DAVA: Distributing Vaccines over Networks under Prior Information

Yao Zhang* B. Aditya Prakash†

Abstract

Given a graph, like a social/computer network or the blogosphere, in which an infection (or meme or virus) has been spreading for some time, how to select the k best nodes for immunization/quarantining immediately? Most previous works for controlling propagation (say via immunization) have concentrated on developing strategies for vaccination *pre-emptively* before the start of the epidemic. While very useful to provide insights in to which baseline policies can best control an infection, they may not be ideal to make *real-time* decisions as the infection is progressing.

In this paper, we study how to immunize healthy nodes, in presence of already infected nodes. Efficient algorithms for such a problem can help public-health experts make more informed choices. First we formulate the *Data-Aware Vaccination* problem, and prove it is NP-hard and also that it is hard to approximate. Secondly, we propose two effective polynomial-time heuristics `DAVA` and `DAVA`-fast. Finally, we also demonstrate the scalability and effectiveness of our algorithms through extensive experiments on multiple real networks including epidemiology datasets, which show substantial gains of up to *10 times* more healthy nodes at the end.

1 Introduction

Given a set of already infected people in a population, what are the healthy people who should be immediately given vaccines to best control the epidemic? How should Twitter decide which accounts to suspend/delete to stop rumors which are already present? Propagation-style processes on graphs/networks are powerful tools for modeling situations of interest in real-life like in social-systems, in cyber-security and in epidemiology. For example diseases spreading over population contact networks, spam/rumors spreading on Facebook/Twitter, malware spreading over computer networks, all are propagation processes. As a result, manipulating and controlling such harmful propagation is an important and natural problem with numerous applications.

In this paper, we concentrate on the problem of how best to distribute vaccines to nodes on a network, when the disease has already spread to certain parts of the graph. Intuitively we want to study how to best build a 'wall' against an emerging and already spreading contagion. We assume the vaccines 'immunize' the nodes completely i.e. they are removed from the network immediately. Most previous work in immunization (see Section 7 for a survey) tries to give algorithms and policies for so-called *pre-emptive* immunization, in which we are looking for the best baseline policies for controlling an epidemic, before the epidemic has started. As discussed later, these papers immunize against an assumption that the epidemic can start anywhere at a random point. In practice, this assumption is almost never true—e.g., travelers or people working with animals (say in case of avian flu) are more likely to get infected and hence an epidemic is much more likely to start from them. Hence while such baseline policies give us good guidelines, they may not be ideal for making *real-time* decisions, given that an epidemic has already affected sets of people. Hence efficient algorithms for such a *Data-Aware Vaccination* problem can help policy makers react and make better policy choices to changing conditions. The problem has clear applications to social and cyber systems as well: which user-accounts should be disabled in Facebook to best stop an active spam message? which computer nodes should install the patches first in presence of malware attacks?

Our main contributions include:

1. *Problem Formulation and Hardness Results*: We formulate the *Data-Aware Vaccination* problem as an combinatorial optimization problem on arbitrary graphs, and prove it is NP-hard and also hard to approximate within an absolute error (see Section 4).

2. *Effective Heuristics*: We present effective polynomial-time heuristics `DAVA` and `DAVA`-fast for general graphs, and `DAVA`-tree, an optimal algorithm for m-trees (see Section 5).

3. *Experimental Evaluation*: Finally, we present extensive experiments against several competitors on multiple real datasets (including large epidemiological social-contact graphs), and demonstrate the efficacy and scalability of our algorithms. Our algorithms outperform other algorithms by up to 10 times in both magnitude and running-time.

*Department of Computer Science, Virginia Tech. Email: yaozhang@cs.vt.edu.

†Department of Computer Science, Virginia Tech. Email: badityap@cs.vt.edu.

The rest of the paper is organized as follows. Section 2 presents some preliminaries while Section 3 sets up the Data-Aware Vaccination problem. Section 4 discusses the computational complexity of our problem, Section 5 presents our algorithms and Section 6 presents experimental results on several datasets. We finally conclude in Section 8.

2 Preliminaries

We give preliminaries in this section. Table 1 lists the main symbols used in the paper. There is an underlying contact network G (between people/computers/blogs etc.) on which the contagion (disease/virus/meme etc.) spreads. For ease of exposition, we assume our graph is weighted and undirected, but all our methods and machinery can be naturally generalized to directed graphs.

Table 1: Terms and Symbols

Symbol	Definition and Description
DAV	Data-Aware Vaccination problem
SIR	Susceptible-Infected-Recovered Model
IC	Independent Cascade Model
$G(V, E)$	graph G with nodes set V and edges set E
I_0	infected nodes set
$p_{u,v}$	propagation probability from node u to v
δ	curing probability for the SIR model
k	the budget (i.e., #nodes to give vaccines to)
S	set of nodes selected for vaccination
$\sigma_{G,I_0}(S)$	the expected number of infectious nodes at the end (footprint)
$\sigma'_{G,I_0}(S)$	the expected number of healthy nodes at the end
$\gamma_S(j)$	the expected benefit of $\sigma'_{G,I_0}(*)$ when adding j into S

We use two well-known and popular discrete-time virus propagation models to model the virus spreading on the network: the so-called Susceptible-Infected-Recovered (SIR) model, and its special case the Independent Cascade (IC) model. *SIR* is a fundamental model which has been extensively used in epidemiology [1, 13, 19] to model mumps (or chicken-pox) like infections[1]. A node in the network is in one of three states: susceptible (healthy), infected and recovered. In each time-step, every infected node u tries to infect each of its healthy neighbors v *independently* with probability $p_{u,v}$ (the weight on each edge). The healthy neighbors who are successfully attacked are considered to be infected from the next time-step. In addition, at each time-step each infected node has a curing probability δ to become recovered (from the next step). Once recovered, a node does not participate in the epidemic further. The process begins when some initial nodes are infected and terminates when no infected nodes remain.

IC is a well-known model [14] which is used to abstract viral marketing and related meme processes. In contrast to SIR, in the IC model, each infected node gets precisely one chance to infect ('activate') its neighbors with the edge-weight probability (in effect the curing probability $\delta = 1$ here). We will first describe our algorithm on this model. Subsequently, extending our algorithm to handle the general SIR case is reasonably straightforward (which we explain in Section 5.4).

[1]Generalizing our results to other disease models like the flu-like SIS model is interesting future work.

3 Problem Formulation

We are now ready to formulate our problem formally. We are given a fixed-set of nodes I_0, which are all the infected nodes at the start of the process. If we give a vaccine to a *healthy* node v, it means that it cannot be infected by its neighbors at any time, effectively *removing* it from the graph. We are given a graph $G(V, E)$, the set I_0 and a budget of k vaccines. We want to find the 'best' set S of healthy nodes that should be given vaccines at the beginning. As the propagation model is stochastic, denote $\sigma_{G,I_0}(S)$ to be the expected total number of infected nodes during the whole process (the 'footprint'), given that I_0 nodes were infected at the start, and S was the vaccinated set. The best set S is the one which minimizes $\sigma_{G,I_0}(S)$. Formally,

PROBLEM 1: *Data-Aware Vaccination* problem $DAV(G, I_0, P, \delta, k)$.

GIVEN: *A graph $G(V, E)$ with node set V and edge set E, the initially infected node set I_0, SIR model with propagation probability on each edge $\{i, j\}$ $p_{i,j} \in P$ and curing probability δ, and an integer (budget) k.*

FIND: *A set S of k nodes from $V - I_0$ to distribute vaccine to minimize $\sigma_{G,I_0}(S)$, i.e.*

$$S^* = \operatorname*{argmin}_S \sigma_{G,I_0}(S) \quad s.t. \quad |S| = k \tag{3.1}$$

Comment 1 Define $\sigma'(\cdot)$ to be the expected number *healthy* nodes at the end i.e. $\sigma'_{G,I_0}(S) = |V| - \sigma_{G,I_0}(S)$. Given the same set S and an integer k, clearly minimizing $\sigma_{G,I_0}(S)$ is equivalent to maximizing $\sigma'_{G,I_0}(S)$ (as nodes can either be infected or healthy). In this paper, for ease of description we adopt this alternate form (of maximizing $\sigma'_{G,I_0}(S)$).

Comment 2 Clearly the problem is trivial when $k \geq |N(I_0) - I_0|$, where $N(I_0)$ is the set of immediate neighbors of I_0 in the graph G (as we can just vaccinate all of these nodes, and the disease will stop). In reality, this is never the case, as vaccines are expensive and networks are huge. For example, for $k = 10$, in our

experiments, we found that $|N(I_0) - I_0|$ ranged from 10 to 250 times our budget. Hence, we will implicitly assume that $k < |N(I_0) - I_0|$.

4 Complexity of the DAV problem

We discuss the complexity of the Data-Aware Vaccination problem next. In summary, we first prove that *on general graphs*, the DAV problem is NP-hard, then show that DAV problem is also hard to approximate.

4.1 Hardness result Unfortunately, our problem is NP-hard. We will reduce it from the MINIMUM K-UNION (MINKU) set problem (where we want to minimize the union of k subsets), which was recently proven to be hard [23].

Consider the corresponding decision version of DAV:
PROBLEM 2: Data-Aware Vaccination (Decision Version) $DAV(G, I_0, P, \delta, k, \tau)$:

GIVEN: *A graph $G(V, E)$, the node set I_0, the SIR model with propagation probability $p_{i,j} \in P$ and curing probability δ, and an integer (budget) $k \geq 0$, and $\tau \geq 0$.*

FIND: *Is there a set S of k nodes of G to distribute the vaccine such that $\sigma'_{G,I_0}(S) \geq \tau$?*

THEOREM 4.1. *DAV (Decision Version) is NP-hard.*

Proof. (Brief sketch) We can reduce the MINKU problem to an instance of the DAV problem with $\delta = 1$ (hence the DAV problem under the IC model is also NP-hard).

4.2 Approximability Typically related optimization problems arising in graphs have a submodular structure lending themselves to the near-optimal greedy solution. But unfortunately, our function is *not* submodular.

REMARK 4.1. *$\sigma'_{G,I_0}(S)$ in DAV is not a submodular function.*

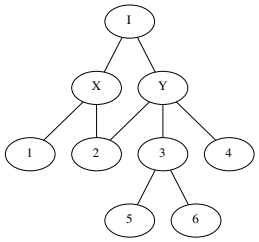


Figure 1: Counter-Example

See Fig. 1. A submodular function has the property that if $A \subseteq B$, then adding an element j into both sets, we should have $f(A \cup \{j\}) - f(A) \geq f(B \cup \{j\}) - f(B)$. Suppose I is infected and $A = \emptyset$ and $B = \{X\}$, we have $\sigma'_{G,I_0}(A \cup \{Y\}) - \sigma'_{G,I_0}(A) = 5$ and $\sigma'_{G,I_0}(B \cup \{Y\}) - \sigma'_{G,I_0}(B) = 8 - 2 = 6$. So $\sigma'_{G,I_0}(S)$ is not a submodular function.

Vinterbo [23] gave a greedy algorithm which can approximate MINKU and its equivalent problem Maximum k-Intersection problem (MAXKI) (where we want to *maximize* the *intersection* of k given subsets) within a constant factor if the cardinality of all the subsets are bounded by a constant. However, in our case the 'subsets' can be very large, and thus the approximation result is not useful. The algorithm we develop is related to Vinterbo's setting in the sense that it is also greedy, though we use different efficient techniques for our particular setting. For the general case, Xavier [24] proved that the MAXKI problem cannot be approximated within a tighter constant $\frac{1}{N^\epsilon}$ where N is the number of subsets and $\epsilon > 0$, under $\mathbf{NP} \not\subset \mathbf{BPTIME}(2^{N^\epsilon})$. Recently Sheih et al [20] proved MAXKI is inapproximable within an absolute error, with a smaller inapproximable gap and under the weaker $\mathbf{P} \neq \mathbf{NP}$ assumption. Using the results in [20], unfortunately our problem is also inapproximable within an absolute error $\frac{1}{2}m^{1-2\epsilon} + O(m^{1-3\epsilon})$ if $m = |V| - |N(I_0) - I_0| - |I_0|$. Here, m means the number of nodes except for the infected nodes and their neighbors.

THEOREM 4.2. *Given any constant $0 < \epsilon < 1/3$, there exists a m_ϵ such that the Data-Aware Vaccination problem with $m > m_\epsilon$, cannot be approximated in polynomial time within an absolute error of $\frac{1}{2}m^{1-2\epsilon} + \frac{3}{8}m^{1-3\epsilon} - 1$ unless **P=NP**.*

Proof. Omitted for brevity.

5 Our Proposed Method

Due to the results in the previous section, we present effective heuristics next. We first describe our methods assuming the IC model in the DAV problem (Section 5.1-5.3), and then extend our algorithm to handle the general SIR case (Section 5.4).

5.1 Merging infected nodes The first observation is that as the reduction suggests, we can merge the all the infected nodes into a *single* 'super infected' node and get an equivalent simplified problem with a single infected node. Intuitively this is because it does not matter how the infected nodes are connected among themselves—all it matters for our problem is how they are connected to *healthy* nodes. If a healthy node has multiple infected neighbors, it will have a new edge probability which would be the logical-OR of the individual probabilities. For example, if a healthy node c has two infected neighbors a and b with edge probabilities p_a and p_b, the new edge probability between I' and c would be $1 - (1 - p_a)(1 - p_b) = p_a + p_b - p_a p_b$.

REMARK 5.1. *Given an instance of the DAV problem (G, I_0, P, k) under IC model, Algorithm 1 outputs an equivalent problem instance (G', I', P', k) where I' is the sole infected node in the new graph G'.*

Algorithm 1 MERGE

Require: Input graph G, infected node set I_0, probability set P

1: $G' = G$
2: Add node I' to G'
3: **for** each node i in I_0 **do**
4: **if** there exists an edge e_{ij} between i and j **then**
5: **if** there is no edge $e_{I'j}$ **then**
6: Add edge $e_{I'j}$ into G'
7: $p_{I'j} \leftarrow p_{ij}$
8: **else**
9: $p_{I'j} \leftarrow p_{I'j} + (1 - p_{I'j})p_{ij}$
10: **end if**
11: Remove e_{ij} from G'
12: **end if**
13: **end for**
14: Remove all nodes in I_0 from G'
15: **return** graph G' and the infected node I'

In Algorithm 1, line 3-13 is used to copy edges from previous infected nodes set I_0 to the new infected node I'. Line 5-10 shows how to assign new propagation probability $p_{I'j}$. Suppose the previous infected nodes set I_0 has E_{I_0} edges in total, Algorithm 1 will take $O(|I_0| + E_{I_0})$ time.

5.2 DAVA-TREE—Optimal solution when the merged graph is a tree Let's call the graph we get after merging (i.e. after Algorithm 1) the *m-graph*. The second important observation is that as we show next, if the *m-graph* of our instance is a tree, then we can get an optimal polynomial time algorithm under IC model, for any edge propagation probability $p_{i,j} \in [0, 1]$. We call the algorithm DAVA-TREE (Data Aware Vaccine Allocation on a tree).

Before we describe our algorithm, define the quantity $\gamma_S(j)$—which is the '*benefit*' of node j to the optimization goal when nodes in S have *already* been removed. It is essentially the expected number of nodes we save after removing j, given that we have already removed nodes from set S. We have:

$$\gamma_S(j) = \sigma'_{G,I_0}(S \cup \{j\}) - \sigma'_{G,I_0}(S) \tag{5.2}$$

Let $\gamma(j) = \gamma_\emptyset(j)$. Algorithm 2 proceeds by computing γ_j efficiently for every neighbor node of I' (in a simple tree traversal) and then taking those neighbors of the infected node I' with top-k $\gamma(\cdot)$ values. Lemma 5.1 proves that this gives us the optimal solution. In short, as there is only one path from any node to any other node in the tree: the optimal solution must be a subset of the immediate neighbors of I' with top k value of $\gamma(j)$.

Algorithm 2 DAVA-TREE

Require: Tree T, infected node I', k and p_{ij}

1: Set I' as the root
2: **for** each neighbor j of I' **do**
3: $\gamma(j) \leftarrow p_{I'j} \times$ calPartial(j)
4: **end for**
5: $S =$ nodes with top-k values of $\gamma(j)$
6: **return** S

Function calPartial(node n)
if n is not a leaf **then**
  benefit $\leftarrow 1$
  for each child i of n **do**
  benefit $\leftarrow$ benefit + calPartial$(i) \times p_{ni}$
  end for
else
  benefit $\leftarrow 1$
end if
return benefit
EndFunction

LEMMA 5.1. (CORRECTNESS OF DAVA-TREE) *If the m-graph $G(V, E)$ is a tree, then we can get optimal solution of the DAV problem under IC model by Algorithm 2.*

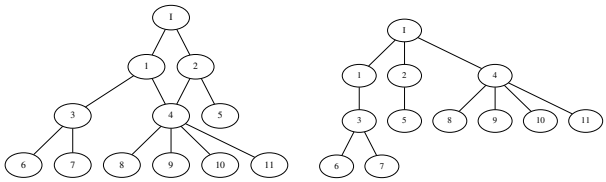
Proof. (Brief sketch) We show the following things: in the optimal set, the chosen nodes must be neighbors of the infected node I'; the benefit of each such node is independent of the rest of the set S; and finally that we correctly calculate $\gamma(j)$.

LEMMA 5.2. (RUNNING TIME OF DAVA-TREE) *Algorithm 2* DAVA-TREE *costs $O(|V| + |E| + klog|V|)$ time in the worst case.*

Proof. Omitted for brevity.

5.3 Arbitrary graphs under IC model What if the m-graph is not a tree? We give an effective heuristic next when m-graphs are arbitrary networks. After the merge algorithm, we can guarantee that a connected graph has only one infected node I'. Intuitively, we need to capture (a) the '*closeness*' of nodes to the infection (represented by I') and at the same time, (b) the importance of the nodes in '*saving*' other nodes. Thus good solutions are composed of nodes which are close to I' and also prevent the infection of many others.

We can still use DAVA-TREE algorithm by generating a tree from the m-graph, rooted at I'. Which tree should we use? People have typically used spanning trees like the Minimum Spanning Tree (MST) in related problems. The problem with MST is that potential solution nodes (for the original graph) can reside at higher depths in the MST—but as we saw in the previous section,



(a) Original Graph. (b) Dominator Tree.

Figure 2: An example of dominator tree. For p=1 and k=1, the optimal solution is node 4. For p=0.5 and k=1, the optimal solution is node 1.

the DAVA-TREE algorithm only chooses nodes which are neighbors of I'. See Figure 2(a) for an example. Let $p = 1$ on all edges, and budget $k = 1$. Then the MST rooted at I' will not have the node 4 as a neighbor of node I', but the optimal solution in this case will exactly be node 4. On the other hand, if $p = 0.5$ and $k = 1$, then it is easy to verify that node 1 will become the optimal solution. Hence, instead, we propose to use the *dominator tree* of the graph and then run DAVA-TREE algorithm on the dominator tree. As we explain later, the dominator tree avoids this problem, by precisely capturing the 'closeness' property required from the solutions.

In graph theory, given a source node I', a node v dominates another node u if every path from I' to u contains v. Node v is the immediate dominator of u, denoted by $v = idom(u)$, if v dominates u and every other dominator of u dominates v. We can build a dominator tree rooted at I' by adding an edge between the nodes u and v if $v = idom(u)$. Dominator trees have been used extensively in studying control-flow graphs. Building dominator trees is a very well-studied topic, with near-linear time algorithms available [4, 16]. Figure 2(b) shows the dominator tree for graph in Figure 2(a). Note that the edges in the dominator tree may not in fact exist in the original graph (compared to say the MST).

Note that we have not specified how to weight the edges of this dominator tree yet. Even the simple unweighted version of the dominator tree has structural properties which are very useful for the DAV problem. As we show in the next Lemma, the *optimal* solution for the original graph can only be a subset of the neighbors of I' in the *unweighted dominator* tree.

LEMMA 5.3. *For the DAV problem, the optimal solution should be the children of root I' in the unweighted dominator tree of the m-graph G.*

Proof. Omitted for brevity.

Note that by building the dominator tree we can reduce the search space substantially without losing any information—we demonstrate this in experiments as well, the number of neighbors of I' in the dominator tree is typically a fraction of the total number of nodes in the original graph. Further, we can prove that if $p = 1$, running DAVA-TREE on dominator tree of m-graph G returns the best *first* node.

LEMMA 5.4. *For the special case when the budget $k = 1$ and propagation probability $p = 1$, running algorithm DAVA-TREE on dominator tree T of m-graph G weighted with $p_{u,v}$ as above, gives the optimal solution.*

Proof. Omitted for brevity.

Weighting the dominator tree DAVA-TREE assumes that the edges in the network denote propagation probabilities. We want to preserve such information (coming from the original graph) in the dominator tree, to make the 'benefit' computation accurate. Hence we weight each edge $\{u, v\}$ in the dominator tree by $p_{u,v}$, the total probability that node u can infect v in the *original graph* (note that u and v may not be neighbors in the original graph).

Lemma 5.4 and the preceding discussion suggest a natural greedy heuristic: find the best single node i using DAVA-TREE on the dominator tree (weighted with $p_{u,v}$ as defined before); then remove node i from the graph; recompute the dominator tree; and repeat till budget k is exhausted. We call this algorithm BASIC.

Speeding up BASIC Unfortunately, computing the probability $p_{u,v}$ for a given pair of nodes u and v in the IC model is #P-complete [6]. It is essentially the canonical $s - t$ connectivity problem in random graphs. We can use Monte-Carlo sampling to get an estimate through simulations, but even that is too slow. Hence we propose to approximate it by using the *maximum propagation path probability* between nodes u and v, which is intuitively the most-likely path through which an infection can spread from node u to v in the original graph.

In the original graph, suppose $p_{pathi}(u, v)$ means the propagation probability from u to v through path i. We define *maximum propagation path probability* $\tilde{p}_{i,j}$ as the maximum value of $p_{pathi}(u, v)$. Here we can use $\tilde{p}_{u,v}$ as the approximate propagation probability for the edge $\{u, v\}$ in the dominator tree. Max. path probability has been used before in context of the influence maximization problem [6], but they need to compute it between all pairs of nodes. On the other hand, in our problem, we need to compute it only for edges in the dominator tree of the graph. In fact, further, it is easy to see that in a dominator tree rooted at I', if $v = idom(u)$, then $\tilde{p}_{v,u} = \frac{\tilde{p}_{I',u}}{\tilde{p}_{I',v}}$. This means we only need to calculate the maximum propagation path probability from root I' to

all other nodes, which is similar to find shortest-paths in graph theory.

Hence a faster algorithm than BASIC would be to assign probabilities $\tilde{p}_{v,u}$ in the dominator tree. We call the complete algorithm DAVA for arbitrary graphs. Pseudocode is given in Algorithm 3.

Algorithm 3 DAVA Algorithm for Arbitrary Networks

Require: Graph G, P, budget k, infected set I_0

$S = \emptyset$
$G' =$ Run MERGE on G and I_0
repeat
  $T =$ Build the dominator tree from G' and assign probabilities $\tilde{p}_{v,u}$
  $v =$ Run DAVA-TREE on T with budget $= 1$
  $S = S \cup v$
  Remove node v from G'
until $|S| = k$
return S

LEMMA 5.5. *(Running time of* DAVA*) Algorithm 3 takes $O(k(|E| + |V|log|V|))$ worst-case time.*

Proof. Omitted for brevity.

DAVA works fine on small graphs, but can be slow on large graphs, as it re-builds the dominator tree k times. Hence we propose an even faster heuristic DAVA-fast, which runs DAVA-TREE on the dominator tree with the full budget k, instead of running it with $k = 1$ after each step. Essentially we are picking neighbor nodes (in the dominator tree) of I' with the top-k $\gamma(\cdot)$ values. Pseudo-code is in Algorithm 4. DAVA-fast performs very well in our experiments, with minimal loss of quality at a fraction of the running time of DAVA.

Algorithm 4 DAVA-fast Algorithm

Require: Graph G, P, budget k, infected set I_0

$S = \emptyset$
$G' =$ Run MERGE on G and I_0
$T =$ Build the dominator tree from G' and assign probabilities $\tilde{p}_{v,u}$
$S =$ Run DAVA-TREE on T with budget $= k$
return S

LEMMA 5.6. *(Running time of* DAVA*-fast) Algorithm 4 takes $O(|E| + |V|log|V|)$ worst-case time.*

Proof. Omitted for brevity.

5.4 Extending to the SIR model Next we explain how to extend our solution to the SIR case. Recall that in the SIR model, as opposed to the IC model, a node u tries to infect its neighbor v multiple times. Suppose Z_u is the random variable denoting the number of time-steps u stays infected until recovery (this is also the number of time-steps that u tries to infect v). The probability that v gets infected by u is $B_{uv} = 1 - (1 - p_{u,v})^{Z_u}$. Note that Z_u has a geometric distribution $Pr(Z_u = z) = (1 - \delta)^{z-1}\delta$, with $E(Z_u) = \frac{1}{\delta}$ (δ is the curing probability). Thus if we force u to be infected for only one time-step before recovering (as in the IC model), then the equivalent probability that u infects v successfully can be approximated by $\beta_{u,v} = 1 - (1 - p_{u,v})^{\frac{1}{\delta}}$ (using Taylor series). Hence, we directly apply our algorithms to the SIR case, by just using an equivalent IC model with $\beta_{u,v}$ as the propagation probability.

6 Experiments

In this section, we give an experimental evaluation of our DAVA and DAVA-fast algorithms.

6.1 Experimental Setup We describe our setup next.

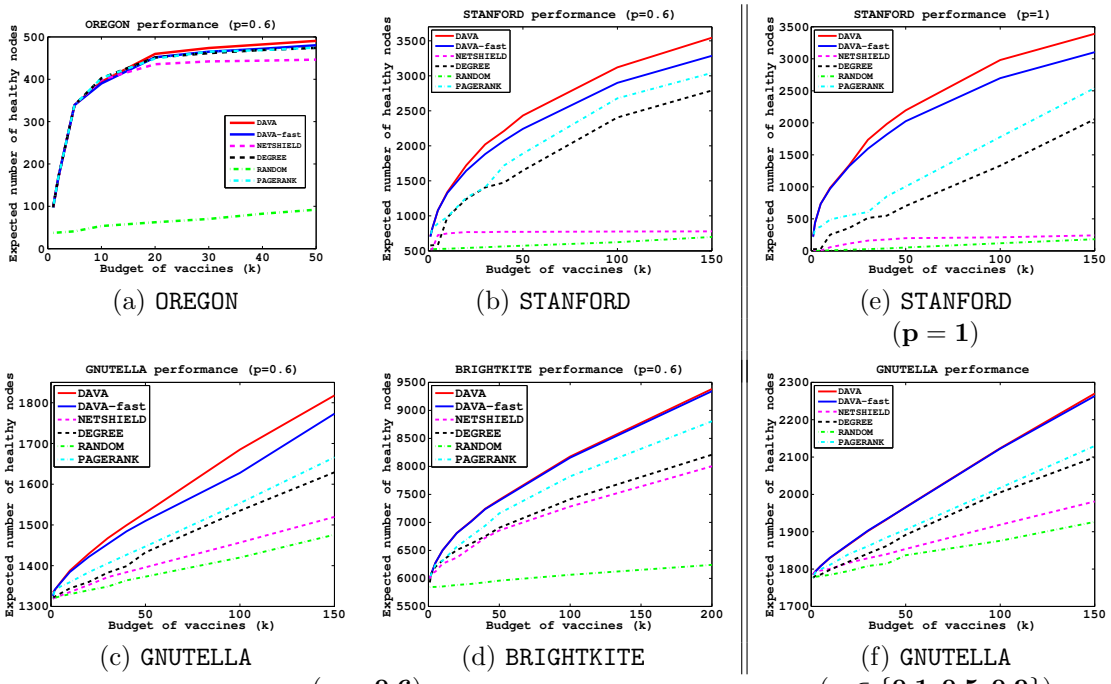
Datasets We run our experiments on multiple real datasets. In addition to trying to pick datasets of various sizes, we also chose them from different domains where the DAV problem is especially applicable. Table 2 shows datasets we used.

Table 2: Datasets

Dataset	Model	Description
OREGON	IC	A Oregon AS router graph containing 633 links among 2172 AS peers.
STANFORD	IC	A Stanford CS hyperlink network containing 8929 nodes and 53829 links.
GNUTELLA	IC	A peer-to-peer network containing 39994 links among 10876 peers.
BRIGHTKITE	IC	A friendship network which consists of 58228 nodes and 214078 edges.
PORTLAND	SIR	A large urban social-contact graph which has been used in national smallpox modeling studies [9]. It contains $0.5million$ people (nodes) and more than $1.6million$ edges (interactions) with contact time between people.
MIAMI	SIR	Another social-contact graph from [9]. It has about $0.6million$ people and $2.1million$ edges.

Settings For the IC model, we use three illustrative settings: (a) Uniform probability $p = 0.6$; (b) Uniform probability $p = 1$; and (c) $p_{u,v}$ uniformly randomly chosen from $\{0.1, 0.5, 0.9\}$ (following literature [6]). We randomly choose 100 nodes to be infected initially as the set I_0.

For the SIR model, since our socio-contact graphs have contact time between people [18], we will use normalized contact time as the attack probability $p_{u,v}$, and set a uniform recovery rate $\delta = 0.6$. As the graphs are larger, we randomly choose 300 nodes to be infected



(a) OREGON (b) STANFORD (e) STANFORD ($\mathbf{p = 1}$)

(c) GNUTELLA (d) BRIGHTKITE ($\mathbf{p = 0.6}$)

(f) GNUTELLA ($\mathbf{p \in \{0.1, 0.5, 0.9\}}$)

Figure 3: Effectiveness for DAVA: IC model on various Real Datasets. (a)(b)(c)(d): $p = 0.6$; (e) $p = 1$ (f) $p \in \{0.1, 0.5, 0.9\}$. Expected number of healthy nodes after distributing vaccines according to different algorithms (i.e. $\sigma'_{G,I'}(S)$) VS budget k. Higher is better. Note that our algorithms consistently outperform the other algorithms by up to 10 times in magnitude. Further DAVA-fast performs almost as well as DAVA, at a fraction of the cost.

initially as the set I_0.

To get the expected number of healthy nodes after immunization, we run the processes 1000 times and take the average.

Baseline Algorithms We compare our algorithms DAVA and DAVA-fast against various other competitors RANDOM, DEGREE, PAGERANK and NETSHIELD [22]. Note that NETSHIELD is a state-of-the-art pre-emptive immunization algorithm, which aims to minimize the epidemic threshold of the graph. We take the top-k healthy nodes according to each algorithm. All our experiments were conducted on a 4 Xeon E7-4850 CPU with 512GB of 1066Mhz main memory, and the algorithms were implemented in Python[2].

6.2 Experimental Results

We describe the results of our experiments next. In short, our results demonstrate DAVA and DAVA-fast get upto 10 *times* better solutions compared to the baselines (resulting in thousands of more healthy nodes). In addition, we also show the scalability of our algorithms on large datasets (DAVA-fast finishes within 20mins for our largest graphs). Finally, we show that DAVA-fast returns solutions which are very comparable to the ones returned by the more expensive DAVA.

6.2.1 Effectiveness for DAV

First of all, the number of the nodes in the first layer of the dominator trees of the graphs were a fraction (ranging from 0.3 to 0.7) of the total number of healthy nodes in the graph. As discussed before, because of Lemma 5.3, this reduces the solution space substantially.

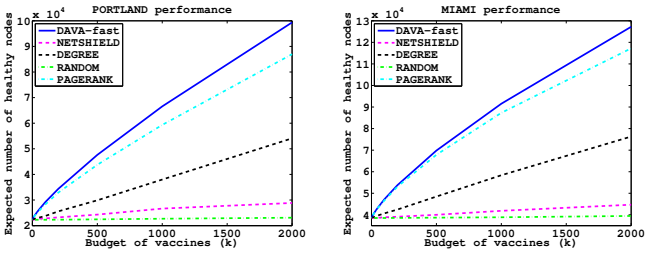
IC model Figure 3 shows our results for the IC model. In all networks, DAVA and DAVA-fast consistently outperform baseline algorithms.

As OREGON had only ~ 600 nodes, we varied k till 50 (roughly 9% of the graph). OREGON has a jelly-fish-type structure, hence for lower k, most algorithms work well by targeting the nodes in the core. But for larger k, the periphery needs to be targeted, and here our algorithms provide the best solution. For bigger networks like GNUTELLA and STANFORD with tens of thousands nodes, the difference in performance of DAVA and DAVA-fast from the other algorithms is clearer. PAGERANK and DEGREE perform well in STANFORD (a web-graph, with many hubs which these baselines exploit). Additionally NETSHIELD performed well only in BRIGHTKITE, demonstrating that the type of solutions change drastically, once we take

[2]Code can be downloaded from http://people.cs.vt.edu/~yaozhang/code/dava

the data of who is infected into account. In all these cases, DAVA and DAVA-fast performed the best.

Our faster heuristic DAVA-fast performed very well in all the networks, getting solutions almost as good as DAVA consistently. On STANFORD, DAVA-fast and DAVA both saved about upto 10 times more nodes than baseline algorithms, yet DAVA-fast took a fraction of the running time of DAVA (see Table 3). Also, the gains from our algorithms reduced as the p value decreased. This is expected as weaker the disease (in spreading), lower is the benefit of vaccinating i.e. lower is the savings gain from carefully selecting important nodes (as removing a node won't further change the expected infections much).



(a) PORTLAND (b) MIAMI

Figure 4: Effectiveness for DAVA: SIR model on Real Datasets. Expected number of healthy nodes after distributing vaccines according to different algorithms (i.e. $\sigma'_{G,I'}(S)$) VS budget k. Higher is better.

SIR model We got similar results under the SIR model on PORTLAND and MIAMI too: see Figure 4. Since PORTLAND and MIAMI contain about $0.5million$ nodes, DAVA was slow and could only allocate < 100 nodes in 1 day and hence we do not show it in the plots (NETSHIELD took 1.5 days to allocate 2000 vaccines while DAVA-fast finished in 20 mins). We notice that the larger k becomes, the better DAVA-fast performs than other algorithms: it saves more than $4,000$ nodes than the second best algorithm PAGERANK, and more than $20,000$ nodes than DEGREE (which is similar to the well-known acquaintance immunization method used in practice [7])!

6.2.2 Scalability Although our algorithms are polynomial-time, we show some running time results to demonstrate scalability. Table 3 shows the running time for DAVA, DAVA-fast and NETSHIELD when $k = 200$. DAVA-fast is much faster than DAVA—it took only 20 seconds to select 200 nodes in GNUTELLA while DAVA takes more than an hour to finish it. For the large-scale datasets like PORTLAND and MIAMI, DAVA-fast took less than 15 minutes to select 200 nodes, while DAVA could not finish in the alloted time. Further, DAVA-fast is up to 10 times faster than NETSHIELD—this is because NETSHIELD has a $O(nk^2)$ complexity (while our algorithms are linear in the budget).

Table 3: Running time(sec.) of NETSHIELD, DAVA and DAVA-fast when $k = 200$. Runs terminated when running time $t > 10$ hours. (shown by '-')

	DAVA-fast	NETSHIELD	DAVA
OREGON	**0.89**	4.9	23.3
STANFORD	**14.2**	74.1	2920.4
GNUTELLA	**20.1**	79.4	4700.5
BRIGHTKITE	**109.3**	246.8	19444.1
PORTLAND	**778.4**	8211.6	-
MIAMI	**1034.2**	11233.9	-

7 Related Work

We discuss the most closely related literature from immunization algorithms next.

Most existing work deals with identifying optimal strategies for vaccine allocation *before* the start of the epidemic. Briesemeister et al [3] and Madar et al [17] focus on immunization of power law graphs. Cohen et al [7] studied the popular *acquaintance* immunization policy (pick a random person, and immunize one of its neighbors at random—which roughly picks nodes according to their degrees), for both the SIS as well as the SIR model. Hayashi et al [12] introduce the SHIR model (Susceptible, Hidden, Infectious, Recovered) to model computers under e-mail virus attack on power-law networks. Kimura et al [15] studied the problem of blocking links in a network. Aspnes et al. [2] studied a game of inoculation given a cost and loss-risk, under *random* starting points. In a similar vein, Chen et al [5] gave a $O(\log n)$ approximation for minimizing 'societal' cost under a deterministic propagation model. Tong et al [21, 22] gave *pre-emptive* node-based and edge-based immunization algorithms for arbitrary graphs, based on minimizing the largest eigenvalue of the graph.

Many works also compare the performance of a limited number of pre-determined sequences of interventions (like school closure, antiviral for treatment) within simulation models [8, 10, 11]. Finally, the most related work is the recent paper by Yaesoubi and Cohen [25] which focuses on the optimal dynamic policies under a simplified flu-like model, and a *homogenous* population (i.e. every node is connected to every other node).

To summarize, none of the above works look into the problem of distributing vaccines under the more realistic setting of *prior information* and on *arbitrary networks*.

8 Conclusion

This paper addresses the problem of immunizing healthy nodes in presence of already infected nodes given a graph like a social/computer network or the blogsphere. The potential applications are broad: from distributing vaccine to control the epidemic, to stopping already present rumor in social media.

We formulated the problem called *Data-Aware Vaccination*, and proved it is NP-hard and also hard to approximate within an absolute error. After that we gave an optimal algorithm `DAVA`-tree for m-trees, and presented two polynomial-time heuristics `DAVA` and `DAVA`-fast for general graphs. We demonstrated the effectiveness and efficiency of our algorithms through extensive experiments on multiple datasets, including epidemiological social contact networks, computer networks and social media networks, on both IC and SIR models. `DAVA` and `DAVA`-fast outperform any other baseline algorithms by up to 10 times in magnitude. In addition we also presented the scalability of `DAVA`-fast on large-scale networks.

Acknowledgments: We would like to thank Virginia Bioinformatics Institute for giving us the PORTLAND and MIAMI datasets. This material is based upon work supported by the NSF under Grant No. IIS-1353346.

References

[1] R. M. Anderson and R. M. May. *Infectious Diseases of Humans*. Oxford University Press, 1991.

[2] J. Aspnes, K. Chang, and A. Yampolskiy. Inoculation strategies for victims of viruses and the sum-of-squares partition problem. *In SODA*, pages 43–52, 2005.

[3] L. Briesemeister, P. Lincoln, and P. Porras. Epidemic profiles and defense of scale-free networks. *WORM 2003*, Oct. 27 2003.

[4] A. L. Buchsbaum, H. Kaplan, A. Rogers, and J. R. Westbrook. A new, simpler linear-time dominators algorithm. *ACM Trans. Program. Lang. Syst.*, 20(6):1265–1296, Nov. 1998.

[5] P.-A. Chen, M. David, and D. Kempe. Better vaccination strategies for better people. In *Proceedings of the 11th ACM conference on Electronic commerce*, EC '10, pages 179–188, New York, NY, USA, 2010. ACM.

[6] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. *KDD*, 2010.

[7] R. Cohen, S. Havlin, and D. ben Avraham. Efficient immunization strategies for computer networks and populations. *Physical Review Letters*, 91(24), Dec. 2003.

[8] J. Dushoff, J. Plotkin, C. Viboud, L. Simonsen, M. Miller, M. Loeb, and D. Earn. Vaccinating to protect a vulnerable subpopulation. *PLoS Med*, 4(5):e174, 2007.

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

[10] N. Ferguson, D. Cummings, C. Fraser, J. Cajka, P. Cooley, and D. Burke. Strategies for mitigating an influenza pandemic. *Nature*, 442(7101):448–452, 2006.

[11] M. E. Halloran, N. M. Ferguson, S. Eubank, I. M. Longini, D. A. T. Cummings, B. Lewis, S. Xu, C. Fraser, A. Vullikanti, T. C. Germann, D. Wagener, R. Beckman, K. Kadau, C. Barrett, C. A. Macken, D. S. Burke, and P. Cooley. Strategies for mitigating an influenza pandemic. *Proceedings of the National Academy of Sciences*, 105(12):4639–4644, 2008.

[12] Y. Hayashi, M. Minoura, and J. Matsukubo. Recoverable prevalence in growing scale-free networks and the effective immunization. *arXiv:cond-mat/0305549 v2*, Aug. 6 2003.

[13] H. W. Hethcote. The mathematics of infectious diseases. *SIAM Review*, 42, 2000.

[14] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Conference of the ACM Special Interest Group on Knowledge Discovery and Data Mining*, New York, NY, 2003. ACM Press.

[15] M. Kimura, K. Saito, and H. Motoda. Minimizing the spread of contamination by blocking links in a network. In *Proceedings of the 23rd national conference on Artificial intelligence*, AAAI'08, pages 1175–1180. AAAI Press, 2008.

[16] T. Lengauer and R. E. Tarjan. A fast algorithm for finding dominators in a flowgraph. *ACM Trans. Program. Lang. Syst.*, 1(1):121–141, Jan. 1979.

[17] N. Madar, T. Kalisky, R. Cohen, D. ben Avraham, and S. Havlin. Immunization and epidemic dynamics in complex networks. *Eur. Phys. J. B*, 38(2):269–276, 2004.

[18] NDSSL. Synthetic Data Products for Societal Infrastructures and Protopopulations: Data Set 2.0. *NDSSL-TR-07-003*, 2007.

[19] D. Shah and T. Zaman. Rumors in a network: Who's the culprit? *IEEE Transactions on Information Theory*, 57(8):5163–5181, 2011.

[20] M.-Z. Shieh, S.-C. Tsai, and M.-C. Yang. On the inapproximability of maximum intersection problems. *Inf. Process. Lett.*, 112(19):723–727, Oct. 2012.

[21] H. Tong, B. A. Prakash, T. Eliassi-Rad, M. Faloutsos, and C. Faloutsos. Gelling, and melting, large graphs by edge manipulation. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, CIKM '12, pages 245–254, New York, NY, USA, 2012.

[22] H. Tong, B. A. Prakash, C. E. Tsourakakis, T. Eliassi-Rad, C. Faloutsos, and D. H. Chau. On the vulnerability of large graphs. In *ICDM*, 2010.

[23] S. A. Vinterbo. Privacy: A machine learning view. *IEEE Trans. on Knowl. and Data Eng.*, 16(8):939–948, Aug. 2004.

[24] E. C. Xavier. A note on a maximum k-subset intersection problem. *Inf. Process. Lett.*, 112(12):471–472, June 2012.

[25] R. Yaesoubi and C. Ted. Dynamic health policies for controlling the spread of emerging infections: Influenza as an example. *PLoS ONE*, 6(9):e24043, 2011.