A Graphlet-based Approach to Finding Dominating Sets

Chris Maheu
University of Notre Dame
cmaheu@nd.edu

Alex Beecham
University of Notre Dame
abeecham@nd.edu

Brent Marin
University of Notre Dame
bmarin@nd.edu

ABSTRACT

This paper presents new a polynomial time algorithm to find small dominating sets in a network. The algorithm uses graphlet degree vectors and degree centrality as heuristics to construct the dominating sets.

Keywords

Dominating Set, Graphlet, Graphlet Degree Vector, Degree, Minimum Dominating Set, NP Hard, Algorithm, Network

1. INTRODUCTION

The problem of finding the smallest dominating set for a network has been shown to be NP-Hard[1]. While it is possible to find a minimum dominating set on a network (see Table 3 under Results) with a low number of nodes and edges, we found the time taken to compute the minimum dominating set grows exponentially. Our approach uses Graphlet Degree Vectors (GDVs) and node degree to build a small dominating set that runs in polynomial time.

2. MOTIVATION

Dominating sets have many uses across multiple types of networks. In social networks, dominating sets can be used to model the spread of information, positivity, and negativity[8]. A small dominating set in this context can be used to influence people to change their mindset about a person, product, or service. Alternatively, it can be used in an emergency to get a message to seek shelter to spread rapidly through a university or city. In communication networks, especially in ad-hoc networks during emergencies, dominating sets can preserve battery life of devices for both transmitting devices and receiving devices[9]. A small dominating set in this instance will allow for more of the energy stored in a battery or created by generators to go towards medical needs and safety equipment.

3. BACKGROUND

Our approach to this problem relies on specific aspects of dominating sets and properties of graphlets in the network. We provide a brief summary of dominating sets and graphlets below.

8.1 Dominating Sets

A dominating set D is a set of nodes in network G such that every node in G is in D or adjacent to a node in D. We define a "dominating node" as a node in the dominating set we are searching for, and a "dominated node" as a node adjacent to a node in that set. A set is a dominating set if it causes every node in G to be dominated or dominating. The problem of finding a dominating set of smallest possible size for a given network has been shown to be NP-Hard[1]. There are several types of dominating sets with additional constraints, such as independent and connected, but we will not be considering these for the scope of this paper, as there are fewer algorithms available for dominating sets in general.

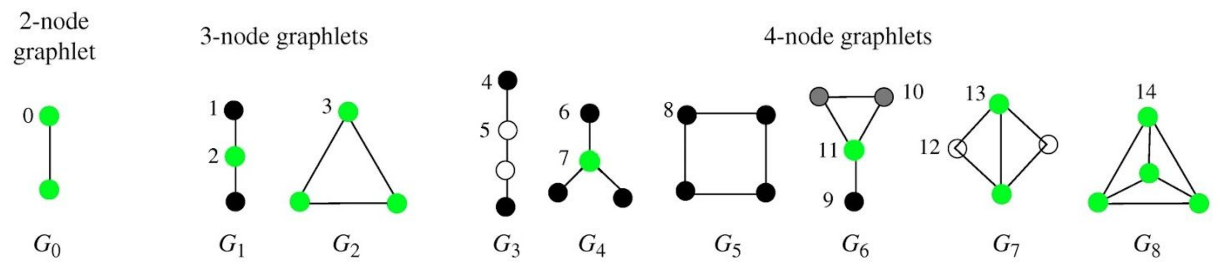

Figure 1: Power Orbits on Graphlets of size n<5 [6]

8.2 Graphlets

A graphlet is a small, connected, induced, non-isomorphic subgraph[6]. There are nine unique graphlets of four nodes or less, and fifteen orbits among these graphlets. An orbit represents nodes in a graphlet that can be mapped to each other by an automorphism, an isomorphism of a network to itself, where an isomorphism between two networks is a bijection of their node sets that preserves their adjacency[7]. Graphlets have a variety of uses, such as computing network similarity or node alignment. Counting the number of times a node touches each orbit provides a strong description of its local neighborhood. The vector containing the orbit counts for a given node in a network is called a Graphlet Degree Vector (GDV).

4. DEFINITIONS

These concepts will play an important role in our algorithm, as they form the basis for how it finds dominating nodes.

8.1 Power Orbits

We define a power orbit as an automorphism orbit that is adjacent to every node in its appropriate graphlet. The power orbits on graphlets with four nodes or fewer are marked as green in Figure 1. These orbits are important for our method because they essentially “dominate” their graphlet, so they are excellent candidates for nodes in a small dominating set.

8.2 Dominating Subgraphs

We define a dominating subgraph as any subgraph of a network such that at least one node in the subgraph must be in every possible dominating set. A given node in a network can be in multiple dominating subgraphs or none at all.

3.2.1 Dominating Subgraphs Can Exist

Assume that network G in Figure 2 has a dominating set that does not include node 1 or node 2. Then the only remaining nodes that can be in the dominating set are nodes 3 and 4. It is easy to see that neither node 3 or 4 can dominate node 1. We have contradicted our assumption and have proved that a dominating set must include node 1 or 2.

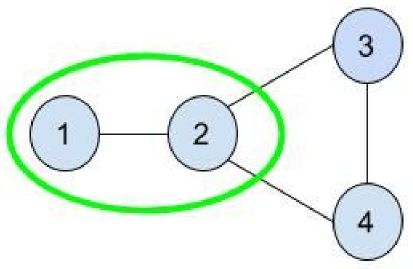

Figure 2: Example Dominating Subgraph

3.2.2 Finding Dominating Subgraphs

If a node is in a power orbit of an N-node graphlet and has degree N-1, it and its neighbors are a dominating subgraph. This is because every neighbor of that node must be in that graphlet. If no node in that graphlet is in the dominating set, then that node will not be dominated or dominating, as described in the section above. This method will not find every dominating subgraph, but it will find enough to generate a small dominating subset.

5. METHODOLOGY

To find a dominating set using power orbits and dominating subgraphs, we first calculate the GDV for every node in the network using the graphlet counting software Orca[10]. Then, we begin searching for the dominating subgraphs to add nodes from them to the dominating set.

8.1 Conditions for Subgraphs

We defined a set of conditions to find dominating subgraphs. Each of these six conditions includes a degree requirement and a GDV orbit requirement corresponding to specific power orbits. For each of these conditions, our algorithm iterates over all node GDVs, searching for nodes that fulfill the conditions. When a power orbit of the specific degree is found, then we know that node and its neighbors form a dominating subgraph. We did not choose every power orbit to be a condition to check because a few of the graphlets will eventually be changed into graphlets we do check when we prune and repeat. Limiting the number of conditions helped our algorithm run faster.

8.2 Building the Dominating Set

When our algorithm finds a dominating subgraph, it must choose one of those nodes to add to the dominating set. First, it attempts to add the node from the subgraph with the highest degree. If multiple nodes are tied for highest degree, the algorithm checks if any of the nodes are already in the dominating set. If so, the subgraph is already fully dominated and can be skipped. If none are already in the dominating set, we must choose a node randomly to add.

8.3 Prune Nodes from the Network

Once this process is complete, it is still possible that some nodes were not in any dominating subgraph, and hence not all nodes are dominated or dominating. Therefore, we remove all nodes from the network that are already in the dominating set or have all or all but one of their neighbors adjacent to a node in the dominating set.

8.4 Repeat until Complete

Once all of these nodes are removed, we repeat the entire algorithm on the new, pruned graph, continually building the dominating set as we make the network smaller and smaller. If we ever reach a point in which our power orbit conditions do not match any nodes in the graph, then we add the node with the highest degree to the dominating set, prune, and try again with the conditions. Eventually, all nodes will be dominated or dominating.

6. BENCHMARKS

To test the effectiveness of our algorithm, we choose a variety of algorithms to compare it with, as well as many different networks to run the algorithms on. We included only algorithms that accomplish the same goal of ours, which is to find a small dominating set, whether or not it is independent or connected.

8.1 Comparable Algorithms

We found several existing algorithms aimed at finding small dominating sets which we chose as benchmarks against our algorithm. We picked benchmarks based on the availability of code for the algorithms or our ability to easily implement the algorithms.

6.1.1 Existing Algorithms

The first benchmark is based on a Microsoft Research SMT solver named Z3. SMT is a generalization of the Satisfiability (SAT) problem, which is known to be NP-Hard[2]. We found an algorithm which attempts to find the minimum dominating set by reducing the problem to SMT and solving with Z3[3]. Our second benchmark is part of an open source network analysis toolkit named NetworkX[4]. This algorithm chooses the first listed node in a graph object that is not already dominating or dominated. This process is repeated until every node is dominating or dominated[5].

6.1.2 Simple, Greedy Algorithms

Our team also implemented two more benchmarks. The first is a greedy algorithm that adds the highest degree node in the network to the dominating set and removes that node's neighbors as possible entries into the dominating set. The second implemented benchmark, which we call reverse greedy, takes the opposite approach. This algorithm adds the node with the lowest degree into the dominating set and removes that node's neighbor(s) from being added to the dominating set.

8.2 Benchmark Networks

Three real-world networks (email, airports, and physics collaboration) and six randomly generated ones (1500 Node GEO, SF, ER and 5000 Node GEO, SF, ER) were used to see the effectiveness of the algorithm described in this paper. The email network[11] has 1005 nodes and 16064 edges. A node is an email account in a large European research institution, and an edge is at least one email sent between those accounts. The airport network[12] has 1575 nodes and 17215 edges. A node is an airport in the US and an edge is one or more flights between that airport. The physics collaboration network[13] has 9877 nodes and 25998 edges. A node is an author, and an edge is a collaboration between two authors on a single paper in the field of high energy physics theory.

7. RESULTS

When looking at the results produced by this algorithm, we considered two factors: runtime and dominating set size. Figure 3 shows the size of the dominating sets found by Z3, our algorithm, NetworkX, a greedy algorithm, and a reverse greedy algorithm on the 3 real world networks. Z3 beats our cardinality results for the airport and email network by small margins. For the physics collaboration network, Z3 was unable to finish as of the writing of this paper. An important finding is that our algorithm beats the greedy and reverse greedy methods in all three cases. This proves the value of dominating subgraphs. Figure 4 shows the results of our algorithm, NetworkX, greedy and reverse greedy on the generated networks. The two most effective methods are our algorithm and the greedy highest degree algorithm. Z3 was not included because it failed to complete in any comparable amount of time for these networks. Unlike the real world networks, in which our method beat the greedy method handily in each instance, the greedy method sometimes produces better results on the randomly generated networks. The difference in the size of the dominating sets is small. This seems to be the case because, on these random networks, the GDV does not have the small graphlet and low degree nodes our method needs to work. This is because, in the case of a network with few or no nodes that match the power orbit conditions we selected, our algorithm becomes the same as the greedy algorithm. Figure 5 shows the run times of the algorithms across all networks. As the runtimes show, our method does not beat its competitors on the randomly generated networks. This is due to the generated networks not having a high number of the graphlets with the matching degrees our algorithm looks for. However, for all the real world networks, our method produces very low run times, even for the 9000 node physics collaboration network.

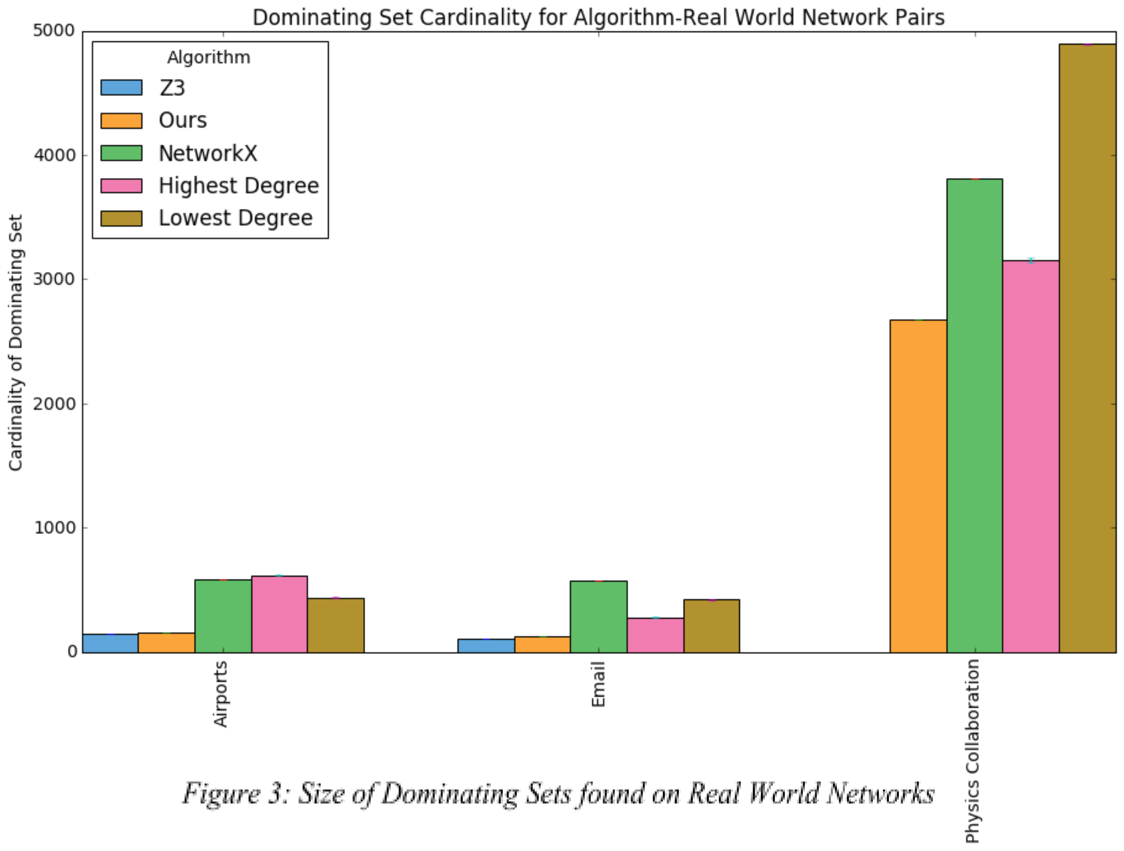

Figure 3: Size of Dominating Sets found on Real World Networks

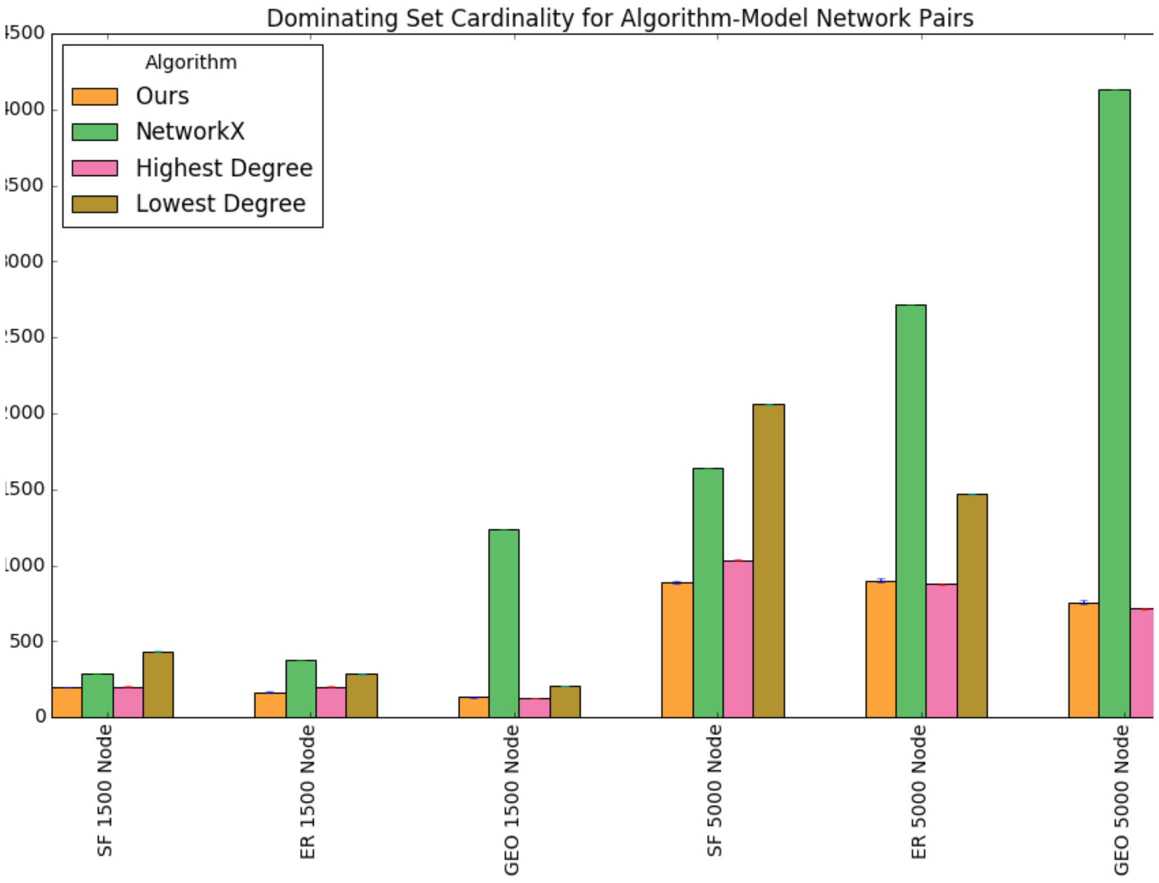

Figure 4: Size of Dominating Set of Randomly Generated Graphs

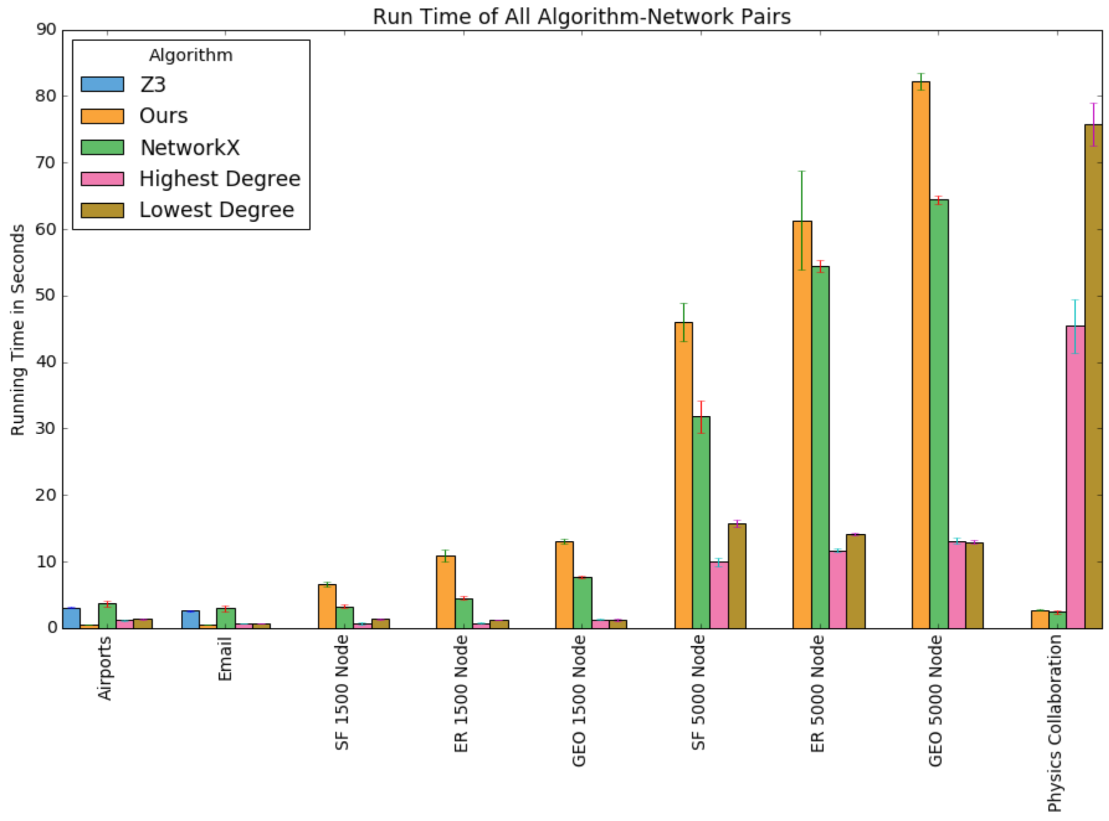

Figure 5: Runtime of All Algorithms on All Networks

8. DISCUSSION

Compared to other algorithms, this one tends to find the smallest dominating sets, but its runtime depends heavily on the type of network. Real-world networks with power-law degree distributions have more nodes that fit one of the conditions for our algorithm but generated network models often do not have as many of these nodes. Therefore, there are many cases in which our algorithm proves to be the most effective in quickly finding a small dominating set.

8.1 Time Analysis

This algorithm runs in a very short length of time, compared to the original NP-Hard problem, due to its heuristic approach to finding the dominating set. This algorithm relies on GDV calculations, which run in $O(V^4)$. Using GDVs in conjunction with node degree (which is calculated in constant time), the algorithm is able to seed the dominating set. From there, selection of the highest node within the seeded area is also done in constant time. In the worst case, there are no dominating subgraphs found and the algorithm will then choose the highest degree node and remove that node and its neighbors. Thus the worst case run time will be $O(V^5)$. This becomes clear when the runtime exceeds that of other methods for the model networks with large numbers of nodes. 5000^5 is a large number, and model networks tend to be the worst case for our algorithm. However, the results on the real-world network, the best case, are much more promising.

8.2 Potential Improvements

The algorithm can be improved in its base case of choosing the highest degree node and removing its neighbors from the graph. It is possible that doing this will remove other nodes of high degree that, if left in, result in a smaller dominating set than currently found. Another area for improvement is experimenting with different centrality measures. Currently, only degree centrality is considered. Perhaps another measure of centrality will produce smaller dominating sets. Alternatively, instead of only considering the degree of the nodes within the dominating subgraph, the degree of their neighbors could be evaluated as well. This might allow the algorithm to make more informed choices than it currently does.

8. CONTRIBUTIONS

Chris Maheu did about 30% of the work. He found a few networks used in the development of the algorithm and developed the idea of a power orbit. He wrote the parts of the code that handed reading in and organizing the network, as well as calculating the GDVs with Orca. He was also a significant contributor in presentation and report development. Alex Beecham did about 30% of the work. Alex proposed the problem of solving dominating sets as a project to the group. He came up with the idea of a dominating subgraph and wrote the first working code that implemented this finding. He also contributed to the class presentation and writing of this report. Brent Marin did about 40% of the work. He developed the pruning part of the algorithm and integrated the team's code into a working product. Brent did network data cleaning and formatting, ran the benchmarks, and contributed to presentation and report writing.

9. REFERENCES

[1] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, 1979.

[2] Stephen A. Cook. 1971. "The complexity of theorem-proving procedures." In Proceedings of the third annual ACM symposium on Theory of computing (STOC '71). ACM, New York, NY, USA, 151-158.

[3] F. Kuhn and T. Moscibroda. "Distributed Approximation of Capacitated Dominating Sets" https://www.microsoft.com/en-us/research/wp-content/uploads/2017/01/SPAA2007b.pdf, 2007.

[4] A. Hagberg, D. Schult, and P. Swart. "NetworkX." https://networkx.github.io/, 2005.

[5] A.H. Esfahanian "Connectivity Algorithms." http://www.cse.msu.edu/~cse835/Papers/Graph_connectivity_revised.pdf

[6] N. Pržulj, D. G. Corneil, and I. Jurisica, "Modeling Interactome: Scale Free or Geometric?," *Bioinformatics* 20, 3508-3515 (2004).

[7] N. Pržulj, "Biological network comparison using graphlet degree distribution." Proceedings of the 2006 European Conference on Computational Biology (ECCB'06), *Bioinformatics,* 23, e177-e183 (2007).

[8] A. Sasireka, A.H. Nandhu Kishore. "Applications of Dominating Set of Graph in Computer Networks." International Journal of Engineering Sciences & Research Technology, January 2014

[9] J. Wu, M. Cardei, F. Dai, and S. Yang. "Extended Dominating Set and Its Applications in Ad Hoc Networks Using Cooperative Communication" http://cgi.di.uoa.gr/~istavrak/Group-Presentations/DomSet/wu05_extDS.pdf

[10] Hocevar T, Demšar J. "A combinatorial approach to graphlet counting." *Bioinformatics*. 2014;30(4):559–565.

[11] Email EU-core Temporal Network (converted to non-temporal network). Stanford Network Analysis Project. https://snap.stanford.edu/data/email-Eu-core.html

[12] Bureau of Transportation Statistics, 2010, Table T-100 id 292 https://toreopsahl.com/datasets/#usairports

[13] High Energy Physics - Theory Collaboration Network. Stanford Network Analysis Project. https://snap.stanford.edu/data/ca-HepTh.html