Cascades and Overexposure in Social Networks: The Budgeted Case

Mohammad T. Irfan
Bowdoin College
Brunswick, Maine, USA
mirfan@bowdoin.edu

Kim Hancock*
IBM
Southbury, Connecticut, USA
Kim.Hancock@ibm.com

Laura M. Friel†
Bowdoin College
Brunswick, Maine, USA
lfriel@bowdoin.edu

ABSTRACT

Influence maximization (IM) has now been a widely studied topic, but only in recent years have studies considered overexposure. Overexposure is usually measured as the negative cost associated with reaching unintended recipients during an information cascade. A polynomial-time algorithm is known for cascades with overexposure when we can seed as many nodes as we want. This paper focuses on overexposure for the budgeted case of seeding, which has received little to no attention. We show that the problem is NP-hard even for restricted cases. For various special cases, we devise provable approximation algorithms, dynamic programming solutions, linear programming solutions, and heuristics. For the general case, we provide a linear programming solution and several fast and effective heuristics, mostly of the greedy flavor. We perform an extensive experimental study using synthetic and real-world networks. We investigate how network properties and model parameters impact our algorithms. It brings out interesting findings like why a low-quality product needs a smarter algorithm, and why certain algorithms do well on some networks but not others.

KEYWORDS

Social Networks; Cascades; Influence Maximization; Overexposure; Dynamic Programming; Linear Programming; Heuristics

ACM Reference Format:
Mohammad T. Irfan, Kim Hancock, and Laura M. Friel. 2022. Cascades and Overexposure in Social Networks: The Budgeted Case. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 9 pages.

1 INTRODUCTION

We have seen this many times: A new product comes to Amazon and starts off with a very high rating. As time goes, its rating slides downhill. The same happens for many restaurant ratings on Yelp and numerous other products and services on various platforms [1–3]. In the realm of marketing, this is commonly attributed to overexposure, which refers to casting the net of advertisement so widely that it catches unintended recipients who do not view the product favorably. Yes, we do want to maximize the spread of our product or service over a social network, but at the same time, we do not want our message to go to overly critical people who would punish us through negative reviews. This gives rise to an intriguing computational question within the general area of cascades or diffusion in social networks. *How do we maximize spread and at the same time minimize reaching unintended recipients, especially under budget constraints?*

This question falls within the well-studied topic of *influence maximization (IM)*, which is broadly defined as finding a small subset of users that can maximize the spread of information in a social network [4–6, 9, 13, 16, 17]. IM has been applied to viral marketing, where a cascade of influence is triggered by initially targeting a few *seed nodes* with the intention of maximizing the spread of influence.

Abebe, Adamic, and Kleinberg (henceforth **AAK**) recently formalized the overexposure problem [1]. Grounded in marketing research, their model takes into consideration the potential drawback of reaching overly critical nodes that can be detrimental to the progress of cascades. In the AAK model, these critical nodes play a role that is non-conducive to cascades, leading to the following binary classification of the nodes. Each node can be an *accepting node* (conducive to cascades) or a *rejecting node* (non-conducive). An accepting node propagates the information to all of its neighbors, whereas a rejecting node does not propagate it to anyone. Naturally, the objective is to find a set of seed nodes that maximizes the difference between the number of accepting and rejecting nodes ultimately reached by the cascade. AAK gave a polynomial-time algorithm for the case of unbounded seeding. In this paper, we study the problem of budgeted seeding.

There are several other studies on overexposure, but all have very different modeling approaches than AAK [1]. Iyer and Adamic [14, 15] show that overambitious seeding can be counterproductive for social media platforms like Facebook. In their model, at each discrete time step, a node's probability of using Facebook (for example) may go up or down by a certain amount depending on the number of friends active on Facebook. When this probability hits 0, the node is permanently removed. They show that overambitious seeding without any consideration for social support is not conducive to sustaining a high level of usage. Whereas AAK explicitly models the negative payoff for reaching rejecting nodes, Iyer and Adamic [14]'s model does not have any notion of rejecting nodes; the focus is rather on increasing the level of user engagement on a social media platform. In this paper, we build upon the AAK model.

A recent study by Cui et al. [8] compares different seeding strategies in the presence of negative word-of-mouth. They show that seeding nodes with high propensities of adoption is more effective than seeding high-degree nodes. Not surprisingly, both of these seeding strategies are better than seeding randomly. Although we consider seeding in this paper, we do not define any propensity measure for adoption. Furthermore, in line with AAK, we model

[*]Hancock worked on this research as an undergraduate student at Bowdoin College.
[†]Friel is as an undergraduate student at Bowdoin College.

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online. © 2022 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

the negative payoff for reaching rejecting nodes, whereas Cui et al. [8] measure only the number of accepting nodes. At a disciplinary level, our take here is more computational, whereas theirs is more focused on management and marketing research.

Another very recent work by Loukides et al. [18] combines overexposure with IM. Their work is particularly interesting because of a very different perspective on overexposure. They define overexposure as the impact to users as a result of receiving certain product information. This impact is negative if they receive the information from too many of their friends. Otherwise, the impact is positive, given that they like the product. Our view on overexposure is fundamentally different. In addition, their goal is to maximize the spread of the product, whereas ours considers the negative effect of reaching rejecting nodes.

Deviating from the IM literature, a recent graph-theoretic work studied the problem of *dual domination* [7]. Given a set of nodes S, a subset $D \subseteq S$ is called a dominating set if each node in D is either in S or has a neighbor in S. Given a number k and a network with nodes partitioned into accepting and rejecting nodes, the dual domination problem asks to find a dominating set that dominates as many accepting nodes as possible while dominating at most k rejecting nodes. Clearly, a dominating set does not model information propagation beyond just one hop, but it is an interesting theoretical concept.

Our work is also very different from several other lines of work within IM, including competitive contagion [12, 13] and IM under positive and negative connections [17].

Before we embark on technical details, we should note here that sometimes the *Groupon effect* (i.e., the decline of rating after a Groupon deal) [2, 3] is mentioned as a motivation for studying overexposure. However, a formal study shows that the Groupon effect is rather complicated due to multiple factors, with a statistical root cause analysis pointing to quality issues [3].

Our Contributions. We study the budgeted case of cascades with overexposure that has been largely left open by AAK [1]. We present a graphical model for this problem that we call "cluster graphs." We show that the problem is NP-hard even for very restricted cases, namely, unweighted cluster graphs, for which we give a 2-approximation. We then consider three classes of the problem: (1) tree-structured cluster graphs, (2) cluster graphs with cycles, and (3) the general case. For tree-structured cluster graphs, we give a polynomial-time algorithm for stars, a polynomial-time dynamic programming algorithm for bounded degree, and a Knapsack-inspired heuristic. For weighted cluster graphs with cycles, we give an integer linear program (ILP) solution. For the general case, we devise an ILP using a different graph model. We also provide several greedy heuristics for the general case. We perform extensive experiments on synthetic as well as real-world networks. We investigate how the network properties and model parameters impact the performance of our algorithms. It brings out interesting findings like why a low-quality product needs a smarter algorithm, and why certain algorithms do well on some networks but not on others.

2 MODELING OVEREXPOSURE

We follow the AAK model [1]. Let $G = (V, E)$ be a network where each node i has a *criticality parameter* θ_i that denotes how sensitive they are about the quality of a product. The quality of a product or the *product appeal* is a model parameter, denoted by ϕ. If $\phi \geq \theta_i$, then node i accepts the product and advertises it to neighbors. If $\phi < \theta_i$, then i rejects it and does not advertise it to neighbors.

In the *overexposure problem*, the overall objective for the marketer is to select a set of seed nodes so as to maximize the objective of reaching as many accepting nodes as possible while reaching as few rejecting nodes as possible. In the *budgeted overexposure problem*, the marketer has a budget of k. The goal is to select a set S of at most k seed nodes to maximize the *payoff*, $\pi(S) = |A| - |B|$, where A and B are respectively the sets of accepting and rejecting nodes reached at the end of the cascade. Note that in the usual IM task, we are only interested in maximizing $|A|$.

In terms of the behavioral aspect of this model, a cascade starts with the seed nodes as the initially reached nodes. Whenever we reach a node, we check whether the node is accepting or rejecting (which depends on the node's criticality parameter and the product appeal). As we mentioned two paragraphs ago, an accepting node advertises the product to its neighbors and a rejecting node does not. As a result, after arriving at any accepting node, we can reach all of its neighbors, irrespective of how critical the neighbors are. In contrast, when we reach a rejecting node, there is no outlet from there. Once the cascade comes to a completion, we count how many accepting and rejecting nodes we have reached. Subtracting the latter from the former gives us the payoff corresponding to the seed set we had selected at the beginning. *Given a number k, we want to select at most k seed nodes to maximize the payoff.*

Example. Fig. 1 gives an illustration of our model. Assume $k = 1$. If we select any one of the green nodes on the left as our seed, the payoff is $2 - 3 = -1$, whereas if we select any one from the right, it is $3 - 2 = 1$. Therefore, selecting one node from the three green nodes on the right is optimal. Note that it is immaterial which one of these three nodes we select.

Modeling Justification. While the rejecting nodes do not propagate information to their neighbors in a local sense, they do inflict a cost in a global sense (e.g., by writing a bad review that everyone can see, not just the neighbors of the rejecting node). The overexposure problem tries to minimize this cost by explicitly accounting for the cost of reaching rejecting nodes in the payoff measure.

Hardness Insight. Further consideration of the overexposure problem shows that it is different from taking out the rejecting nodes and doing influence maximization on the remaining network. Simple examples exist (omitted for space) to illustrate this point. Moreover, one distinctive property of the problem is non-monotonicity. That is, the optimal payoff can oscillate between "high" and "low" as we select more seed nodes. Essentially, non-monotonicity is one major reason why the problem is computationally hard.

3 ALGORITHMS AND HARDNESS RESULTS

Before considering the general case, we first devise a variety of algorithms for several special cases. The following hardness result motivates our consideration of special cases.

Theorem 3.1. *[1] For the overexposure problem with a budget k, it is NP-complete to decide whether there exists a set S of at most k nodes such that $\pi(S) > 0$.*

Since it is NP-complete to decide the sign of the optimal payoff in the budgeted case, we get the following corollary.

Corollary 3.2. *[1] It is NP-hard to approximate the budgeted overexposure problem.*

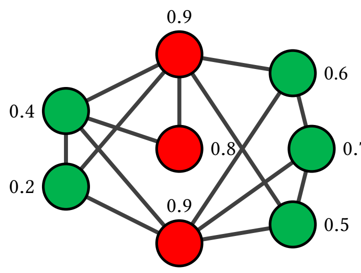

Figure 1: An example with a product appeal of 0.7**. Each node is labeled with its criticality parameter. The green nodes are accepting nodes and red are rejecting.**

This result is in direct contrast with much of the influence maximization literature. For example, the greedy hill-climbing search algorithm has an approximation guarantee for the independent cascade model [16]. One reason the overexposure problem is hard to even approximate is that in this problem, the payoff function does not have the submodularity property. As noted in Section 2, the payoff can even swing up and down as we select more seed nodes. This warrants looking into special cases for approximation algorithms, dynamic programming algorithms, linear programming solutions, and heuristics. After that, we consider the general case and provide a linear programming solution and several fast and effective heuristics, mostly of the greedy flavor.

3.1 Cluster Graph Formulation

The example shown in Fig. 1 illustrates a distinctive network property: All the accepting nodes that are reachable from each other form a cluster of accepting nodes (in short, *cluster*). The implication is that whenever we are able to reach just one accepting node within a cluster, we are able to reach the whole cluster.

We next construct a *cluster graph*. Each node in a cluster graph represents a cluster of accepting nodes of the original graph instance. Each edge (u, v) in a cluster graph represents all of the rejecting nodes that are adjacent to both an accepting node in the cluster represented by u and an accepting node in the cluster represented by v. The resulting cluster graph has weights for both nodes and edges. The weight of a node u in a cluster graph is the number of accepting nodes in its corresponding cluster, and the weight of an edge (u, v) is the number of rejecting nodes it represents. Note that our definition of cluster graph is different from AAK's [1].

Fig. 2 shows an example where the original instance of the overexposure problem is very complex, but the resulting cluster graph is basically a line graph.

To construct the cluster graph, we do an exhaustive BFS traversal of the graph and identify all the reachable accepting nodes from an accepting node without passing through any rejecting node. That is, if we reach a rejecting node, we stop the traversal from that node

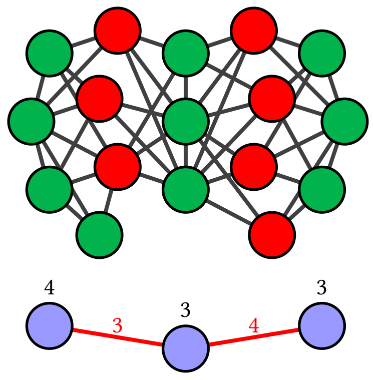

Figure 2: The top network shows the original overexposure instance, where the green nodes are accepting and red are rejecting. The bottom one shows the corresponding cluster graph. Each node of the cluster graph represents a cluster of accepting nodes and each edge represents a wall of rejecting nodes separating two clusters of accepting nodes. The nodes and edges of the cluster graph are weighted accordingly.

and add that node to the list of rejecting nodes connected to the cluster. Once the BFS finishes, we create the cluster graph using two pieces of information: (1) all the clusters of accepting nodes we have identified, and (2) the list of rejecting nodes at the boundary of each cluster of accepting nodes. In particular, for constructing an edge in the cluster graph, we compute the rejecting nodes that are at the shared boundary between two clusters of accepting nodes. A special case arises when a rejecting node is connected to only one cluster of accepting nodes. We call such rejecting nodes *orphaned*. There is no way of representing orphaned rejecting nodes by an edge in the cluster graph (which we build as a simple graph without any self-loop). In this case, we subtract the weight of the cluster-graph node by the number of such orphaned rejecting node.

The motivation behind constructing the cluster graph is two-fold. First, seeding a node in the original overexposure instance is the same as seeding the cluster to which that node belongs. In either case, we reach the same number of accepting nodes, and that number is stored as the node weight in the cluster graph. Henceforth, we will think in terms of seeding the nodes of the cluster graph.

Second, for any cluster-graph node, if we sum up the weights of the edges incident on that node, we obtain the total number of rejecting nodes reached if we seed that node. However, things get complicated when the same rejecting node is at the boundary of more than two clusters. This would lead to accounting errors in terms of double counting the same rejecting node because the edges of a cluster graph only store the number of rejecting nodes, not the identities of the rejecting nodes. To address this, we make the following assumption.

Assumption 3.1. *A rejecting node cannot be at the boundary of more than two clusters of accepting nodes.*

With Assumption 3.1, we can restate the payoff function for the overexposure problem within the context of a cluster graph. Let w_i and w_e be the weights of node i and edge e in the cluster graph, respectively. Following is the payoff for seeding a set S of cluster-graph nodes.

$$\pi(S) = \sum_{i \in S} w_i - \sum_{e: e \text{ has an endpoint in } S} w_e .$$

3.2 Unweighted Cluster Graphs: Hardness and Approximation

Despite Assumption 3.1, we next show that the problem remains NP-hard even for very simplistic instances of cluster graphs, such as unweighted cluster graphs (i.e., $w_i = 1$ and $w_e = 1$ for all nodes i and edges e).

Theorem 3.3. *It is NP-hard to find a set S of k nodes in the cluster graph to maximize the payoff $\pi(S)$, even when the cluster graph is unweighted.*

To prove the theorem, we first state a well-known property of any undirected graph. Denote the number of edges with at least one endpoint in S by $\#incident(S)$, the sum of the degrees of the nodes in S by $DegreeSum(S)$, and the number of edges in the subgraph induced by S by $\#induced(S)$. Furthermore, $degree(v)$ denotes the degree of a node v.

Property 3.1. *Let S be a set of nodes in an undirected graph.*

$$\#incident(S) = DegreeSum(S) - \#induced(S).$$

Proof. Each edge in the subgraph induced by S contributes 2 to $DegreeSum(S)$. Each edge that is outside of the induced subgraph but still has one endpoint in S contributes 1 to $DegreeSum(S)$, and there are $\#incident(S) - \#induced(S)$ such edges. Therefore, $DegreeSum(S) = 2 \times \#induced(S) + \#incident(S) - \#induced(S)$. □

Proof. (Thm 3.3) First, observe that in the overexposure problem on an unweighted cluster graph, the payoff for selecting a set S of k nodes is $\pi(S) = k - \#incident(S)$. Therefore, maximizing the payoff is equivalent to minimizing $\#incident(S)$. We next show that it is NP-hard to find a set S of k nodes to minimize $\#incident(S)$ in a cluster graph.

The reduction is from the known NP-complete problem of deciding whether there is a k-clique in an r-regular graph. Given any instance of that problem, we make a cluster graph instance with the same graph and the same k. We show that there is a k-clique in the clique instance if and only if there is a set S of k nodes in the cluster graph with $\#incident(S) \leq kr - \binom{k}{2}$.

The only if direction follows from Property 3.1. For the reverse direction (if), suppose that there is a set S of k nodes in the cluster graph such that $\#incident(S) \leq kr - \binom{k}{2}$, but there is no k-clique. When there is no k-clique, then for any set T of k nodes, $\#induced(T) < \binom{k}{2}$, which implies that $\#incident(S) > kr - \binom{k}{2}$ (by Property 3.1). This is a contradiction. □

As argued above, maximizing the payoff by selecting a set S of k nodes is the same as minimizing $\#incident(S)$. Therefore, we can treat the problem as minimizing the cost of selecting S, defined as $\#incident(S)$. This leads to a simple approximation algorithm.

Theorem 3.4. *There exists a 2-approximation algorithm for minimizing the cost of selecting k nodes in an unweighted cluster graph.*

Proof. We show that selecting a set $U = \{u_1, ..., u_k\}$ of the k lowest degree nodes has a cost at most $2 \times$ OPT, where OPT is the minimum cost due to selecting a set $W = \{w_1, ..., w_k\}$. Using Property 3.1, $DegreeSum(W) = \#incident(W) + \#induced(W) \leq 2 \times \#incident(W)$. Assume the elements in U and W are sorted by degree in non-descending order. The cost of selecting $U \leq DegreeSum(U) = \sum_{i=1}^{k} degree(u_i) \leq \sum_{i=1}^{k} degree(w_i) \leq 2 \times \#incident(W) = 2 \times \text{OPT}$. □

Related Graph Theory Problems. Dense and sparse subgraph problems have received a great deal of attention over the last few decades. These problems are closely related to overexposure on unweighted cluster graphs. Goldschmidt and Hochbaum [11] investigate the problem of finding a maximum set of nodes such that the number of incident edges is at most k. They give a 3-approximation for this problem in the weighted case. This approximation algorithm has been improved to 2-approximation by Gandhi et al. [10], who also study another related problem that asks to find the minimum number of edges such that at least k vertices are touched by these edges. See [20] and references therein for other related problems.

3.3 Weighted Cluster Graphs: Trees

We start with weighted stars as a special case.

Polynomial-Time Algorithm for Stars.

Theorem 3.5. *There exists a polynomial-time algorithm for solving the budgeted overexposure problem for weighted cluster graphs that are stars.*

Proof. Let v be the hub node of the star. In the optimal solution, we can either select v or not select v.

For the case of selecting v, all the edges of the star graph must also be selected. That is, all rejecting nodes at the boundary between the hub and any spoke is reached. We then select $k - 1$ highest weight nodes among the spokes of the star and calculate the final payoff. For the case of not selecting v, we greedily select at most k spokes. For each spoke we calculate its node weight and subtract its edge weight from it. We select the most profitable spokes up to k spokes (in order to avoid negative payoff). □

We next go beyond stars and consider weighted cluster graphs that are tree-structured. We present an optimal dynamic programming (DP) algorithm and a Knapsack-inspired heuristic for this case. We first present the DP algorithm.

DP Algorithm for Trees. We describe the idea of the DP algorithm without the very lengthy pseudocode. Consider a subtree rooted at v. Suppose we can allocate at most t seeds at this subtree. At node v, there are two alternatives: (1) select v, or (2) do not select v. We explore both alternatives as follows.

When we select v, we can allocate at most $t - 1$ seeds to the next-level subtrees (i.e., subtrees rooted at the children of v). We explore all possible ways of allocating a total of $t - 1$ seeds among these subtrees and get the payoff for each possibility recursively. For the second alternative of not selecting v, we can allocate at most t seeds to the subtrees rooted at the children of v. Again, we explore all possible seeding and get the payoff for each possibility. At node v, we calculate the best payoff for each of the two alternatives while making sure that no edge is double counted.[1] For each alternative (select v or not), we record a *witness vector* for the optimal payoff that consists of the following information for each child of v: (1) Are we selecting that child of v? (2) How many seeds are allocated to the subtree rooted at that child? Due to the recursive nature of this phase of computation, information flows upstream: from the leaves to the root. We implement memoization to avoid recomputation.

At the root node, for $t = 0, ..., k$, we compare the payoffs for the two alternatives of selecting the root or not selecting it when we seed t nodes in the whole tree. We select the best seeding level $t \le k$ and output whether or not we select the root. We also look up the corresponding witness vector and follow the information in the witness vector. We repeat this to output an optimal solution for the tree case.

The computational bottleneck of this DP algorithm is due to exploring all possible ways of allocating a certain number of seeds to the next-level subtrees as described two paragraphs ago. This is a well-known combinatorial problem known as the *stars and bars problem*. For k seeds and Δ being the maximum number of children of any node, there are $\binom{k+\Delta-1}{\Delta-1}$ or $O\left(\frac{e(n+k-1)}{k-1}\right)^{\Delta-1}$ ways of distributing k seeds among Δ subtrees. Therefore, we have the following result.

Theorem 3.6. *The DP algorithm for solving budgeted overexposure problem runs in polynomial time for tree-structured cluster graphs of bounded degree.*

Knapsack-Inspired Heuristic. We devise a fast heuristic based on the well-known dynamic programming (DP) solution to the Knapsack problem. In this heuristic, we first perform a topological sorting of the nodes of the tree so that any descendent would come before ancestors. We then select at most k nodes to "maximize" the payoff in the usual Knapsack DP fashion: For each node in the ordering and for each seeding level $\le k$, we compute the "optimal" solution up to that node. At the last node, we compare the payoffs for all seeding levels and choose the best one. We then trace back to compute the solution.

Although this heuristic is fast with a running time of $O(kn)$, we have examples showing that it does not compute the optimal solution. The main reason is that due to the ordering of the nodes, information coming from an earlier subtree may "clog" or "constrict" the seeding in another subtree.

3.4 Weighted Cluster Graphs with Cycles

We present an integer linear program (ILP) solution for weighted cluster graphs with cycles. We have the following setup: n is the number of nodes in the cluster graph, m is the number of edges in the graph, and k is the maximum number of seed nodes. In the ILP, $x_i = 1$ if we select node i, and $x_i = 0$ otherwise. The ILP makes sure that $y_e = 1$ whenever one of the endpoints of the edge e is selected. We denote the set of edges incident on a node i as $I(i)$. Finally, w_i and w_e denote the weights of node i and edge e, respectively. We have the following ILP.

[1]For example, for the case of selecting v, we also select all edges connecting v to v's children. In that case, if a child u of v is also selected as part of the optimal solution, we subtract the weight of (v, u) from the payoff at v so that it is not double counted.

ILP for Cluster Graphs With Cycles.

$$\begin{array}{lll} \text{Max} & \sum_{i=1}^{n} x_i w_i - \sum_{e=1}^{m} y_e w_e & \\ \text{s. t.} & & \\ & \sum_{i=1}^{n} x_i \le k & \\ & x_i \le y_e, & \forall i \text{ and } \forall e \in I(i) \\ & x_i \in \{0,1\}, & \forall i \\ & y_e \ge 0, & \forall e \end{array}$$

3.5 General Overexposure Instances

We now turn to the most general problem instances, which are provably hard. To address general instances, we inevitably have to move away from the cluster graph formulation because we want to relax Assumption 3.1. Due to the value of a compact representation, we formulate a *bipartite graph* corresponding to the original instance. We take each cluster of accepting nodes and add it as a single node, called *cluster node*, to the bipartite graph. We also add each rejecting node as a node to the bipartite graph. We then add directed edges from each rejecting node to all the cluster nodes for which the rejecting node is at the boundary of the cluster in the original instance. Put simply, rejecting nodes are now given *their own node in the bipartite graph*, whereas previously in the cluster graph, they were incorporated in the edges. Fig. 3 illustrates the translation of the original instance to a bipartite graph. It can be verified that the bipartite graph formulation preserves all the necessary information for computing an optimal solution.

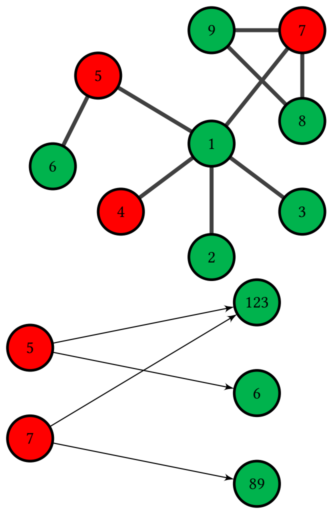

Figure 3: Bipartite graph (bottom) derived from original instance (top). Accepting nodes are green, rejecting nodes red. The rejecting node 4 is orphaned and is internalized using the same procedure described in Section 3.1.

We next present an ILP-based optimal solution for general instances. The ILP works on the bipartite graph $G = (V_R, V_C, E)$, where V_R is the set of all m rejecting nodes, V_C is the set of all n cluster nodes, and E is the set of directed edges. In this ILP, y_r represents whether a rejecting node r is counted due to selecting an adjacent cluster node.

ILP for General Instances.

$$\begin{array}{lll} \text{Max} & \sum_{i=1}^{n} x_i w_i - \sum_{r=1}^{m} y_r & \\ \text{s. t.} & & \\ & \sum_{i=1}^{n} x_i \leq k & \\ & x_i \leq y_r, & \forall\ (r,i) \in E \\ & x_i \in \{0,1\}, & \forall i \in V_C \\ & y_r \geq 0, & \forall r \in V_R \end{array}$$

We next present a series of greedy heuristics for the general instances. Although these are described for the bipartite graph setting, we have implemented these for cluster graphs as well.

Strawman Greedy. This strawman heuristic simply selects the k highest-weight clusters nodes. It does not take into consideration the rejecting nodes connected to these clusters.

Edge-Aware Greedy. This heuristic adds cluster nodes to the seed set one at a time, optimizing the payoff at each iteration in isolation. Essentially, the first iteration picks a single cluster-node i with the highest payoff, where the payoff of i is w_i- number of rejecting nodes adjacent to i. Then, that node is removed from the graph along with all of its incoming edges and its adjacent rejecting nodes (to avoid double counting). The algorithm continues this way for k iterations, returning a set of greedily picked seed nodes. It runs in $O(kn\Delta)$ time, where Δ is the maximum indegree of a cluster node.

Forward-Thinking Greedy. This heuristic uses one-iteration forward thinking by selecting a cluster node that would be advantageous for the next iteration. For example, when the algorithm evaluates a cluster-node i, it considers the combined effect of selecting every other cluster-node j (not previously selected) along with i. At each iteration, it selects the best cluster node in this fashion and then removes the selected node together with all the rejecting nodes adjacent to it. It runs in $O(kn^2\Delta)$ time.

4 EXPERIMENTS

We have extensively tested the algorithms designed for the three classes of our problem: (1) tree-structured weighted cluster graphs, (2) weighted cluster graphs with cycles, and (3) general instances.

4.1 Data

Our suite of networks includes synthetically generated network instances as well as real-world data from the Stanford Large Network Dataset (SNAP). Following is a brief snapshot.

(1) Barabasi-Albert networks: These networks incorporate two important aspects of real-world networks: dynamic growth and preferential attachment. Although these networks can model power-law degree distributions, their clustering properties are not that strong.
(2) Erdos Renyi random graphs: An Erdos-Renyi graph $G(n,p)$ is a random graph with n nodes where each possible pair of nodes is connected by an edge with probability p. These graphs are known for their small-world properties but fall short when it comes to either clustering or real-world degree distribution.
(3) Watts-Strogarz small-world networks: This model produces networks with small-world properties, exhibiting short average path lengths and high clustering. The generation process is usually known as *rewiring*. Unfortunately, the rewiring process cannot generate power-law degree distribution.
(4) Facebook network: This dataset consists of "circles" (or "friends lists") from Facebook, collected from survey participants using the Facebook app. The dataset includes node features (i.e., profiles), circles, and ego networks [19]. This network consists of 4,039 nodes and 88,234 edges.

For the synthetic networks, we generate 25 instances each for varying model parameters. We vary the number of nodes among 500, 1000, 2000, and 5000. We test with three different product appeals: 0.25, 0.5, and 0.75. The criticality parameters of the nodes are uniformly distributed between 0 and 1.

4.2 Creating Instances for Different Classes

While the synthetic and real-world networks can be easily transformed into bipartite graph instances for the general case of our problem, devising instances for the cluster graphs (with or without cycles) need further processing.

For cluster graphs with cycles, we first make sure that Assumption 3.1 is satisfied. If a rejecting node is connected to more than two clusters of accepting nodes, we remove edges from the original instance to make sure that it is connected to two clusters only. We then create a cluster graph instance.

Now, to create instances of cluster graphs without cycles (i.e., trees), we first create a cluster graph from the original instance without any consideration for Assumption 3.1. For this, we compute a maximum spanning tree for the cluster graph using Kruskal's algorithm. This method discards some edges, but tries to preserve the edges with large weights in order to guarantee a maximum weight tree. Once we arrive at a cluster graph without any cycle, it must be the case that Assumption 3.1 is satisfied (otherwise, there would have been a cycle among the three or more clusters that share the same rejecting node).

For each original instance of synthetic network, we generate an instance of a cluster graph with cycles and an instance of cluster graphs without cycles. Therefore, for either case, we have 25 instances for each configuration of model parameters.

4.3 Shorthand Notation

We use the following shorthand notation throughout this section.

BA	Barabasi-Albert preferential attachment networks
ER	Erdos-Renyi random graphs
FB	Facebook network
WS	Watts-Strogatz small-world networks

Clus-ILP	ILP for cluster graphs with cycles
Edge-Gr	Edge-aware greedy algorithm for the general case
Forw-Gr	Forward-thinking greedy algorithm for the general case
Gen-ILP	ILP for the general case
Knap-Gr	Knapsack-inspired heuristic for tree cluster graphs
Straw-Gr	Strawman greedy algorithm for the general case

4.4 Payoff Comparison

We first note that all algorithms for the general case can be applied to the case of cluster graphs with cycles, and all algorithm for the latter can be applied to tree-structured cluster graphs. Here, we mostly compare performances of the algorithms for the tree cluster graph case so that we can compare the widest array of algorithms. We have extensively replicated these experiments for the other classes of our problem and have not found any remarkable qualitative differences. Regarding implementation, we have used Python in general and the PuLP library for the ILPs.

Fig. 4 shows a comparison of among the algorithms for different types of networks. Across the board, the two greedy algorithms—edge-aware and forward-thinking—closely approximate the optimal solution obtained by the ILPs. We should note here that we could not run our DP algorithm for these large instances (5000 nodes in each synthetic network) because its running time goes off the chart. For smaller instances, however, DP finishes quickly and matches the optimal solutions of ILPs, as expected. As a side note, the Clus-ILP matches Gen-ILP because both are optimal for tree cluster graphs.

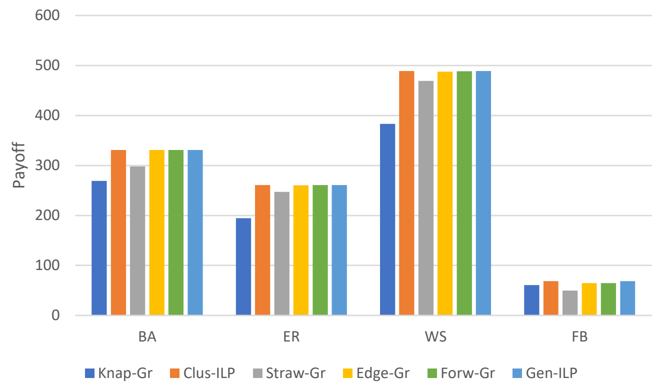

Figure 4: Comparison of payoff values. The plot shows that the greedy algorithms like the forward-thinking greedy closely approximate the optimal solution calculated by the ILPs. Here, the type of instances is tree-structured cluster graphs. Model parameters: product appeal of 0.5, 5000 nodes in each synthetic network, 100 seed nodes.

4.5 Running Time Comparison

We have used a computer with Intel Core i7-8750H processor and 16GB of RAM. Fig. 5 compares the running times of different algorithms based on the tree cluster graph case. We see similar behavior in other classes of problem instances. Not surprisingly, forward thinking greedy is the slowest, albeit faster than DP, which is not shown here. The reason is that it uses a look ahead and this look ahead operation necessitates an additional computation compared to other greedy algorithms. Interestingly, the ILPs run very fast on the synthetic network instances.

4.6 Effects of Varying Seed Set Size

As shown in Fig. 6, increasing the seed set size increases the payoff in general. This behavior does not necessarily ease the theoretical concern of payoffs fluctuating up and down as we select more seed nodes. This is because the algorithms select at most k seeds and are not required to select exactly k seeds. Interestingly, for this reason, the Knap-Gr stagnates after selecting around 70 seeds.

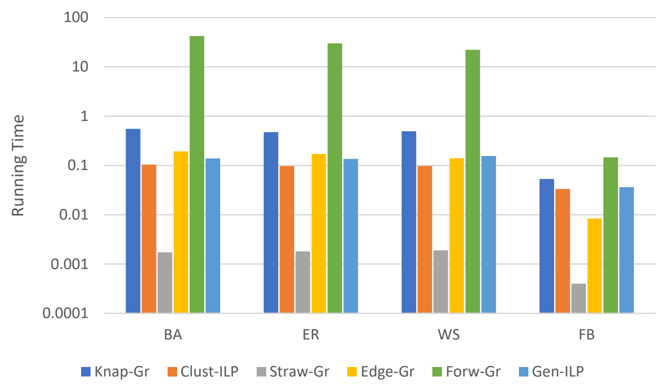

Figure 5: Running time comparison. The plot shows that the forward-thinking greedy is the slowest. More interestingly, the ILPs are fast in practice. On synthetic networks, they closely match the running time of the edge-aware greedy algorithm. The type of instances is tree-structured cluster graphs. Model parameters: product appeal of 0.5, 5000 nodes in each synthetic network, 100 seed nodes.

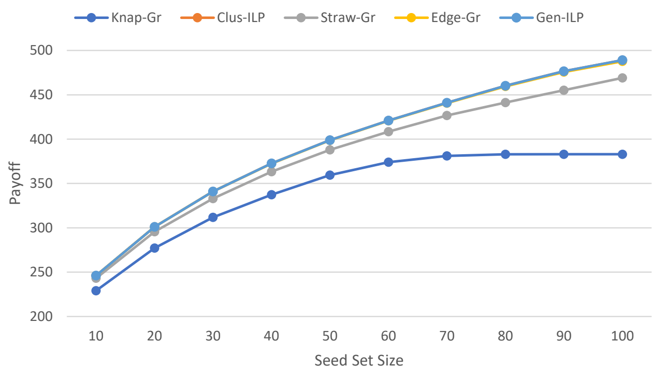

Figure 6: The effects of varying seed set size for Watts-Strogatz instances within the class of tree-structured cluster graphs. On the top line, Gen-ILP and Clus-ILP completely overlap and are closely matched by Edge-Gr. Model parameters: product appeal of 0.5, 5000 nodes in total.

4.7 Effects of Network Structures

When greedy and ILP algorithms are run on general instances, it turns out that the strawman greedy performs quite well on Barabasi–Albert networks compared to Watts–Strogatz networks, selecting seeds nearly optimally at a fraction of the runtime. This is shown in Fig. 7. This is due to the difference between the two network structures. The existence of high degree nodes, also known as "celebrity" nodes, in Barabasi–Albert networks means that these networks lead to a few very large clusters in addition to many small clusters. This lends itself quite well to a simple greedy approach. In contrast, we have observed that Watts–Strogatz networks lead to many medium and small sized clusters comparatively. This warrants more intelligent seed selection. [2]

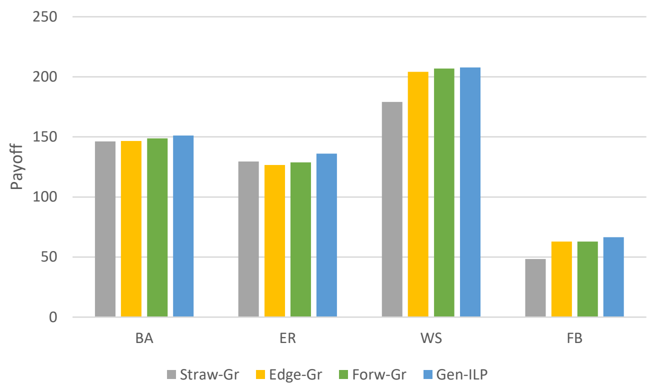

Figure 7: Comparison of payoffs. The model parameters are the same as Fig. 4, except that this plot is for the general instances whereas Fig. 4 is for trees. Straw-Gr works better for BA than for WS due to cluster size distribution.

4.8 Effects of Product Appeal

As shown in Fig. 8, for low product appeal (0.25)—meaning low quality product—the strawman greedy algorithm performs the poorest, followed by edge-aware greedy. When we allow negative payoffs, both of these algorithms come up with negative payoffs. On the other spectrum, when the product appeal is high (0.75)—meaning high quality product—these greedy algorithms perform quite well. Although Fig. 8 shows the case of Watts-Strogatz networks, we observe similar behavior for the other types of networks. Not surprisingly, Fig. 9 shows that all else remaining same, lower product quality demands more computational time, which is due to the emergence of a high number of rejecting nodes fragmenting the network into many clusters. These observations lead to an interesting finding:

Low quality products require smarter algorithms to avoid overexposure, whereas high quality products can be coupled with extremely simple and fast greedy algorithms.

5 CONCLUSION

In this paper, we have investigated the problem of budgeted overexposure problem. We have shown that the problem is NP-hard even for a very restricted case. We have presented an assortment of approximation algorithms, polynomial-time algorithms, heuristics, and linear programming solutions for several classes of the problem. We have also performed an extensive experimental study based on synthetic networks like Barabasi-Albert preferential attachment networks, Erdos-Renyi random graphs, and Watts-Strogatz small-world networks as well as real-world networks. We have investigated how model parameters and network properties impact the performance of our algorithms. Extending the AAK model [1] to more complex models like the linear threshold model and independent cascade model is an interesting future direction, particularly because the problem is provably hard even for very simple models.

[2]We should note that the strawman greedy is not that effective for tree cluster graphs. This is because the graph structure is fundamentally altered for creating the tree cluster graphs, as described in Section 4.2.

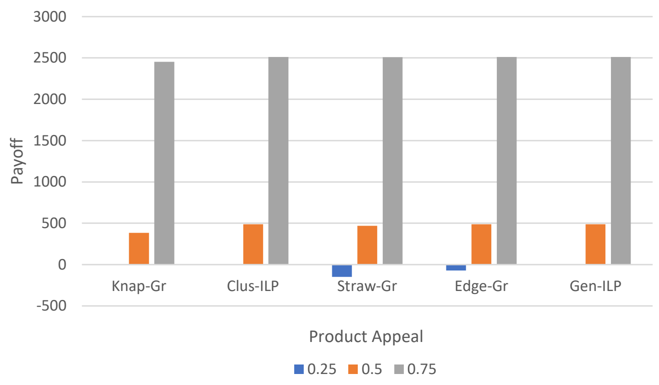

Figure 8: The effects of varying product appeals on payoff. This plot is based on Watts-Strogatz instances within the class of tree-structured cluster graphs. Model parameters: product appeal of 0.5, 5000 nodes in total.

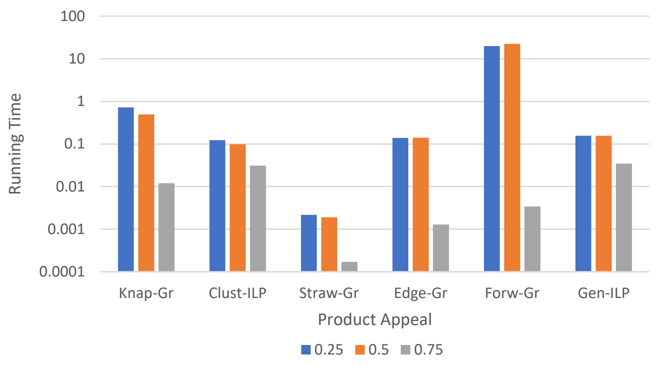

Figure 9: The effects of varying product appeals on running time. This plot is based on Watts-Strogatz instances within the class of tree-structured cluster graphs. Model parameters: product appeal of 0.5, 5000 nodes in total.

ACKNOWLEDGMENTS

We thank Drs. Cibele Freire and Eric Ramos of Bowdoin for their help with the graph theory part and the reviewers for helpful suggestions. We are also grateful for support from NSF Award IIS-1910203.

REFERENCES

[1] Rediet Abebe, Lada A. Adamic, and Jon Kleinberg. 2018. Mitigating Overexposure in Viral Marketing. In *Thirty-Second AAAI Conference on Artificial Intelligence*. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17343

[2] John W. Byers, Michael Mitzenmacher, and Georgios Zervas. 2012. Daily deals: prediction, social diffusion, and reputational ramifications. In *Proceedings of the fifth ACM international conference on Web search and data mining (WSDM '12)*. Association for Computing Machinery, New York, NY, USA, 543–552. https://doi.org/10.1145/2124295.2124361

[3] John W. Byers, Michael Mitzenmacher, and Georgios Zervas. 2012. The groupon effect on yelp ratings: a root cause analysis. In *Proceedings of the 13th ACM Conference on Electronic Commerce (EC '12)*. Association for Computing Machinery, New York, NY, USA, 248–265. https://doi.org/10.1145/2229012.2229034

[4] Wei Chen, Chi Wang, and Yajun Wang. 2010. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '10)*. Association for Computing Machinery, New York, NY, USA, 1029–1038. https://doi.org/10.1145/1835804.1835934

[5] Wei Chen, Yajun Wang, and Siyu Yang. 2009. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '09)*. Association for Computing Machinery, New York, NY, USA, 199–208. https://doi.org/10.1145/1557019.1557047

[6] Wei Chen, Yifei Yuan, and Li Zhang. 2010. Scalable Influence Maximization in Social Networks under the Linear Threshold Model. In *2010 IEEE International Conference on Data Mining*. 88–97. https://doi.org/10.1109/ICDM.2010.118 ISSN: 2374-8486.

[7] Gennaro Cordasco, Luisa Gargano, and Adele Anna Rescigno. 2019. Dual Domination. In *Combinatorial Algorithms (Lecture Notes in Computer Science)*, Charles J. Colbourn, Roberto Grossi, and Nadia Pisanti (Eds.). Springer International Publishing, Cham, 160–174. https://doi.org/10.1007/978-3-030-25005-8_14

[8] Fang Cui, Hai-hua Hu, Wen-tian Cui, and Ying Xie. 2018. Seeding strategies for new product launch: The role of negative word-of-mouth. *PLOS ONE* 13, 11 (Nov. 2018), e0206736. https://doi.org/10.1371/journal.pone.0206736 Publisher: Public Library of Science.

[9] Pedro Domingos and Matt Richardson. 2001. Mining the Network Value of Customers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California) *(KDD '01)*. Association for Computing Machinery, New York, NY, USA, 57–66. https://doi.org/10.1145/502512.502525

[10] Rajiv Gandhi, Samir Khuller, and Aravind Srinivasan. 2004. Approximation algorithms for partial covering problems. *Journal of Algorithms* 53, 1 (Oct. 2004), 55–84. https://doi.org/10.1016/j.jalgor.2004.04.002

[11] Olivier Goldschmidt and Dorit S. Hochbaum. 1997. k-edge subgraph problems. *Discrete Applied Mathematics* 74, 2 (April 1997), 159–169. https://doi.org/10.1016/S0166-218X(96)00030-3

[12] Sanjeev Goyal, Hoda Heidari, and Michael Kearns. 2019. Competitive contagion in networks. *Games and Economic Behavior* 113 (2019), 58–79. https://doi.org/10.1016/j.geb.2014.09.002

[13] Xinran He, Guojie Song, Wei Chen, and Qingye Jiang. 2012. Influence Blocking Maximization in Social Networks under the Competitive Linear Threshold Model. In *Proceedings of the 2012 SIAM International Conference on Data Mining (SDM)*. Society for Industrial and Applied Mathematics, 463–474. https://doi.org/10.1137/1.9781611972825.40

[14] Shankar Iyer and Lada A. Adamic. 2019. The Costs of Overambitious Seeding of Social Products. In *Complex Networks and Their Applications VII (Studies in Computational Intelligence)*, Luca Maria Aiello, Chantal Cherifi, Hocine Cherifi, Renaud Lambiotte, Pietro Lió, and Luis M. Rocha (Eds.). Springer International Publishing, Cham, 273–286. https://doi.org/10.1007/978-3-030-05414-4_22

[15] Shankar Iyer and Lada A. Adamic. 2019. When can overambitious seeding cost you? *Applied Network Science* 4, 1 (Dec. 2019), 1–24. https://doi.org/10.1007/s41109-019-0146-z Number: 1 Publisher: SpringerOpen.

[16] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '03)*. Association for Computing Machinery, New York, NY, USA, 137–146. https://doi.org/10.1145/956750.956769

[17] Yanhua Li, Wei Chen, Yajun Wang, and Zhi-Li Zhang. 2013. Influence diffusion dynamics and influence maximization in social networks with friend and foe relationships. In *Proceedings of the sixth ACM international conference on Web search and data mining (WSDM '13)*. Association for Computing Machinery, New York, NY, USA, 657–666. https://doi.org/10.1145/2433396.2433478

[18] Grigorios Loukides, Robert Gwadera, and Shing-Wan Chang. 2020. Overexposure-Aware Influence Maximization. *ACM Transactions on Internet Technology* 20, 4 (Oct. 2020), 39:1–39:31. https://doi.org/10.1145/3408315

[19] J. McAuley and J. Leskovec. [n.d.]. *Social circles: Facebook*. https://snap.stanford.edu/data/ego-Facebook.html

[20] Rémi Watrigant, Marin Bougeret, and Rodolphe Giroudeau. 2012. *The k-Sparsest Subgraph Problem*. report. https://hal-lirmm.ccsd.cnrs.fr/lirmm-00735713