Zhan *et al. J Big Data (2017) 4:2*
DOI 10.1186/s40537-016-0061-4

Journal of Big Data

RESEARCH Open Access

Vaccination allocation in large dynamic networks

Justin Zhan[1*†], Timothy Rafalski[1†], Gennady Stashkevich[2†] and Edward Verenich[2†]

*Correspondence:
justin.zhan@unlv.edu
†All the authors have equal contributions
[1] Department of Computer Science, University of Nevada, 4505 Maryland Parkway, Las Vegas, NV 89154, USA
Full list of author information is available at the end of the article

Abstract

Network infections that are already in progress cause challenges to those officers trying to preserve those nodes not yet infected. Static solutions can take advantage of global knowledge of the network to produce quick and approximate answers for those members who should be vaccinated. In dynamic situations however, small changes can severely alter those static solutions making them irrelevant. Yet in dynamic situations it can not be known with certainty which small changes will affect the solution and those that will not. Computational resources are wasted recalculating a global solution for the entire network, when a local recalculation may be enough. This paper presents a dynamic node vaccination solution that seeks to take advantage of these local recalculations.

Keywords: Dynamic networks, Immunization, Dominance tree

Background

Event propagation over a network is a complex and frequently studied human phenomena [1–3]. Historical examples of information exchange between humans in a social network, past and present, can be seen during colonial expansion of the British Empire [4], memes passed between friends on any assortment of social networks on the internet [1] and transmission of human or animal pathogens such as the virus H1N1 over airways [5]. In the information age computer networks can mirror these types of information exchange, with event information being passed along from node to node when network neighbors communicate through various network protocols [6].

This information can be useful, whether in the form of postal delivery of early colonial directives and parcels [4], neutral, shared irrelevant Facebook posts [7], or detrimental, such as human pathogens or computer attacks [5, 8]. In this paper we will use consistent terminology of a negative event for the purpose of visualization for the reader. Some examples may include: how to spread, as in infection, or how to prevent the spread, as in vaccination, of this negative event, or attack. Malicious attacks can take many forms, such as, malware, viruses, undesirable information, or even memes that might violate a networks posting policy [8, 9]. In terms of this paper "vaccination"? is defined as a set of nodes that can be inoculated, removed, and/or protected from these malicious attacks. The main assumption is that if these specific sets of nodes are given pre-emptive

measures then the attack will only be effective on a minimum number of nodes in the network.

The development and growth of networks has led to a diverse number of academic domains with shared properties and/or unique properties. Network traversal by means of graph theory [10], network communication protocols [6], static networks [11], dynamic networks [9] and network security [12] are network domains with similar terminology and visualizations. If a similarity exists across such diversity then it is possible to explore a similar solution that models important aspects borrowed from similar domains. This paper examines networks with nodes that are sparsely connected that may have a hierarchical structure that can be exploited using parts of graph theory to develop a solution.

Large networks can suffer disastrous results if an attack is allowed to migrate over a network. Various problems have been considered in the literature regarding this issue. Stopping a known attack at a source node before it can infect its neighbor nodes is a trivial problem. Once the attack spreads beyond the source node, however, its complexity rises as more nodes are exposed and infected through available edges [13]. Another trivial situation for an active infection network is if the amount of vaccine available is much larger than the number of non-infected adjacent nodes to any infected nodes. The more typical case occurs when non-infected nodes outnumber the amount of vaccine by a large degree [11]. This limitation causes a cost-benefit solution for which choices must be made in order to create an outcome that is favorable to the network administrators. Giving system administrators a variety of implementations allows them to choose the best option, or possible combinations of near optimal solutions, for their system [14].

Several solutions exist to prevent the spread of attacks over closed static systems; however these solutions have limited application to dynamic systems. An effective solution for arbitrary static networks with sparse connectivity is called Data-Aware Vaccine Allocation [11] or DAVA. DAVA is a framework that treats infected sub-graphs as super nodes and all its infected node neighbors as rooted bidirectional sub-trees that has a representational optimal solution in the form of a set of adjacent neighbors of the infected root super node. The framework determines this set by using a benefit system derived from a probability of propagation set, an infection set, and the main network graph. Thoroughly discussed by Zhang and Prakash the DAVA algorithm has been proven effective for "vaccine" distribution over an arbitrary network given prior information [11]. However this solution has limited application if new nodes are being added to the network over a consistent time interval.

Large networks that are dynamic have vulnerabilities because of the changes made to the network in real time [9]. Complications arise when new nodes vary in their importance to the networks due to location of edges and varying probability to infection [8]. For example a node could be added to the network that has edges to nodes that were not connected before or that create a cycle that was not present before the edge was added. This addition may or may not change the overall benefit of the sub-tree that contains the node due to probability changes or dominance changes caused by this addition. In a static system this addition or removal now would represent a new graph which requires the entire network to be examined from the start. This paper seeks to make the distinction that these new nodes will not necessarily alter the current benefit solution of its

associated sub-trees thus would not alter the optimal solution; it would require less work than recalculating the entire tree with new benefits and optimal solution. As a bonus, overall network information would be gained that could be applied to machine learning to enhance the information about this network. This gain might be exploited when examining local or global solutions.

Similarly to a multi-task learning problem [2], dynamic network node vaccination, or removal, has a main task, namely the optimal solution and then sub tasks to calculate and determine the overall benefit for top-k vaccination order subgroups. Individual nodes that have outlier tendencies might alter any current branch benefit, thus altering the top-k benefit order. It is from this concept that the idea of local solutions are of greater interest than global solutions for dynamic networks. If these local changes could be determined to have little or no changes to the benefits of the global sub-tree to which the local solution belongs, then it maybe possible to exploit this information to make an algorithm more efficient.

This paper presents a dynamic network algorithm, VAILDN, that is able to receive nodes sequentially from a network and then devise an optimal set of vaccination nodes to ensure maximum protection of the network. While performing effectively on static solutions DAVA group of algorithms slow down due to the fact that the global solution always must be determined, even though changes may occur only for local solutions. This paper will show that it is possible to adjust the optimal solution without examining the entire network which will save time over static algorithms. In addition this paper will show that VAILDN has the same level of accuracy as do DAVA type algorithms.

Dynamic solutions using dominator trees also present challenges that are not considered when using static solutions. These challenges include the addition or removal of edges to the graph that may redraw the dominator tree if a cycle is created or destroyed. In a dynamic environment it is possible that children may be detected before the parents in terms of the overall graph. This will not affect the global solution however, as edges are added, local connections and dominance are built that could be exploited for time saving when any parent does arrive.

In this paper first related works are discussed in "Preliminaries" section. The main algorithm is broken into three phases and presented in "Experiments" section. "Results" section describes the experiments in which this algorithm was compared to random baseline algorithms and vaccination methods for static networks. The final sections discuss the experimental results and the conclusions, including suggestions for future work.

Related work

This paper combines several resources into a cohesive solution that would be able to deal with active infections in a large dynamic network. Static solutions for networks having prior information were examined [15] and interesting solutions are present that could be expanded to dynamic networks by using a multi-task approach. Topics for consideration include, epidemiology, multi-task learning, data-aware vaccination allocation, dominance trees, network hierarchy, and large dynamic network optimization.

The topic of population vaccination has been well studied with a variety of models based on different approaches. Sometimes the global solution may not be obvious [16] or maybe difficult to obtain [17]. A localized approach could target specific groups

that are known to have higher propagation probabilities or more frequent contact with infected. A global static approach could be too costly due to number of nodes in the network [14]. A dynamic solution that examines edges could be beneficial in calculating the optimal solution [16].

In references to "Data-Aware Vaccination Allocation" [11] this paper demonstrates that the Data Aware Vaccination problem is NP-hard using a reduction from the MinKU set problem [18]. In order to demonstrate the variety of the problem and the general application of DAVA two common discrete-time models were used for virus infection, the susceptible-infected-removed (SIR) model and the immune response (IR) model. The SIR model [19] used for the general case, is based on chicken-pox-like infections, during which a time cycle consists of two phases, an independent infection probability and a recovery probability δ. If an infection is transferred from an infected node to a healthy node successfully (the probability of the infection is given by the weights of the edges) then that healthy node will be infected during next round. If recovery of an infected node occurs, then that node is "cured" and no longer will able to participate in the epidemic. The second model, the IR model [11], is a specific case of the SIR model during which the interaction between an infected and a healthy node happens only once, and the recovery probability, $\delta = 1$. The solution algorithms suggested by Zhang and Prakash the DAVA, DAVA-prune, and DAVA fast algorithms were demonstrated to perform better within static networks in which an infection already was in progress when compared with five different static network algorithms.

This paper presents the propagation probabilities based on population simulations. Due to the related natures of networks and human communities [20] it was worth exploring to simulate how human communities interact in order to assign non-random meaning to the propagation. A normalization of a population simulation yields sufficient data values to test the research. People can be considered nodes and their contact can be considered edges. Much like human populations contact among nodes is dynamic and should benefit from a dynamic solution. Contacts of certain individuals may have more weight than others thus the importance of calculating a local solution needs to be stressed. For example the elderly or infirm might be of concern only to their local group associations rather than to the network as a whole [17].

Network exploration and node importance are two fundamental concepts to this work. Research in the field of network exploration has resulted in many good algorithms and two popular methods include depth-first-search (dfs) [21] and breadth-first-search (bfs) [10]. Dfs can be used to maximize propagation results as well as to define subdominators and dominators [11] as a dominator tree is built. Dfs is an effective tool for network exploration when the entire network is known and is relatively sparse. Highly connected graphs will benefit more from bfs due to the exploration of individual levels before the depth of each sub-tree is explored. This distinction is important for this study because the assumption for a dominance tree is based around the fact that the nodes are sparsely connected. Dominator trees are used to discover the interconnectedness of nodes in a sparse network [21]. A dominator tree will yield sub-trees that show the relative connection between nodes, thus can yield maximum solutions based on those sub-trees [11]. If the structure of a dominator tree is correct [22] a new relational mapping of a network can be built quickly to determine which parent nodes singularly expose their children

to infection. A dominator tree shows that if there is a non-unique cycle between two nodes x and y then x and y will be the only common nodes to all paths. As nodes in a network become more and more connected, dominance will lose its significance as many nodes begin to share the same dominator. This is why the assumption was made in this study that the graphs had some hierarchical structure present and that they were sparsely connected.

The building of the dominator tree becomes more challenging in a dynamic setting. Shortest path algorithms were shown to be useful to determine dominance in linear time [23]. In a dynamic setting it is not simply the shortest path alone which determines dominance. Any edge that creates a cycle between nodes in a sub-tree will upset the dominance and create the need to recalculate dominance for any nodes who are siblings to any node in the cycle. Another consideration for dominance is when an edge is removed the original dominance must be restored to all nodes that were either in the shortest path to the dominator or those neighbors of the shortest path nodes.

A multi-task solution is necessary for large dynamic networks due to the number of sub-problems created by sequential node entry into the network. Collaborative Online Multi-task Learning (COML) [2], an important aspect considered in this study was written to deal with the discovery of individual distributions that may not be properly recognized by the main task. It is the dynamic nature of the sub-trees that are formed with sequential node entry into the network that can form these subtasks. COML demonstrates that subtasks can be learned effectively at the same time as the main task. Thus information about individuals and the group are maintained independent of each other. Predictions made from using these learned distributions could be useful in making classifications for unknown instances that enter the system.

One of the main libraries used for implementation was the networkX library. This package is a library of graph objects and a variety of associated functions beneficial to this study. It was possible to use the graph representations as a cornerstone for the building, analysis, and traversal of the network. This graph could then be dissected into its representative dominance tree in order to yield the final solution [24].

Preliminaries

Notation

For the infectionSetMerge algorithm

- graph represents a networkx graph object
- infectedSet is a list of nodes all who can propagate the infection
- InfectedSuper is the name for the node, I'_x, that will represent the node with all infected edges and will serve as the root of the dominance tree
- x is each member of the infected set
- y is a variable that represents the other vertex that shares an edge with x
- $p_{I_x,y}$ is the propagation probability of that edge
- newGraph is the resultant graph with all infected nodes merged into one super infected node

For the BuildTree algorithm

- graph is the newGraph from infectedSetMerge with the infected super node infectedSuper
- changedsub-trees is a list of sub-trees from the super node that have been altered by the addition or removal of an edge
- Tree and domTree are the dominance tree being built with the infected super node as root
- Tree.benefit is the associated benefit dictionary that is based off of domTree

For the solutionDecision algorithm

- k is the number of available vaccines
- The table of benefits is retrieved from the dominance tree.

Algorithm outline

This paper outlines three main phases used in the study to implement the process of node detection which would maximize the number of "saved" nodes if an attack already is in progress. These three phases which were incorporated into the final algorithm are:

*Phase*1 The InfectionSetMerge phase,during which all connected infected nodes are combined into one supernode that will server as the root in Phase 2.
*Phase*2 The Tree Building phase, during which the nodes are entered sequentially and the sub-trees are built, propagation probabilities are set, and sub-trees are marked if changed; and
*Phase*3 The Benefit and Solution Decision phase during which, if necessary, the benefits are recalculated and compared to the current solution set. If there is a difference all benefits are sorted and the solution set of the top-k benefits are chosen.

InfectionSetMerge

In many cases in which a negative event is occurring, many nodes might be present whose neighbors are infected. Edges between any number of these infected nodes in a path can be condensed into a single node. For purposes of this experiment once a node was infected it remained as such. The purpose of this phase was to condense the Infection Set into a single node, which then was used as the root for the master tree from which the optimal solution was derived. The algorithm for building this single node can be stated as follows: if there is a edge between two infected nodes then a union of those two nodes is performed. The infection propagation then must be updated to reflect the propagation of both nodes. Once this has been completed a single node is returned. As further research continues in this field, it might be possible to consider the possibility of multiple non-connected infection nodes, which would be put into a final Infection Set where each set would have its optimal solution. Algorithm 1 for InfectionSetMerge is expressed as follows:

Algorithm 1 InfectionSetMerge

Input: networkX graph **graph**, set of infected nodes **infectedSet**
Output: networkX graph with merged infected nodes into node **infectedSuper**

add I'_z to I'
for x in **infectedSet do**
  for each neighbor of x **do**
    if x is a neighbor of I'_x **then**
      $p_{I'_x,y} \leftarrow p_{I'_x,y} + (1 - p_{I'_x,y})p_{infectedSet_x,y}$
    else
      add the edge (I'_x, y)
      $p_{I'_x,y} \leftarrow p_{infectedSet_x,y}$
    end if
  end for
  remove node from **infectedSet**
end for
if exists an edge between **infectedSet** and itself **then**
  remove the edge
end if

Tree Building

The Tree Building phase has two main goals using a similar idea structure to a multitask learning model [8]. Nodes are entered sequentially and processed for each goal. In the initial stages this algorithm must calculate and store an entire collection of propagation probabilities. Then the dominance tree is built using relations between nodes. Due to the nature of dominance, while the graph contains the edges of the tree, new edges now are in the tree that represent dominance only; these new nodes do not actually exist in the original graph. This can complicate the real time solution due to the fact that edges that are added may create new edges in the tree thus altering dominance while edges that are removed may cause some of these dominance tree edges to represent an incorrect level of dominance that must be found and corrected. As the tree is being built any changed sub-tree from the infected super node was tracked in order to recalculate benefit for Phase 3. Algorithm 2 for BuildTree is as follows:

Algorithm 2 BuildTree

Input: networkX graph **graph**, node **infectedSuper**, list of edges **edgeList**
Output: dominance tree with corrected dictionary of benefits **domTree**, list **changedsub-tree**

for each edge from **edgeList do**
  determine if the edge was added or removed
  adjust dominance tree for edge
  mark relevant sub-trees as changed
  add sub-trees to **changedsub-tree**
end for
if tree exists and **changedsub-tree** is empty **then**
  calculate benefit for each sub-tree of **infectedSuper**
else
  change benefit of any sub-tree in **changedsub-tree**
end if

Solution Decision

The last step keeps track of all the current benefits of sub-trees as nodes are added and/or removed in real time. This means that the solution set can be achieved by sorting the benefits sub-trees that have root nodes that are the neighbors of the infected super node

and then returning the solution set. It is assumed that there always will be changes to the sub-trees as nodes are added and removed. This step must be performed each time but at a minimal cost. Algorithm 3 for SolutionDecision is as follows:

Algorithm 3 SolutionDecision

Input: table of benefits, number of vaccinations available **k**
Output: dictionary of top-k benefits **solution**
1: sort the table of benefits
2: solution = select top-k benefits
3: **return solution**

Vaccine allocation in large dynamic network

Given a set of sequentially input nodes, $n(x, y)$, where x is a vector of neighbors, and y is the probability of an infection, an active infection set I, and a budget k of vaccination allocation can be an output of the optimal vaccination solution set. This protects the maximum number of nodes that can be achieved only when changes to the top-k benefit sub-trees have been detected. The overall solution combines the three phases from the above subsections. The time saved in performing the benefit checks as new nodes are added to the network will make this program more efficient than running the optimal solution program by itself, especially as the network grows in size.

Algorithm 4 Vaccine Allocation in Large Dynamic Network

Input: network graph **N**, list of changed edges **edgeList**, list of infected nodes **infectionSet**, number of vaccinations available **k**
Output: dictionary with sorted top-k benefits **solution**
1: newGraph = *InfectionSetMerge*(**N**, **infectionSet**)
2: tree = *BuildTree*(newGraph,**edgeList**)
3: **solution** = *SolutionDecision*(**tree.benefits**, **k**)
4: **return solution**

Complexity analysis

The time complexity analysis for Phase 1 mainly is dependent on the number of nodes in the infections set. Each recursion from InfectionSetMerge Line 14 would be a subset of the main Infection Set thus the time complexity would be $O(|I_V| + |I_E|)$. BuildTree is based upon taking the graph, building the dominance tree representation of the graph and then calculating the dictionary of benefits for the individual sub-trees. This makes the worst time $O(|V| + |E|)$ with the additional constant costs of calculating the benefits. At this point, there is a significantly large difference between a static approach and a dynamic approach. Static algorithms must repeat this process for each benefit needed regardless if it is new or not. In the real time solution the average time is dependent on the number of sub-trees changed by the addition or removal of an edge. There are constant multipliers that are dependent on the number of sub-trees changed; however in sparsely connected graphs this number always should be much lower than the total of the sub-trees in the graph. Also another factor that may change the solution is how close the sub-trees are to the infected supernode. Changes to nodes close to the infected super node will require rebuilding large sections of the current dominance tree. Again,

the overall changes to the graph should not be close to the infected set (Note For the Gnutella set only 400 edges are adjacent to the infected super node out of 20,000 edges in total). The third phase is a linear sorting of the benefits dictionary. A rare but worst case for sorting for this would be O(n); however this is an extreme exception [25]. The average case is O(1).

Experiments

The goal of this experimentation was to answer the following questions:

- How does the algorithm developed in this study compare speed and accuracy with the baseline algorithms?
- How does the speed of this algorithm compare with DAVA Prune and DAVA fast?
- How does the accuracy of this algorithm compare to DAVA fast?

Settings

The baseline algorithms used were: k-random, k-random adjacent, and k-random tier 1. The algorithm k-random chooses the k-random nodes for vaccination. The adjacent k-random algorithm randomly chooses nodes that are neighbors of the infected super node using the graph as a basis. k-random tier 1 randomly chooses nodes that are subtrees of the infected super node. Due to dominance, in the study there have might have been edges between the infected supernode and its neighbors that were not present in the graph. In addition the DAVA, DAVA-fast, and DAVAprune algorithms were implemented for comparison. Due to the real time nature of the data the original code as presented by Li et al. [2] was not used for tests in this study. However the algorithms were maintained by means of an implementation that used their paper as a guide. If a dominance tree was needed for an algorithm, k-random tier 1 or DAVA-prune for example, the tree was built using the BuildTree algorithm from this paper in order to maintain consistency and reduce variability. The first set of tests used the IR model and a propagation of 1. Later tests used a variety of propagations, either 0.5 or 0.9, in order to simulate the propagation of negative information across the network.

The tests were ran at the Supercomputing Center, Cherry Creek, at University of Nevada Las Vegas (UNLV). The trials were run on a single node with two Intel Xeon E5-2697v2 12core, 128Gb ram [26].

Three main sources of data were used (Table 1). The first was a random driver that created a network of random size with a random number of infected nodes. The number of nodes that were adjacent to the infected node also were randomized. For each random run those adjacent to the infected super node were slightly different; however the results were consistent for all the trials. The second dataset was obtained from the Stanford Network Analysis Project (SNAP), which is the Gnutella peer to peer network from 2002 with 6301 nodes and 20,777 edges [27]. The third set was an undirected graph of emails, also from SNAP, and consisted of 36,692 nodes and 183,831 edges [27]. While the first data source was used to test the concept, the other networks provided real world solution sets with scalabilities that could be compared to other networks.

The trial driver used to test the algorithms attempted to simulate a real time network. The network was created, either by means of the random driver or by using a .csv file of

edges obtained from SNAP. Periods then were created to simulate changes in the network over time. For the purposes of this experiment a dynamic solution was presented in which changes might occur to the network as the solution was being run. Changes that occured that may affect the network included both adding and removing edges. The changes for each period were randomized for each trial but were kept consistent while the algorithms were run. Changes were made to the graph and then a copy of that graph was passed to each of the algorithms. The changes were chosen randomly using standard randomization functions (Python™). The percent of node change and variance for each period were fixed in order to demonstrate the concept and to reduce difference in performance. The time and infection results were averaged over 100 runs for each algorithm.

Results

Figure 1a shows the time comparison for the baseline algorithms, DAVA types, and VAILDN for the Email-Enron set, and Fig. 1b is the time comparison for the baseline algorithms, DAVA types, and VAILDN for the Gnutella set. The complete rebuilding of the dominance tree accumulated more and more costs as the number of vaccines were increased. This cost increase grew as the networks grew. The slope for the VAILDN algorithm demonstrates a slower growth as the network increased in size. Figure 1c represents the time comparison for the baseline algorithms, DAVA types, and VAILDN with the random set.

Figure 2a, the Gnutella set, and Fig. 2b, the random set, present a zoom in of the random algorithms and VAILDN. These figures demonstrate how VAILDN's cost saving measures are consistent when compared to simply randomly choosing k nodes for vaccination and choosing k random nodes that are adjacent to the infected super node. The overall results show that using the Enron set, the Gnutella set, and random set VAILDN performs faster than the DAVA types and with similar times to the k-random algorithms.

As shown in Fig. 3 even when the propagation is changed there is little change in the performance of the various algorithms. Because benefit calculation is similar for each of the algorithms not much change was expected. The benefit calculation is similar in all algorithms thus improvements were seen in which the sub-trees are recalculated if a change in the sub-tree was made.

Figure 4 shows the number of infected nodes that result once the k vaccinations have been delivered. Figure 4a is the Enron set, Fig. 4b is the Gnutella set, and Fig. 4c is the random set. In all cases VAILDN has similar numbers to the DAVA algorithms. All these outperform the random algorithms. There was a loss in the accuracy of VAILDN as the network got larger when compared to the DAVA algorithms but this loss decreases as

Table 1 Dataset information

Set name	Total nodes	Infected set	infectedSuper adj
Random	4000	300	525
Gnutella	6301	53	401
Email-Enron	36,692	200	1932

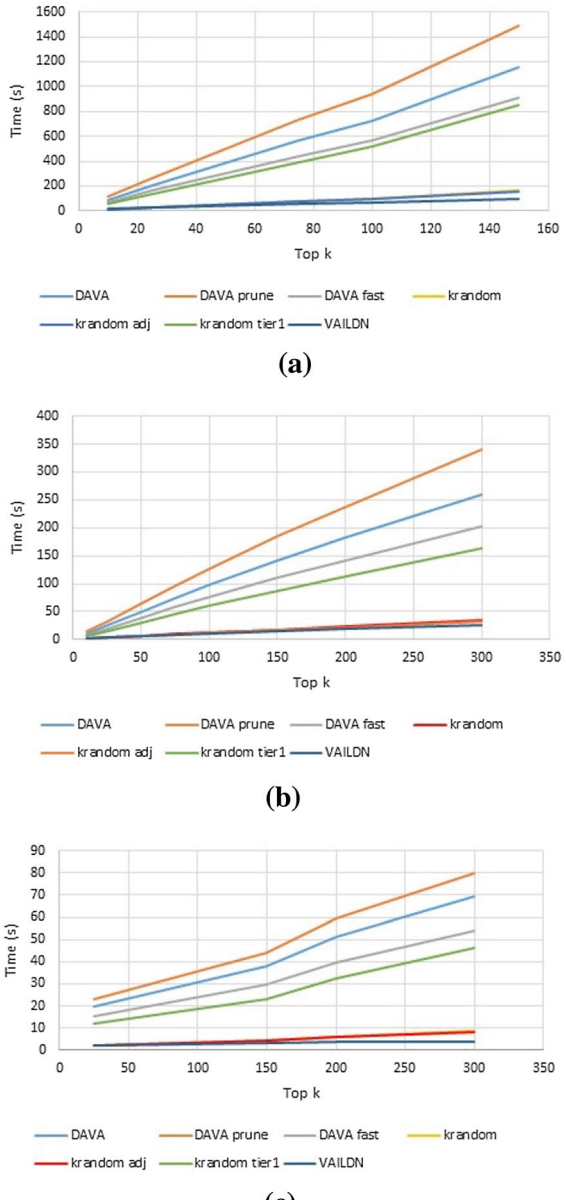

Fig. 1 Top-k vaccines and the average time in seconds per run. **a** Enron set. **b** Gnutella set. **c** Random set

the size of the network increases. In the Gnutella set the loss is about 2 and 0.04% in the Enron set. This groups still outperforms the random algorithms.

Discussion

These experiments demonstrate how a real time solution for a dynamic network can outperform a static solution. VAILDN is able to compute the solution as edges are examined in real time. This suggests that a local solution to the vaccination problem may be a better overall solution as the network grows in size or has a global solution that is difficult to calculate. Random vaccination is fast but it is not as effective as VAILDN.

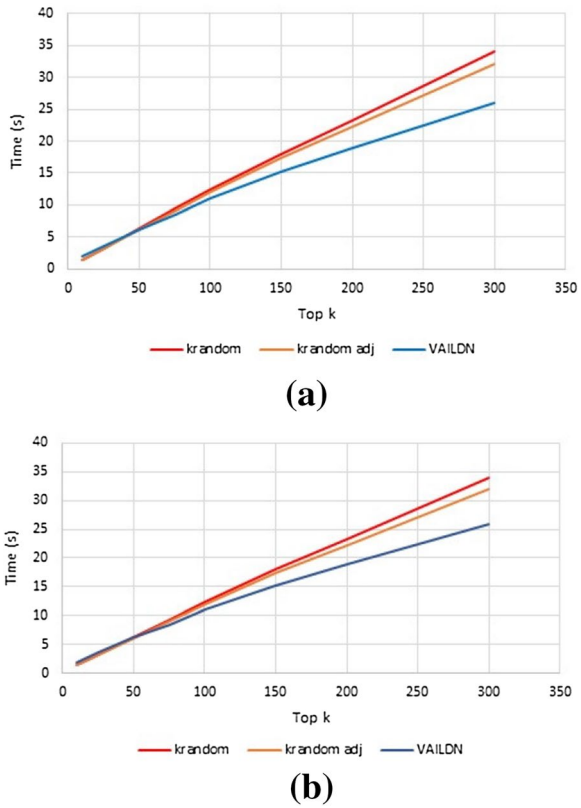

Fig. 2 Time comparison zoom of Gnutella and Random. **a** Gnutella set. **b** Random set

The main assumption for algorithms that use dominance as a basis is that either there are sparse connections or there is some overall hierarchical structure to the network. Network analysis in a static situation can reveal this type of information with many of the basic tools already established, such as depth first or breadth first searches. Dynamic networks, however present a complex field of parameters that can confound a single method of finding a solution. A dynamic network that is relatively stable might mimic a static environment; however edges can still be presented that could disrupt the assumptions of dominance or sparsity.

In VAILDN adding edges allowed the exploitation of the current dominance tree in order to establish new dominance of siblings and children. The removal of edges presented a specific challenge because after removing an edge the current tree structure gave little information about how to restore dominance. This is in contrast to adding edges where the dominance tree structure is already present. The method that was used relied on determining the siblings of each level for all the grandchildren of the common dominator. Removing edges was the most time consuming part of the algorithm, if it could be improved the efficiency will increase even more.

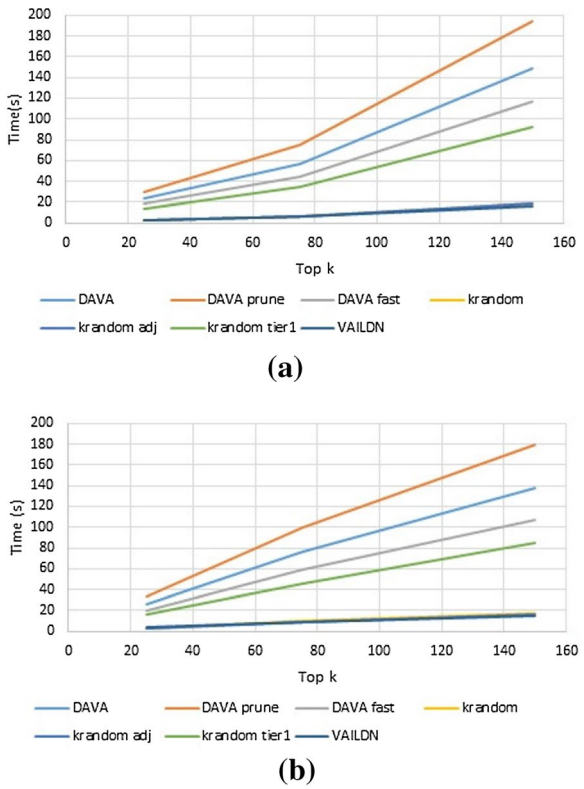

Fig. 3 Average runtime of Gnutella set at different propagations. **a** Gnutella set at p = 0.5. **b** Gnutella set at p = 0.9

In Fig. 5 VAILDN still compares similarly to the DAVA algorithms with regards to accuracy. The change in the weight calculation does little to affect the benefit solution. These results suggest that VAILDN is still beneficial with varying weight differences.

Conclusion and future work

A real time solution is able to deal with the reality of a modern and changing network dynamic. A solution can be determined quickly, as only the information local to the incoming, or removed, edge is considered. The accuracy of VAILDN is comparable to the static solution with diminishing losses. This means a network can make real time decisions about how the information is being distributed over the network. Dynamic networks offer difficult challenges with regard to issue detection. This algorithm is able to determine a real time solution with the assumption that the network is sparse.

Densely connected graphs create a low level dominance tree that makes finding the optimal solution challenging. To compensate for this a few future directions are open for exploration. This solution can be further refined by adding a factor that will calculate how much effect an edge propagation might have on its sub-tree solution. This will allow no change to the solution set if a small benefit is added, thus saving time. This could represent highly unlikely, yet present, connections between network members.

To continue this research, community clusters, rather than dominance trees, might be another structure for detecting the optimal solution. Even if a community is densely

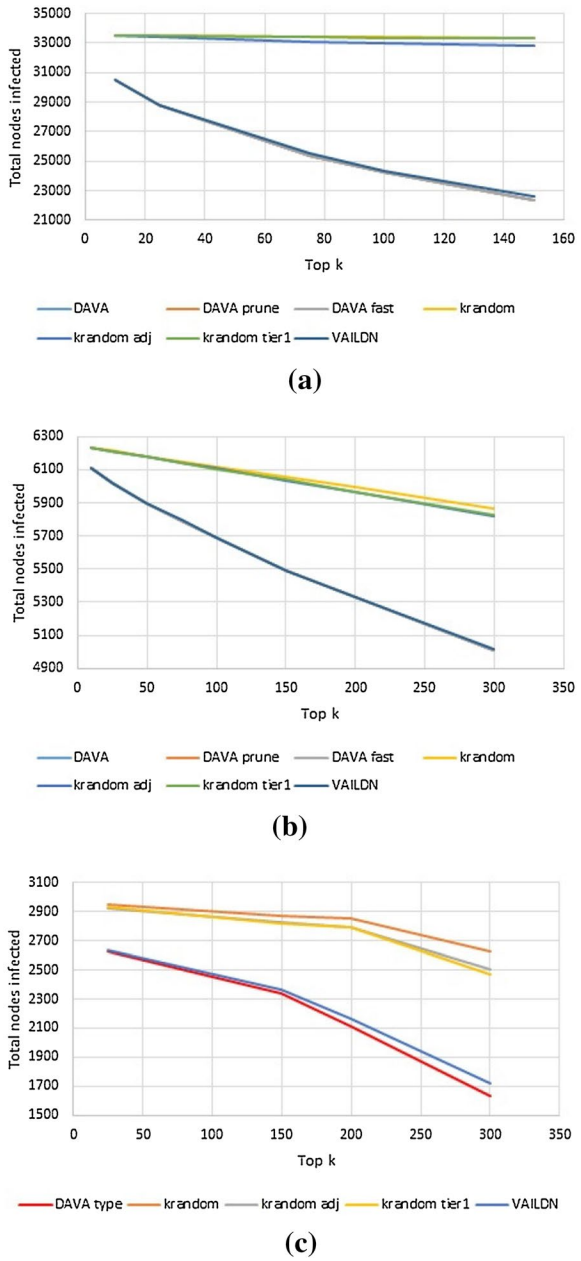

Fig. 4 Comparison of average infected nodes. **a** Enron set. **b** Gnutella set. **c** Random set

connected, the communities themselves maybe sparsely connected. This implies that with community detection, a local outbreak solution maybe able to be determined quickly offering a better solution than the global solution. A machine learning approach could be applied to determine which solution, dominance tree or community clustering, presents the most beneficial to the host network in that configuration.

This paper used terminology that indicated the information that was propagated was negative in nature (Table 2). This may not always be the case. The development of

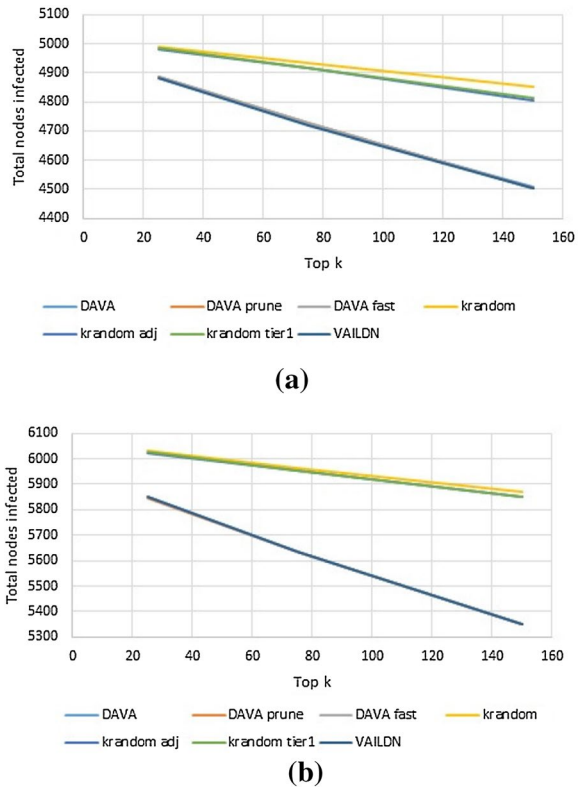

Fig. 5 Comparison of average infected nodes with changed propagation. **a** Gnutella set at p = 0.5. **b** Gnutella set at p = 0.9

Table 2 Acronym list

Acronym	Meaning
bfs	Breadth-first-search
COML	Collaborative Onlinie Multi-task Learning
DAVA	Data Aware Vaccine Allocation
dfs	Depth-first-search
IR	Immune Response (model)
NP-Hard	Non-deterministic Polynomial-time Hard
SIR	Susceptible-Infected-Removed (model)
SNAP	Stanford Network Analysis Project
UNLV	University of Nevada, Las Vegas

algorithms for information distribution with limited resources may be able to build on this algorithm as a real time solution for information distribution. The real time solution presented here is a heuristic solution that demonstrates the value of information propagation over a dynamic network.

Author details
[1] Department of Computer Science, University of Nevada, 4505 Maryland Parkway, Las Vegas, NV 89154, USA. [2] Air Force Research Laboratory, 26 Electronics Parkway, Rome, NY 13341, USA.

Acknowledgements
We gratefully acknowledge the United States Department of Defense (W911NF-14-1-0119, W911NF-13-1-0130, W911NF-16-1-0416), the National Science Foundation (1560625, 1137443, 1247663, 1238767), United Healthcare Foundation (#1592), Oak Ridge National Laboratory (#4000144962), and National Consortium for Data Science for their support. Through these financial support, the following articles have been published [28–42].

Authors' contributions
JZ, TR, GS, and EV developed the concept for the manuscript. JZ and TR drafted the manuscript. JZ, GS, and EV revised the manuscript. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Received: 26 October 2016 Accepted: 9 December 2016
Published online: 13 January 2017

References

1. Leskovec J, Backstrom L, Kleinberg J. Meme-tracking and the dynamics of the news cycle. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '09. New York: ACM; 2009. p. 497–506.
2. Li G, Hoi SCH, Chang K, Liu W, Jain R. Collaborative online multitask learning. IEEE Trans Knowl Data Eng. 2014;26(8):1866–76.
3. Wu Y, Deng S, Huang H. Information propagation through opportunistic communication in mobile social networks. Mob Netw Appl. 2012;17(6):773–81.
4. Nkwi WG, de Bruijn M. "Human Telephone Lines": flag post mail relay runners in british southern cameroon (1916–1955) and the establishment of a modern communications network. Int Rev Soc Hist; 59:211–35, 12 2014. Copyright-Copyright © Internationaal Instituut voor Sociale Geschiedenis 2014; Last updated—2015-07-25; SubjectsTermNotLitGenreText-Cameroon; United Kingdom–UK.
5. Hwang GM, Mahoney PJ, James JH, Lin GC, Berro AD, Keybl MA, Goedecke DM, Mathieu JJ, Wilson T. A model-based tool to predict the propagation of infectious disease via airports. Travel Med Infect Dis. 2012;10(1):32–42.
6. Memon I. A secure and efficient communication scheme with authenticated key establishment protocol for road networks. Wirel Pers Commun. 2015;85(3):1167–91.
7. Pempek TA, Yermolayeva YA, Calvert SL. College students' social networking experiences on facebook. J Appl Dev Psychol. 2009;30(3):227–38.
8. Cai F, Qingfeng H, LanSheng H, Li S, Xiao-Yang L. Virus propagation power of the dynamic network. EURASIP J Wirel Commun Netw. 2013;2013(1):1–14.
9. Wang J, Liu Y, Deng K. Modelling and simulating worm propagation in static and dynamic traffic. IET Intell Transp Syst. 2014;8(2):155–63.
10. Merrill D, Garland M, Grimshaw A. High-performance and scalable gpu graph traversal. ACM Trans Parallel Comput. 2015;1(2):14:1–30.
11. Zhang Y, Prakash BA. Data-aware vaccine allocation over large networks. ACM Trans Knowl Discov Data. 2015;10(2):20:1–32.
12. Battiti R, Cigno RL, Sabel M, Orava F, Pehrson B. Wireless LANs: from warchalking to open access networks. Mob Netw Appl. 2005;10(3):275–87.
13. Emmert-Streib F, Dehmer M. Exploring statistical and population aspects of network complexity. PLoS ONE. 2012;7(5):1–17.
14. Dewri R, Ray I, Poolsappasit N, Whitley D. Optimal security hardening on attack tree models of networks: a cost-benefit analysis. Int J Inform Secur. 2012;11(3):167–88.
15. Yaesoubi R, Cohen T. Dynamic health policies for controlling the spread of emerging infections: influenza as an example. PLoS ONE. 2011;6(9):1–11.
16. Cohen R, Havlin S, ben Avraham D. Efficient immunization strategies for computer networks and populations. Phys Rev Lett. 2003;91:247901.
17. Dushoff J, Plotkin JB, Viboud C, Simonsen L, Miller M, Loeb M, et al. Vaccinating to protect a vulnerable subpopulation. PLoS Med. 2007;4:05.
18. Vinterbo SA. Privacy: a machine learning view. IEEE Trans Knowl Data Eng. 2004;16(8):939–48.
19. Hethcote HW. The mathematics of infectious diseases. SIAM Rev. 2000;42:599.
20. NDSSL. Synthetic data products for societal infrastructures and protopopulations: Data set 2.0. ndssl-tr-07-003. 2007. http://ndssl.vbi.vt.edu/Publications/ndssl-tr-07-003.pdf.
21. Lengauer T, Tarjan RE. A fast algorithm for finding dominators in a flowgraph. ACM Trans Program Lang Syst. 1979;1(1):121–41.
22. Georgiadis L, Tarjan RE. Dominator tree certification and divergent spanning trees. ACM Trans Algorithms. 2015;12(1):11–42.
23. Buchsbaum AL, Kaplan H, Rogers A, Westbrook JR. Corrigendum: a new, simpler linear-time dominators algorithm. ACM Trans Program Lang Syst. 2005;27(3):383–7.
24. Aric A, Hagberg, Daniel A. Schult, Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In: Proceedings of the 7th Python in Science Conference (SciPy2008). Pasadena; 2008. p. 11–5.
25. python. Time complexity. 2015. https://wiki.python.org/moin/TimeComplexity. Accessed 1 Aug 2016.
26. NSCEE. How to run jobs with pbs/pro on cherry-creek. 2016. http://wiki.nscee.edu. Accessed 1 Aug 2016.

27. Leskovec J, Krevl A. SNAP Datasets: Stanford large network dataset collection. 2014. http://snap.stanford.edu/data.
28. Chopade P, Zhan J (2016) Towards a framework for community detection in large networks using game-theoretic modeling. IEEE Trans Big Data (accepted)
29. Pirouz M, Zhan J, Tayeb S (2016) An optimized approach for community detection and ranking. J Big Data 3:22
30. Gan W, Lin C-W, Chao H-C, Zhan J (2016) Data mining in distributed environment: a survey. WIREs Data Min Knowl Disc (accepted)
31. Wu JM-T, Zhan J, Lin JC-W (2017) An ACO-based approach to mine high-utility itemsets. Knowl-Based Syst 116:102–113
32. Lin JC-W, Yang L, Fournier-Viger F, Wu M-T, Hong T-P, Wang LS-L, Zhan J (2016) Mining high-utility itemsets based on particle swarm optimization. J Eng Appl Artif Intell 55(C):320–330
33. Lin JC-W, Wu T-Y, Fournier-Viger P, Lin G, Zhan J, Voznak M (2016) Fast algorithms for hiding sensitive high-utility itemsets in privacy-preserving utility mining. J Eng Appl Artif Intell 55(C):279–284
34. Lin JC-W, Li T, Fournier-Viger P, Hong T-P, Wu JM-T, Zhan J (2016) Efficient mining of multiple fuzzy frequent itemsets. Int J Fuzzy Syst 18:1-9
35. Lin JC-W, Li T, Fournier-Viger P, Hong T-P, Zhan J, Voznak M (2016) An efficient algorithm to mine high average-utility itemsets. Adv Eng Inform 30(2):233–243
36. Lin JC-W, Liu Q, Fournier-Viger P, Hong T-P, Voznak M, Zhan J (2016) A sanitization approach for hiding sensitive itemsets based on particle swarm optimization. J Eng Appl Artif Intell 53(C):1–18
37. Zhan J, Gudibande V, Parsa SPK (2016) Idenfication of Top-K influential communities in large networks. J Big Data 3:16
38. Pirouz M, Zhan J (2016) Optimized relativity search: node reduction in personalized page rank estimation for large graphs. J Big Data 3:12
39. Selim H, Zhan J (2016) Towards shortest path identification on large networks. J Big Data 3:10
40. Chopade P, Zhan J (2016) Structural and functional analytics for community detection in large-scale complex networks. J Big Data 2(1):1
41. Fang X, Zhan J (2016) Sentiment analysis using product review data. J Big Data 2(1):1
42. Zhan J, Fang X (2014) A computational framework for detecting malicious actors in communities. Int J Inf Priv Secur Integr 2(1):1–20