Positive Influence Dominating Set Generation in Social Networks

Akshaye Dhawan, Matthew Rink

Department of Mathematics and Computer Science
Ursinus College
Collegeville, PA
adhawan@ursinus.edu, marink@ursinus.edu

Abstract

Current algorithms in the Positive Influence Dominating Set (PIDS) problem domain are focused on a specific type of PIDS, the Total Positive Influence Dominating Set (TPIDS). We have developed an algorithm specifically targeted towards the non-total type of PIDS. In addition to our new algorithm, we adapted two existing TPIDS algorithms to generate PIDS. We ran simulations for all three algorithms, and our new algorithm consistently generates smaller PIDS than both existing algorithms, with our algorithm generating PIDS approximately 5% smaller than the better of the two existing algorithms.

***Index Terms*—Greedy Algorithms; Social Networks; Dominating Sets;**

I. Introduction

Social networks have grown immensely in size and popularity in the last decade. The explosion of these networks has led to a great deal of research pursuing algorithms [1] that provide insight into the structure and properties of these networks. Social networks are often represented as mathematical graphs with users of the network being modeled as nodes in a graph and the relationships between users being modeled as edges between nodes.

An important algorithmic problem in these network is that of modeling the spread of information within a group. Social science research has shown that individuals are strongly influenced by the ideas of those around them. Various studies in the context of drinking [2] and smoking [3] have shown that the likelihood of an individual to participate in these activities increases as their peers engage in these activities.

Within the context of social networks, recent work [4] [5] [6] has looked at modeling the spread of positive behavior in a social network. For example, a campaign to eat healthier or to not binge drink can only gain traction if a key number of users propagate this message to others using their influence. This problem has been defined as the *Positive Influence Dominating Set* problem (PIDS).

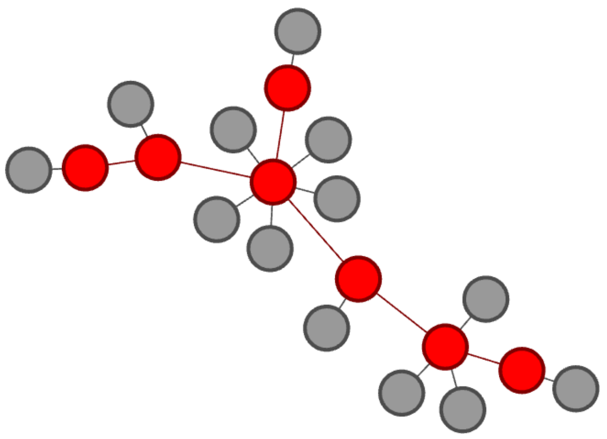
PIDS vs. TPIDS A Positive Influence Dominating Set (PIDS) is a subset P of a graph $G = (V, E)$ such that for each node $u \in V$, if less than some fraction of the neighbors of u are not in P, then $u \in P$. In the literature on social networks this fraction is fixed at 0.5 [4].

Figure 1a shows an example of a PIDS on a graph. The nodes in the set P are shown in red. As can be seen from the figure, every gray node has at least half its neighbors in the set P.

A Total Positive Influence Dominating Set (TPID) is similar to a PIDS, with a single alteration. While in a PIDS P, the nodes in P are exempt from the requirement of needing half their neighbors to be in P, in a TPIDS this is not true. Therefore, in a TPIDS T, *every* node in V has half of its neighbors in T, including the nodes in T itself. Note the difference between Figure 1a and Figure 1b.

The difference in definition between TPIDS and PIDS is simple, but it is sufficiently different that algorithms that are suited to creating one may not be suited for creating the other. In this paper we tackle the problem of computing a PIDS.

The remainder of this paper is as follows. Section II discusses the literature on Positive Influence Dominating Sets in social networks. In Section III we present our algorithm for computing a PIDS that we call AltGreedy. In Section IV we present simulation results on PIDS generation that compare our work to algorithms in the literature. Finally, we conclude and discuss future work in Section V.


(a) PIDS on the graph

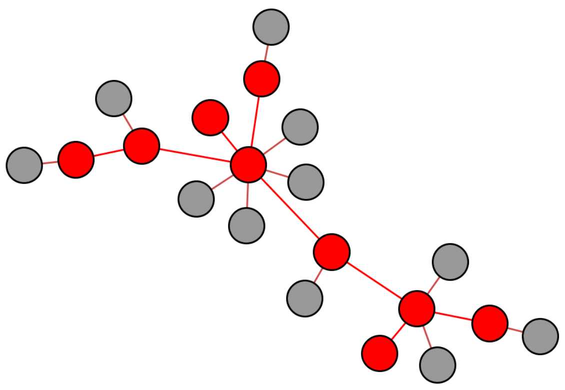

(b) TPIDS on the graph

Fig. 1: Example of PIDS and TPIDS on a power-law graph

II. Related Work

While the topic of dominating sets has several years of research behind it in particular for backbone formation in Wireless Sensor Networks [7] [8] [9], the problem of Positive Influence Dominating Sets (PIDS) in social networks has been studied relatively recently. The problem was introduced in [4] within the context of looking for a solution to social issues. An example the paper cites is the issue of binge drinking on college campuses. Intervention programs may not have the budget to advertise to or take in all binge drinkers. It is then desirable to select a subset of the binge drinkers such that if a binge drinker is not directly participating in an intervention program, at least half of their binge drinking friends are. For the problem application of binge drinking intervention, Wang et. al. [4] categorized nodes as positive, neutral, or negative. They collapsed neutral and negative into a single negative category, and then used the two remaining categories (positive and negative) as a starting point for their PIDS computation. Nodes that are initially positive (in the notation of the paper, the nodes in C) are able to influence the other nodes they are connected to. The subset P that is output by the algorithm proposed in [4] is not a PIDS by itself, but $P \cup C$ will be the complete PIDS of the graph. The strategy to select a PIDS used by Wang et al. in [4] is to repeatedly select a 1-dominating set of the nodes $V - P \cup C$ that have yet to be dominated, and then dominate the nodes in $P \cup C$ by selecting the highest degree neighbors of those nodes. The notion of the initial positive compartment C is one that is generally not seen beyond this paper. Wang et al. do mention that a graph with $C = \emptyset$ is one where every node is initially negative, which is the assumption most PIDS research has used. It is important to note that in this paper ([4]), the definition provided for PIDS is what is later called a TPIDS by [6], and so for the purposes of the current work we use the definitions in [6], which is what the definition for PIDS and TPIDS presented in Section I follow.

In [10], a followup paper to [4], the authors present a greedy algorithm to generate a TPIDS. Their algorithm is based on a selection function $f()$ defined as:
$$f(P) = \sum_{u \in V} min(h(u), n_P(u))$$

The strategy for this algorithm, henceforth referred to as WangGreedy [10] is to greedily choose the node that maximizes their evaluation function f. This function sums the number of neighbors that each node has in the TPIDS-in-progress P (given by the function $n_P(u)$), with the stipulation that each node can only contribute up to half its neighbors to the sum. The function $h(u)$ computes half the neighbors of u.

We now take a look at how WangGreedy's evaluation function f behaves when generating TPIDS in comparison to generating PIDS. When generating a TPIDS, the node that maximizes f is simply the node that touches the most unsatisfied nodes. To see this, consider first that the TPIDS-in-progress P is empty initially. Then the node that maximizes f will be the one with the highest degree, since at first f is 0 because no nodes have any neighbors in P and therefore all nodes are unsatisfied. Now consider the case where P already contains some nodes. Then the node u that maximizes f will also be the one that touches the most unsatisfied nodes, since nodes that are already satisfied by P cannot contribute any more to the sum in f because of the min limitation, therefore the only way f will increase is if u touches unmet nodes.

We adapt WangGreedy to generate PIDS by altering its evaluation function:

$$f(P) = \sum_{u \in V-P} min(h(u), n_P(u))$$

We make these changes because it accurately represents the notion that in comparison to a TPIDS, a PIDS does *not* require the nodes in P to also have half their neighbors in P. However, when adapted to the requirements for PIDS, WangGreedy no longer chooses the nodes that touch the most unmet nodes. This is because the evaluation function f no longer considers nodes in P.

The authors of [11] take a very different greedy approach to generating a TPIDS. Their approach is to value a node u based on how needy u's neighbors are. This has the effect of valuing nodes that might not necessarily connect to a large number of unsatisfied nodes, but perhaps instead connect to a few very needy nodes. Need is tracked by checking if a node has at least half its neighbors in P or if $n_P(u) < \left\lceil \frac{deg(u)}{2} \right\rceil$. We refer to this algorithm as RaeiGreedy in our comparison.

III. AltGreedy: A Greedy Algorithm for Positive Influence Dominating Set Generation

In this section, we present our algorithm for constructing a Positive Influence Dominating Set (PIDS) which we call AltGreedy. Since both WangGreedy [10] and RaeiGreedy [11] were designed with Total Positive Influence Dominating Sets (TPIDS) in mind, we seek to develop an algorithm specifically targeted towards generating a PIDS efficiently. This is because the PIDS problem is a more accurate representation of spreading a positive message since nodes participating in spreading the message are already considered to be *influenced* and thus do not need half their neighbors to repeat the message back to them. This is also more closely in line with the traditional definition of a dominating set whereby nodes in the set do not need to be dominated themselves.

A. Notation

Here we define the notation necessary for our algorithm. Let $G = (V, E)$ be a graph G with nodes V and edges E. The nodes V typically represent users on the social network and the edges E represent relationships. For the purpose of this research the graph G is undirected. Let P be our PIDS on G. For some node $u \in V$, we define the following notation.

Notation	Definition
$d(u)$	degree of node u
$h(u)$	$\left\lceil \frac{d(u)}{2} \right\rceil$
$n(u)$	the set of the neighbors of u
$n_P(u)$	$n(u) \cap P$

B. Algorithm: AltGreedy

We first define the following functions $s()$ and $g()$ on a node u :

$$s(u) = \begin{cases} 1 & \text{if } n_P(u) \geq h(u) \text{ or } u \in P \\ 0 & \text{otherwise} \end{cases}$$

$$g(u) = s(u) + \sum_{w \in n(u)} 1 - s(w)$$

Our first function, $s(u)$, represents whether a node is satisfied under the definition of a PIDS. A node is said to be satisfied when at least half its neighbors are in P.

Our second function, $g(u)$, tallies the number of unsatisfied neighbors a given node u touches. To this tally, we add 1 if u itself is unsatisfied. We do this because as noted in the definition of PIDS, a node can be satisfied either by having half of its neighbors in our PIDS P, or by being in P itself. Thus we increment the tally by 1 to indicate that selecting this node comes with the added benefit of satisfying itself instantly.

Our algorithm which we call AltGreedy can be written as follows.

```
1: $P \leftarrow \emptyset$
2: compute $g(u)$ value for all $u \in V$
3: **while** $P$ is not a PIDS **do**
4:     select $u \in V - P$ to maximize $g(u)$
5:     and set $P \leftarrow P \cup \{u\}$
6:     revise $g(w)$ values for all $w \in V - P$
7: **end while**
8: **return** $P$
```

Our algorithm's strategy for generating a PIDS is to be greedy and select the node that is connected to the most unsatisfied nodes, and add that node to the PIDS under-construction represented by the set P. To accomplish this, after creating our empty set P that will become our PIDS at termination, we compute g values for all the nodes in our graph V. Our loop simply chooses the node with the maximum g value, adds that to P, and recomputes the g values for the remaining nodes that are not in P (denoted in Line 6 by the nodes in $V - P$). This is very similar in flow to the algorithm presented by Raei et al. [11], but we are substituting RaeiGreedy's cover-degree criterion with our g function as the basis of selecting nodes to add in V.

We chose to use the strategy of selecting the node that is connected to the most unsatisfied nodes, taking into account whether the node in question is unsatisfied or not, because we are specifically targeting PIDS generation with our algorithm. Both algorithms WangGreedy [10] and RaeiGreedy [11] are aiming to construct a TPIDS. Section IV examines the performance of those algorithms when compared to AltGreedy via several experiments. For fairness we test the algorithms to generate both a PIDS and a TPIDS.

Complexity Analysis: Since the AltGreedy algorithm is structured much like that of RaeiGreedy [11], we are able to reference their time complexity proof to show that our algorithm is $O(n^2)$.

The loop in line 3 will run at most n times (where n is the number of nodes in the graph), since we can only add as many nodes to our PIDS P as we have altogether, and each iteration of the loop adds a node from V to P.

The first thing we do in the loop is to check if P is a PIDS or not, and only continue if the latter is true (this also takes place on line 3). In the worst case, we need to evaluate every node to check if it satisfies the PIDS conditions, so the complexity of this step is $O(n)$.

Lines 4 and 6 of our algorithm both take $O(n)$, since we will have to choose from and revise first n nodes, then $n-1$ nodes, etc. Line 5 takes constant time ($O(1)$).

Therefore, our algorithm's time complexity is $O(n^2)$.

IV. Simulation Results

In order to evaluate performance, all three algorithms - WangGreedy, RaeiGreedy and AltGreedy were tested by having them generate both a Positive Influence Dominating Set (PIDS) and a Total Positive Influence Dominating Set (TPIDS) on random scale-free graphs. The simulations were designed to allow us to compare our algorithms with those in [11]. Both WangGreedy and RaeiGreedy were modified in the PIDS experiments to stop when a PIDS has been achieved.

A. SNAP Library

We used the Stanford Network Analysis Platform (SNAP) [12] to implement and perform our simulations. The $C++$ SNAP library allowed us to easily generate scale-free networks using their implementation of the Barabasi-Albert model [13]. Using SNAP allowed us to quickly implement the algorithms in question and scale to larger graphs easily. Scale-free networks are a type of graph that follow a power-law degree distribution. This means that there are few nodes with relatively high degree (which can be viewed as *hubs*), and many nodes with low degree. [13] and [1] showed that social networks tend to form scale-free networks. Thus for our experimental work, we model social networks by randomly generating graphs using the Barabasi-Albert model of preferential attachment.

B. Simulations

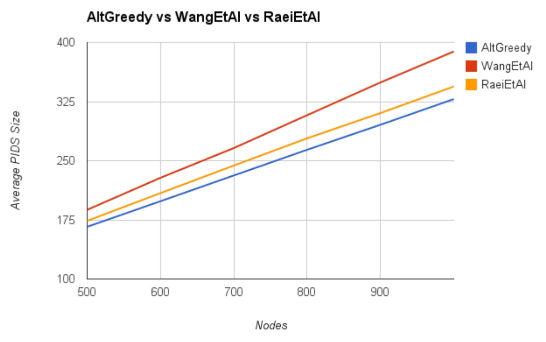
Using the SNAP library to generate graphs with power-law degree distribution (via the Barabasi-Albert algorithm), we ran several simulations to compare the PIDS and TPIDS generated by the three algorithms - WangGreedy, RaeiGreedy, and AltGreedy.

1) PIDS Generation: PIDS were generated on graphs of node sizes 500-1000 (increments of 100), all with an average degree of 10. A total of 1000 random graphs were generated using the Barabasi-Albert model for each data point in this experiment. The average size of the PIDS generated by the algorithms during this simulation are displayed in Figure 2. As can be seen from the figure , AltGreedy out performs WangGreedy and RaeiGreedy by generating a PIDS that is about 10% smaller than the other algorithms. Figure 2b shows what percentage of the total nodes in the graph participate in the PIDS. Again, as expected AltGreedy uses $2-5\%$ fewer nodes in the PIDS.

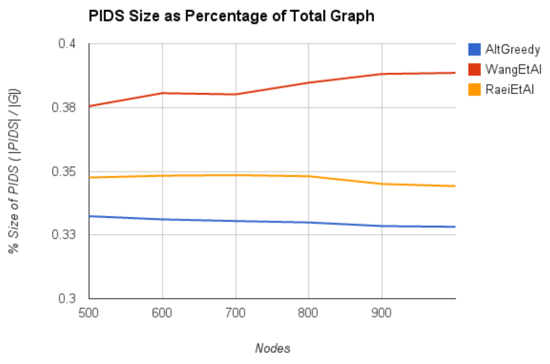
In addition to these results, we ran the RaeiGreedy (since this represents the state-of-the-art in the literature and performs better than WangGreedy) and AltGreedy algorithms on data sets of larger graphs of with 5000 to 25000 nodes in increments of 5000, all with an average degree 10. The results for this can be seen in Figure 3. The trend seen in smaller graphs is preserved with AltGreedy generating a smaller PIDS and using fewer nodes than RaeiGreedy.

2) TPIDS Generation: We also ran simulations for the Total Positive Influence Dominating Set (TPIDS) generation. To construct a TPIDS, the function s described in Section III-B, can also be adapted to test satisfaction under TPIDS, by simply removing the $u \in P$ qualification, since in a TPIDS *every* node needs to have at least half of its neighbors in our TPIDS subset. As expected, on average the size of the TPIDS generated were larger than that of the PIDS for the same graph. Interestingly, TPIDS and PIDS exhibit opposite behaviors for certain tests. To replicate an experiment from [11], we used all three algorithms to generate both TPIDS (as in the original experiment) and also PIDS for graphs of a fixed size 200, while incrementing the average degree by 2.

For TPIDS, the data corresponds to the results published in [11], in that the higher the average degree, the smaller the average TPIDS. However, the opposite behavior is exhibited when we generate a PIDS, i.e., a higher average degree means we select more nodes. This behavior makes sense, because when generating a TPIDS (T), not only do the nodes not in T need to have half of their neighbors in T, but so do the nodes in T itself. When there is a low



(a) Average PIDS size



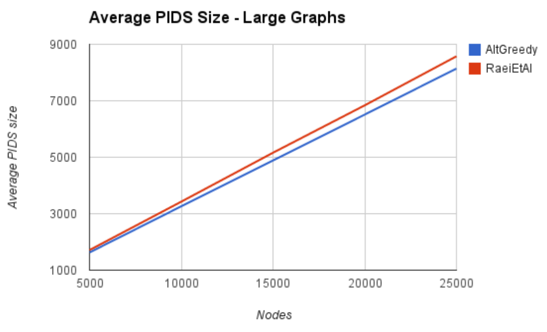
(b) Average PIDS $\%$

Fig. 2: PIDS Results for $\#$ nodes $n = 500 \rightarrow 1000$, 100 iterations per size

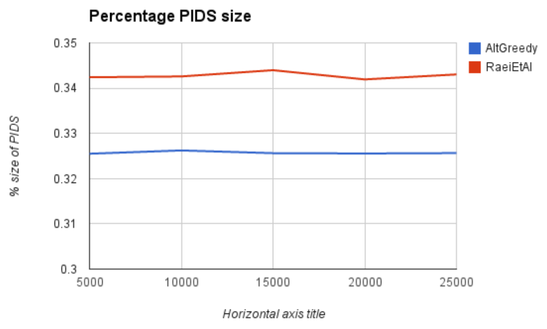
average degree, and the graphs follow a power-law degree distribution, the graphs have only a few nodes with high degree and many nodes with low degree. This explains why there is a reverse trend in TPIDS and PIDS when the average degree is small.

V. Conclusion and Future Work

Our results show that our algorithm produces a smaller Positive Influence Dominating Set (PIDS) than Wang-Greedy and RaeiGreedy; however it produces a Total Positive Influence Dominating Set (TPIDS) of a similar size as those generated by WangGreedy and RaeiGreedy. This suggests that any further algorithmic development in this area must be tested against both TPIDS and PIDS, since their different qualifications require different approaches. For future work, we plan to further investigate the relationship between average degree and the PIDS and TPIDS size. In addition to this, our PIDS selection function g could be refined through further experimentation. Also, extensions of these algorithms to directed graphs that



(a) Average PIDS size



(b) Average PIDS $\%$

Fig. 3: PIDS Results for $\#$ nodes $n = 5000 \rightarrow 25000$

represent social networks like Twitter will be also be useful since these networks allow users to influence others without the reverse being true. In such networks, the nodes that are hubs become even more crucial to the spread of positive influence since typically a celebrity is followed by many users but in turn follows only a few other nodes. This can result in interesting variations of criteria for adding a node to a PIDS/TPIDS.

References

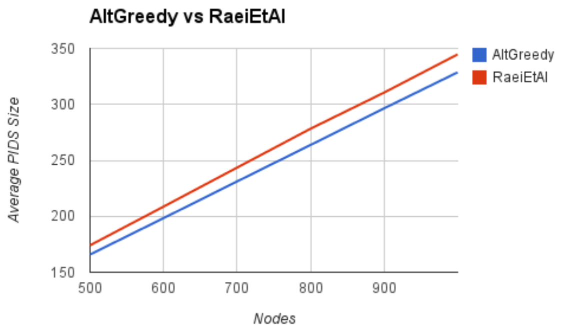
[1] S. Eubank, V. Kumar, M. V. Marathe, A. Srinivasan, and N. Wang, "Structural and algorithmic aspects of massive social networks," in *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2004, pp. 718–727.

[2] J. B. Standridge, R. G. Zylstra, and S. M. Adams, "Alcohol consumption: an overview of benefits and risks." *Southern Medical Journal*, vol. 97, no. 7, pp. 664–672, 2004.

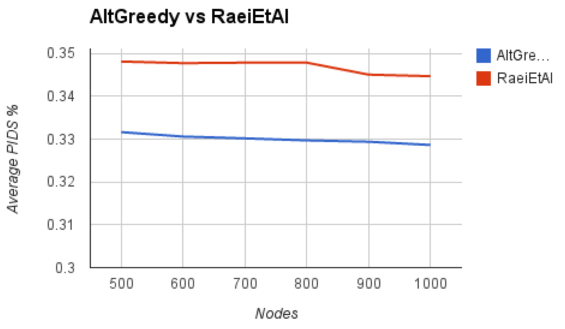
[3] K. G. Hill, J. D. Hawkins, R. F. Catalano, R. D. Abbott, and J. Guo, "Family influences on the risk of daily smoking initiation," *Journal of Adolescent Health*, vol. 37, no. 3, pp. 202–210, 2005.

[4] F. Wang, E. Camacho, and K. Xu, "Positive influence dominating set in online social networks," in *Combinatorial Optimization and Applications*. Springer, 2009, pp. 313–321.

[5] W. Zhang, W. Wu, F. Wang, and K. Xu, "Positive influence dominating sets in power-law graphs," *Social Network Analysis and Mining*, vol. 2, no. 1, pp. 31–37, 2012.

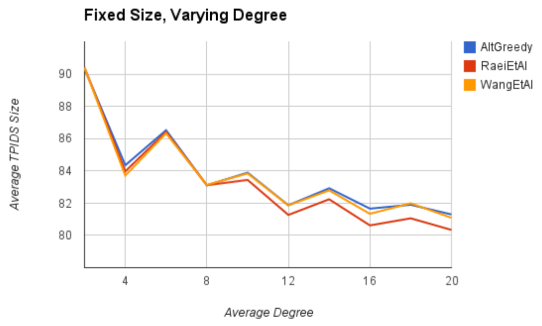


(a) Average PIDS size

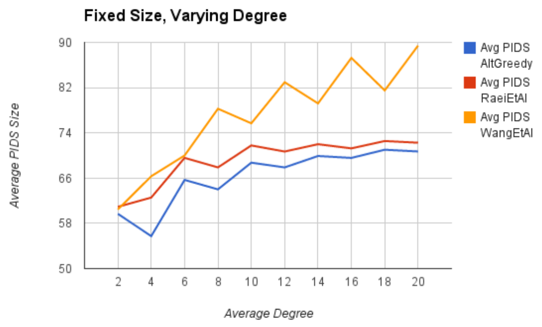


(b) Average PIDS %

Fig. 4: PIDS Results for # nodes $n = 500 \rightarrow 1000$, 1000 iterations per size



(a) Average TPIDS Size



(b) Average PIDS Size

Fig. 5: PIDS and TPIDS results for fixing number nodes $n = 200$, varying average degree $2 \rightarrow 20$ in increments of 2

[6] T. N. Dinh, Y. Shen, D. T. Nguyen, and M. T. Thai, "On the approximability of positive influence dominating set in social networks," *Journal of Combinatorial Optimization*, vol. 27, no. 3, pp. 487–503, 2014.

[7] W. Wu, H. Du, X. Jia, Y. Li, and S. C.-H. Huang, "Minimum connected dominating sets and maximal independent sets in unit disk graphs," *Theoretical Computer Science*, vol. 352, no. 13, pp. 1 – 7, 2006.

[8] A. Dhawan, M. Tanco, and N. Scoville, "A distributed greedy algorithm for constructing connected dominating sets in wireless sensor networks," in *SENSORNETS 2014 - Proceedings of the 3rd International Conference on Sensor Networks, Lisbon, Portugal, 7 - 9 January, 2014*, 2014, pp. 181–187.

[9] J. Yu, N. Wang, G. Wang, and D. Yu, "Connected dominating sets in wireless ad hoc and sensor networks–a comprehensive survey," *Computer Communications*, vol. 36, no. 2, pp. 121–134, 2013.

[10] F. Wang, H. Du, E. Camacho, K. Xu, W. Lee, Y. Shi, and S. Shan, "On positive influence dominating sets in social networks," *Theoretical Computer Science*, vol. 412, no. 3, pp. 265–269, 2011.

[11] H. Raei, N. Yazdani, and M. Asadpour, "A new algorithm for positive influence dominating set in social networks," in *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*. IEEE Computer Society, 2012, pp. 253–257.

[12] J. Leskovec and R. Sosivc, "SNAP: A general purpose network analysis and graph mining library in C++," http://snap.stanford.edu/snap, Jun. 2014.

[13] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of modern physics*, vol. 74, no. 1, p. 47, 2002.