Approximability of Adaptive Seeding under Knapsack Constraints

Aviad Rubinstein
UC Berkeley
aviad@cs.berkeley.edu

Lior Seeman
Cornell Univeristy
lseeman@cs.cornell.edu

Yaron Singer
Harvard University
yaron@seas.harvard.edu

May 6, 2015

Abstract

Adapting Seeding is a key algorithmic challenge of influence maximization in social networks. One seeks to select among certain available nodes in a network, and then, adaptively, among neighbors of those nodes as they become available, in order to maximize influence in the overall network. Despite recent strong approximation results [25, 1], very little is known about the problem when nodes can take on different activation costs. Surprisingly, designing adaptive seeding algorithms that can appropriately incentivize users with heterogeneous activation costs introduces fundamental challenges that do not exist in the simplified version of the problem.

In this paper we study the approximability of adaptive seeding algorithms that incentivize nodes with heterogeneous activation costs. We first show a tight inapproximability result which applies even for a very restricted version of the problem. We then complement this inapproximability with a constant-factor approximation for general submodular functions, showing that the difficulties caused by the stochastic nature of the problem can be overcome. In addition, we show stronger approximation results for additive influence functions and cases where the nodes' activation costs constitute a small fraction of the budget.

1 Introduction

Information diffusion is a fundamental process in which interpersonal interactions between individuals generate large word-of-mouth cascades in a social network. In many cases the dynamics of this system can be engineered by seeding important nodes in the network. The algorithmic challenge of selecting individuals who can serve as early adopters of a new idea, product, or technology in a manner that will trigger a large cascade in the social network is known as *influence maximization*. The problem was first posed by Domingos and Richardson [7, 24] and elegantly formulated and further developed by Kempe, Kleinberg, and Tardos [16], whose result showed that for natural classes of influence models, influence maximization can be reduced to maximizing a monotone submodular function under a cardinality constraint. Since this seminal work, a broad range of algorithmic methods have been developed for this problem [19, 21, 5, 20, 2].

Challenge in influence maximization. Although there has been substantial progress on the problem throughout the past decade, influence maximization remains a challenging task. In many applications, one only has access to a small *sample* of the network in which individuals have low influence potential, and algorithms that select their users from such samples are severely handicapped. In marketing applications, for example, merchants often reward users who visit their online store, or who have engaged in other ways (subscribe to a mailing list, follow the brand,

install an application etc.). If we think of such as being randomly sampled from the network, it follows that a high-degree user is a rare event simply because the degree distributions of social networks are heavy-tailed. Influence maximization techniques are based on selecting high-degree users (not necessarily the *highest* degree), and their application on such samples is therefore ineffective.

Adaptive seeding. To overcome the problem of low degree nodes a new approach to information dissemination in social networks was recently suggested in [25]. Rather than spend the entire budget on the nodes in the sample, one can spend a fraction of the budget on the sample in order for them to attract their friends, wait for the friends to become accessible, and optimize influence on the friends with the remaining budget. The idea is to leverage high degree neighbors that may be connected to the sample. The intuition for why neighbors may be more influential comes from Scott Feld's *friendship paradox* (*"your friends have more friends than you"*) [10]. Recent experiments in large online social networks support this claim and show that a large fraction of users have less friends than the average number of friends their friends have [29, 13]. Recent work shows that in general models of social networks, asymptotic gaps exist with constant probability [18]. This implies that applying two-stage approaches on graphs that have such properties can lead to substantial improvement over current techniques for influence maximization. This has been confirmed through experiments on the Facebook social network using implementation of scalable adaptive seeding algorithms [14].

The model. Adaptive-Seeding is a two-stage stochastic optimization problem. We are given a node-weighted bipartite graph which consists of a set X and its neighbors $\mathcal{N}(X)$; the node weights signify costs, and each node $v_i \in \mathcal{N}(X)$ is also associated with some probability p_i, which signifies the probability that v_i will be realized if one or more of its neighbors in X are selected. We are also given a budget B and a combinatorial function $f : 2^{\mathcal{N}(X)} \to \mathbb{R}$. In the first stage we select a subset of nodes $S \subseteq X$, which in turn will cause each of their neighbors v_i to be realized independently with probability p_i. In the second stage, we use the remaining budget $B - c(S)$ (where $c(S)$ denotes the cost of the selected set S) to choose a subset of the realized neighbors. The goal is to find a subset of nodes $S \subseteq X$ which can maximize the objective function of the second stage, in expectation over all realizations of neighbors of S. Beyond its original motivation, Adaptive-Seeding is a simple way to capture a fundamental kind of stochastic process: Given a distribution of the consequences of decisions we make in the present, which actions would lead to lucrative opportunities in the future?

Incentives in adaptive seeding. In many cases different individuals require different incentives to become activated. In practice, it may be that nodes with high influence potential require larger incentives to become activated.

Are good approximation guarantees obtainable in adaptive seeding when nodes have heterogeneous activation costs?

In the past, only the unweighted case of the problem (often called the case of *cardinality constraints*) has been considered, giving strong approximation guarantees for various classes of monotone submodular functions [25, 1, 14]. The weighted case where nodes can take on different costs to become activated captures a broad scope of applications, though is far more complex. As we now explain, the complexity of adaptive seeding is due to two components of the problem: the optimization over two layers captured by the bipartite graph, and its stochastic nature. Both these

components are aggravated tremendously in the weighted case, as discussed next. To be consistent with optimization literature we will refer to heterogeneous costs as adaptive seeding under *knapsack constraints*.

1.1 The difficulty in activation costs

Two-layers. In the cardinality case the difficulty in optimizing over two layers can be quite elegantly overcome by a simple algorithm that obtains an approximation arbitrarily close to $1 - 1/e$ for any monotone submodular function [1]; essentially matching that for the standard (single-stage) optimization problem of maximizing submodular functions. Knapsack constraints are much harder; in fact, it is easy to show that under knapsack constraints, even the very simple *cardinality functions* cannot be approximated within a factor better than $1 - 1/e$ unless P=NP.

Costs $\cap$ probabilities. The second difficulty comes from the fact that the known techniques for solving adaptive seeding largely rely on non-adaptive algorithms which (approximately) optimize the non-adaptive objective, meeting the budget in expectation over the probabilities of the second stage and the randomizations of the algorithm. Since the algorithm needs to a priori divide its budget between the first and second stage, before the realizations occur, the weights create havoc over the probabilistic calculations. Specifically, non-adaptive algorithms can lead to arbitrarily bad adaptive solutions as they may select high-cost nodes that appear with small probability, while not having enough budget to actually seed them when the realization of the second stage occurs.

1.2 Our results

In this paper we prove upper and lower bounds for Adaptive-Seeding under knapsack constraints. Our strong lower bound mathematically articulates the first of the two difficulties discussed (it applies to the deterministic case when all probabilities are one), while the upper bounds indicate that, despite all, something can be salvaged, and in some cases tight approximation bounds are achievable.

Tight lower bound for cover functions. Our lower bound applies to the case in which all probabilities are one. We show that, unless P = NP, no polynomial-time algorithm can obtain an approximation factor better than $1 - 1/e^{1-1/e}$, and that this bound is tight. It is interesting to contrast our hardness result with the algorithm in [1], which achieves a $(1 - 1/e)$-approximation in the case of cardinality constraints. For cardinality constraints one can overcome the difficulty of the two-layer problem by observing that in any computationally hard instance, *each node in the first layer covers a large number of second layer nodes.* Therefore spending small fractions of the budget on the first layer adds value while the lost budget is negligible for the second layer. When the nodes can take different costs, this idea breaks: the cost of a top layer node can be high, which can make it more efficient to spend the budget on the second layer.

Constant factor approximation for monotone submodular functions. Our second main result addresses the difficulty that arises from the stochastic nature of the problem under knapsack constraints. Our main result is a constant factor approximation for any monotone submodular function under knapsack constraints. Monotone submodular functions are fundamental in the context of influence maximization as they are arguably the most general class of functions that capture natural influence models [16, 21]. We first match the lower bound by giving a $(1 - 1/e^{1-1/e})$-approximation for any monotone submodular function for the non-adaptive relaxation of the problem. We then

construct a procedure which, given an approximation algorithm for the non-adaptive version of the problem returns an adaptive policy, with a loss of a constant factor. When costs are arbitrarily small in comparison to the budget, we can convert the non-adaptive algorithm to an adaptive algorithm which approaches a $(1 - 1/e)(1 - 1/e^{1-1/e})$ approximation ratio. This bound consists of the (optimal) non-adaptive approximation and a $(1-1/e)$-adaptivity gap, which is tight.

Additive functions. Additive functions are relatively simple, yet rich enough to include important classes of influence functions like the voter model [7, 8]. For additive functions we give a $(\frac{1-1/e}{2})$-approximation algorithm which can be improved to a tight $1-1/e$-approximation when second-stage costs are arbitrarily small.

1.3 The Adaptive Seeding Model

At this point, it is worth to pause and briefly discuss the adaptive seeding model. We add this discussion for completeness and similar exposes can be found in [25, 14, 18].

The adaptive seeding framework has been designed to allow effective application of influence maximization. In many applications the goal is to spread influence through individuals who express interest in an application or product. As an example, one can consider users who purchase goods through an online retailer. A standard assumption in the influence maximization literature is that the network and the influence function are known to the entity activating the cascade. This assumption is quite realistic as the network is often public data, or can alternatively be purchased through business agreements with the social network application. The models for the spread of influence are also assumed to be known (the parameters can be fit from cascade data) and the submodular function is a reasonable proxy for the spread of influence. Simple functions such as coverage or sum of degrees can be used as reasonable measures, and are both monotone submodular functions.

The key challenge that adaptive seeding addresses is that the retailer does not have immediate access to influential users. Although the retailer is assumed to have access to the entire social network, it cannot offer a reward to users who have not explicitly expressed interest in the product. This limitation is either due to privacy regulations of social networking applications, or simply due to social norms.

The adaptive seeding framework seeks to recruit the neighbors of core users (those who initially purchase the product or engaged with the retailer) to become influencers using the budget (free samples). Unlike the process of influence, the goal is not to try to influence neighbors to forward the information without incentives, and the assumption is that the neighbors have a standard bayesian utility model with no externalities. That is, every neighbor has a probability of being interested to become an early adopter, and this probability does not depend on the number of friends who also made a purchase. This is modeled by having probabilities on the nodes that can be activated if a node has a single neighbor in X that has been selected in the initial step by the algorithm. For submodular functions, without loss of generality, one can also consider models where the probabilities are on the edges, and every node in X selected by the algorithm activates the neighbor with the probability encoded on the edge. In this case the same adaptive seeding model can be used with a modified monotone submodular function.

1.4 Related work

The Adaptive Seeding model was introduced in [25], motivated by the question of influence maximization in social networks formalized by Kempe, Kleinberg, and Tardos [16]. The problem considered was under cardinality constraints, and the main result applies to a strict subclass of

submodular functions. In a recent paper [1], a different technique is introduced, called *locally-adaptive policies*. In the cardinality case, the gap between the optimal adaptive policy and the optimal non-adaptive policy can be shown to be $(1 - 1/e)$. For special cases of submodular functions a matching approximation exists, which together with the adaptivity gap of $(1 - 1/e)$ yields a $(1 - 1/e)^2$-approximation for the adaptive problem. For general submodular functions however, it seems that we cannot obtain a $(1 - 1/e)$-approximation to the non-adaptive policy under cardinality constraints. The focus of [1] is therefore to consider a different class of policies all together called *locally-adaptive policies*, which are shown to provide a $(1 - 1/e)^2$-approximation for any monotone submodular function. In this work, the above challenge is irrelevant. In the case of knapsack constraints, we obtain a tight approximation for non-adaptive policies that match the $\left(1 - 1/e^{1-1/e}\right)$-lower bound for the deterministic variant of the problem. Our main upper bound is therefore achieved via an algorithm for non-adaptive policies. The difficulty in obtaining the upper bound is due to the combination of knapsack constraints and the stochastic nature of the problem, which does not exist in the cardinality case.

Hardness of approximation. Our lower bound generalizes seminal techniques developed by Feige [9] who showed it is NP-hard to approximate MAX-COVER within a factor better than $1-1/e$.

Stochastic optimization. There has been extensive work on stochastic combinatorial optimization. The main models considered are two-stage with recourse minimization ([15], [23], [12], [26, 27]) and maximization of a stochastic objective under a budget constraint ([17], [6], [11]). In contrast, adaptive seeding is a two-stage model where the actions in the first stage affect the realizations in the second stage, and the two stages are tied together by the joint budget constraint.

Submodular Maximization. The classes of functions we focus on in this paper are *monotone submodular functions*. Nemhauser et al. [22] show a simple greedy algorithm that achieves a $(1-1/e)$-approximation for maximizing monotone submodular functions under cardinality constraints. For knapsack constraints, Sviridenko [28] shows that a slight variation on the greedy algorithm achieves the same approximation ratio. The notion of non-adaptivity we employ in this paper is closely related to multilinear extensions of submodular functions. We use results by Calinescu et al. [3] and Chekuri et al. [4] to bound the adaptivity gap between non-adaptive and adaptive policies.

2 Preliminaries

ADAPTIVE-SEEDING is a two stage process where in the first stage we are given an initial set of nodes $X \subseteq V$ which can be seeded, and in the second stage each neighbor v_i of the seeded nodes appears with some independent probability p_i. The input to the problem can be succinctly described through a bipartite graph $G = (V, E)$ that encodes the connections between X and its neighbors, denoted $\mathcal{N}(X)$, a value oracle to a combinatorial function $f : 2^{\mathcal{N}(X)} \to \mathbb{R}_+$, probabilities vector $\mathbf{p} \in [0, 1]^{|\mathcal{N}(X)|}$ describing the independent probabilities of every node realizing in the second stage, a cost function $c : X \cup \mathcal{N}(X) \to \mathbb{R}$ describing the cost of seeding a node in the first stage or selecting a node in the second stage once it realizes, and a budget B.

The goal of an adaptive policy is to select a subset $S \subseteq X$ of cost no larger than B in the first stage s.t. the function f is maximized under the remaining budget, in expectation over all possible realizations of neighbors of S. That is, for any $S \subseteq X$ let $S_R = \text{argmax}\{f(T) : T \subseteq R \cap \mathcal{N}(S), c(S) + c(T) \leq B\}$ where R is a realization of the neighbors

of X. For brevity we use p_R to denote the probability of R to realize. The optimal adaptive seeding solution is:

$$S^* := \text{argmax}_{S \subseteq X, c(S) \leq B} \Big\{ \sum_R p_R \cdot f(S_R) \Big\}.$$

3 The Lower Bound

In this section, we show a nontrivial (beyond $(1-1/e)$) hardness of approximation for the Adaptive Seeding problem. Furthermore, our hardness of approximation holds even in the special case that: (1) the instance is deterministic (i.e. the probabilities on all the nodes are set to 1); and (2) the submodular function we want to optimize is a coverage function. In particular, for deterministic two-stage seeding, our result is tight. Formally, we show that it is NP-hard to approximate the following problem:

Definition 3.1. Two-Stage Max-Cover with Knapsack *(2S-MCK) optimization problem consists of three sets: X, which we call the* top layer*, $Y = \mathcal{N}(X)$ is the* middle layer*, and a* bottom layer *Z of elements to be covered under knapsack constraints. (The top layer corresponds to the first stage nodes in* Adaptive-Seeding*, while the middle layer corresponds to the second stage). Let A_y be the sets of nodes in Z covered by $y \in Y$. The goal is to to solve the following restricted variant of* Adaptive-Seeding*:*

$$\max_{S,T} |\cup_{y \in T} A_y|$$
$$\sum_{x \in S} c(x) + \sum_{y \in T} c(y) \leq B$$
$$S \subseteq X, T \subseteq \mathcal{N}(S) \ .$$

We can now state our main hardness result.

Theorem 3.2. *It is NP-hard to approximate 2S-MCK within any factor better than $(1-1/e^{1-1/e}) \approx 0.469$.*

Notice that together with the results of Section 4, we have a tight characterization of the approximation bound for the non-adaptive relaxation of the problem defined in the following section, and in particular for the special case where there are no probabilities.

The rest of this section is devoted to the proof of Theorem 3.2. In Section 3.1, we expose our main idea: composing two instances of Max-Cover. As a warmup, we use this idea to obtain a weaker result, namely (≈ 0.502)-hardness for 2S-MCK. Then, in Section 3.2 we modify our construction and introduce a more delicate analysis of the Max-Cover problem. Together we obtain the tight (≈ 0.469)-hardness of approximation.

3.1 Warmup: 0.502-hardness of approximation

In this section we sketch a simpler yet somewhat weaker construction that gives a $(1-1/e)\left(1-e^{-e/(e-1)}\right) \approx 0.502$-hardness of approximation. This warmup is an important exposition of our first main idea: We show that the two-stage optimization problem is hard to approximate by composing two hard instances of Max-Cover - one for each stage. Informally, covering more elements in the first instance, corresponds to having access in the second stage to more copies of the second instance.

It is well known that it is NP-hard to cover more than $(1-1/e)$ of the optimal solution of Max-Cover, i.e. we would have access to $(1-1/e)$ of the copies of the second instance. However, this leaves us with more budget to spend on each copy. On average, the budget for each copy would be greater by a factor of $e/(e-1)$ than that spent by the optimal solution. It is not hard to show that this calculation gives a $(1-1/e)\left(1-e^{-e/(e-1)}\right) \approx 0.502$ hardness of approximation.

Theorem 3.3. *It is NP-hard to approximate 2S-MCK within any factor better than* $(1-1/e)\left(1-e^{-e/(e-1)}\right) \approx 0.502$.

Proof Sketch: Consider a hard instance of Max-Cover on (X, U) where X is a family of subsets of the universe U. X will coincide with the top layer of nodes in the construction.

The middle layer Y will consist of $|U|$ disjoint sets called *pieces*, $Y_1, \ldots, Y_{|U|}$. For each $x \in X$, $\mathcal{N}(x)$ is the union of all pieces corresponding to the elements in the set $x \subseteq U$.

Now for the final layer: The bottom layer Z also decomposes into disjoint pieces $Z_1, \ldots, Z_{|U|}$. For each $u \in U$, we have another hard instance of Max-Cover (V_u, W_u), where we identify the subsets V_u with the nodes in Y_u, and the elements W_u with the nodes in Z_u. This means that $y_{u,v}$ is in the neighborhood $\mathcal{N}(x)$ for every x that contains u and for every $v \in V_u$. Finally $z_{u,w}$ is in the neighborhood $\mathcal{N}(y_{u,v})$ whenever v contains w in instance (V_u, W_u). A visual description of the warmup construction appears in the appendix.

The budget is chosen such that when we compose two good instances of Max-Cover, we have exactly enough budget to seed enough nodes from X and Y to cover all the nodes in Y and Z, respectively. We set the costs $c(x), c(y)$ of the nodes such that, on a composition of two bad instances of Max-Cover, the optimal solution splits the budget between the first and second stage in the same proportions as does the optimal solution for two good instances (this step is described in more detail in Section 3.2).

Now, given bad instances of Max-Cover, we can cover nodes $y_{u,v} \in Y$ that correspond to at most $(1-1/e)$ of the subsets $u \in U$. This corresponds to having access to $(1-1/e)$ of the copies of the second instance of Max-Cover.

On each reachable copy, on average, we have $(e/(e-1))$-times more budget to spend than is required to cover all the nodes in a good instance. Therefore, by a refinement of Feige's hardness result (see Theorem 3.4 for more details), we can cover at most $\left(1-e^{-e/(e-1)}\right)$ of the nodes in each reachable copy. Multiplying by the number of reachable a copies, we get the factor $(1-1/e)\left(1-e^{-e/(e-1)}\right)$. □

3.2 Tight hardness of approximation

In this section, we improve our construction to obtain the tight $\left(1-1/e^{1-1/e}\right) \approx 0.469$ hardness of approximation. Intuitively, we construct an instance where the (easy-to-find) solution, cannot spread the budget evenly between the reachable copies of the second instance. In particular, any solution must, for each set in the first instance, spend the same budget across all the second-stage copies that correspond to that set.

Some elements in the first-stage instance are covered by more subsets than others. We show that in every solution, the budget spent on each second-stage instance is proportional to the number of subsets that cover the corresponding first-stage element. Therefore, *the budget is no longer distributed evenly across the copies of the second instance.*

Delicate analysis of Max-Cover

Feige [9] proved that it is NP-hard to approximate Max-Cover within any factor better than $(1-1/e)$. For our hardness result, in order to leverage on the added hardness of having a 3-layer

instance as opposed to the MAX-COVER bipartite graph, we want to compose copies of Feige's construction. It is not clear how to apply Feige's result as a black box. Instead, we explore some special characteristics of this construction.

Let us recall some properties of Feige's construction. In his hard instance of MAX-COVER, the universe can be decomposed into a large number of partition systems B_r for each $r \in R$. Each partition system consists of L partitions of k subsets each, for some large constant L. The different partitions in each system B_r are *orthogonal* in the sense that given subsets of different partitions, the relative size of their intersections is exactly the product of their relative sizes. In other words, the events that a random element is covered by each of those subsets are *independent*. Each available subset of the MAX-COVER instance, denoted $S_{(q,a,i)}$, corresponds to a union of subsets in partition system B_r, one for each r corresponding to (q, i).

For completeness, in a *good* instance it is possible to cover all the elements with k' disjoint subsets $S_{(q,a,i)}$. Recall that this corresponds to exactly k subsets for each partition system B_r.

On the soundness side, Feige proves that for any *bad* instance, and any choice of subsets, for all but $\epsilon/3$ of the partition systems B_r's, either more than $3k'/\epsilon$ of the subsets in B_r are covered, or no two subsets in the same partition are covered. Informally, in bad instances, the relative number of elements in any cover behaves like a union of independent events of probability $1/k'$ each.

The soundness above suffices to show that it is NP-hard to cover more than $(1 - 1/k')^{ak'} + \epsilon$ of the elements with budget ak' (which would suffice for our warmup construction). However, since the different partitions are exactly orthogonal, we can deduce the following stronger corollary, which also bounds the number of elements covered m times, for any constant m.

Theorem 3.4. *(Essentially [9]) For any constants $M, \epsilon > 0$, it is NP-hard to distinguish between:*

- ***Completeness:*** *an instance where it is possible to cover every element exactly once with budget k'; and*
- ***Soundness:*** *an instance where with budget ak', for every $m \leq M$, it is impossible to cover more than $\Pr[\mathbf{Pois}(a) \geq m] + \epsilon$ of the elements at least m times.*

The reduction

We are now ready to present our full construction for the proof of Theorem 3.2. The main difference in this construction as opposed to the warmup from Section 3.1, is in the role played by the middle layer, Y. In the warmup construction, each middle-layer node corresponds to a pair of: one element from the first instance of MAX-COVER, and one subset from the second instance; here, per contra, each middle-layer node corresponds to a pair of subsets, one from each instance of MAX-COVER. This forces the uneven budget spread across copies of the second instance of MAX-COVER.

Formally, we compose two hard instances of MAX-COVER. Let (X, U) be the first instance: i.e. X is a family of subsets of the universe U. X will coincide with the top layer of nodes in the construction. We denote the second instance of MAX-COVER by (V, W) (subsets V over universe W).

Observe that it is NP-hard to distinguish between a composition of two *good* instances of MAX-COVER (where all the elements in each instance can be covered) and a composition of two *bad* instances (i.e. ones where with the same budget only $(1 - 1/e)$ of the elements in each instance can be covered).

Now, each node $y_{x,v} \in Y$ in the middle layer corresponds to a pair of subsets $(x, v) \in X \times V$. Each middle-layer node $y_{x,v}$ is covered only by the corresponding top-layer node x. $y_{x,v}$ covers all the bottom-layer nodes $z_{u,w} \in Z$ that satisfy both $u \in x$ and $w \in v$.

For each $u \in U$, we denote by (V_u, W_u) the restriction of the entire instance to elements $\{z_{u,w} \colon w \in W\}$ and their parents. Each subset $v_u \in V_u$ is a subset of every (x, v) such that $u \in x$, and each element w_u corresponds to $z_{u,w}$. Thus the number of v_u's in any cover is equal to the number of covered (x, v)'s such that $u \in x$. In particular, if we spread the budget evenly across x's, this results in a skewed distribution of budget across (V_u, W_u)'s. A visual description of this construction appears in the appendix.

Finally set the costs $c(x), c(y)$ of the nodes, such that an optimal cover for the entire set costs aB on the top layer, and $(1-a)B$ on the middle layer, where a is some constant to be defined soon. Suppose that we can find a cover that spends bB on the top layer, and $(1-b)\,B$ on the middle layer.

Completeness: By the choice of parameters, if we composed two good instances of Max-Cover, then it is possible to cover all the bottom-layer nodes $z_{u,w} \in Z$ using budget B.

Soundness: Let M be some large constant. By Theorem 3.4, for any solution (S, T) on a composition of two bad instances of Max-Cover, and for any $m \leq M$, the number of u's for which S contains m or more top-layer nodes $x \ni u$, is at most $\Pr[\mathbf{Pois}(b/a) \geq m] + \epsilon$. For each $x \in S$, we are left with $\frac{1-b}{b}B$ budget, on average, to spend on middle-layer nodes $y_{x,v} \in \mathcal{N}(x)$. So on average, for each u covered by m of the x's in S, the optimal cover will contain $\frac{1-b}{b}mB$ subsets $v_u \in V_u$. Applying Theorem 3.4 again, we get that on the bad instance it is impossible to cover more than $\left(1 - \exp\left(-\frac{1-b}{b} \cdot \frac{a}{1-a} \cdot m\right) + \epsilon\right)$ of the elements $w_u \in W_u$.

Therefore, for a and b as above, it is NP-hard to approximate 2S-MCK to any factor better than

$$\begin{aligned} f(a,b) &= \sum_{m=0}^{\infty} \Pr[\mathbf{Pois}(b/a) = m] \left(1 - \exp\left(-\frac{1-b}{b} \cdot \frac{a}{1-a} \cdot m\right)\right) + 2\epsilon \\ &= \sum_{m=0}^{\infty} \frac{(b/a)^m}{m!} \exp^{-b/a} \left(1 - \exp\left(-\frac{1-b}{b} \cdot \frac{a}{1-a} \cdot m\right)\right) + 2\epsilon\,. \end{aligned}$$

Now, we could calculate what choice of a give us the strongest result. Instead, let $b(a)$ be the value of b that maximizes f given a. Since $b(a)$ is continuous on $[0,1]$, it must have a fixed point $a^* = b(a^*)$. Notice that when $a \to 0$, $b(a)$ is small, yet greater than a. Similarly, when $a \to 1$, $b(a) < a$. Therefore, a^* is some constant bounded away from 0 and 1. Set the costs such that $a = a^*$ to get NP-hardness for a factor of:

$$\begin{aligned} f(a^*, a^*) &= \sum_{m=0}^{\infty} \frac{\exp(-1)}{m!} (1 - \exp(-m)) + 2\epsilon \\ &= 1 - \sum_{m=0}^{\infty} \frac{\exp(-m)}{m!} \exp(1/e) \frac{\exp(-1)}{\exp(-1/e)} + 2\epsilon \\ &= 1 - \left(\sum_{m=0}^{\infty} \Pr[\mathbf{Pois}(1/e) = m]\right) \cdot \left(1/e^{1-1/e}\right) + 2\epsilon \\ &= 1 - 1/e^{1-1/e} + 2\epsilon. \end{aligned}$$

□

4 An Optimal Non-Adaptive Approximation Algorithm

In this section we present a $(1 - 1/e^{1-1/e})$-approximation algorithm for monotone submodular functions for the randomized-and-relaxed non-adaptive objective defined in [25]. Given the lower bound from the previous section, this algorithm is optimal for this non-adaptive objective. In the next section we show how to use the algorithm presented here to construct an approximation algorithm for the optimal *adaptive* policy. We start off with a few definitions.

Randomized-and-relaxed non-adaptive policies. As we mentioned in the introduction, our algorithm for ADAPTIVE-SEEDING relies on the solution of a fractional, non-adaptive relaxation to the problem. A *randomized-and-relaxed non-adaptive* policy (henceforth *non-adaptive* for short) is a set $S \subseteq X$ and vector $\mathbf{q} \in [0,1]^{|\mathcal{N}(X)|}$ which is strictly positive only if $v_i \in \mathcal{N}(S)$. The policy decides on $(S, \mathbf{q})$ prior to the realization of the second stage. In the first stage the set S is selected; in the second stage, once the neighbors of S realize, the policy selects every node $v_i \in \mathcal{N}(S)$ that realized with probability $\mathbf{q}_i$. The expected value of this policy denoted $F(S, \mathbf{q})$ is:

$$F(S, \mathbf{q}) = \sum_{T \subseteq \mathcal{N}(S)} \Big(\Pi_{i \in T} \mathbf{p}_i \mathbf{q}_i \Pi_{i \notin T} (1 - \mathbf{p}_i \mathbf{q}_i) \Big) f(T)$$

A *feasible* non-adaptive policy is a pair $(S, \mathbf{q})$ which meets the budget in expectation, i.e. a policy for which $C(S, \mathbf{q}) \leq B$, where $C(S, \mathbf{q}) = c(S) + \sum_{i \in \mathcal{N}(S)} c(i) \mathbf{p}_i \mathbf{q}_i$. When we refer to a non-adaptive policy we always mean a feasible policy.

Densities. The *marginal value* of some policy $(H, \mathbf{r})$ to an existing policy $(S, \mathbf{q})$ is denoted by $F_{S,\mathbf{q}}(H, \mathbf{r})$ and defined as $F_{S,\mathbf{q}}(H, \mathbf{r}) = F(S \cup H, \mathbf{q} \vee \mathbf{r}) - F(S, \mathbf{q})$. Similarly, the marginal cost is $C_{S,\mathbf{q}}(H, \mathbf{r}) = C(S \cup H, \mathbf{q} \vee \mathbf{r}) - C(S, \mathbf{q})$. The *marginal density* of a policy is given by:

$$D_{S,\mathbf{q}}(H, \mathbf{r}) = \begin{cases} F_{S,\mathbf{q}}(H, \mathbf{r}) / C_{S,\mathbf{q}}(H, \mathbf{r}) & \text{if } (S, \mathbf{q}) \neq (H, \mathbf{r}) \\ 0 & \text{otherwise} \end{cases}$$

Finally, we let $\mathbf{w}$ denote the *expected costs* vector, defined as the element-wise multiplication of $\mathbf{c}$ and $\mathbf{p}$ (i.e. the costs and probabilities of the model).

4.1 The algorithm

The algorithm for optimizing over non-adaptive policies called DENSESTASCENT is a greedy algorithm which iteratively adds nodes in X and fractional values to their neighbors. We formally describe the algorithm below, followed by a brief explanation.

Algorithm 1 DensestAscent

Input: $f : 2^{\mathcal{N}(X)} \to \mathbb{R}_+$, budget B, ϵ.

$S^* \leftarrow \emptyset$,$\mathbf{q}^* \leftarrow \mathbf{0}$.
for $S_0 \subseteq X, |S_0| \leq 1/\epsilon$ **do**
  $S \leftarrow S_0$, $\mathbf{q} \leftarrow \mathbf{0}$.
  while $C(S, \mathbf{q}) < B$ **do**
    $(S, \mathbf{q}) \leftarrow$ AddDensest$(S, \mathbf{q})$
  end while
  if $F(S, \mathbf{q}) \geq F(S^*, \mathbf{q}^*)$ **then**
    $S^* \leftarrow S, \mathbf{q}^* \leftarrow \mathbf{q}$
  end if
end for
return $(S^*, \mathbf{q}^*)$

The algorithm considers all subsets of X of size at most $1/\epsilon$ and greedily completes each set until exhausting the budget. The algorithm then takes the completed solution which has the maximal value, out of all $n^{O(1/\epsilon)}$ solutions. The greedy completion is done via calls to the AddDenesest procedure. This procedure adds nodes in X and assigns values in $[0, 1]^{|\mathcal{N}(X)|}$ to the vector $\mathbf{q}$ in a manner that maximizes *density* (i.e. bang-per-buck). More specifically, AddDensest finds an approximately densest addition of first-stage node $x \in X$ and probabilities over second-stage nodes to the vector $\mathbf{q}$. It enumerates over all first-stage candidates $x \in X$. For each x, it greedily adds the second-stage neighbors of x that maximize the marginal density, until either the budget is exhausted or until the marginal density can no longer be improved.

Algorithm 2 AddDensest

for $x \in X$ s.t. $C\left(S \cup \{x\}, \mathbf{q}\right) \leq B$ **do**
  $\mathbf{q}_x \leftarrow$ Saturate$\left(x, (S, \mathbf{q}), B - C\left(S \cup \{x\}, \mathbf{q}\right)\right)$
end for
return argmax $D_{S,\mathbf{q}}(x, \mathbf{q}_x)$

Saturate

Input: $x \in X$, current solution $(S, \mathbf{q})$, remaining budget b

$\mathbf{r} \leftarrow \mathbf{q}$
while $b > 0$ **do**
  $j \leftarrow \text{argmax}_i \frac{F_{S,\mathbf{r}}(x, \mathbf{e}_i)}{\mathbf{w}_i}$
  if $\frac{F_{S,\mathbf{r}}(x, \mathbf{e}_j)}{\mathbf{w}_j} \geq D_{S,\mathbf{q}}(x, \mathbf{r})$ **then**
    $r_j \leftarrow \min\{1, \frac{b}{\mathbf{w}_j}\}$
    $b \leftarrow b - r_j \mathbf{w}_j$
    else break
  end if
end while
return $\mathbf{r}$

We first prove that the procedure AddDensest above obtains a $(1-1/e-O(\epsilon))$-approximation to the density. We then use this property to establish the algorithm's performance guarantee.

Lemma 4.1. AddDensest *returns a* $(1 - 1/e)$*-approximation to the maximal marginal density.*

Proof. We have standard arguments that show that greedy algorithms like AddDensest can approximate monotone submodular functions within $1-1/e$. However, the *marginal density is neither*

monotone nor submodular[1]. Therefore we prove our lemma by arguing about a similar algorithm that maximizes the marginal *value* as opposed to the elusive density.

For the sake of analysis, suppose that we knew the first-stage node x^* and cost $b^* \leq b$ of the density-maximizing feasible addition $\mathbf{r}^*$. Consider a modified version $\overline{\text{SATURATE}}$ of Subroutine SATURATE, which has Line 4 removed, and runs until a budget of b^* is exhausted. Denote the output of the modified algorithm $\overline{\text{SATURATE}}$ by $(x^*, \bar{\mathbf{r}})$.

$\overline{\text{SATURATE}}$ is the standard greedy algorithm for maximizing the multilinear fractional relaxation of submodular f with budget b^*. By standard arguments, we achieve a $(1 - 1/e)$-approximation to the *marginal value*:

$$F_{S,\mathbf{q}}(x^*, \bar{\mathbf{r}}) \geq (1 - 1/e) F_{S,\mathbf{q}}(x^*, \mathbf{r}^*).$$

Since $(x^*, \bar{\mathbf{r}})$ has the same cost b^* as the optimal solution $(x^*, \mathbf{r}^*)$, it also gives a $(1 - 1/e)$-approximation for the density:

$$D_{S,\mathbf{q}}(x^*, \bar{\mathbf{r}}) \geq (1 - 1/e) D_{S,\mathbf{q}}(x^*, \mathbf{r}^*).$$

Of course, $\overline{\text{SATURATE}}$ is an imaginary subroutine - in reality we don't know b^*. Nonetheless, since our original algorithm, SATURATE, adds nodes as long as the marginal density increases, and by submodularity once the density decreases it never increases again, it must do at least as well as $\overline{\text{SATURATE}}$. Let $(x^*, \mathbf{r})$ denote the output of SATURATE when running with node x^*. Then,

$$D_{S,\mathbf{q}}(x^*, \mathbf{r}) \geq D_{S,\mathbf{q}}(x^*, \bar{\mathbf{r}}).$$

To complete the proof, we next show that the marginal density is always maximized by a single first-stage node $x^* \in X$ and its neighbors (as opposed to a subset of X). Given a solution $(O, \mathbf{v})$ of optimal marginal density, associate each second-stage node in $\mathcal{N}(O)$ with one of its neighbors in O, and remove all other edges from the graph. For any $x \in O$, let $\mathbf{v}(x)$ denote the restriction of $\mathbf{v}$ to neighbors of x. Then we have,

$$D_{S,\mathbf{q}}(O, \mathbf{v}) \leq \frac{\sum_{x \in O} F_{S,\mathbf{q}}(x, \mathbf{v}(x))}{\sum_{x \in O} C_{S,\mathbf{q}}(x, \mathbf{v}(x))} \leq \max_{x \in O} D_{S,\mathbf{q}}(x, \mathbf{v}(x)).$$

Let $x^* = \arg\max_{x \in O} D_{S,\mathbf{q}}(x, \mathbf{v}(x))$. Since we enumerate over all $x \in X$, wel in particular consider x^*. □

Building on Lemma 4.1, we now analyze Algorithm DENSESTASCENT and prove the approximation guarantee.

Theorem 4.2. *For any constant $\epsilon > 0$,* DENSESTASCENT *is a $(1 - 1/e^{1-1/e} - O(\epsilon))$-approximation algorithm of the optimal non-adaptive policy for general submodular functions.*

Proof. We now show that this $(1 - 1/e)$-approximation for ADDDENSEST translates to a $(1 - 1/e^{1-1/e} - \epsilon)$-approximation guarantee for DENSESTASCENT. Had we knew, that at every iteration the addition returned by ADDDENSEST is also a $(1 - 1/e)$-approximation to the marginal density *of the entire optimal policy*, then we would be done. However, there is a complication due to the knapsack constraints: what if some of the second-stage nodes used by the optimal policy are infeasible given the remaining budget?

To overcome this obstacle we introduced in Line 2 of Algorithm DENSESTASCENT a partial enumeration over all subsets $S \subseteq X$ of size $1/\epsilon$.

[1] Adding more nodes may increase the cost and decrease the density - thus non-monotone; adding yet more nodes may again decrease the density, but to a lesser degree - hence also not submodular.

Let $(O, \mathbf{v})$ be the optimal solution. Consider some arbitrary ordering on the nodes in O and based on this ordering, for every $o \in O$ define the vector $\mathbf{v}(o)$ to have the values of $\mathbf{v}$ in all indices j for which $j \in \mathcal{N}(o)$ and o is the highest ordered neighbor of j. Now consider the iteration in which S_0 is the set of the $1/\epsilon$ nodes in O with the highest combined value according to this decomposition (or when $S_0 = O$ if $|O| \leq 1/\epsilon$).

Now look at the first iteration of the algorithm in which there is an element $(o, \mathbf{v}(o))$ that if added the solution would violate the budget constraint. Let the solution before that iteration be $(S, \mathbf{q})$. Note that if $o \in S$ then by submodularity there exists a partial solution with at least the same marginal density and thus in this iteration we still get at least the same marginal density as of the optimal solution. Otherwise, let $O' = O \setminus \{o\}$ and $\mathbf{v}' = \mathbf{v} - \mathbf{v}(o)$. It is easy to see that from our choice of S_0 we have that $F(O', \mathbf{v}') \geq (1-\epsilon)F(O, \mathbf{v})$. Now also notice that $C(S, \mathbf{q}) \geq C(O', \mathbf{v}')$ because adding $(o, \mathbf{v}(o))$ to $(S, \mathbf{q})$ violates the budget constraint.

Since at every previous iteration we used a $(1-1/e)$-approximation of the densest contribution,

$$F(S, \mathbf{q}) \geq (1 - 1/e^{1-1/e})F(O', \mathbf{v}') \geq (1 - 1/e^{1-1/e})(1-\epsilon)F(O, \mathbf{v}).$$

□

5 Adaptive Policies

In this section we develop an approximation algorithm for the optimal adaptive policy, which uses the non-adaptive algorithm from the previous section. At a high-level, one would like to take a non-adaptive policy $(S, \mathbf{q})$ and use S as the adaptive policy. Showing that S is a good approximation to the optimal adaptive policy would then follow by making two arguments:

- First, that the value of the optimal non-adaptive policy is an approximation of the value of the optimal adaptive policy.
- Second, that the value obtained from the non-adaptive objective can be approximately obtained in expectation over the second-stage realizations.

The first point follows as an easy consequence of a Lemma from [1] which bounds the gap between the optimal non-adaptive policy to the optimal adaptive policy (this bound is tight). Let OPT_{NA} be the optimal value of the non-adaptive objective. We include the proof for completeness.

Lemma 5.1. *$OPT_{NA} \geq (1 - \frac{1}{e})OPT$.*

Proof. Let S be the set chosen by the optimal adaptive policy and let T_i be the set chosen by this policy in realization R_i. For a set $T \subseteq \mathcal{N}(S)$ let α_T be the total probability that T is chosen by the adaptive policy in the second stage. That is, $\alpha_T = \sum_{i \in \{i | T = T_i\}} p(R_i)$. For $i \in \mathcal{N}(S)$ let $\mathbf{r}_i$ be the probability that i is seeded over all realizations in the second stage. That is, $\mathbf{r}_i = \sum_{\{T | i \in T\}} \alpha_T$. Now note that $\mathbf{r}_i \leq \mathbf{p}_i$ as an item can't be seeded with a higher probability than the probability it realizes. Thus, there exists $\mathbf{q} \in [0,1]^{|\mathcal{N}(S)|}$ such that for every $i \in \mathcal{N}(S)$ $\mathbf{r}_i = \mathbf{q}_i \mathbf{p}_i$. Since in each realization the cost of element chosen is at most $B - c(S)$, we also know that $\mathbf{r}^T \mathbf{w} < B - c(S)$. Thus, $(S, \mathbf{q})$ is a feasible non-adaptive policy.

Consider the function:

$$f^+(S, \mathbf{q}) = \max\{ \sum_{T \subseteq \mathcal{N}(S)} \alpha_T f(T) | \sum_T \alpha_T = 1; \alpha_T \geq 0; \sum_T \alpha_T 1_T = \mathbf{r}\}.$$

Obviously, $\sum_T \alpha_T = 1$ and $\texttt{OPT} = \sum_T \alpha_T f(T)$. Thus α is a valid parameter which $f^+(S, \mathbf{q})$ optimizes over which means $f^+(S, \mathbf{q}) \geq \texttt{OPT}_A$. By a consequence of a lemma due to [3] we know that $F(S, \mathbf{q}) \geq (1 - 1/e) f^+(\mathbf{q})$ and thus $OPT_{NA} \geq F(S, \mathbf{q}) \geq (1 - 1/e)\texttt{OPT}$. □

The second part however, introduces a fundamental challenge. The problem is that the non-adaptive policy meets the budget in expectation over the probabilities of the second stage (the $\mathbf{p}$ in the model) and the randomizations of the policy (the $\mathbf{q}$ values). Since the policy needs to a priori divide its budget between the first stage and second stage, before the realizations occur, it can select high-cost nodes that appear with small probability, and may never have enough remaining budget to actually seed them when the realization of the second stage occurs. As an example, consider one node in each stage, both of cost 2, a budget of $B = 3$, and the objective function $f(T) = |T|$. A non-adaptive policy can buy the second stage node with probability $\mathbf{q}_1 = 1/2$ and obtain an expected value of $1/2$, while the value of the optimal adaptive policy is 0. In this example a non-adaptive policy is completely meaningless.

5.1 A special case: small costs

An important special case is the one where no node costs too much with respect to the general budget. While in many applications different nodes might have much different costs compared to one another it is very likely that none of them has a cost that is a constant fraction of the budget. The two-layered aspect of our problem continues to be an obstacle, and in particular the lower bound construction from Section 3 applies here. The small costs, however, make it possible to convert non-adaptive policies into adaptive ones, with a small loss. An additional factor of $(1-1/e)$ is incurred due to Lemma 5.1. For the special case of small costs, we get a strong guarantee.

Theorem 5.2. *Suppose that the cost of every node is at most a δ-fraction of the budget: $c(x) \leq \delta B$. Then there is a $(1-1/e)(1-1/e^{1-1/e})(1-O(\delta^{1/3}))$-approximation algorithm to the optimal adaptive policy.*

We use the following lemmas in our proof. The first lemma shows that we can reduce the cost of a non-adaptive policy by a constant fraction without losing too much in its value.

Lemma 5.3. *Given a non-adaptive policy $(S, \boldsymbol{q})$ of cost B, we can construct, for any $\epsilon \geq \delta$, a policy $(S_\epsilon, \boldsymbol{q}_\epsilon)$ of cost $(1-\epsilon)B$ Such that $F(S_\epsilon, \boldsymbol{q}_\epsilon) \geq (1-4\epsilon)F(S, \boldsymbol{q})$.*

Proof. If most of the budget is spent on the second stage, i.e. $C(\mathbf{q}) \geq B/2$, we can save ϵB on this stage by taking $\mathbf{q}_\epsilon = (1-2\epsilon)\mathbf{q}$. By submodularity, $F(S, \mathbf{q}_\epsilon) \geq (1-2\epsilon)F(S, \mathbf{q})$.

Otherwise, most of the budget is spent on the first stage: $c(S) > B/2$. We arbitrarily associate each second-stage node with one of its parents in S. F now induces a submodular function over subsets of S. We greedily remove nodes from S (together with the associated second-stage nodes) until the total cost of the removed first-stage nodes reaches ϵB. Denote the remaining non-adaptive policy $(S_\epsilon, \mathbf{q}_\epsilon)$. Because we assumed that each node has cost at most $\delta B \leq \epsilon B$, the total cost of the *remaining first-stage nodes* is at least $c(S_\epsilon) \geq c(S) - 2\epsilon B \geq (1-4\epsilon)c(S)$. Thus, by submodularity, $F(S_\epsilon, \mathbf{q}_\epsilon) \geq (1-4\epsilon)F(S_\epsilon, \mathbf{q}_\epsilon)$. □

The next lemma shows that given a non-adaptive policy that does not use some constant fraction of its budget, we can find an adaptive policy of almost the same value that never over spends.

Lemma 5.4. *If $\delta < \frac{1}{2}$, then for any feasible non-adaptive policy $(S, \boldsymbol{q})$ such that $C(S, \boldsymbol{q}) < (1-\delta^{1/3}-\delta)B$, we can find an adaptive policy of cost B with value $\geq \left(1-\delta^{2/3}\right) F(S, \boldsymbol{q})$.*

Proof. In the first stage we seed S. In the second stage, for any realized set $R \subseteq \mathcal{N}(S)$ (where each item $i \in \mathcal{N}(S)$ is realized with probability $\mathbf{p}_i\mathbf{q}_i$), we seed the set $T = R$ if it is feasible (i.e. if $c(R) \leq B_2 = B - c(S)$), and let T be the empty set otherwise. We next compute the probability of any node j to be in T given that it is in R:

$$\Pr\left[j \in T \mid j \in R\right] = \Pr\left[c(R) \leq B_2 \mid j \in R\right] = 1 - \Pr\left[c(R \setminus \{j\})| > B_2 - c(j)\right]$$

Notice that because the entire non-adaptive policy fits in budget $(1-\delta^{1/3}-\delta)B$, the expected cost of R is at most $B_2 - (\delta^{1/3}+\delta)B$. Recall also that $c(j) \leq \delta B$. Therefore,

$$\Pr\left[j \in T \mid j \in R\right] \geq 1 - \Pr\left[c(R \setminus \{j\})| > \mathbf{E}\left[c(R \setminus \{j\})\right] + \delta^{1/3}B\right]$$

Applying Hoeffding's inequality, we have that

$$\Pr\left[j \in T \mid j \in R\right] \geq 1 - e^{-\frac{2(\delta^{1/3}B)^2}{\delta B^2}} = 1 - e^{-2/\delta^{1/3}} > 1 - \delta^{2/3}$$

(For the last inequality, observe that for any $x > 0$, $x = x \cdot e^{1/x} \cdot e^{-1/x} > x \cdot (1/x) \cdot e^{-1/x} = e^{-1/x}$.)

Finally, by submodularity,

$$\mathbf{E}[f(T)] \geq \left(1-\delta^{2/3}\right)\mathbf{E}[f(R)] = \left(1-\delta^{2/3}\right)F(S,\mathbf{q}).$$

□

We are now ready to prove Theorem 5.2.

Proof. Given the results from Section 4, we can find $(S,\mathbf{q})$ that approximate the optimal non-adaptive policy:

$$F(S,\mathbf{q}) \geq (1-1/e^{1-1/e}-\epsilon)\texttt{OPT}_{NA}.$$

In Lemmas 5.3 and 5.4 we prove that given a non-adaptive policy $(S,\mathbf{q})$, we can find an adaptive policy $A(S,\mathbf{q})$, such that

$$A(S,\mathbf{q}) \geq (1-O(\delta^{1/3}))F(S,\mathbf{q}).$$

Together with Lemma 5.1 we have:

$$\begin{aligned} A(S,\mathbf{q}) &\geq (1-O(\delta^{1/3}))F(S,\mathbf{q}) \geq (1-O(\delta^{1/3}))(1-1/e^{1-1/e})\texttt{OPT}_{NA} \\ &\geq (1-O(\delta^{1/3}))(1-1/e^{1-1/e})(1-1/e)\texttt{OPT}. \end{aligned}$$

□

5.2 The general case: arbitrarily large costs

In general, when the costs can be an arbitrary large fraction of the budget (w.l.o.g. we assume costs are always bounded by B), non-adaptive policies are still the main workhorse, though a far more nuanced treatment is required. The main idea for overcoming obstacles like the one described before is to consider the optimal adaptive policy as several restricted policies, where each one of these restrictions can be approximated either by using the non-adaptive relaxation, or through an adaptive policy which can be found using sampling and standard techniques for submodular maximization. Roughly speaking, we show that any adaptive policy can be broken into three components: a policy that uses at most two nodes on the first stage; a policy that uses at most two nodes on the second

stage; and a policy whose nodes have bounded costs. We then find an approximate solution for each of those restricted classes.

Consider some arbitrary association of every second-stage node $y \in \mathcal{N}(S)$ with one parent $x(y) \in X$. Given an adaptive policy, let $\overline{c}(x)$ denote the expected cost of x and all the second-stage neighbors associated with x selected by this policy. Notice that $\sum_{x \in S} \overline{c}(x) \leq B$. In particular, there are at most two nodes in X for which $\overline{c}(x) > B/3$. Furthermore, in every realization, any feasible policy can seed at most two second-stage nodes y such that $c(y) > B/3$.

Definition 5.5. *We say that an adaptive policy is* nice *if every first-stage node satisfies* $\overline{c}(x) \leq B/3$*, and every second-stage node* y *satisfies* $c(y) \leq B/3$*.*

We denote the optimal value of a nice adaptive policy by $\texttt{OPT}_{nice}$. Similarly $\texttt{OPT}_{TP}$ denotes the optimal value of a policy that seeds at most two nodes on the first stage, and $\texttt{OPT}_{TC}$ denotes the value of the optimal policy that, on any realization, seeds at most two nodes on the second stage. The following lemma follows from submodularity and the discussion above.

Lemma 5.6. $\mathit{OPT} \leq \mathit{OPT}_{nice} + \mathit{OPT}_{TP} + \mathit{OPT}_{TC}$.

Both $\texttt{OPT}_{TC}$ and $\texttt{OPT}_{TP}$ can be well approximated by reducing the problem to maximization of submodular functions under budget constraints, with approximation ratios of $\left(\frac{1-1/e}{2}\right)$ and $(1-1/e)$, respectively. We begin with the two second stage nodes case. The idea is that we can instead solve for the best adaptive policy with one second stage node and that the objective in that case is submodular.

Lemma 5.7. *We can find in polynomial time an adaptive policy that achieves value* $\left(\frac{1-1/e}{2}\right) \mathit{OPT}_{TC}$.

Proof. By submodularity, $\texttt{OPT}_{TC} \leq 2\texttt{OPT}_{SC}$, where the latter is a maximum over all policies that only seed a single second stage node.

In order to approximate $\texttt{OPT}_{SC}$, we enumerate over partitions of the budget $B_1 + B_2 = B$. Notice that we can assume without losing generality that the budget for the second stage, B_2, is exactly equal to the price of some second stage node. Thus we only enumerate over polynomially-many partitions.

Once we fix the budget partition, let $f_{B_2}(S)$ be the expectation of the maximum value of a single, feasible second stage node in $\mathcal{N}(S)$. It is easy to see that f_{B_2} is monotone submodular: Adding another node x to S adds available nodes to $\mathcal{N}(S)$; similarly, if $S \subseteq S'$, then x adds more nodes to S than to S', i.e. $(\mathcal{N}(S' \cup x) \setminus \mathcal{N}(S')) \subseteq (\mathcal{N}(S \cup x) \setminus \mathcal{N}(S))$.

Finally, for any S we can compute the exact value of $f_{B_2}(S)$. Order the nodes $y \in \mathcal{N}(S)$ according to $f(y)$, and compute, for each y, the probability that we seed it: this is the probability it realizes, but all the nodes with higher utility do not. Therefore we can approximate $\texttt{OPT}_{SC}$ to within $(1-1/e)$. □

The next Lemma shows we can approximate the optimal policy that only seeds two first stage nodes.

Lemma 5.8. *We can find in polynomial time an adaptive policy that achieves value* $(1-1/e-\epsilon)\,\mathit{OPT}_{TP}$.

Proof. Enumerate over all pairs of first-stage nodes, and sample second-stage realizations for each pair. For each pair and realization, we need to solve a submodular maximization problem over the feasible realized neighbors. By repeating this enough times for each pair we can approximates the value of its value to within $(1-1/e-\epsilon)$. Thus, choosing the best pair gives us the desired result. □

To approximate $\texttt{OPT}_{nice}$, the restriction to a nice policy by itself does not suffice to bound the adaptivity gap of the non-adaptive relaxation. Instead, we observe that any nice policy can be decomposed into five nice policies, each of which uses an expected budget of at most $B/3$. Let $\texttt{OPT}_{nice}^{B/3}$ be the optimal *nice* adaptive policy that uses at most $B/3$ budget.

Lemma 5.9. $\mathit{OPT}_{nice} \leq 5\mathit{OPT}_{nice}^{B/3}$.

Proof. We decompose S into five subsets $S_1, \ldots, S_5$, each with expected cost $\overline{c}(S_i) = \sum_{x \in S_i} \overline{c}(x) \leq B/3$. Order the nodes in S, and iteratively add nodes to S_1 while $\overline{c}(S_1) \leq B/3$; add the next nodes to S_2 while $\overline{c}(S_2) \leq B/3$, and so on. By the nice condition on the first-stage nodes (i.e. $\overline{c}(x) \leq B/3$), we get that $\overline{c}(S_1) + \overline{c}(S_2) > B/3$ as well as $\overline{c}(S_3) + \overline{c}(S_4) > B/3$. Therefore, we can place all the remaining nodes in S_5. □

Finally, we must show that we can approximate $\texttt{OPT}_{nice}^{B/3}$.

Lemma 5.10. *We can find in polynomial time an adaptive policy that achieves value* $\left(\frac{1-1/e}{2}\right)\left(1 - e^{-(1-1/e)}\right) \cdot \mathit{OPT}_{nice}^{B/3}$.

Proof. Apply Algorithm DensestAscent to a modified instance of the problem, with budget $B/3$, and without all the nodes of cost greater than $B/3$. Let $\left(S^{B/3}, \mathbf{q}^{B/3}\right)$ be the output of DensestAscent on the modified instance. Then,

$$F\left(S^{B/3}, \mathbf{q}^{B/3}\right) \geq \left(1 - e^{-(1-1/e)}\right) \texttt{OPT}_{NA}^{B/3} \geq \left(1 - e^{-(1-1/e)}\right)(1 - 1/e)\, \texttt{OPT}_{nice}^{B/3}$$

Finally, it remains to show how to convert $\left(S^{B/3}, \mathbf{q}^{B/3}\right)$ to a feasible adaptive solution. Let $\hat{T}$ be the set of realized nodes in $\mathcal{N}\left(S^{B/3}\right)$ (according to both $\mathbf{p}$ and $\mathbf{q}$). We seed $T = \hat{T}$ whenever $c(\hat{T}) + c(S) \leq B$, and otherwise $T = \phi$.

Since the expected cost of $(S, \hat{T})$ is only $B/3$, we have that for each $y \in \mathcal{N}\left(S^{B/3}\right)$, $\Pr\left[y \in T \mid y \in \hat{T}\right] \geq 1/2$. Therefore by submodularity, the value of this adaptive policy is at least

$$\mathbf{E}\left[f\left(T\right)\right] \geq (1/2)\,(1 - 1/e)\left(1 - e^{-(1-1/e)}\right) \texttt{OPT}_{nice}^{B/3}.$$

□

Combining these results gives us the following theorem.

Theorem 5.11. *There is a constant factor (≈ 0.0259) approximation to the optimal adaptive policy that can be found in polynomial time.*

6 Additive Functions

In this section we show that for the special case in which the function is additive, i.e. $f(T) = \sum_{i \in T} f(i)$, one can obtain better approximation guarantees by solving for the adaptive policy directly. While this functions are simple they already capture important classes of influence models like the voter model [7, 8].

We first claim that in this case no algorithm can do better than $1 - 1/e$. We give a simple reduction from the Max-Cover problem: set all the costs of X to 1 and all the costs of Y to 0, and let the additive function $f : 2^{\mathcal{N}(X)} \to \mathbb{R}$ be $f(T) = |T|$. In this case the Adaptive-Seeding problem is equivalent to finding $\lfloor B \rfloor$ nodes in X that maximize cover on the universe of $\mathcal{N}(X)$.

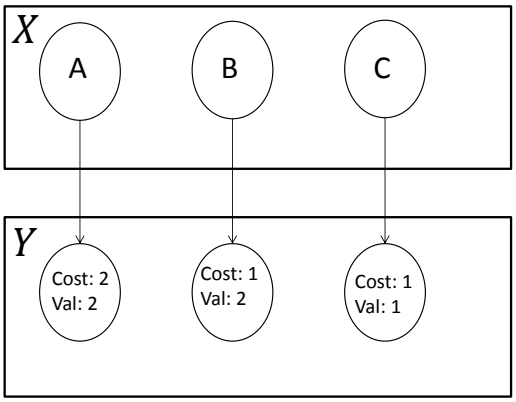


Figure 1: An example of a graph where Adaptive-Seeding is not submodular when the cost are not uniform.

Claim 6.1. *No polynomial-time algorithm can approximate the* Adaptive-Seeding *problem with additive functions under knapsack constraints within a factor better than* $1 - 1/e$*, unless P=NP.*

Note that the above bound holds even for the deterministic case, i.e. when all probabilities are one. The algorithm sketched here provides a matching upper bound. When the ratio (denoted δ) between the highest cost of a node in the *second stage* and the budget used in the second stage by the optimal solution is sufficiently small, the algorithm is a near-optimal matching upper bound for the Adaptive-Seeding problem. More generally, we have the following result.

Theorem 6.2. *For any* $\epsilon > 0$*, there is a* $\max\{1/2, 1-\delta\}(1-1/e-\epsilon)$*-approximation algorithm for the best adaptive policy when the function is additive.*

The main idea in this upper bound is to consider the following fractional relaxation:

$$\mathcal{H}_{R,B_2}(S) = \max_{\mathbf{q}} \Big\{ \sum_{i=1}^{n} v_i q_i \ : \ \sum_{i=1}^{n} c_i q_i \leq B_2 \ ; \ q_i \leq \mathbf{1}_{\{i\}}^{\intercal} \mathbf{1}_{\mathcal{N}(S)\cap R} \ \ \forall i \in [n] \ ; \ \mathbf{q} \in [0,1]^n \Big\}$$

It is important to note that the definition of $\mathcal{H}_{R,B_2}$ as an optimal *fractional* solution is crucial. The following example shows that defining $\mathcal{H}_{R,B_2}$ as the optimal *integral* solution breaks submodularity. Let $\mathcal{H}'_{R,B_2}$ be defined as $\mathcal{H}_{R,B_2}$ where the constraint $\mathbf{q} \in [0,1]^n$ from the definition of $\mathcal{H}_{R,B_2}$ is replaced with $\mathbf{q} \in \{0,1\}^n$. We show that $\mathcal{H}'_{R,B_2}(\cdot)$ is not submodular. Consider the example in figure 1 and the function $\mathcal{H}'_{R,2}$ where the edges represent the edges of G. It is easy to see that $\mathcal{H}'_{R,2}(A) = 2$ and $\mathcal{H}'_{R,2}(\{A,C\}) = 2$ and also that $\mathcal{H}'_{R,2}(\{A,B\}) = 2$ and $\mathcal{H}'_{R,2}(\{A,B,C\}) = 3$. But that means that adding C to the larger set $\{A,B\}$ increases the value by more than adding it to the smaller set $\{A\}$. We next show that H_{R,B_2} is indeed monotone submodular.

Lemma 6.3. *For any realization* $R \subseteq \mathcal{N}(X)$ *and budget* $B_2 \in [0,B]$ *the function* $\mathcal{H}_{R,B_2}$ *as defined above is monotone submodular.*

Proof. We first notice that for every set $S \subset X$ we can view $\mathcal{H}_{R,B_2}(S)$ as an linear program, and let $\mathbf{q}^*(S)$ denote an optimal solution for $\mathcal{H}_{R,B_2}(S)$. For any $S_1, S_2 \subseteq X$, consider the sum of the solutions $\mathbf{r} = \mathbf{q}^*(S_1 \cap S_2) + \mathbf{q}^*(S_1 \cup S_2)$. We show that there are also feasible solutions to the LPs $\mathcal{H}_{R,B_2}(S_1)$ and $\mathcal{H}_{R,B_2}(S_2)$ whose sum is $\mathbf{r}$. By linearity of the objective function, this proves that

$$\mathcal{H}_{R,B_2}(S_1) + \mathcal{H}_{R,B_2}(S_2) \geq \mathcal{H}_{R,B_2}(S_1 \cap S_2) + \mathcal{H}_{R,B_2}(S_1 \cup S_2).$$

Notice that in each coordinate i, $\mathbf{r}_i$ can be split into $\mathbf{r}_i = \mathbf{r}_i^1 + \mathbf{r}_i^2$, such that $\mathbf{r}_i^j$ is non-zero only if $i \in \mathcal{N}(S_j)$. However, solutions $\mathbf{r}^1$ and $\mathbf{r}^2$ may still not be feasible since they may violate the budget constraint.

To ensure budget-feasibility, notice that if $\mathbf{q}_i^*(S_1 \cap S_2) > 0$ then $i \in \mathcal{N}(S_1)$. Thus we can require that for every i, $\mathbf{r}_i^1 \geq \mathbf{q}_i^*(S_1 \cap S_2)$. We label this partition $\left(\mathbf{r}^{1+}, \mathbf{r}^{2-}\right)$, and define $\left(\mathbf{r}^{1-}, \mathbf{r}^{2+}\right)$ in a similar fashion. (That is, for every i, $\mathbf{r}_i^{2+} \geq \mathbf{q}_i^*(S_1 \cap S_2)$.) Notice that $c(\mathbf{r}^{1-}), c(\mathbf{r}^{2-}) \leq c(\mathbf{q}^*(S_1 \cup S_2)) \leq B_2$, i.e. they are both feasible.

For each LP, $\mathcal{H}_{R,B_2}(S_1)$ and $\mathcal{H}_{R,B_2}(S_2)$, we have one feasible solution and one infeasible solution. (The infeasible solution is wlog - otherwise we are done.) Therefore, there is some convex combination $\mathbf{r}^{1*} = \alpha\mathbf{r}^{1-} + (1-\alpha)\mathbf{r}^{1+}$ that exactly satisfies the budget constraint $c(\mathbf{r}^{1*}) = B_2$. Therefore, $\mathbf{r}^{2*} = \alpha\mathbf{r}^{2-} + (1-\alpha)\mathbf{r}^{2+}$ also satisfies the budget constraint $c(\mathbf{r}^{2*}) = c(\mathbf{r}) - c(\mathbf{r}^{1*}) \leq B_2$. By convexity, both $\mathbf{r}^{1*}$ and $\mathbf{r}^{2*}$ continue to satisfy the rest of the constraints in each LP; they are therefore feasible solutions. Finally, by linearity $\mathbf{r}^{1*} + \mathbf{r}^{2*} = \mathbf{r}$. □

A corollary of the above lemma is that the function $\mathcal{H}_{B_2}(S) = \sum_R p_R \mathcal{H}_{R,B_2}(S)$ is also monotone submodular, since submodular functions are closed under addition. Note that for every set S we can evaluate $\mathcal{H}_{B_2}(S)$ to within any desired accuracy by sampling. Thus, the algorithm in [28] gives us an approximation arbitrarily close to $1 - 1/e$ solution for every value of B_2.

Lemma 6.4. *For any $\epsilon > 0$ and any $0 < B_2 \leq B$ there is a $(1 - 1/e - \epsilon)$ approximation algorithm for:*

$$\max_S \left\{ \mathcal{H}_{B_2}(S) \ : \ S \subseteq X; \ c(S) + B_2 \leq B \right\} \tag{1}$$

We next show that given a solution to problem (1) with B_2 we have an adaptive policy with a loss of at most $\max\{1/2, 1 - \delta_{B_2}\}$ where δ_{B_2} is the ratio of the highest cost of any second stage element and B_2.

Lemma 6.5. *If S is a solution to problem 1, then the adaptive policy that seeds S at the first stage has value at least* $\max\{1/2, 1 - \delta_{B_2}\}\mathcal{H}_{B_2}(S)$.

Proof. We claim that if we use the same set S returned by our algorithm but restrict the second stage to using only integral solutions, then in all the realizations there is an integral solution of value at least $\max\{1/2, 1 - \delta_{B_2}\}$ of the optimal fractional solution. Thus, our adaptive policy gets at least $\max\{1/2, 1 - \delta_{B_2}\}$ of the value of $\mathcal{H}_{B_2}(S)$.

To see this is true, we first notice that without loss of generality an optimal fractional solution has only one fractional entry. Thus we can get at least half the value by taking either only the element associated with the fractional entry or taking all the other elements in the solution. Moreover, by the linearity of the function we know that the contribution of that element to the value of the solution can't be more than $(1 - \delta_{B_2})$ of the solution's value and thus by removing it we get an integral solution of value $(1 - \delta_{B_2})$ of the fractional solution in each realization. □

6.1 Proof of Theorem 6.2

We are now ready to complete the proof of Theorem 6.2. First notice that the value of the optimal solution of problem (1) has a higher value than the value of the optimal adaptive policy. This is because the adaptive policy is a candidate solution for this problem and considering fractional solutions can only increase the value of the solution.

Let c_m be the cost of the smallest cost element in $\mathcal{N}(X)$. The algorithm is as follows: For every $B_2 \in \{c_m, (1+\xi)c_m, (1+\xi)^2 c_m \dots B\}$ we remove all the items in $\mathcal{N}(X)$ whose cost is larger than B_2. We then run the greedy algorithm to find the best solution for $\mathcal{H}_{B_2}$ and compute the value of the adaptive policy (we can approximate its value to any desired accuracy) that seeds the same set S as in Lemma 6.5. We then choose the set S that gave the highest values of all the sets we considered.

The optimal adaptive policy is some set $O \subseteq X$. Let $B_2^* = B - c(O)$. Assume first that in one of the iterations we check B_2^* and find a solution to $\mathcal{H}_{B_2^*}$. O is also a valid solution to $\mathcal{H}_{B_2^*}(\cdot)$, and thus if we solve $\mathcal{H}_{B_2^*}$ we get a solution which is at least a $\max\{1/2, 1-\delta\}(1-1/e-\epsilon')$ approximation of the value of O, for any desired $\epsilon' > 0$. As this is one of our candidate solutions and we choose the one with highest expected value, we know that we get a solution whose value is at least $\max\{1/2, 1-\delta\}(1-1/e-\epsilon')$ of the value of the optimal solution.

In case the algorithm did not use B_2^* exactly in any iteration, we know that it used a B_2 s.t. $B_2^*/(1+\xi) \leq B_2 \leq B_2^*$. With such a value of B_2 we can still choose the same set O as in the optimal solution. It is also easy to see that the value of $\mathcal{H}_{B_2}(O)$ is at least a $(1-\xi)\mathcal{H}_{B_2^*}(O)$, and thus the algorithm must find a solution S such that $\mathcal{H}_{B_2}(S) \geq (1-1/e-\epsilon')/(1-\xi)\mathcal{H}_{B_2^*}(O)$. By Lemma 6.5 and the observation that $\mathcal{H}_{B_2^*}(O)$ is at least the value of the optimal adaptive policy, we know that if we seed S we get an adaptive policy with value at least $\max\{1/2, 1-(1+\xi)\delta\}(1-1/e-\epsilon')/(1-\xi)$ of the optimal adaptive policy. It is easy to check that if we choose ϵ' and ξ to be small enough we get the desired bound.

As we choose the best solution out of all possible values of B_2 we know that the solution we choose as at least this value which completes our proof □

7 Acknowledgments

Aviad Rubinstein was supported by NSF grants CCF-0964033 and CCF1408635, and by Templeton Foundation grant 3966. This work was done in part while Aviad was at the Simons Institute for the Theory of Computing.

Lior Seeman was supported by NSF grant CCF-1214844, ARO grant W911NF-14-1-0017, by the Multidisciplinary University Research Initiative (MURI) program administered by the AFOSR under grant FA9550-12-1-0040, and by Simons Foundation grant 315783.

Yaron Singer was supported by NSF grant CCF-1301976 and a Google Faculty Research Award.

References

[1] A. Badanidiyuru, C. Papadimitriou, A. Rubinstein, L. Seeman, and Y. Singer. Submodular adaptive seeding. Manuscript, 2015.

[2] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In *SODA*, pages 946–957, 2014.

[3] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a submodular set function subject to a matroid constraint. In *IPCO*, pages 182–196. 2007.

[4] C. Chekuri, J. Vondrák, and R. Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. In *STOC*, pages 783–792, 2011.

[5] N. Chen. On the approximability of influence in social networks. In *SODA*, pages 1029–1037, 2008.

[6] B. C. Dean, M. X. Goemans, and J. Vondrák. Approximating the stochastic knapsack problem: The benefit of adaptivity. In *FOCS*, pages 208–217, 2004.

[7] P. Domingos and M. Richardson. Mining the network value of customers. In *KDD*, pages 57–66, 2001.

[8] E. Even-Dar and A. Shapira. A note on maximizing the spread of influence in social networks. *Inf. Process. Lett.*, 111(4):184–187, 2011.

[9] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[10] S. L. Feld. Why your friends have more friends than you do. *American Journal of Sociology*, pages 1464–1477, 1991.

[11] A. Gupta, R. Krishnaswamy, V. Nagarajan, and R. Ravi. Approximation algorithms for stochastic orienteering. In *SODA*, pages 1522–1538, 2012.

[12] A. Gupta, M. Pál, R. Ravi, and A. Sinha. Boosted sampling: approximation algorithms for stochastic optimization. In *STOC*, pages 417–426, 2004.

[13] N. O. Hodas, F. Kooti, and K. Lerman. Friendship paradox redux: Your friends are more interesting than you. In *ICWSM*, 2013.

[14] T. Horel and Y. Singer. Scalable methods for adaptively seeding a social network. In *WWW*, 2015.

[15] N. Immorlica, D. Karger, M. Minkoff, and V. S. Mirrokni. On the costs and benefits of procrastination: approximation algorithms for stochastic combinatorial optimization problems. In *SODA*, pages 691–700, 2004.

[16] D. Kempe, J. M. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *KDD*, pages 137–146, 2003.

[17] J. Kleinberg, Y. Rabani, and É. Tardos. Allocating bandwidth for bursty connections. *SIAM Journal on Computing*, 30(1):191–217, 2000.

[18] S. Lattanzi and Y. Singer. The power of random neighbors in social networks. In *WSDM*, 2015.

[19] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. M. VanBriesen, and N. S. Glance. Cost-effective outbreak detection in networks. In *KDD*, pages 420–429, 2007.

[20] M. Mathioudakis, F. Bonchi, C. Castillo, A. Gionis, and A. Ukkonen. Sparsication of influence networks. In *KDD*, 2011.

[21] E. Mossel and S. Roch. On the submodularity of influence in social networks. In *STOC*, pages 128–134, 2007.

[22] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions ii. *Math. Programming Study 8*, pages 73–87, 1978.

[23] R. Ravi and A. Sinha. Hedging uncertainty: Approximation algorithms for stochastic optimization problems. In *IPCO*, pages 101–115. 2004.

[24] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD*, pages 61–70, 2002.

[25] L. Seeman and Y. Singer. Adaptive seeding in social networks. In *FOCS*, 2013.

[26] D. B. Shmoys and C. Swamy. Stochastic optimization is (almost) as easy as deterministic optimization. In *FOCS*, pages 228–237, 2004.

[27] D. B. Shmoys and C. Swamy. An approximation scheme for stochastic linear programming and its application to stochastic integer programs. *Journal of the ACM (JACM)*, 53(6):978–1012, 2006.

[28] M. Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Oper. Res. Lett.*, 32(1), 2004.

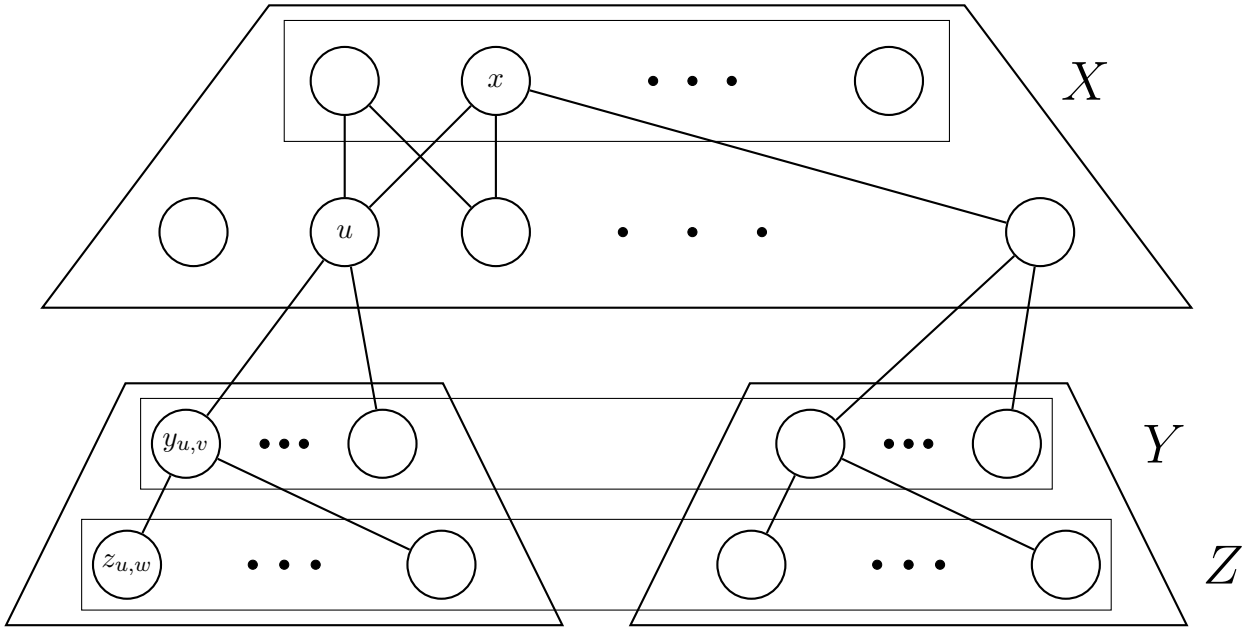
[29] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow. The anatomy of the facebook social graph. *CoRR*, abs/1111.4503, 2011.

Appendix

A Illustrations for the Main Reduction

A.1 Illustration for the warmup reduction

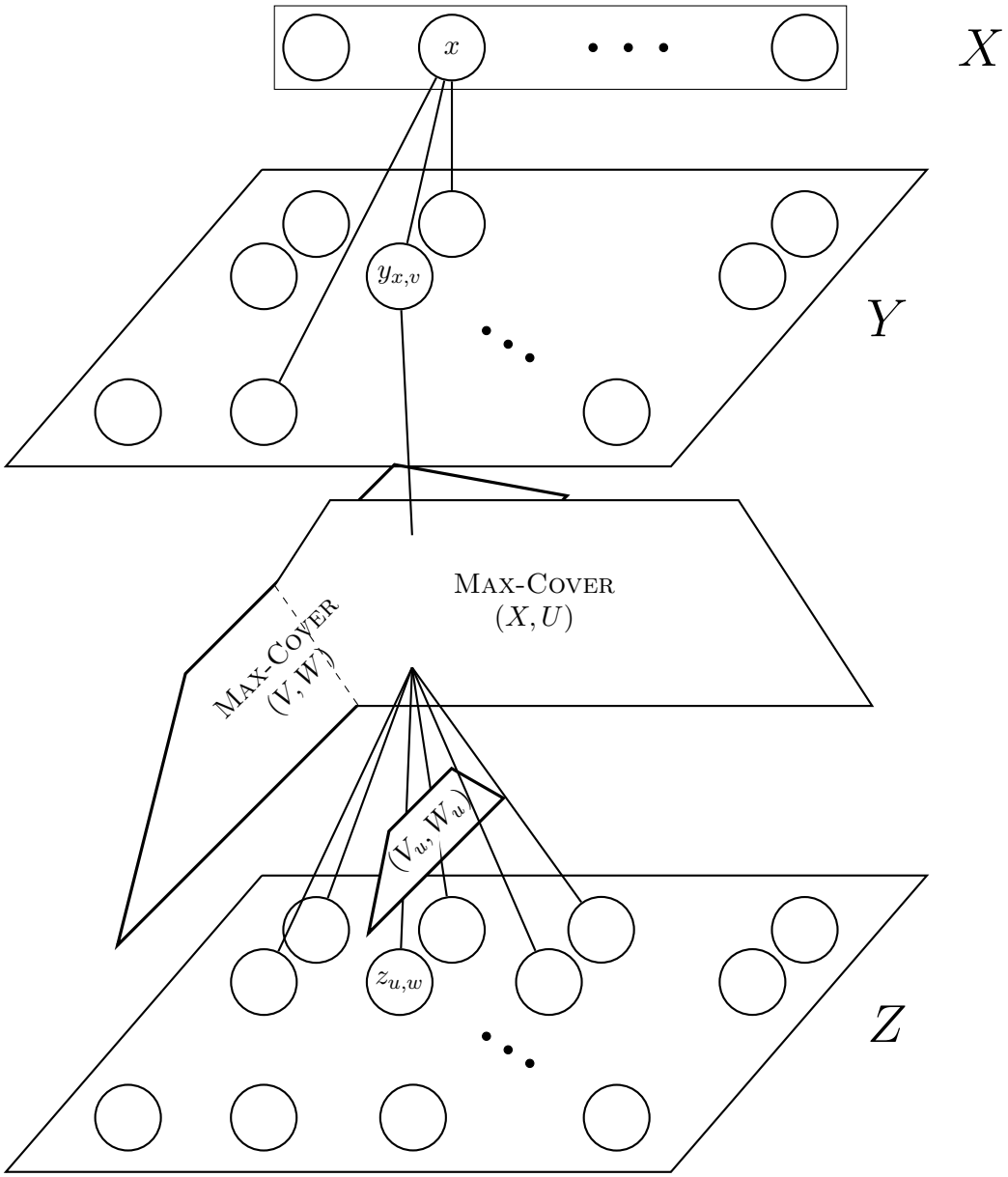
Figure 2: Construction of the warmup instance



Each trapezoid represents an instance of Max-Cover. The subsets of the top instance correspond to the nodes $x \in X$ in the top layer. The elements in the top instance, $u \in U$ are virtual in our construction: they do not correspond to any nodes in the 2S-MCK instance. This top instance of Max-Cover determines the connections between the top and middle layers: Node $y_{u,v} \in Y$ is connected to node x iff $u \in x$.
For each $u \in U$, we have a disjoint instance (V_u, W_u) of Max-Cover. This lower instance of Max-Cover determines the connections between the middle and bottom layers: The middle layer nodes $y_{u,v} \in Y$ correspond to subsets in V_u. The bottom layer nodes, $z_{u,w} \in Z$ correspond to elements in W_u. Node $z_{u,w}$ is connected to $y_{u,v}$ iff $w \in v$.

A.2 Full illustration for the tight reduction

Figure 3: Construction of the hard instance for the tight reduction



Each node $x \in X$ is connected to a row of nodes $y_{x,v} \in Y$. Each node $y_{x,v} \in Y$ corresponds to a products of subsets from two orthogonal Max-Cover instances: $x \in X$ and $v \in V$. Each node $z_{u,w} \in Z$ corresponds to a products of elements from two orthogonal Max-Cover instances: $u \in U$ and $w \in W$. $y_{x,v}$ is connected to $z_{u,w}$ iff both $u \in x$ and $w \in v$. Finally, notice that for each $u \in U$ we have an induced copy (V_u, W_u) of (V, W).