1 Introduction

"Viral Marketing" is a term first introduced in 1997 by Steve Jurvetson and Time Draper[1]. It is an effective way of advertising a product to a large group of people by having little groups eventually spread awareness to your product. Companies make a commercial with this in mind. Many over-the-top and absurd commercial having been making rounds across the internet [2]. First, being seen by a few people who were entertained by the commercial enough for them to hit the share button creating a network to other people who will eventually see the commercial shared by their friend and promptly clicking the share button themselves. Thus, repeating the cycle and expanding the network.

Online social networks are a good medium to advertise on. Viral Marketing on online social networks is a very cost-effective technique that most companies today utilize. For only spending a few dollars, the payoff can be amazing[1]. In 1996, this strategy was used by the once popular e-mail service Hotmail: Every mail sent via Hotmail contained advertisement for the service. A total of $50,000 was spent on this campaign. Hotmail expanded into 12 million users in just 18 months[1].

Recently, the video game company Epic Games, used this tactic to advertise their new battle royale game "Fortnite" and it gave astounding results. The number one video streaming site, Youtube, is a very large network and resource to put to use. They sponsored Youtubers with hundred to millions of subscribers to promote their game[3]. Youtuber Evan Fong "VanossGaming" was one of the content creators Epic Games sponsored for advertising the beta release of Fortnite [4]. He currently has 23 million subscribers(August 2018). The video, titled "Fortnite Funny Moments-Launcing the Rocket!(Gameplay)", has garnered 10 million views in a span of a year.

With this in mind, finding the set of people to target would be ideal. Target Set Selection is basically finding the smallest set of people in a social network to influence all of the people in said network[6, 5, 7]. This network can be represented with a graph G with each person as a node/vertex V. Connecting each node/vertex V is an edge E. Connected vertices are considered neighbors n. The degree $d(v)$ of each node is how many neighbors a node has. The threshold $k(v)$ of each node is a factor to gauge the suceptability of a node from being influenced. The neighbor $N(v)$ is the set of neighbors of the node.

1.1 Two Basic Diffusion Models

Two basic diffusion models have been introduced in 2003 by Kempe, Kleinberd and Tardos[6]. These models are improved upon by works proceeding it. These two models are considered in order to further understand the basics and foundations of the models the algorithm is based/taken from.

In the following models, as information spreads through the network nodes will be referred to as either *active* or *inactive*. The chance of the node to be active increases as more of its neighbor become active. Inactive nodes can become active nodes but the other way around is not to be considered.

1.1.1 Linear Threshold Model

In this model, a node v is influence by each neighbor n according to a $weight\, b_{v,w}$ such that $\sum_{w\, neighbor\, of\, v} b_{v,w} \leq 1$[6]. Each node v is assigned a threshold k uniformly at random from interval [0,1], [1,$\frac{d(v)}{2}$] or [1,10] [5]. The threshold range varies when applied to different variations of social network problem. In this model, the threshold lies in the interval [0,1].

This represents the fraction of neighbors of a node to be active in order for that node to be active itself. A weight is assigned to each of their neighbor. If the cumulative weight of that node's neighbors is the same or more as their threshold, the node itself will become active. Thresholds are selected at random because of the lack of knowledge of the tendencies of the node [6].

1.1.2 Independent Cascade Model

This model uses probablity instead of thresholds. Nodes have a certain probability pertaining to how a single interaction from an active node to an inactive node can go. The active can either activate its neighbor or not. Once the inactive node is influenced, it can do the same to its neighbors with their own probabilities and the cycle continues for each time step until no more nodes can be activated[6].

1.2 Other threshold models

We study other threshold models since the model in this study uses thresholds as a way to gauge how easy or how hard it is to influence/activate a certain node. Learning the tendencies and limitations of other thresholds models

will help in understanding the tendencies of the threshold function used in the study. This study has a threshold function that uses real numbers in the set [1,2,...,10].

1.2.1 General Threshold Model

To be able to express the notion that a node v's tendency to be active can be based on an arbitrary monotone function of the set of neighbors of v is a monotone threshold function f_v that maps subsets of v's neighbor set to real numbers in [0,1], with the condition that $f_v() = 0$. The diffusion process follows the general structure of the previously discussed Linear Threshold Model. However, v becomes active in time step if $f_v(S) \geq t_v$, where S is the set of neighbors of v are active in the previous time step[6].

1.2.2 Majority Thresholds

One important and well-studied threshold model is the majority threshold. This is where a vertex becomes active if at least half of its neighbors are active $[1, \frac{d(v)}{2}]$[5]. This can be applied to networks that have voting systems and use the majority rule.

1.2.3 Small Thresholds

This case is when all the thresholds are small constants. However, this threshold becomes more complex as the threshold increases from $k = 1$. Past researches have studied that, for any $k \leq 2$, the Target Set Selection Problem is NP-hard.A theorem constructed by Chen et.al(2009) states that the Target Set Selection problem is NP-hard when the thresholds are at most 2. In order to prove NP-hardness we need to define a few terms [8].

- NP or Non-deterministic polynomial time is the set of languages that can verified in polynomial time, or equivalently, that can be solved in polynomial time by a "guessing computer", whose guesses are guaranteed to produce an output of "yes" if at all possible.

- A problem is NP-hard if an algorithm for solving it can be translated into one for solving any NP-problem.

- Boolean Variables are variables having a value of TRUE represented by x_i and FALSE represented by the negation of x_i which is $\bar{x_i}$. They can

be interchanged if $\bar{x_i}$ is defined as TRUE. Let $x_1, x_2, ..., x_n$ be boolean varibles.

- A literal is either one of the variables x_i or its negation $\bar{x_i}$.
- Clause is a set of literals.
- A Clause is SATISFIABLE if there is an assignment of truth values to the variables $x_1, x_2, ..., x_n$ so that each clause is satisfied.
- 3SAT is a set of problems with each clause having the logical OR 3 boolean variables. Each clause is then connected by the logical AND. Example: $(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3)$

The Target Set Selection problem is NP-hard when the threshold k for every vertex v is at most 2

Proof: We reduce 3SAT in order to prove that the TSS is NP hard. Given a formula $\phi(x_1, x_2, ..., x_n) = C_1 \wedge C_2 \wedge ... \wedge C_n$. 3SAT has at most 3 literals for each clause. With this in mind, we can assume that both x_i and $\bar{x_i}$ will appear atleast once in the formula. We create a graph $G = (V, E)$ Fig.1.(The left gadget is if x_i occurs twice and $\bar{x_i}$ occurs once, the middle one if x_i occurs once and $\bar{x_i}$ occurs twice, and the right one if both variables occur once). The numbers in the nodes are their corresponding thresholds. For each clause C_j, we add a vertex w_j with a threshold of 1. Let $W = w_1, w_2, ..., w_m$. If $x_i \in C_j$, we connect v_i to w_j. If $\bar{x_i} \in C_j$, we connect v_i' to w_j. Notice that v and v' in the graph have thresholds equal to their degree.

We claim that ϕ is satisfiable if and only if the optimal target size of G has a size of n.

We assume that ϕ is satifiable. Define a target set.

$$S = \{u_i | 1 \leq i \leq n : x_i = true\} \cup \{u_i' | 1 \leq i \leq n : x_i = false\} \qquad (1)$$

Note that $|S| = n$. If $u_i \in S$, we can see that v_i and other nodes between v_i *and* u_i becomes active. The same can be said for u_i' and v_i'. Since we've made the assumption that ϕ is satisfiable, by the construction of G, all vertices in W become active, which in implies that v_i and v_i' become active. After this, vertices in G become active.

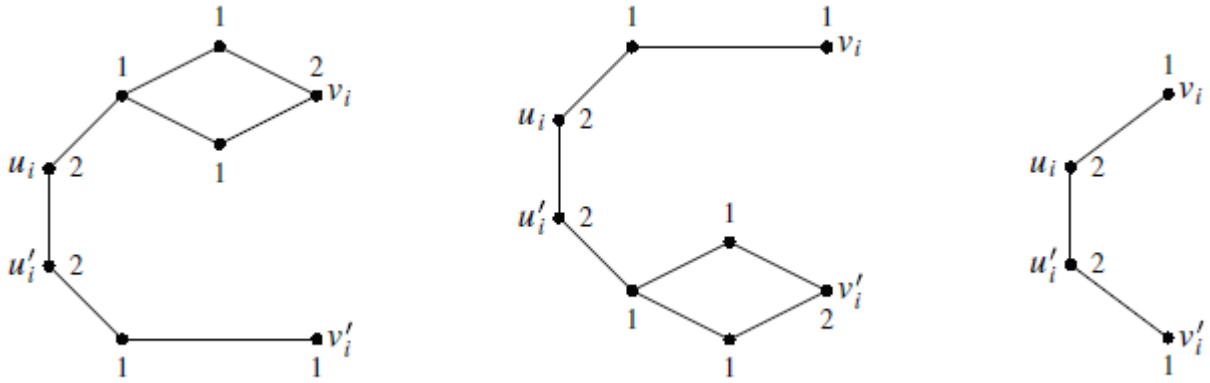

Figure 1: Gadget for each variable x_i

Let T be the the target set. Suppose that $|T| \leq n$. For any $i = 1, 2, ..., n$, the threshold of u_i and u_i' is 2. This implies that at least one of them is in T. Hence, $|T| = n$ and T does not contain any vertex other than u_i, u_i'.

To make w_j active, at least of its neighbors have to be active before it. Asumme without loss of generality, v_i is active before w_j. To make v_i itself active, we need to have u_i active. This implies that $u_i \in T$. In summary, if all the vertices in W are active by targeting T, the assignment corresponding to T satisfies ϕ. Hence, ϕ is satisfiable if and only if the optimal target set of G has the size n, which would imply that the Target Set Selection problem is NP-hard.[5]

For the instance of $k = 1$ it can be solved trivially. Take any graph G and set of all node's thresholds $k = 1$. Activate any of the nodes in that graph and you will create a chain reaction of activating **every** node in that graph. So, with $k = 1$ we can take any one node from that graph that is our target set.[5].

1.2.4 Unanimous Thresholds

This settings is considered the most influence-resistant of the previously mentioned[5]. All the nodes have unanimous thresholds i.e, the threshold of each node is the same as it's degree. This threshold can be applied to an ideal virus-resistant network, where a vertex is infected only if all of its neighbors are infected with the virus. With this in mind, this threshold can

be used in constructing robust virus resistant network structures[5].

1.3 Target Set Selection Algorithms compared in this study

1.3.1 TIP-DECOMP

- d_i^{in}= degree of vertex v_i
- At lines 1-3, a for loop is used for computing the $k_i's$ for each vertex v_i. k_i is defined as $k_i = [t(v_i) \cdot d_i^{in}]$
- At lines 4-6, a for loop is used to compute for the distribution or $dist_i$. This will later be used as distinguishing what the current v_i is to be used in the inner while loop.
- At line 7 a FLAG is instantiated as a boolean variable which will be used in the while loop for identifying the target set selection. We can see it being used in line 8.
- The for loop in line 9-13 is for selecting the vertex v_i where the result of the degree and the threshold is minimal.
- Lines 14-16 are for escaping the main while loop. If this condition is met, it means that the procedure is done.
- The else statement in 16-25 is for removing the vertex where the degree and the threshold is almost the same(line 17).
- The inner for loop in lines 18-24 is used for updating the distributions in the neighborhood of v_j.
- The process returns the reduced vertex set with the vertices removed if necessary. This is the target set.

TIP-DECOMP or Tipping Decomposition is model based on the idea of node "tipping" when a node adopts a property or behavior if a number of his neighbors currently exhibit the same. It is a type of Target Set Selection Algorithm.

The algorithm above inputs a threshold function t and the social network G and outputs the network with vertices removed based off the condition in the algorithm.[9].

Algorithm 1 TIP-DECOMP

Require: Threshold function, t and social network $G = (V, E)$
Ensure: V'

1: **for all** *vertex* v_i **do**
2: compute k_i
3: **end for**
4: **for all** *vertex* v_i **do**
5: $dist_i = d_i^{in} - k_i$
6: **end for**
7: FLAG=TRUE
8: **while** FLAG=TRUE **do**
9: **for** $v_i \in V$ **do**
10: **if** $v_i = min(dist_i)$ **then**
11: $v_i =$ current v
12: **end if**
13: **end for**
14: **if** $dist_j = \infty$ **then**
15: FLAG=FALSE
16: **else**
17: Remove v_i from G
18: **for all** $v_j \in n_i^{out}$ **do**
19: **if** $dist_j > 0$ **then**
20: $dist_j = dist_j - 1$
21: **else**
22: $dist_j = \infty$
23: **end if**
24: **end for**
25: **end if**
26: **end while**
27: **return** All nodes left in G

1.4 VirAds

- ρ = The influence factor is a decimal between 0-1 that is multiplied to the degree of current vertex v to get $n_v^{(a)}$
- r_v = the round in which the vertex v is activated
- $n_v^{(e)}$ = the number of new active edges after adding v into the seeding
- $n_v^{(a)}$ = the number of extra active neighbors v needs in order for it to activate.
- r_v^i = the number of activated neighbors of v up to round i where i=1..d
- Lines 1-2 are for initialization. Line 1 is initialization of variables with values while line 2 is initialization of zero variables.
- Line 3 shows that all inactive vertices will be activated or considered in the algorithm
- Line 4-6 finds the value where the maximum occurs in the sum of the degree of v and it's effectiveness.
- u is added to P in line 7
- A queue is initialized by adding the vertex and its corresponding round, which at the start is zero (u, r_v)
- Lines 9-11 updates the neighbors of u, reducing their degree by 1. If less than zero is reached, it goes back to zero.
- At the start of the while loop in line 13 (t, r_t) is set as the head of the queue. This is obtained by the pop function.
- For all neighbors w of the current node t, another for loop is used for checking if the neighbors of t have reached the threshold defined by $\rho \cdot d_w$ AND $round\ r_w \geq the\ max\ round\ d$ AND the index i of the for loop doesn't exceed the max rounds d.
- This then leads to the thresholds of the neighbors of the that neighbor w being decremented.

Algorithm 2 VirAds Algorithm

Require: Graph $G = (V, E), 0 < \rho < 1, d \in N^+$
Ensure: A small $d - seeding$

$n_v^{(e)} = d(v), n_v^a = \rho \cdot d(v), r_v = d + 1, v \in V$
$r_v^i = 0, i = 0..d, P = \emptyset$
while there exist inactive vertices **do**
    repeat
        $u = argmax_{v \notin P}\{n_v^{(e)} + n_v^{(a)}\}$
        Recompute $n_v^{(e)}$ as the number of new active edges after adding u.
    until $u = argmax_{v \notin P}\{n_v^{(e)} + n_v^a\}$
    $P = P \cup \{u\}$
    Initialize a queue: $Q = \{(u, r_v)\}$
    for all $x \in N(u)$ **do**
        $n_x^{(a)} = max\{n_x^a - 1, 0\}$
    end for
    while $Q \neq \emptyset$ **do**
        $(t, r_t) = Q.pop()$
        for all $w \in N(t)$ **do**
            for all $i = r_t$ *to* $\min\{r_t - 1, r_w - 2\}$ **do**
                $r_w^i = r_w^i + 1$
                if $(r_w^{(i)} \geq \rho \cdot d_w) \wedge (r_w \geq d) \wedge (i + 1 < d)$ **then**
                    for all $x \in N(w)$ **do**
                        $n_x^{(a)} = \max\{n_x^{(a)} - 1, 0\}$
                    end for
                    $r_w = i + 1$
                    if $w \notin Q$ **then**
                        $Q.push(w, r_w)$
                    end if
                end if
            end for
        end for
    end while
end while

- In line 21 the round of the neighbor w is incremented and if that current $w \notin Q$ then it is added to the queue and the while loop starts again.

VirAds selects in each step the vertex u with the highest effectiveness which is defined as $n_u^{(e)} + n_a^{(e)}$. Which is basically the vertex v which can activate the most number of *edges*. It also considers the vertex with the most active neighbors. After that, the algorithm updates the measures for all the remaining vertices. VirAds will make the selection based on the information within d-hop neighbor around the considered vertices rather than only one-hop neighbor as in the degree-based heuristic. When a vertex u is selected, it causes a chain-reaction and activate a sequence of vertices or lower the rounds in which vertices are activated[10].

The algorithm utilizes a queue. A queue is a data structure which is FIFO. Imagine, a line at any commercial establishment where the first who came is the first who's served.

References

[1] S. Jurvetson, "What exactly is viral marketing," *Red Herring*, vol. 78, pp. 110–112, 2000.

[2] R. J. Larson, "The rise of viral marketing through the new media of social media," 2009.

[3] M. Abuljadail, N. C. Bi, A. Fisher, C. Y. Joa, K. Kim, X. Wen, and F. R. Zhang, *The Audience and Business of YouTube and Online Videos*. Rowman & Littlefield, 2018.

[4] L. Ha, "Most popular youtube channels," *The Audience and Business of YouTube and Online Videos*, p. 135, 2018.

[5] N. Chen, "On the approximability of influence in social networks," *SIAM Journal on Discrete Mathematics*, vol. 23, no. 3, pp. 1400–1415, 2009.

[6] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '03. New York, NY, USA: ACM, 2003, pp. 137–146. [Online]. Available: http://doi.acm.org/10.1145/956750.956769

[7] P. Shakarian and D. Paulo, "Large social networks can be targeted for viral marketing with small seed sets," in *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, ser. ASONAM '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 1–8. [Online]. Available: http://dx.doi.org/10.1109/ASONAM.2012.11

[8] O. Chartrand, *Applied and Algorithmic Graph Theory*, 1993.

[9] P. Shakarian and D. Paulo, "Large social networks can be targeted for viral marketing with small seed sets," in *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*. IEEE Computer Society, 2012, pp. 1–8.

[10] T. N. Dinh, H. Zhang, D. T. Nguyen, and M. T. Thai, "Cost-effective viral marketing for time-critical campaigns in large-scale social networks,"

IEEE/ACM Transactions on Networking (ToN), vol. 22, no. 6, pp. 2001–2011, 2014.