

Overexposure in Viral Marketing

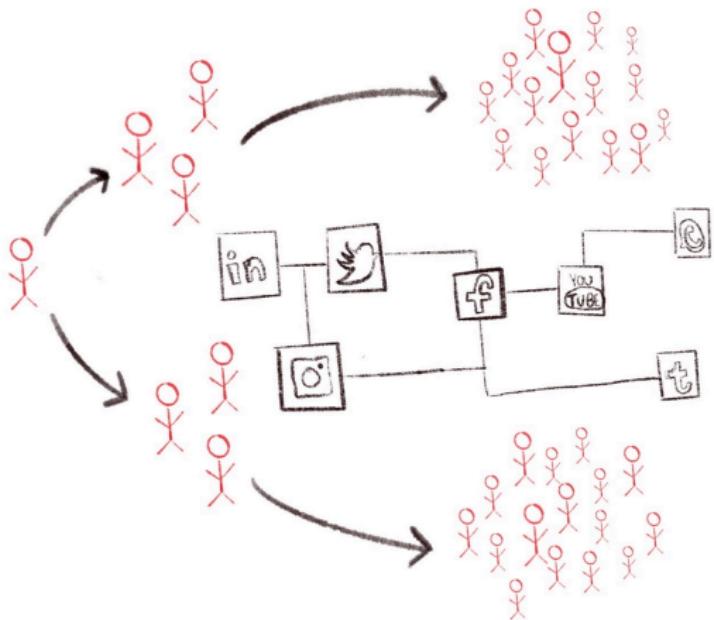
Kim Hancock

khancock@bowdoin.edu

Honors Project Advisor: Professor Irfan

12/15/2020

First, some preliminaries...



Motivation for this project

- Information cascades in social networks
- Application: viral marketing
- Caution: “too much cascade” may be bad!
 - We call this *overexposure*



Central Research Question

How do we select a set of “initial adopters” that will maximize “payoff” of a cascade?



Preliminary Set-up

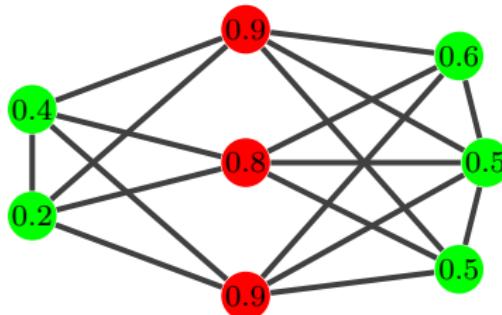


Figure: Graph Set-Up

We consider a graph $G = (V, E)$ where $|V| = n$ and $(u, v) \in E$

- A product has an “appeal” parameter $\phi \in [0, 1]$.
- Here, $\phi = 0.7$
- Each node i in the network has a criticality parameter θ_i
- If $\theta_i < \phi$ they accept the product
- If $\theta_i \geq \phi$ they reject it



Original Graph to Cluster Graph

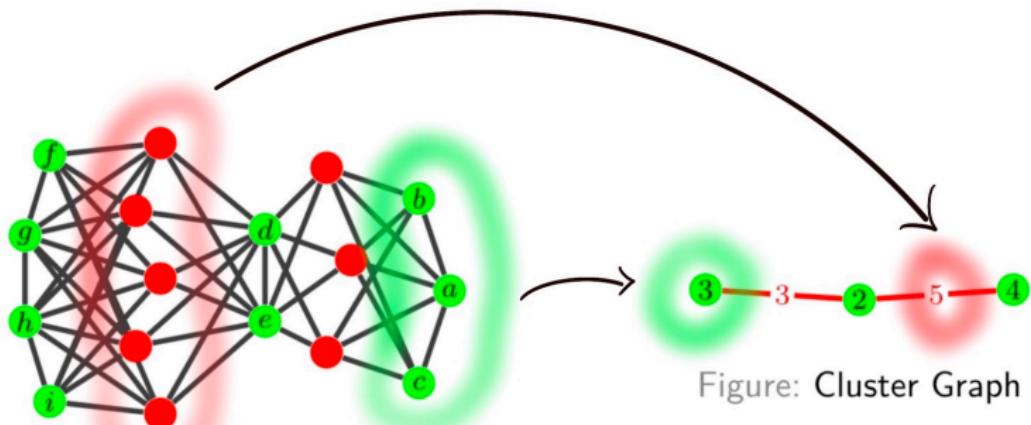


Figure: Individual Nodes

Figure: Cluster Graph



Original Graph to Cluster Graph

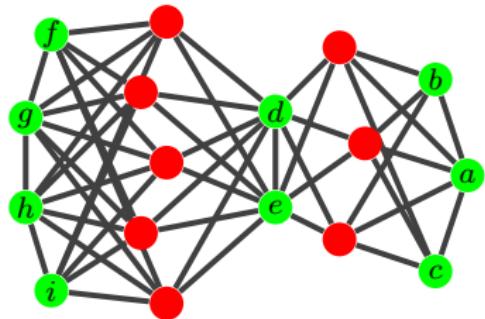


Figure: Individual Nodes



Figure: Cluster Graph

We identify *clusters of accepting nodes* which are groups of nodes surrounded by rejecting nodes. Each group of nodes becomes a cluster node in our new graph!

Previous Works

Models of propagation were first studied in the context of social networks by Kempe 2003 and experiment with both a *threshold* and *cascade* model of influence maximization.

More recent work by Cui et al. 2018 and Iyer and Adamic 2019 also use the cascade model but do not include overexposure explicitly.



Previous Works

Abebe et al. 2018 considered overexposure for the first time.
They proved you could find an optimal seed set in polynomial time
in the **unbudgeted** case

Theorem 1 (Budgeted Seeding Abebe et al. 2018)

The decision problem of whether there exists a seed set S with $|S| \leq k$ and $\pi(S) > 0$ is NP-complete.

We want to solve this budgeted case:

- Efficient algorithms for special cases
- Heuristics for general case



Our Algorithms



Overview of Algorithms we Have Designed so Far

	Greedy	Recursive DP	Tree Decomposition	Integer Linear Program
Overview	Greedily pick k highest payoff clusters	Recurse on up to k seeds, consider n choose k seeds	When we have cycles, do a tree decomposition	Get an exact answer an integer linear program
Pros	Fast, easy to implement	Gives exact answer for trees	Works on general graph	Exact answer
Cons	Inaccurate	Doesn't work in general case	Approx. factor not proven	Exponential run time



Greedy Approaches

Simple Greedy Algorithm

We pick the k highest payoff clusters!

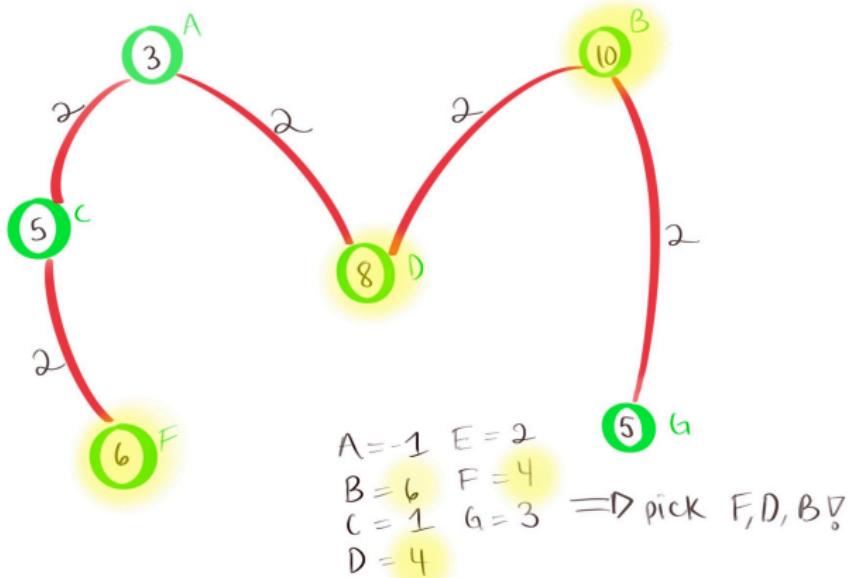
Knapsack Dynamic Programming

We simply create a memoization table and pick clusters iteratively, recursing on *take* and *dont_take*



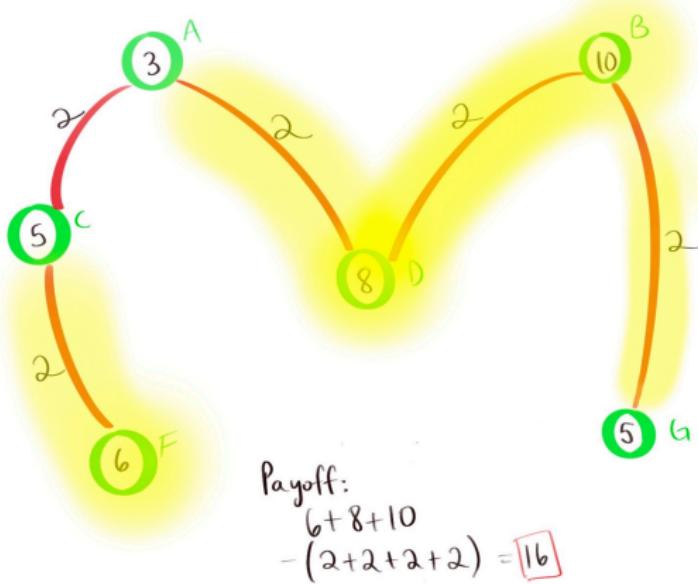
Greedy Example on Cluster Graph

Let's let $k = 3$ and pick the highest clusters x_i such that $x_i - \sum_e x_e$ is maximized



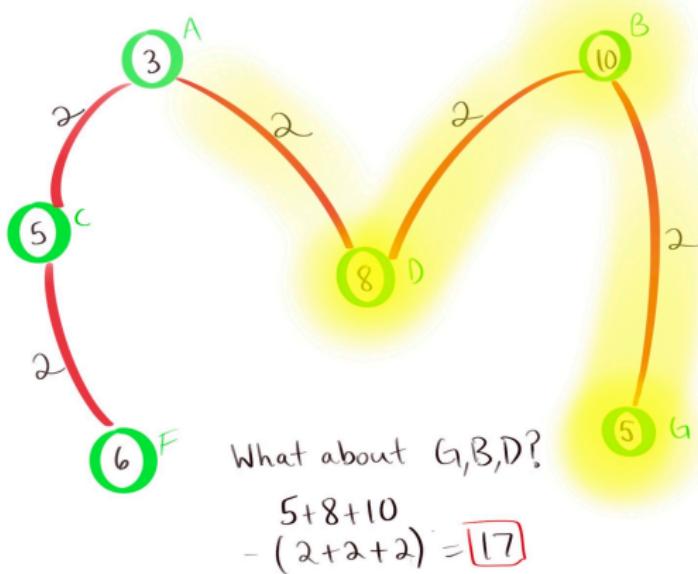
Greedy Example on Cluster Graph

Let's let $k = 3$ and pick the highest clusters x_i such that
 $x_i - \sum_e x_e$ is maximized



Greedy Example on Cluster Graph

But, we can easily see that there is a higher payoff!



Greedy Example on Cluster Graph

We thus conclude that the greedy approach does not lead to an optimal solution! Neither does knapsack dynamic programming, as there is not optimal substructure. The order in which we consider each cluster matters, and determining an optimal order would be timely.



Linear Program on Cluster Graph

To get to an optimal solution in exponential time, we implemented a linear program.

$$\text{objective function} = \sum_{i=1}^n x_{i,i} * w_{i,i} - \sum_{j=1}^n x_{e,j} * w_{e,j}$$

Subject to constraints:

$1 \geq x_{i,j} \geq 0$, such that $x_{i,j} \in \mathbb{Z}$ for all $i, j \leq n$

If $w_{i,j} > 0$, then $x_{i,j} \geq x_{i,i}$ for all $i, j \leq n$ and $i \neq j$

$\sum_{i=1}^n x_{i,i} \leq k$, where k is the number of nodes we are choosing to seed.



Recursive Dynamic Programming

To approximate a better solution, we designed the following recursive dynamic programming algorithm:

Algorithm 2 Dynamic Programming Algorithm

```
1. procedure DPG( $G = (V, E, W)$ ,  $A$ , source, storePayoff/ $f$ , witness)
2. > source is the root of subtree, storePayoff stores maximized payoffs
3. > preComputed, True.case = precomputed, False.case = False
4. > maxSum = sum of all children of source
5. > takeChild[i] = maxPayoff[i]/maxPartition[i] for each child i of source
6. > sum = 0
7. > opt.allocation = Null
8. > opt.takeChild[i] = Null
9. > partitions = Set of all possible ways of dividing
10. > k - 1 subtrees among the children of source.
11. > maxSum = 0
12. > opt.allocation = Null
13. > opt.takeChild[i] = Null > take or leave the child node when seeds are allocated optimally?
14. > for p in partitions: do
15.   sum = 0
16.   Define a map takeChildi from each child of source to a boolean indicating if it is taken (True) or not (False).
17.   for each child i of source: do
18.     Call DPG( $G, p, i, \text{storePayoff}, \text{witness}$ ).
19.     if storePayoff[i][p][i][True] >= storePayoff[i][p][i][False] then
20.       sum += storePayoff[i][p][i][True]
21.     else
22.       sum += storePayoff[i][p][i][False]
23.   end for
24.   if sum > maxSum then
25.     maxSum = sum
26.     opt.allocation = allocation
27.     opt.takeChild[i] = takeChildi
28.   end if
29. end for
30. > Case I. Take source:
31. > If not preComputed, True.case then
32.   partitions = Set of all possible ways of dividing k seeds among the children of source.
33.   maxSum = 0
34.   opt.allocation = Null > allocation of seeds corresponding to max sum of payoffs of children
35.   opt.takeChild[i] = Null > take or leave the child node when seeds are allocated optimally?
36.   for p in partitions: do
37.     sum = 0
38.     for each child i of source: do
39.       Define a map takeChildi from each child of source to a boolean indicating if it is taken (True) or not (False).
40.       for each child i of source: do
41.         Call DPG( $G, p, i, \text{storePayoff}, \text{witness}$ ).
42.         if storePayoff[i][p][i][True] >= storePayoff[i][p][i][False] then
43.           sum += storePayoff[i][p][i][True]
44.         else
45.           sum += storePayoff[i][p][i][False]
46.       end for
47.     end for
48.     if sum > maxSum then
49.       maxSum = sum
50.       opt.allocation = allocation
51.       opt.takeChild[i] = takeChildi
52.     end if
53.   end for
54.   > Case II. Leave source:
55.   > If not preComputed, True.case then
56.     partitions = Set of all possible ways of dividing k seeds among the children of source.
57.     maxSum = 0
58.     opt.allocation = Null
59.     opt.takeChild[i] = False
60.     for each child i of source: do
61.       for each child j of source: do
62.         if takeChildi[j] == True then
63.           maxSum = maxSum + storePayoff[i][p][i][True]
64.         else
65.           maxSum = maxSum + storePayoff[i][p][i][False]
66.         end if
67.       end for
68.     end for
69.     if maxSum >= maxSum then
70.       maxSum = maxSum
71.       opt.allocation = allocation
72.       opt.takeChild[i] = takeChildi
73.     end if
74.   end for
75.   > update maxSum, opt.allocation, and
76.   > opt.takeChild[i] if needed.
77.   > end for
78.   > storePayoff[source][k][True] = maxSum
79.   > witness[source][k][True] = ...
80.   > storePayoff[source][k][False] = maxSum
81.   > witness[source][k][False] = ...
```



Recursive Dynamic Programming

Okay, let's break this down....

- Only works when the input graph is a tree
- Gives an exact answer (which is awesome!)
- Exponential in the number of seeds
- Compute all possible ways of seeding k clusters among j subtrees, where j is the number of child nodes.

Recursive Calls

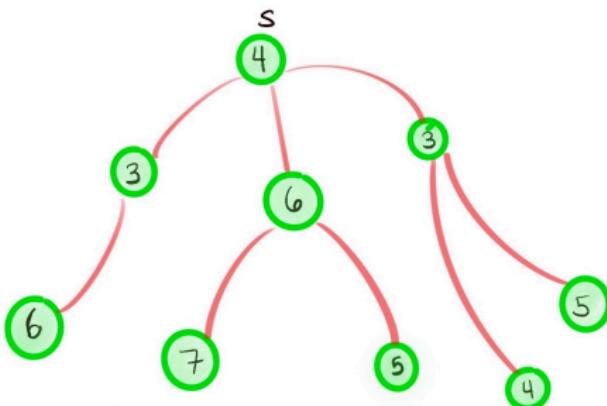
$$DP(G, p[i], i, \text{storePayoff}, \text{witness})$$

Where

- $p[i]$ is the number of seeds we are allocating to the subtree rooted at node i
- storePayoff is our memoization table, and witness is a vector storing which seeds we are taking



Recursive Dynamic Programming



So, $k=4$ means we allocate 4 seeds among the 3 children of source node!

$\{\{2,2,0\}, \{2,1,1\}, \{4,0,0\}, \{1,1,2\}$, etc...

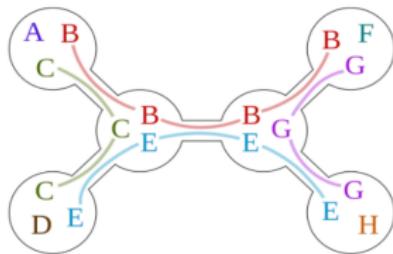
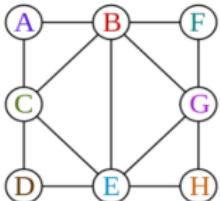
So, we have $\binom{n+k-1}{n-1}$ ways of allocating k seeds among n subtrees. For this example, that is $\binom{6}{2} = 15$ ways!



Tree Decomposition

What do we do when we have cycles in our input graph?

Answer: Tree decomposition!

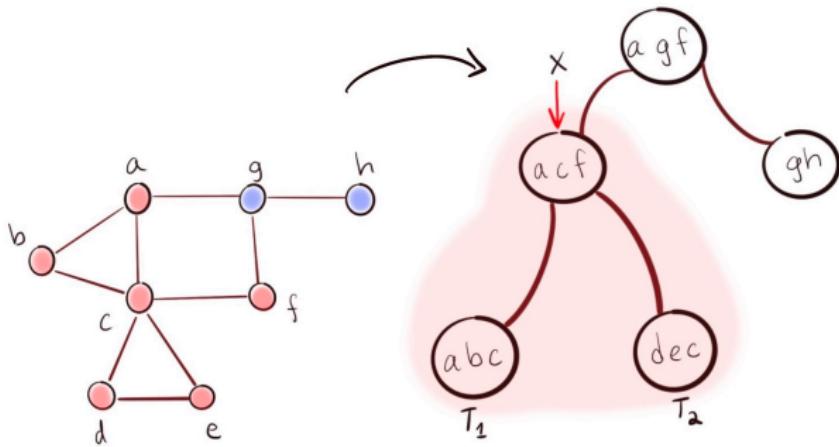


Tree Decomposition

For every edge $(u, v) \in G$ there must be some bag X_i which includes both u and v

If X_i and X_j contain v , then all nodes between them contain v

Tree Decomposition Algorithm



The optimal value at the subroot X for including a subset of nodes in the subtree is computed by considering the optimal values in T_1 and T_2 .



Details of Recursive Formulation

Then, we have two recursive calls. The first is the max payoff for selecting $S \subseteq X_i$:

$$A(S, i, k) = \max_p \left[w_S - w_{E_S} + \sum_j \left[B(S \cap X_j) - w_{S \cap X_j} + w_{E_{S \cap X_j}} \right] \right] \quad (1)$$

where w_S is the total sum of the weights in S and w_{E_S} is the sum of the edges incident on S . We also have that p is where we enumerate all possible seeds.

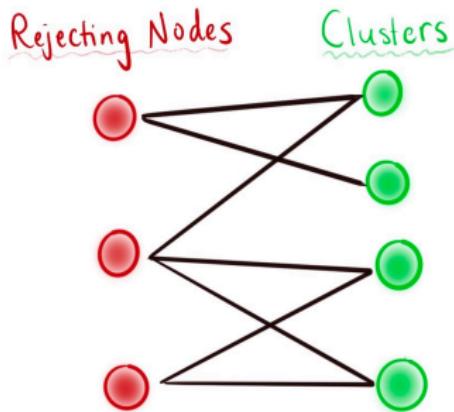
Finally, we have our second recursive call:

$$B(T, j, i) = \begin{cases} -\infty & |T| > l_j \\ \max(A(T', i)) & |T| \leq l_j \end{cases} \quad (2)$$

where $T \subseteq T' \subseteq X_j$. Thus, T' is a superset of $T \subseteq X_i \cap X_j$.



Generalized Linear Program



Generalized Linear Program

$$\text{Max} \left(\sum_{i \in C} w_i x_i - \sum_{r \in R} y_r \right)$$

such that:

- 1) $\sum_i w_i x_i \leq K$ BUDGET
- 2) $x_i \in \{0, 1\}$ TAKE or NOT
- 3) $y_r \geq x_i \quad \forall (r, i) \in G$

IF WE TAKE A CLUSTER, MUST TAKE ALL RET NODES ITS ATTACHED TO



Future Work

General Case

Improve linear program to work in the most general case

Real-life examples

Compare the outcomes of these algorithms on real life networks of
> 2000 nodes

Formally Writing Proofs

We are confident these algorithms work but want to prove
approximation factors



Special Thanks

Special thanks to the computer science department at Bowdoin and professor Irfan for giving me the opportunity to do this project!



Bibliography



Bibliography I

-  Abebe, Rediet, Lada A Adamic, and Jon Kleinberg (2018). "Mitigating overexposure in viral marketing". In: *Thirty-Second AAAI Conference on Artificial Intelligence*.
-  Cui, Fang, Hai-hua Hu, Wen-tian Cui, and Ying Xie (2018). "Seeding strategies for new product launch: The role of negative word-of-mouth". In: *PloS one* 13.11, e0206736.
-  Iyer, Shankar and Lada A Adamic (2019). "When can overambitious seeding cost you?" In: *Applied Network Science* 4.1, p. 38.
-  Kempe Kleinberg, Tardos (2003). "Maximizing the Spread of Influence through a Social Network". In: *ACM*.