing information diffusion in networks. He has published several papers in top conferences and journals such as KDD, ICDM, SDM and TKDD.

Arvind Ramanathan is a senior staff scientist and the team lead for Integrative Systems Biology in the Computational Science and Engineering Division and the Health Data Sciences Institute at Oak Ridge National Laboratory. His research interests are at the intersection of high performance computing and biological/health data science. He is interested in developing data-driven machine learning approaches for public health surveillance. He has published over 30 papers, and his work has been highlighted in the popular media. More information about his group and research interests can be found at http://ramanathanlab.org.

Anil Vullikanti is a Professor at the Department of Computer Science and the Biocomplexity Institute & Initiative at the University of Virginia. His research interests include the broad areas of approximation and randomized algorithms and dynamical systems, and their applications to computational epidemiology, wireless networks, and interdependent infrastructures. His papers have been nominated for best paper awards at Supercomputing 2016 and AAAI 2013. He is the recipient of the Virginia Tech College of Engineering Faculty Fellow Award 2017, the Biocomplexity Institute of Virginia Tech Excellence in Research Award 2017, DOE Early Career award in 2010 and the NSF CAREER award in 2009.

Laura Pullum is a senior research scientist in the Computer Science and Mathematics Division at Oak Ridge National Laboratory (ORNL), with over 30 years of experience. Her research is in software-based system dependability and intelligent systems in a variety of domains. She currently conducts research in the evaluation, verification and validation of machine learning, artificial intelligence and complex computing systems. Prior to joining ORNL, she worked in industry, at a non-profit research institute, as a visiting professor, and as a small business owner. Dr. Pullum has authored over 100 publications including books, book chapters, and peer-reviewed papers; holds a patent; serves on technical advisory boards; serves on the standards working group for IEEE P1012- 201X Standard for System, Software and Hardware Verification and Validation; and serves as the IEEE Computer Society representative to the IEEE Smart Grid. She is a senior member of the IEEE Computer Society. Dr. Pullum also serves on the ASME V&V 50 Committee on Verification and Validation of Data-Driven and Hybrid Models and serves on the ISO/IEC JTC 1/SC 42 Technical Advisory Group on Artificial Intelligence. Dr. Pullum holds a BS in Math; masters degrees in Operations Research, Business Administration, and Geology; and a doctorate in Systems Engineering and Operations Research.

B. Aditya Prakash is an Assistant Professor in the Computer Science Department at Virginia Tech. He graduated with a Ph.D. from the Computer Science Department at Carnegie Mellon University in 2012 and obtained his B.Tech (in CS) from the Indian Institute of Technology (IIT) Bombay in 2007. He has published one book, more than 70 refereed papers in major venues, holds two U.S. patents and has given five tutorials (SDM 2018, SDM 2017, SIGKDD 2016, VLDB 2012 and ECML/PKDD 2012) at leading conferences. His work has also received a best paper award and four best-of-conference selections (ICDM 2017, ASONAM 2013, CIKM 2012, ICDM 2012, ICDM 2011) and multiple travel awards. His research interests include Data Mining, Applied Machine Learning and Databases, with emphasis on big-data problems in large real-world networks and time-series. His work has been funded through grants/gifts from the National Science Foundation (NSF), the Department of Energy (DoE), the National Security Agency (NSA), the National Endowment for Humanities (NEH) and from companies like Symantec. Tools developed by his group have been in use in many places including ORNL, Walmart and Facebook. He received a Facebook Faculty Gift Award in 2015 and the NSF CAREER award in 2018 and was named as one of '2017 AI Ten to Watch' by IEEE Intelligent Systems. He is also an affiliated faculty member at the Discovery Analytics Center at Virginia Tech.

Affiliations

Yao Zhang[1] · Arvind Ramanathan[3] · Anil Vullikanti[2] · Laura Pullum[3] · B. Aditya Prakash[1]

Arvind Ramanathan
ramanathana@ornl.gov

Anil Vullikanti
vsakumar@virginia.edu

Laura Pullum
pullumll@ornl.gov

B. Aditya Prakash
badityap@cs.vt.edu

1 Department of Computer Science, Virginia Tech, Blacksburg, USA

2 Department of Computer Science, Biocomplexity Institute and Initiative, University of Virginia, Charlottesville, USA

3 Oak Ridge National Laboratory, Oak Ridge, USA