Scalable Vaccine Distribution in Large Graphs given Uncertain Data

Yao Zhang
Department of Computer Science
Virginia Tech
yaozhang@cs.vt.edu

B. Aditya Prakash
Department of Computer Science
Virginia Tech
badityap@cs.vt.edu

ABSTRACT
Given an noisy or sampled snapshot of a network, like a contact-network or the blogosphere, in which an infection (or meme/virus) has been spreading for some time, what are the best nodes to immunize (vaccinate)? Manipulating graphs via node removal by itself is an important problem in multiple different domains like epidemiology, public health and social media. Moreover, it is important to account for uncertainty as typically surveillance data on who is infected is limited or the data is sampled. Efficient algorithms for such a problem can help public-health experts take more informed decisions.

In this paper, we study the problem of designing vaccine-distribution algorithms under an uncertain environment, with known information consisting of confirmed cases as well as a probability distribution of unknown cases. We formulate the NP-Hard Uncertain Data-Aware Vaccination problem, and design multiple efficient algorithms for factorizable distributions (including a novel sub-quadratic algorithm) which naturally take into account the uncertainty, while providing robust solutions. Finally, we show the effectiveness and scalability of our methods via extensive experiments on real datasets, including large epidemiological and social networks.

Categories and Subject Descriptors
H.2.8 [Database Management]: Database Applications—Data mining

Keywords
Graph Mining; Uncertainty; Immunization; Diffusion

1. INTRODUCTION
What is the best way to distribute vaccines to prevent the spread of diseases on a socio-contact network? Most previous works (see Related Work) for controlling propagation have concentrated on developing strategies for vaccination (node/edge removal) pre-emptively before the start of an epidemic. While very useful to provide insights in to which baseline policies can best control an infection, they may not be ideal to help make real-time decisions as the infection is progressing. Consider also social media and cyber security. Popular phrases or links or rumors are re-posted/re-tweeted on Facebook/Twitter, ‘infecting’ followers to do the same. How should Twitter decide which accounts to suspend/delete to stop active rumors/spam/malware as much as possible? Which machines should install patches first, in presence of malware attacks? All these problems can be thought of as immunization/vaccination in a network, in presence of already infected nodes [46].

However, in reality contagions usually spread over uncertain environments and the sources of such uncertainty are many. For example, in public health, due to the so-called multi-layered surveillance pyramid [39, 16, 30] at each layer the number of detected infections is a fraction of the infections in the layer below it. Hence the total detected infections at the top of the pyramid is a fraction of the actual infections in the population at the bottom. Another example is the likelihood ratios used in diagnostic testing [13]. For each a person who gets the negative test outcome, she has some probability that her test was a false-negative. In social media, as externals we rarely get access to the complete cascade. Researchers usually have access to only a uniform sample of cases (e.g. the Twitter API). In Facebook, most users keep their activity and profiles private. Moreover, if only because of the extreme velocity of social media data, one has to resort to using only a sample of the data. Hence this implies that we will have to make do with only an uncertain snapshot.

In this paper, we study the problem of how to best distribute vaccines to nodes in large networks, in presence of uncertain prior information. Our goal is not to fill-in the missing information; instead we want to take robust decisions in presence of uncertain information. Our contributions include:

1. Problem Formulation: We formulate the Uncertain Data-Aware Vaccination problem, which takes into account multiple natural uncertainty models arising from social media and epidemiology.

2. Efficient Algorithms: As the problem is NP-hard and hard to approximate within absolute error, we develop multiple polynomial-time algorithms of varying efficiency, namely (a) SAMPLE-Cas, based on the sample average approximation; and (b) EXPECT-MAX, a faster hybrid algorithm which leverages the so-called...
expected graph and two complementary approaches to estimate benefits.

3. Extensive Experiments: We demonstrate the effectiveness and scalability of our algorithms on multiple real datasets including large epidemiological and social networks, over different uncertainty distributions and initial conditions. Our algorithms outperform several other competitor algorithms, getting substantial gains in both number of nodes saved, and running time.

The rest of the paper is organized as follows. Section 2 presents some preliminaries while Section 3 sets up the Uncertain Data-Aware Vaccination problem, and discusses the computational complexity of our problem. Section 4 presents our algorithms and Section 5 presents experimental results on several datasets. We give related work in Section 6, and finally conclude in Section 7.

2. PRELIMINARIES

Table 1 lists the main symbols used in this paper. There exists an underlying contact network \( G \) on which the contagion (disease/virus/meme etc.) can spread. We assume that our network is weighted and undirected, but all our methods can be naturally generalized to directed graphs.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( UDAV )</td>
<td>Uncertain Data-Aware Vaccination problem</td>
</tr>
<tr>
<td>IC</td>
<td>Independent Cascade Model</td>
</tr>
<tr>
<td>SIR</td>
<td>Susceptible-Infected-Recovered Model</td>
</tr>
<tr>
<td>footprint</td>
<td>number of infected nodes at the end</td>
</tr>
<tr>
<td>benefit</td>
<td>number of nodes saved</td>
</tr>
<tr>
<td>( G(V,E) )</td>
<td>graph ( G ) with nodes set ( V ) and edges set ( E )</td>
</tr>
<tr>
<td>( \beta_{i,j} )</td>
<td>propagation probability from node ( i ) to ( j ) (weight over edges)</td>
</tr>
<tr>
<td>( p_i )</td>
<td>probability that ( i ) is infected at the start</td>
</tr>
<tr>
<td>( k )</td>
<td>the budget (i.e., the number of vaccines available)</td>
</tr>
<tr>
<td>( S )</td>
<td>set of nodes to give vaccines to</td>
</tr>
<tr>
<td>( E_S(F) )</td>
<td>the expected footprint after vaccinating ( S )</td>
</tr>
<tr>
<td>( \delta_Z(S) )</td>
<td>given graph ( Z ), the expected benefit of vaccinating ( S ) in ( Z )</td>
</tr>
<tr>
<td>( l )</td>
<td>number of samples</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>percentage of nodes that have ( p_i &gt; 0 ) in ( U )</td>
</tr>
</tbody>
</table>

We use two widely used propagation models to describe how the virus spreads on the network: the Independent Cascade (IC) model and the Susceptible-Infected-Recovered (SIR) model. SIR is a well-known epidemiological model to model mumps-like infections [17, 2]. A node in this model can be healthy (susceptible), infectious or recovered. When a node \( u \) becomes infected at the timestamp \( t \), it will try to infect each of its direct healthy neighbors \( v \) with the propagation probability \( \beta_{u,v} \). If \( u \) succeeds, \( v \) will become infectious at the timestamp \( t + 1 \). At the end of each timestamp \( t \), each infected node \( u \) has a curing probability \( \rho \) to become ‘recovered’ at the next timestamp \( t + 1 \). Once recovered, \( u \) will never be infected further. The process stops when no additional node becomes infectious. The IC model [21], a special case of SIR, has been extensively studied in the social media to model the viral marketing. Unlike SIR, a node \( u \) in IC has only single chance to infect its healthy neighbors (hence the curing probability, \( \rho = 1 \) here).

3. OUR PROBLEM FORMULATION

3.1 Uncertainty model

In this paper, we are concerned with the scenario when we know the underlying contact network, but we do not know the exact current infected state of the network. One source of uncertainty is public-health surveillance [39, 16, 34, 30, 12, 5]. Generally there are three types of surveillance: population-based, health provider-based and lab-based. Although different types of surveillance may have different probabilities to miss the truly infected person, we can simply use a set of probabilities \( P \) (over the nodes) to model such uncertainty. Another example is the likelihood ratios used in diagnostic testing; each a person has a probability \( p \) that her test was a false-negative. In Twitter, each relevant ‘infected’ tweet can be modeled as having some probability of being missed (because of uniform samples [32]).

Table 2 summarizes common probability distributions models \( U \) we use in this paper to model the uncertainty in observed infections. Each gives the probability of a node \( i \) not observed as infected being truly infected. We focus on fully factorizable distributions (over nodes) for simplicity.

Hence, if \( G_i \) denotes a particular configuration of infections in the network (i.e. a ‘possible world’), then \( \Pr(G = G_i) = \prod_{i \in I} p_i \prod_{h \in H} (1 - p_h) \) where \( I \) and \( H \) are the set of infected and healthy nodes in \( G_i \), and the probabilities \( p_i \) for any node \( i \) come from \( U \).

3.2 Problem Definition

Now we are ready to state our problem formally. We assume that a contagion can travel in principle from any node to any other node i.e. the graph is connected (strongly connected if directed). We are given a fixed-set \( I_0 \) with infected nodes, and an uncertainty model \( U \) as above. We are also given a budget \( k \) of vaccines. Giving a vaccine to a node renders it immune to the virus and hence it can not get infected further (effectively removing it from the network). Our goal is to find the ‘best’ set \( S \) of nodes to vaccinate to minimize the spread of the contagion, which can be measured by the so-called ‘footprint’, the number of infected nodes at the end. A subtle point is that vaccination is meaningful only for healthy nodes. Hence when we select a node-set \( S \), not all the nodes in \( S \) can be vaccinated (removed) in all possible sampled graphs: if a node \( i \) is infected in a possible world \( G_i \), then it can not be vaccinated in \( G_j \), and it does not give us any benefit there.

More specifically, suppose \( S \) is the set of nodes selected initially for vaccination and \( G_i \) is a particular realization (‘world’) sampled from \( U \). There only exist infected or healthy nodes in \( G_i \). Denote \( S_i \subseteq S \) as the subset of nodes

\(^1\)Extending our results to more general forms e.g. distributions factorizing over groups of nodes being infected is interesting future work.
distribution $\pi_i$ is the probability to be infected for each node. The probability to be infected for each node is proportional to its degree, i.e., people with larger number of connections have higher probabilities to be infected.

The probability to be infected for each node is proportional to its degree, i.e., people with larger number of connections have higher probabilities to be infected.

in $S$ that are healthy in $G_i$—these are the nodes which will be vaccinated in $G_i$. Denote $\sigma_{G_i}(S_i)$ as the expected number of infected nodes after running the epidemiology model e.g., IC, on $G_i$ starting from the infected nodes in $G_i$, but after removing nodes in set $S_i$. Let $F$ be the random variable denoting the number of infected nodes after choosing set $S$ under $U$. Then $E_S(F) = \sum_{G_i \in U} \Pr(G_i) \sigma_{G_i}(S_i)$ and we are trying to find the best set $S$ to minimize $E_S(F)$. Formally:

**Problem 1:** Uncertain Data-Aware Vaccination Problem: $\text{UDAV}(G, U, I_0, k)$.

**Given:** A graph $G(V,E)$ with node set $V$ and edge set $E$, the uncertainty model $U$, the infected node set $I_0$, propagation probability on each edge $(i,j)$ $\beta_{i,j}$, and an integer (budget) $k$.

**Find:** A set of nodes $S^* = \text{argmin} \ E_S(F)$ s.t. $|S| = k$.

Note that as vaccination will be applied only to healthy nodes in a possible world, this formulation also naturally generalizes the corresponding deterministic version of this problem (data-ware vaccination problem studied in DAV [46]).

**Complexity.** $\text{UDAV}$ is NP-hard, and cannot be approximated within an absolute error since its deterministic counterpart $\text{DAV}$ is itself NP-hard, and cannot be approximated within an absolute error [46].

### 4. OUR PROPOSED METHODS

**Overview.** In this section, we first present a sampling algorithm $\text{SAMPLE-CAS}$ for $\text{UDAV}$, which is a stochastic algorithm under the SAA framework. However, $\text{SAMPLE-CAS}$ is not scalable to large networks. Hence, we propose two faster algorithms: $\text{EXPECT-DOM}$ and $\text{EXPECT-EIG}$, which are based on the expected graph and measuring benefits of vaccinations. After analyzing the performance of $\text{EXPECT-DOM}$ and $\text{EXPECT-EIG}$, we show that these two algorithms are complementary w.r.t. the support of the uncertainty model, and hence we present a hybrid algorithm called $\text{EXPECT-MAX}$ with sub-quadratic running time.

We assume the $\text{GENERAL}$ model everywhere in this section (as the rest in Table 2 are just special cases of $\text{GENERAL}$). Further we describe the algorithms assuming the IC model first (Section 4.1 and 4.2)—later, we will discuss how to extend to the SIR model (Section 4.3).

#### 4.1 The Sample-Cascade Algorithm

**Main Idea.** Since $\text{UDAV}$ is a stochastic optimization problem, we try to apply the SAA (Sample Average Approximation) [22] framework to solve it. The idea is to reduce the stochastic optimization problem to the deterministic version by sampling the uncertainty distribution to generate a finite number of deterministic cases. Unfortunately, as we mentioned in the previous section, even the deterministic version of $\text{UDAV}$ is NP-hard. Hence we leverage the solution in [46], which utilizes a spanning tree called dominator tree, and then find a suitable sub-modular structure to solve $\text{UDAV}$ approximately.

**Details.** Let $\delta_{G_i}(S_i)$ be the expected benefit after vaccinating the healthy node set $S_i$ in a graph $G_i$, i.e.:

$$\delta_{G_i}(S_i) = \sigma_{G_i}(\emptyset) - \sigma_{G_i}(S_i)$$

(1)

So

$$E_S(F) = \sum_{G_i \in U} \Pr(G_i) \sigma_{G_i}(S_i)$$

$$= \sum_{G_i \in U} \Pr(G_i)(\sigma_{G_i}(\emptyset) - \delta_{G_i}(S_i))$$

(2)

Since $\sum_{G_i} \Pr(G_i)\sigma_{G_i}(\emptyset)$ is constant, $\text{UDAV}$ (Problem 1) can be rewritten as:

$$S^* = \text{argmax} \sum_{G_i} \Pr(G_i)\delta_{G_i}(S_i)$$ s.t. $|S| = k$.

So we need to compute $\delta_{G_i}(S_i)$ for each $G_i$, which is essentially the deterministic problem on graph $G_i$. Hence we re-purpose the solution from [46]: first merge all the infected nodes in $G_i$ into a super node $I_0$ ($I_0$ is infected). If a healthy node has multiple infected neighbors, $I_0$ will connect to the node with the probability that is the logical-OR of the individual probabilities (so if a node $u$ has two infected neighbors $x$ and $y$, $\beta_{I_0,u} = 1 - (1 - \beta_{x,u})(1 - \beta_{y,u})$). Secondly, build a dominator tree $\text{Dom}_{G_i}$ on this merged graph, and properly weight it. Briefly, given a source node $I_0$, a node $v$ dominates another node $u$ if every path from $I_0$ to $u$ contains $v$. Node $v$ is the immediate dominator of $u$, denoted by $v = \text{idom}(u)$, if $v$ dominates $u$ and every other dominator of $u$ dominates $v$. We can build a dominator tree rooted at $I_0$ by adding an edge between the nodes $u$ and $v$ if $v = \text{idom}(u)$ (totally in near-linear time [7, 26]). Finally, we approximate $\delta_{G_i}(S_i)$ as $\delta_{\text{Dom}_{G_i}}(S_i)$ (i.e. the benefit after removing nodes in $S_i$ from $\text{Dom}_{G_i}$).

In fact, can further prove that the real benefit of removing $S_i$ from graph $G_i$ is lower-bounded by $\delta_{\text{Dom}_{G_i}}(S_i)$.

**Lemma 1.** (Lower Bound of $\delta_{G_i}(S_i)$) The nodes we can save from $G$ must be greater than nodes we save from its dominator tree, that is, $\delta_{\text{Dom}_{G_i}}(S_i) \leq \delta_{G_i}(S_i)$ (where the inequality is saturated when the merged graph is a tree).

**Proof.** (Sketch) For any node $u$ in $S_i$, the benefit we can get on $G_i$ is at least all nodes under the subtree of $u$ in the dominator tree of $G_i$ (because there is no path from $I_0$ to those nodes). Hence, $\delta_{\text{Dom}_{G_i}}(S_i) \leq \delta_{G_i}(S_i)$. □

Let $Q(S) = \sum_{G_i} \Pr(G_i)\delta_{\text{Dom}_{G_i}}(S_i)$; then using Lemma 1, we get $Q(S) \leq \sum_{G_i} \Pr(G_i)\delta_{G_i}(S_i)$. The gap between $Q(S)$

<table>
<thead>
<tr>
<th>Name</th>
<th>Distribution</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIFORM</td>
<td>$p_i = p$</td>
<td>All nodes have identical probability to be infected. Can be thought of sampling rate in case of Twitter API [32].</td>
</tr>
<tr>
<td>SURVEILLANCE</td>
<td>$p_i \in \mathcal{P}$</td>
<td>Each node takes a probability from $\mathcal{P}$ (which is a finite set of probabilities like ${0.1, 0.5, 0.9}$). See Surveillance pyramid [39, 34, 30].</td>
</tr>
<tr>
<td>PROP-DEG</td>
<td>$p_i \propto d_i$</td>
<td>The probability to be infected for each node is proportional to its degree, i.e., people with larger number of connections have higher probabilities to be infected.</td>
</tr>
<tr>
<td>GENERAL</td>
<td>$p_i$</td>
<td>Each node has its own infected probability.</td>
</tr>
</tbody>
</table>
and \( \sum_{G_i} \Pr(G_i) \delta_{G_i}(S) \) depends on the structure of the graph (if the merged-graph is a tree, there is no gap). Hence, Lemma 1 suggests that we can use \( Q(S) \) to approximate \( \sum_{G_i} \Pr(G_i) \delta_{G_i}(S) \). Hence we formulate Problem 2 next, to approximate UDAV (Problem 1).

**Problem 2:** Given: \( G(V,E), U, I_0, \) and \( k \).

Find: A set of nodes \( S^* = \text{argmax}_S Q(S) \) s.t. \( |S| = k \).

Interestingly, \( Q(S) \) is a submodular function, while \( \delta_{G_i}(S) \) is not submodular [46].

**Lemma 2.** (Submodularity of \( Q(S) \)) \( Q(S) \) is a submodular function.

**Proof.** (Sketch) First of all, we prove that given a set \( S \), \( \delta_{\text{Dom} G_i}(S) \) is a submodular function of \( S \) (this can be done by a case analysis). Then \( Q(S) \) is a submodular because the linear combination of submodular functions is still a submodular function. \( \square \)

We now apply the SAA framework: sample \( l \) graphs \( G_1, G_2, \ldots, G_l \) from \( U \) and define \( Q_l(S) = \frac{1}{l} \sum_{i=1}^l \delta_{\text{Dom} G_i}(S) \). As \( Q_l(S) \) is a submodular function, we apply the greedy algorithm [33] to obtain the \((1-1/e)\)-approximation for Problem 2 (under the \( l \) samples). We call this algorithm **SAMPLE-CAS** (Algorithm 1). Note that we can speed up Algorithm 1 using CELF optimization [27].

### Algorithm 1 The SAMPLE-CAS algorithm

**Require:** Input \( G, U, I_0, k \) and \( l \)

1. Sample \( G_1, \ldots, G_l \) from \( U \) and \( G \)
2. Merge infected nodes into \( I_i^0 \) for each \( G_i \)
3. Build dominator trees \( \text{Dom} G_1, \ldots, \text{Dom} G_l \) rooted at \( I_i^0 \) for \( G_i \)
4. \( S = \emptyset \)
5. for \( i \leftarrow 1 \) to \( k \) do
6. \( a^* = \text{arg max}_a \frac{1}{l} \sum_{i=1}^l \delta_{\text{Dom} G_i} \)
7. Remove \( a^* \) from each of \( \text{Dom} G_1, \ldots, \text{Dom} G_l \)
8. \( S = S \cup \{a^*\} \)
9. end for
10. return \( S \)

**Lemma 3.** (Running Time of SAMPLE-CAS) The time complexity for Algorithm 1 is \( O((l(|V| + k|E| + |V| \log |V|)) \).

**Proof.** (Sketch) Sample \( G_i \) needs \( O(|V|) \) time, and build \( l \) dominator trees and weight it need \( O(|V| \log |V|) \) time. Selecting a node a needs \( l(|E| + |V|) \) time. So in general the time complexity is \( O((l(|V| + k|E| + |V| \log |V|)) \).

**How many samples?** The next lemma estimates the number of samples \( l \) needed so that \( Q_l(S) \) is a good estimate of \( Q(S) \).

**Lemma 4.** (Number of samples) For any \( \epsilon > 0 \), to estimate \( Q(S) \) within absolute error \( \epsilon \) with probability \( \gamma = 1 - 2e^{-\frac{2\epsilon^2}{\Delta^2}} \), we need \( l \geq \frac{\Delta^2\ln \frac{1}{\gamma}}{2\epsilon^2} \), where \( \Delta \) is the upper bound for \( \delta_{\text{Dom} G_i}(S) \) in the dominator tree.

**Proof.** (Sketch) It follows from using the well-known Hoeffding’s Inequality [31]. \( \square \)

As \( \Delta \) can be \( O(|V|) \), Lemma 4 shows that we need worst-case \( O(|V|^2) \) samples to get accurate estimates. Hence SAMPLE-CAS does not scale to large networks.

### 4.2 Expect-Max: a faster algorithm

Since SAMPLE-CAS is not scalable for large networks, we next develop another faster algorithm **EXPECT-MAX**. We give the main idea, and then describe the details in the subsequent subsections.

**Main Idea.** We first formulate an equivalent problem which uses the concept of a so-called ‘expected graph’ \( G_E \). Based on that, we propose two different methods **EXPECT-DOM** and **EXPECT-EG** measuring expected benefits of vaccinations. We show that these two methods are in fact complementary, and hence then propose **EXPECT-MAX**, which is sub-quadratic in running time (in nodes/edges).

#### 4.2.1 Expected Graph: An Equivalent Formulation

Here, we formulate an equivalent formulation of Problem 1 based on the concept of an ‘expected graph’.

**Definition 1 (Expected Graph):** The expected graph \( G_E \) is constructed as follows: start with \( G \); add a ‘super node’ \( I_0 \); connect \( I_0 \) to any node \( i \) where \( p_i > 0 \) with the edge weight \( \beta_{I_0,i} = p_i \) (\( p_i \in U \), the uncertainty model); and then mark all nodes except \( I_0 \) as healthy nodes.

As we show next, this construction transforms the uncertainty model from nodes to edges without losing any information. Hence, we can focus on a single graph \( G_E \) instead of sampling graphs (the main reason why SAMPLE-CAS was slow).

More specifically, we show in Lemma 5, an equivalent formulation of Problem 1 based on expected graph \( G_E \), under **GENERAL** and for budget \( k = 1 \). The main idea is that, crucially, as **GENERAL** is factorizable (i.e. for a particular configuration \( G_j \), \( \Pr(G \equiv G_j) = \prod_{a \in F} p_a \prod_{b \in H} (1-p_b) \), see Section 3 for details), after running the first step of the diffusion model on the expected graph, we will get the same configurations like sampling from the uncertainty model in Problem 1. A subtle point is that Lemma 5 also takes into account the fact that nodes can not be vaccinated in all ‘possible-worlds’ (wherever they are already infected), by correcting the estimate got from \( G_E \) by an appropriate factor.

**Lemma 5.** (Equivalent formulation of UDAV when \( k = 1 \)) When the budget \( k = 1 \), for the UDAV problem, the best node \( a^* = \text{arg max}_a \mathbb{E}(a) \) can be equivalently written as \( a^* = \text{arg max}_a (1-p_a) \delta_{G_E}(\{a\}) \).

**Proof.** (Sketch) We first prove \( \delta_{G_E}(a) = \sum_{G_i} \Pr(G_i) \delta_{G_i}(a) \) using Definition 1 and the factorizability of **GENERAL**. Based on this, we can prove that \( \mathbb{E}(a) = \sum_{G_i} \Pr(G_i) \sigma_{G_i}(0) - (1-p_a) \delta_{G_E}(\{a\}) \), hence minimizing \( \mathbb{E}(a) \) is equivalent to maximizing \( (1-p_a) \delta_{G_E}(\{a\}) \).

Lemma 5 shows that when the budget \( k = 1 \), we can get an equivalent formulation of Problem 1 based on the expected graph. Furthermore, note that UDAV is a stochastic problem, while Lemma 5 is based on calculating ‘benefits’ \( \delta_{G_E}(\{a\}) \) on a deterministic graph \( G_E \). Next we propose two heuristics to estimate \( \delta_{G_E}(\{a\}) \) on \( G_E \) which are complementary methods based on \( \alpha \), the support of the uncertainty model (see Section 4.2.4 for more details).

#### 4.2.2 The Expect-Dom Algorithm

One of the ways we can estimate the benefits is by using our Lemma 1 on \( G_E \). The main idea is that we estimate \( \delta_{G_E}(\{a\}) \) by its lowerbound \( \delta_{\text{Dom} G_i}(\{a\}) \) via the dominator tree on the expected graph. Motivated by the equivalent
formulation of Problem 1 (Lemma 5), we propose that at each step select a node with the maximum value of \((1 - p_a)\delta_{Dom_{G_a}}(\{a\})\) after building the dominator tree \(Dom_{G_a}\) of \(G_a\). We call this algorithm \textsc{Expect-Dom} (Algorithm 2).

**Algorithm 2 The \textsc{Expect-Dom} algorithm**

**Require:** Input \(G, U, I_0\) and \(k\)

1: Construct \(G_{G_a}\)
2: \(S = \emptyset\)
3: Build a dominator tree \(Dom_{G_a}\) on \(G_a\)
4: for \(i \leftarrow 1\) to \(k\) do
5: \(a^* = \arg\max_a(1 - p_a)\delta_{Dom_{G_a}}(\{a\})\)
6: \(S = S \cup \{a^*\}\)
7: Remove \(a^*\) from \(G_a\)
8: end for
9: return \(S\)

**Lemma 6.** (*Running Time of \textsc{Expect-Dom}*) The time complexity for Algorithm 2 is \(O(k(|V| + |E|) + |V| \log |V|)\).

**Proof.** (Sketch) Creating an expected graph \(G_{G_a}\) costs \(O(|V|)\) time, building a dominator tree and weight it need \(|V| \log |V|\) time. Updating dominator tree costs \(O(|V| + |E|)\) time. Hence, the time complexity of \textsc{Expect-Dom} is \(O(k(|V| + |E|) + |V| \log |V|)\). □

### 4.2.3 The \textsc{Expect-Eig} Algorithm

Another approach we propose is to estimate \(\delta_{G_a}(\{a\})\) is via the change in the largest eigenvalue of \(G_a\), \(\lambda_1(a)\), after removing node \(a\). The largest eigenvalue of the adjacency matrix of a graph is related to the so-called ‘epidemic threshold’ of the graph under several epidemic models [37, 36]. If the largest eigenvalue is very small, a virus will get extinguished quickly. Next we will explain why \(\Delta \lambda_1(a)\) is crucial to the benefits. In addition to that, we will show how to estimate \(\delta_{G_a}(\{a\})\) using the greedy algorithm in [43] as well.

**Justification of \(\Delta \lambda_1(a)\).** Let \(\lambda_i/u_i\) be the \(i\)-th largest eigenvalue/eigenvector of \(G_{G_a}\), and \(f_i\) be the vector of probability of each node being infected at time \(t\). The next lemma will show that the expected number of newly infected nodes is upper-bounded by a function of \(\lambda_1\). Hence, reducing \(\lambda_1\) (maximizing \(\Delta \lambda_1(a)\)) by removing node \(a\), can effectively minimize the expected number of newly infected nodes, and eventually minimize \(E_\alpha(F)\) (the expected maximum of infected nodes at the end). According to Equation \(2\) (in Section 4.1), minimizing \(E_\alpha(F)\) is equivalent to maximizing the benefit \(\delta_{G_a}(\{a\})\). Hence we can estimate \(\delta_{G_a}(\{a\})\) using \(\Delta \lambda_1(a)\).

**Lemma 7.** The expected number of newly infected nodes at timestep \(t+1\), is upper-bounded by \(h = e'(\sum_{j=1}^{V} \lambda_j^i u_j j')f_1\). Furthermore, \(h \leq \lambda_1 e'(\sum_{j=1}^{V} u_j j')f_1\) where \(e = (1, \ldots, 1)'\) and \(f_1 = (p_1, \ldots, p_n)'\) (the initial infection probabilities of the nodes, which essentially comes from the uncertainty model).

**Proof.** (Sketch) First, following steps of Lemma 1 in [36], we can get that the expected number of newly infected nodes at timestep \(t + 1\) is upper-bounded by \(e'(\sum_{j=1}^{V} \lambda_j^i u_j j')f_1 \leq \lambda_1 e'(\sum_{j=1}^{V} u_j j')f_1 = h_1\) \((f_1\) essentially comes from the uncertainty model). We first demonstrate that this inequality \((h \leq h_1)\) saturates when \(f_1\) is parallel to \(u_1\). Then, we will show how to maximize our chance to achieve this, which will lead us to the discussion about the performance of \textsc{Expect-Eig} in terms of \(\alpha\).

**4.2.4 The Hybrid Algorithm: \textsc{Expect-Max}**

Although \textsc{Expect-Dom} and \textsc{Expect-Eig} are both fast algorithms compared to \textsc{Sample-Cas}, they may not work well all the time. Next we will discuss how uncertainty models affect their performances, and present a hybrid algorithm combining both of them.

**Discussion about \textsc{Expect-Dom}**. Denote \(\alpha\) as the support of the uncertainty model (the percentage of nodes that are possibly infected). When \(\alpha = 0\), the \textsc{UDAV} problem becomes exactly the \textsc{DAV} problem [46] (the deterministic case of \textsc{UDAV}) and \textsc{Expect-Dom} reduces to the algorithm in [46], which was shown to perform well. However consider the opposite case \(\alpha = 1\). In this case, \(I_0\) connects to the rest of nodes. Hence the dominator tree of \(G_a\) becomes a star. For any node \(a\), \(\delta_{Dom_{G_a}}(a)\) will only depend on the propagation probability from \(I_0\) to \(a\) (i.e., \(p_a\)). We cannot utilize any other information from the original graph, hence we would choose nodes essentially randomly. This also gives us the intuition that as \(\alpha\) increases, the performance of \textsc{Expect-Dom} will become worse (a fact we demonstrate in experiments as well).

**Discussion about \textsc{Expect-Eig}**. As we discussed in Section 4.2.3, the expected number of newly infected nodes at timestep \(t + 1\) is upper-bounded by \(h = e'(\sum_{j=1}^{V} \lambda_j^i u_j j')f_1 \leq \lambda_1 e'(\sum_{j=1}^{V} u_j j')f_1 = h_1\) \((f_1\) essentially comes from the uncertainty model). We first demonstrate that this inequality \((h \leq h_1)\) saturates when \(f_1\) is parallel to \(u_1\). Then, we will show how to maximize our chance to achieve this, which will lead us to the discussion about the performance of \textsc{Expect-Eig} in terms of \(\alpha\).

**Lemma 9.** (*\(h-h_1\) Gap*) As the inner product of \(u_1\) and \(f_1\) increases, \(h_1 - h\) decreases. When \(f_1\) is parallel to \(u_1\), \(h_1 = h\).
PROOF. (Sketch) As \( u'_1 f_1 \) increases, \( f_1 \) becomes more parallel to \( u'_1 \), and \( u'_1 f_1 \) (\( j \neq 1 \)) becomes smaller (because \( u_1 \) and \( u'_1 \) are orthogonal). Hence \( h_2 - h_1 \) decreases. And when \( u'_1 f_1 = 0 \) (\( j \neq 1 \)), \( h_1 = h \). □

This shows that closer the uncertainty model is to \( u_1 \), the better bound \( h_1 \) is of \( h \): as a result of which we expect \( \Delta \lambda_1 (a) \) to become a better estimate, and hence \( \text{EXPECT-EIG} \) to perform better. How is this related to \( a' \)? The following analysis shows a preliminary justification. Apriori we do not know the graph, hence we do not know \( u_1 \); so reasonably we can assume it is randomly uniformly picked from a \( n \)-dimensional space. Let us denote \( x \) as the random variable of the first eigenvector. To make \( f_1 \) more parallel to \( x \), we need to maximize the expectation of \( f'_1 x \) (i.e., \( E_x[f'_1 x] \)). It is not hard to see that as we increase \( \alpha \), \( E_x[f'_1 x] \) will increase.

**Lemma 10. (Expected gap) When \( \alpha \) increases, \( E_x[f'_1 x] \) increases as well.**

**Proof.** (Sketch) \( E_x[f'_1 x] = f'_1 E_x[x] \), and all elements in \( E_x[x] \) are non-negative. As \( \alpha \) increases, more elements in \( f'_1 \) become non-zero, hence \( E_x[f'_1 x] \) increases as well. □

Lemma 10 suggests that when \( \alpha \) increases, we expect \( f_1 \) and \( u_1 \) to become more parallel, and so the gap to decrease, as a result of which \( \Delta \lambda_1 (a) \) becomes a better estimate. Thus even this preliminary analysis immediately suggests that as \( \alpha \) becomes larger, \( \text{EXPECT-EIG} \) should perform better. Again we demonstrate this through experiments as well.

**The Expect-Max Algorithm.** The above discussion suggests a complementary picture: when \( \alpha \) is low, we expect \( \text{EXPECT-DOM} \) to be better, and when \( \alpha \) is high, we expect \( \text{EXPECT-EIG} \) to be better. Unfortunately, we don’t know exactly when which algorithm is better: this likely depends not only on \( \alpha \) but also the graph, and the distribution. However, we can still leverage this insight to propose a hybrid algorithm called \( \text{EXPECT-MAX} \), which maintains the scalability and quality of \( \text{EXPECT-DOM} \) and \( \text{EXPECT-EIG} \). \( \text{EXPECT-MAX} \) chooses either \( \text{EXPECT-DOM} \) or \( \text{EXPECT-EIG} \) based on their performances, that is,

\[
S_{\text{EXPECT-MAX}} = \arg\max_{s \in \{S_{\text{EXPECT-DOM}}, S_{\text{EXPECT-EIG}}\}} E_S(F)
\]

Comment. \( S \) is the output either of \( \text{EXPECT-DOM} \) or \( \text{EXPECT-EIG} \), and \( E_S(F) \) can be obtained by via simulation of the IC model (not sampling from the uncertainty model). Also note that \( \text{EXPECT-MAX} \) is not the greedy algorithm that picks one node from either \( \text{EXPECT-DOM} \) or \( \text{EXPECT-EIG} \) in each step. Instead, it chooses \( S \) just once after running \( \text{EXPECT-DOM} \) and \( \text{EXPECT-EIG} \). Hence the time complexity for \( \text{EXPECT-MAX} \) is \( O(k(|V| + |E|) + |V| \log |V| + T) \) where \( T \) is the time to run IC model (which should be sub-quadratic in edges).

### 4.3 Extending to SIR model

Note that in SIR, the footprint is the total number of recovered nodes at the end (in contrast to the IC model). Nevertheless, leveraging the method in [46], we can directly extend our algorithms to SIR model by changing IC model to SIR model with the propagation probability \( 1 - (1 - \beta_{i,j})^{\frac{1}{a'}} \). This does not change any of our algorithms/results.

## 5. Experiments

We present a detailed experimental evaluation in this section.

### 5.1 Experimental Setup

We briefly describe our setup next. We implemented the algorithms in Python\(^2\), and conducted the experiments using a 4 Xeon E7-4850 CPU with 512GB of 1066Mhz main memory.

**Datasets.** We ran our experiments on multiple datasets using both IC and SIR. Table 3 summarizes the datasets, which were chosen for their size as well as the applicability to the UDAV problem (from social media to epidemiology).

1. **KARATE** is a social network of friendships with 34 members in a karate club at a US university in the 1970s [45].
2. **OREGON**\(^4\) is the Oregon AS router graph collected from the Oregon router views. The contagion here can be thought of malware and computer-network viruses, which we want to control by shutting-off or patching relevant routers.
3. **STANFORD**\(^4\) is the Stanford CS hyperlink network, in which a web page links to another page. Contagions here can be false information spreading through the webpages, and we want to prevent their spread by posting true information at strategic web pages.
4. **GNUTELLA**\(^4\) is a peer-to-peer network showing the snapshot of the Gnutella P2P file sharing network from August 2012. Similar to OREGON, we can control the spread of malware and harmful files by patching some important peers.
5. **BRIGHTKITE**\(^5\) is a friendship network from a location-based social networking service provider Brightkite. As friends regularly frequent the same places, such location-based networks can be useful for the public-health.
6. **PORTLAND** and **MAMI** are social-contact graphs based on detailed microscopic simulations of large US cities. Edge weights here represent the expected contact time between people. Versions of these have been used in national smallpox and influenza modeling studies using the SIR model [15].

| Table 3: Datasets |
|-------------------|-----------------|-----------|---------|
| **Dataset**      | **Nodes(V)**    | **Edge(E)**| **Model**|
| KARATE           | 34              | 156       | IC      |
| OREGON           | 633             | 2172      | IC      |
| STANFORD         | 8929            | 53829     | IC      |
| GNUTELLA         | 10876           | 39994     | IC      |
| BRIGHTKITE       | 59228           | 0.2 million | IC   |
| PORTLAND         | 0.5 million     | 1.6 million | SIR   |
| MAMI             | 0.6 million     | 2.1 million | SIR   |

**Uncertainty models.** We used three types of uncertainty models: (a) **UNIFORM**: \( p = 0.6 \); (b) **SURVEILLANCE**: \( p_i \) uniformly randomly chosen from \( \{0, 0.5\} \) for each node \( i \) (following different levels of the surveillance pyramid, e.g. 10%\(^6\))

\(^2\)Code can be downloaded from http://people.cs.vt.edu/~yaozhang/code/udav

\(^4\)http://topology.eecs.umich.edu/data.html

\(^5\)http://www.cise.ufl.edu/research/sparse/matrices/

\(^6\)GNUTELLA and BRIGHTKITE are from http://snap.stanford.edu/data/index.html
of the total population is infected and is in the hospital, and only ~33% of infected people go to a hospital, which together imply 50% of the total population is infected and does not go to a hospital; (c) PROP-DEG: \( p_i = \frac{d_i}{d_{max}} \) for each node \( i \) (\( d_{max} \) is the maximum degree of the graph \( G \)).

**Parameters.** For IC model, \( \beta_{u,v} \) is uniformly randomly chosen from \{0.1, 0.5, 0.9\} [9]. For SIR model, we use the normalized contact time as the propagation probability \( \beta_{u,v} \), and set a uniform curing probability \( \rho = 0.6 \). We uniformly randomly pick 5% of nodes as infected nodes. For SAMPLE-Cas, we set the number of samples \( i = 200 \). For robustness, each data point we show is the mean of 1000 runs of the diffusion/epidemiological model.

**Baselines.** We compare our algorithms against various intuitive and non-trivial competitors to better judge their performance. Recall that \( I_0 \) is the infected node set (\( I_0 = \{ u | u \in V, p_u = 1 \} \)). Let us denote \( W = V - I_0 \) (so \( W \) is the set of nodes that are not certainly infected at the start).

1. **OPTIMAL:** a brute-force algorithm that tries all combinations of possible solutions. As it is very slow, we only run it on very small graph (KARATE).
2. **RANDOM:** uniformly randomly select \( k \) nodes from \( W \).
3. **DEGREE:** choose the top-\( k \) nodes from \( W \) according to their weighted degree.
4. **PAGERANK:** pick the top-\( k \) healthy nodes from \( W \) with the highest pagerank. We use the restart probability of 0.15.
5. **PER-PRANK:** we first merge all infected nodes into one supernode as the preferred node, and then choose the top-\( k \) nodes from \( W \) with the highest personalized pagerank with respect to the supernode [18]. We use the restart probability of 0.15.
6. **DAVA-fast:** This is a fast immunization algorithm [46], which aims to control the epidemic in presence of already infected nodes (without uncertainty in the data). We apply DAVA-fast as if any node from \( W \) on \( G \) is a healthy node. We take the top-\( k \) nodes from \( W \) according to the algorithm.

### 5.2 Results

In short, we demonstrate that SAMPLE-Cas and EXPECT-MAX outperform other baselines on all datasets. SAMPLE-Cas provides very accurate results, but does not scale to large networks, while EXPECT-MAX is fast, scalable and effective. We also show the behaviors of EXPECT-DOM and EXPECT-EIG as \( \alpha \) varies.

#### 5.2.1 Accuracy of SAMPLE-CAS

First of all, we compare SAMPLE-Cas with OPTIMAL on KARATE to demonstrate its accuracy (because OPTIMAL is too slow, we chose KARATE so that we can run OPTIMAL completely). As Figure 1(a) shows, for all uncertainty models, SAMPLE-Cas saves at least 90% of nodes compared to OPTIMAL no matter how \( k \) changes. We also found as expected, SAMPLE-Cas’s performance gets better as number of samples increases (not shown here).

#### 5.2.2 Justification of EXPECT-MAX

We compared EXPECT-DOM with EXPECT-EIG as \( \alpha \) changes on multiple datasets under three uncertainty models (see Figure 1(b) and (c)). For all networks, as expected from our discussion in Section 4.2.4, clearly as \( \alpha \) increases, EXPECT-EIG becomes better while EXPECT-DOM becomes worse. In addition to that, there does exist a ‘cross-over point’ for each network where the algorithms switch in performance (\( R = 1 \) in Figure 1(b) and (c)). However, this cross-over point is different for different networks and for different distributions, which is the reason why we propose the EXPECT-MAX algorithm (as we do not know exactly when we should use either EXPECT-DOM or EXPECT-EIG as \( \alpha \) changes).

#### 5.2.3 Effectiveness of SAMPLE-CAS and EXPECT-MAX

Figure 2(a), (b), (d) and (e) show experimental results under IC model for UNIFORM. In all networks, SAMPLE-Cas and EXPECT-MAX consistently outperform other competitors. OREGON contains only 600 nodes, hence we varied \( k \) till 50. Due to a jelly-fish type structure of OREGON, for lower \( k \), most algorithms perform well by targeting the nodes in the core. However, for larger \( k \), SAMPLE-Cas provides the best solution, while EXPECT-MAX outperforms other competitors as well, getting solutions almost as good as SAMPLE-Cas. For GNUTELLA, STANFORD and BRIGHTKITE (much larger than OREGON), the difference of SAMPLE-Cas and EXPECT-MAX from the other algorithms is clearer: they save up to 2.5 times the nodes than other algorithms, yet EXPECT-MAX took a fraction of the running time of SAMPLE-Cas (see Ta-
Figure 2: Effectiveness ($\alpha = 0.5$, UNIFORM). Expected number of healthy nodes after distributing vaccines vs. budget $k$. Higher is better. (a), (b), (d), (e): IC model; (c), (f): SIR model. Sample-Cas and Expect-Max outperform other baseline algorithms.

Table 4: Running times (sec.) when $k = 100$ and $t = 200$ ($\alpha = 0.5$). Runs terminated when running time $t > 24$ hours. (shown by `-`)

<table>
<thead>
<tr>
<th></th>
<th>SAMPLE-CAS</th>
<th>EXPECT-MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>OREGON</td>
<td>241.6</td>
<td>3.1</td>
</tr>
<tr>
<td>STANFORD</td>
<td>3401.7</td>
<td>45.2</td>
</tr>
<tr>
<td>GNUTELLA</td>
<td>4221.1</td>
<td>59.8</td>
</tr>
<tr>
<td>BRIGHTKITE</td>
<td>19072.0</td>
<td>371.5</td>
</tr>
<tr>
<td>PORTLAND</td>
<td>-</td>
<td>6930.2</td>
</tr>
<tr>
<td>MIAMI</td>
<td>-</td>
<td>9231.4</td>
</tr>
</tbody>
</table>

Note: Although DAVA-fast contains information of infected nodes, it doesn’t perform well (especially on STANFORD) because it fails to take into account the uncertainty model.

We got similar result under SIR model on PORTLAND and MIAMI for UNIFORM (see Figure 2(c) and (f)). Since PORTLAND and MIAMI have more than 0.5million nodes, SAMPLE-CAS did not finish even in a day, and we do not show it on the plots. We notice that the larger $k$ becomes, the better EXPECT-MAX performs than other competitors. When $k = 2000$, the difference of EXPECT-MAX from other algorithms is clearer: it saves more than 10,000 nodes than the second best algorithm DAVA-fast.

For SURVEILLANCE and PROP-DEG, the results are the same: SAMPLE-CAS and EXPECT-MAX always outperform other algorithms (see Figure 3). We do not show the plots of other datasets and other values of $\alpha$ due to lack of space, but the results are similar: SAMPLE-CAS and EXPECT-MAX provide the best solution.

5.2.4 Scalability

Although both SAMPLE-CAS and EXPECT-MAX are polynomial time (in particular EXPECT-MAX is subquadratic in nodes and edges), we show some running time results to evaluate scalability. Table 4 shows the running times of our algorithms under UNIFORM. EXPECT-MAX is much faster than SAMPLE-CAS. EXPECT-MAX takes only 45 seconds on STANFORD while SAMPLE-CAS takes about an hour.

The larger networks are, the faster EXPECT-MAX is than SAMPLE-CAS. On BRIGHTKITE with 60K nodes, EXPECT-MAX is more than 50 times faster than SAMPLE-CAS. Furthermore, on the largest network MIAMI, EXPECT-MAX takes about 2.5 hours to select 100 nodes while SAMPLE-CAS did not finish even in one day. Hence, EXPECT-MAX is scalable for large networks.

6. RELATED WORK

We now review the most closely related work here.

Stochastic Optimization. Extensive surveys and textbooks [40, 20] exist on this topic. Dyer et al. [14] showed that two-stage stochastic programming problems are \#P-hard. Sample Average Approximation (SAA) is a well-known framework to approach these problems, which yields strong approximation results [22, 41]. We leveraged the SAA framework for SAMPLE-CAS in this paper. However, it is not scalable to large networks because of its computational complexity.

Handling Uncertainty. Many studies in epidemiology try to estimate the total infections using the surveillance pyramid [39, 16, 34, 30, 12, 5]. More generally, missing data in
networks is an important yet relatively poorly understood problem. A line of work in databases studies several querying problems on uncertain graphs, including the k-nearest neighbors query [35], discovering reliable subgraphs [19] and efficient subgraph search [44]. Another related line of work studies the effect of sampling on measured structural properties [11, 23, 4] or network construction [25, 29]. Correcting for the effects of missing data in cascades in general has not seen much attention—the exceptions are Sadikov et al. [38] (who try to correct metrics like cascade size for sampling), and Adiga et al. [1] (who study the effect of more general noise in the network structure on metrics like expected footprint in the IC and LT models). Here we study a specific algorithmic task (immunization) under uncertainty in observed infections.

Immunization Algorithms. Most existing studies focus on finding optimal strategies for vaccine allocation under perfect information [10, 6, 28, 8]. Using game theory, Aspnes et al. [3] developed inoculation strategies for victims of viruses under random starting points. Kuhlman et al. [24] studied two formulations of the problem of blocking a contagion through edge removals under the model of discrete dynamical systems. Tong et al. [43, 42] and Prakash et al. [36] proposed various node-based and edge-based immunization algorithms based on minimizing the largest eigenvalue of the graph. Zhang and Prakash [46] studied the problem of immunizing healthy nodes in presence of already infected nodes.

To summarize, none of the above works studies the problem of distributing vaccines given uncertain surveillance data.

7. CONCLUSIONS

This paper addresses the problem of distributing vaccines given uncertain data over large networks with applications to cascade-like processes on networks in several areas. The main contributions are:

(a) Problem Formulation: Motivated by multiple natural uncertainty models from social media and epidemiology, we first formulated the Uncertain Data-Aware Vaccination (UDAV) problem.

(b) Efficient Algorithms: Due to its computational complexity, we presented two main novel algorithms: SAMPLE-CAS and EXPECT-MAX. SAMPLE-CAS is an accurate stochastic algorithm under the SAA framework, while EXPECT-MAX is a fast hybrid algorithm with sub-quadratic time complexity, which utilizes the expected graph and two complementary methods to estimate benefits.

(c) Extensive Experiments: Experimental results demonstrate that our algorithms outperform several other baseline algorithms on multiple diverse real datasets (from social, cyber, and epidemiological domains) over multiple different uncertainty models.

Future work can include extending our results to other models in epidemiology such as SIS (where nodes can get infected multiple times), and generalizing EXPECT-MAX to any uncertainty distribution (not just factorizable distributions).

Acknowledgements. The authors would like to thank the anonymous reviewers for their comments, and Virginia Bioinformatics Institute for sharing with us the PORTLAND and MIAMI datasets. This material is based upon work supported by the NSF under Grant No. IIS-1353346 and by the Maryland Procurement Office under contract H98230-14-C-0127. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the respective funding agencies.

8. REFERENCES


