

Pricing strategies for viral marketing on Social Networks

David Arthur¹, Rajeev Motwani¹, Aneesh Sharma², and Ying Xu¹

¹ Department of Computer Science, Stanford University

² Institute for Computational and Mathematical Engineering, Stanford University
{`darthur,rajeev,aneeshs,xuying`}@cs.stanford.edu

Abstract. We study the use of viral marketing strategies on social networks that seek to maximize revenue from the sale of a single product. We propose a model in which the decision of a buyer to buy the product is influenced by friends that own the product and the price at which the product is offered. The influence model we analyze is quite general, naturally extending both the Linear Threshold model and the Independent Cascade model, while also incorporating price information. We consider sales proceeding in a cascading manner through the network, i.e. a buyer is offered the product via recommendations from its neighbors who own the product. In this setting, the seller influences events by offering a cash-back to recommenders and by setting prices (via coupons or discounts) for each buyer in the social network. This choice of prices for the buyers is termed as the seller's strategy.

Finding a seller strategy which maximizes the expected revenue in this setting turns out to be NP-hard. However, we propose a seller strategy that generates revenue guaranteed to be within a constant factor of the optimal strategy in a wide variety of models. The strategy is based on an *influence-and-exploit* idea, and it consists of finding the right trade-off at each time step between: generating revenue from the current user versus offering the product for free and using the influence generated from this sale later in the process. We also show how local search can be used to improve the performance of this technique in practice.

1 Introduction

Social networks such as Facebook, Orkut and MySpace are free to join, and they attract vast numbers of users. Maintaining these websites for such a large group of users requires substantial investment from the host companies. To help recoup these investments, these companies often turn to monetizing the information that their users provide for free on these websites. This information includes both detailed profiles of users and also the network of social connections between the users. Not surprisingly, there is a widespread belief that this information could be a gold mine for targeted advertising and other online businesses. Nonetheless, much of this potential still remains untapped today. Facebook, for example, was valued at \$15 billion by Microsoft in 2007 [13], but its estimated revenue in 2008

was only \$300 million [17]. With so many users and so much data, higher profits seem like they should be possible. Facebook’s Beacon advertising system does attempt to provide targeted advertisements but it has only obtained limited success due to privacy concerns [16]. This raises the question of how companies can better monetize the already public data on social networks without requiring extra information and thereby compromising privacy. In particular, most large-scale monetization technologies currently used on social networks are modeled on the sponsored search paradigm of contextual advertising and do not effectively leverage the networked nature of the data.

Recently, however, people have begun to consider a different monetization approach that is based on selling products through the spread of influence. Often, users can be convinced to purchase a product if many of their friends are already using it, even if these same users would be hard to convince through direct advertising. This is often a result of personal recommendations – a friend’s opinion can carry far more weight than an impersonal advertisement. In some cases, however, adoption among friends is important for even more practical reasons. For example, instant messenger users and cell phone users will want a product that allows them to talk easily and cheaply with their friends. Usually, this encourages them to adopt the same instant messenger program and the same cell phone carrier that their friends have. We refer the reader to previous work and the references therein for further explanations behind the motivation of the influence model [4, 6].

In fact, many sellers already do try to utilize influence-and-exploit strategies that are based on these tendencies. In the advertising world, this has recently led to the adoption of *viral marketing*, where a seller attempts to artificially create word-of-mouth advertising among potential customers [8, 9, 14]. A more powerful but riskier technique has been in use much longer: the seller gives out free samples or coupons to a limited set of people, hoping to convince these people to try out the product and then recommend it to their friends. Without any extra data, however, this forces sellers to make some very difficult decisions. Who do they give the free samples to? How many free samples do they need to give out? What incentives can they afford to give to recommenders without jeopardizing the overall profit too much?

In this paper, we are interested in finding systematic answers to these questions. In general terms, we can model the spread of a product as a process on a social network. Each node represents a single person, and each edge represents a friendship. Initially, one or more nodes is “active”, meaning that person already has the product. This could either be a large set of nodes representing an established customer base, or it could be just one node – the seller – whose neighbors consist of people who independently trust the seller, or who are otherwise likely to be interested in early adoption.

At this point, the seller can encourage the spread of influences in two ways. First of all, it can offer cashback rewards to individuals who recommend the product to their friends. This is often seen in practice with “referral bonuses” – each buyer can optionally name the person who referred them, and this person

then receives a cash reward. This gives existing buyers an incentive to recommend the product to their friends. Secondly, a seller can offer discounts to specific people in order to encourage them to buy the product, above and beyond any recommendations they receive. It is important to choose a good discount from the beginning here. If the price is not acceptable when a prospective buyer first receives recommendations, they might not bother to reconsider even if the price is lowered later.

After receiving discount offers and some set of recommendations, it is up to the prospective buyers to decide whether to actually go through with a purchase. In general, they will do so with some probability that is influenced by the discount and by the set of recommendations they have received. The form of this probability is a parameter of the model and it is determined by external factors, for instance, the quality of the product and various exogenous market conditions. While it is impossible for a seller to calculate the form of these probability exactly, they can estimate it from empirical observations, and use that estimate to inform their policies. One could interpret the probabilities according to a number of different models that have been proposed in the literature (for instance, the Independent Cascade and Linear Threshold models), and hence it is desirable for the seller to be able to come up with a strategy that is applicable to a wide variety of models.

Now let us suppose that a seller has access to data from a social network such as Facebook, Orkut, or MySpace. Using this, the seller can estimate what the real, true, underlying friendship structure is, and while this estimate will not be perfect, it is getting better over time, and any information is better than none. With this information in hand, a seller can model the spread of influence quite accurately, and the formerly inscrutable problems of who to offer discounts to, and at what price, become algorithmic questions that one can legitimately hope to solve. For example, if a seller knows the structure of the network, she can locate individuals that are particularly well connected and do everything possible to ensure they adopt the product and exert their considerable influence.

In this paper, we are interested in the algorithmic side of this question: Given the network structure and a model of the purchase probabilities, how should the seller decide to offer discounts and cashback rewards?

1.1 Our contributions

We investigate seller strategies that address the above questions in the context of expected revenue maximization. We will focus much of our attention on *non-adaptive* strategies for the seller: the seller chooses and commits to a discount coupon and cashback offer for each potential buyer before the cascade starts. If a recommendation is given to this node at any time, the price offered will be the one that the seller committed to initially, irrespective of the current state of the cascade.

A wider class of strategies that one could consider are *adaptive strategies*, which do not have this restriction. For example, in an adaptive strategy, the seller could choose to observe the outcome of the (random) cascading process

up until the last minute before making very well informed pricing decisions for each node. One might imagine that this additional flexibility could allow for potentially large improvements over non-adaptive strategies. Unfortunately, there is a price to be paid, in that good adaptive strategies are likely to be very complicated, and thus difficult and expensive to implement. The ratio of the revenue generated from the optimal adaptive strategy to the revenue generated from the optimal non-adaptive strategy is termed the “adaptivity gap”.

Our main theoretical contribution is a very efficient non-adaptive strategy whose expected revenue is within a constant factor of the optimal revenue from an *adaptive* strategy. This guarantee holds for a wide variety of probability functions, including natural extensions of both the Linear Threshold and Independent Cascade models³. Note that a surprising consequence of this result is that the adaptivity gap is constant, so one can make the case that not much is lost by restricting our attention to non-adaptive policies. We also show that the problem of finding an optimal non-adaptive strategy is NP-hard, which means an efficient approximation algorithm is the best theoretical result that one could hope for.

Intuitively, the seller strategy we propose is based on an *influence-and-exploit* idea, and it consists of categorizing each potential buyer as either an *influencer* or a *revenue source*. The influencers are offered the product for free and the revenue sources are offered the product at a pre-determined price, chosen based on the exact probability model. Briefly, the categorization is done by finding a spanning tree of the social network with as many leaves as possible, and then marking the leaves as revenue sources and the internal nodes as influencers. We can find such a tree in near-linear time [7, 10]. Cashback amounts are chosen to be a fixed fraction of the total revenue expected from this process. The full details are presented in section 3.

In practice, we propose using this approach to find a strategy that has good global properties, and then using local search to improve it further. This kind of combination has been effective in the past, for example on the k-means problem [1]. Indeed, experiments (see section 8 in the appendix) show that combining local search with the above influence-and-exploit strategy is more effective than using either approach on its own.

1.2 Related work

The problem of *social contagion* or spread of influence was first formulated by the sociological community, and introduced to the computer science community by Domingos and Richardson [2]. An influential paper by Kempe, Kleinberg and Tardos [5] solved the *target set selection* problem posed by [2] and sparked interest in this area from a theoretical perspective (see [6]). This work has mostly been limited to the *influence maximization* paradigm, where influence has been taken to be a proxy for the revenue generated through a sale. Although similar

³ More precisely, the strategy achieves a constant-factor approximation for any *fixed* model, independent of the social network. If one changes the model, the approximation factor does vary, as made precise in Section 3.

to our work in spirit, there is no notion of price in this model, and therefore, our central problem of setting prices to encourage influence spread requires a more complicated model.

A recent work by Hartline, Mirrokni and Sundararajan [4] is similar in flavor to our work, and also considers extending social contagion ideas with pricing information, but the model they examine differs from our model in a several aspects. The main difference is that they assume that the seller is allowed to approach arbitrary nodes in the network at any time and offer their product at a price chosen by the seller, while in our model the cascade of recommendations determines the timing of an offer and this cannot be directly manipulated. In essence, the model proposed in [4] is akin to advertising the product to arbitrary nodes, bypassing the network structure to encourage a desired set of early adopters. Our model restricts such direct advertising as it is likely to be much less effective than a direct recommendation from a friend, especially when the recommender has an incentive to convince the potential buyer to purchase the product (for instance, the recommender might personalize the recommendation, increasing its effectiveness). Despite the different models, the algorithms proposed by us and [4] are similar in spirit and are based on an *influence-and-exploit* strategy.

This work has also been inspired by a direction mentioned by Kleinberg [6], and is our interpretation of the informal problem posed there (of moving influence to edges). Finally, we point out that the idea of cashbacks has been implemented in practice, and new retailers are also embracing the idea [8,9,14]. We note that some of the systems being implemented by retailers are quite close to the model that we propose, and hence this problem is relevant in practice.

2 The Formal Model

Let us start by formalizing the setting stated above. We represent the social network as an undirected graph $G(V, E)$, and denote the initial set of adopters by $S^0 \subseteq V$. We also denote the *active* set at time t by S^{t-1} (we call a node *active* if it has purchased the product and *inactive* otherwise). Given this setting, the recommendations cascade through the network as follows: at each time step $t \geq 1$, the nodes that became active at time $t - 1$ (i.e. S^0 for $t = 1$, and $u \in S^{t-1} \setminus S^{t-2}$ for $t \geq 2$) send recommendations to their currently inactive friends in the network: $N^{t-1} = \{v \in V \setminus S^{t-1} \mid (u, v) \in E, u \in S^{t-1} \setminus S^{t-2}\}$. Each such node $v \in N^{t-1}$ is also given a price $c_{v,t} \in \mathbb{R}$ at which it can purchase the product. This price is chosen by the seller to either be full price or some discounted fraction thereof.

The node v must then decide whether to purchase the product or not (we discuss this aspect in the next section). If v does accept the offer, a fixed cashback $r > 0$ is given to a recommender $u \in S^{t-1}$ (note that we are fixing the cashback to be a positive constant for all the nodes as the nodes are assumed to be non-strategic and any positive cashback provides incentive for them to provide recommendations). If there are multiple recommenders, the buyer must choose only one of them to receive the cashback; this is a system that is quite standard

in practice. In this way, offers are made to all nodes $v \in N^{t-1}$ through the recommendations at time t and these nodes make a decision at the end of this time period. The set of active nodes is then updated and the same process is repeated until the process quiesces, which it must do in finite time since any step with no purchases ends the process.

In the model described above, the only degree of freedom that the seller has is in choosing the prices and the cashback amounts. It wants to do this in a way that maximizes its own expected revenue (the expectation is over randomness in the buyer strategies). Since the seller may not have any control over the seed set, we are looking for a strategy that can maximize the expected revenue starting from any seed set on any graph. In most online scenarios, producing extra copies of the product has negligible cost, so maximizing expected revenue will also maximize expected profit.

Now we can formally state the problem of finding a revenue maximizing strategy as follows:

Problem 1. Given a connected undirected graph $G(V, E)$, a seed set S^0 , a fixed cashback amount r , and a model \mathbf{M} for determining when nodes will purchase a product, find a strategy that maximizes the expected revenue from the cascading process described above.

In this work, we restrict attention to policies that choose a single price for a node (as against a price per edge) as otherwise this would lead to strategizing by the buyers. We are particularly interested in non-adaptive policies, which correspond to choosing a price for each node in advance, making the price independent of both the time of the recommendation and also the state of the cascade at the time of the offer. Our goal will be threefold: (1) to show that this problem is NP-hard even for simple models \mathbf{M} , (2) to construct a constant-factor approximation algorithm for a wide variety of models, and (3) to show that restricting to non-adaptive policies results in at most a constant factor loss of profit.

To simplify the exposition, we will assume the cashback $r = 0$ for now. At the end of Section 4, we will show how the results can be generalized to work for positive r , which should be sufficient incentive for buyers to pass on recommendations.

2.1 Buyer decisions

In this section, we discuss how to model the probability that a node will actually buy the product given a set of recommendations and a price. We use a very general model in this work that naturally extends the most popular traditional models proposed in the influence maximization literature, including both Independent Cascade and Linear Threshold.

Consider an abstract model \mathbf{M} for determining the probability that a node will buy a product given a price and what recommendations it has received. To model the lack of information about individual node preferences, we will assume that each node's value for the product is drawn from a known distribution. Further, the distribution is dependent on the neighbors of the node that already own

the product. Thus, we will assume that the value of the product for a node with degree d , $k \geq 1$ of whose neighbors already have the product, is drawn at random from a distribution specified by the cumulative distribution function $F_{k,d}(\cdot)$, i.e. $F_{k,d}(t)$ is the probability that the node's value is less than t . Note that such a node will buy the product at an offer price x with probability $1 - F_{k,d}(x)$. For technical reasons, it is convenient to work with the inverse of $1 - F_{k,d}(x)$, which we call $C_{k,d}(x)$, i.e. if a node with degree d and k active neighbors is offered the product at price $C_{k,d}(x)$, it buys with probability x .⁴ We allow the model M to be fairly general, imposing only the following conditions:

1. We assume that the seller has full information about the different functions $C_{k,d}(\cdot)$ for all $k \in \{1, 2, \dots, d\}$ and $d \in \{1, 2, \dots, n\}$.⁵
2. The functions $C_{k,d}(\cdot)$ have bounded domain, which we assume to be $[0, 1]$ for all k, d w.l.o.g. This only means that a node will never pay more than full price for the product, where the full price is assumed to be 1. Without an assumption like this, the seller could potentially achieve unbounded revenue on a network, which makes the problem degenerate.
3. We also assume that $C_{k,d}(1) = 0$ and $C_{k,d}(0) = 1$ for any k, d . In other words, a node will always accept the product and recommend it to friends if it receives a recommendation with price 0. Since nodes are given positive cash rewards for making recommendations, this condition is true for any rational buyer. Also, we assume that each node has non-trivial interest in the item and hence a node's value will never exceed the full price of the product.
4. All the functions $C_{k,d}(\cdot)$ are assumed to be differentiable at 0 and 1.

Now, given a model M that satisfies the stated assumptions, we make the following observations (the proof is in the appendix):

Lemma 1. *For a cascading process proceeding according to a model M that satisfies the above assumptions, the following properties hold:*

1. *If the social network is a single line graph with S^0 being the two endpoints, the maximum expected revenue for the optimal choice of prices is at most a constant L .*
2. *There exist constants f, c, q so that if more than fraction f of a given node's neighbors recommend the product to the node at cost c , the node will purchase the product with probability q .*

Intuitively, the first property states that each prospective buyer on a social network should have some chance of rejecting the product (unless it's given to them for free), and therefore the maximum revenue on a line is bounded

⁴ It is sometimes useful to consider functions $p(\cdot)$ that are not one-to-one. These functions have no formal inverse, but in this case, c can still be formally defined as $C(x) = \max_y |1 - F_{k,d}(y)| \geq x$.

⁵ The different functions can be approximated in practice by running experiments and observing people's behavior. Also, this general model encompasses a wide variety of models, many of which require estimating a much smaller number of parameters.

by a geometric series, and is therefore constant. The second property rules out extreme inertia, for example the case where no buyer will consider purchasing a product unless almost all of its neighbors have already done so.

Next, we define a parameter that captures how complicated the model M is, and our final approximation bound will be in terms of this parameter. Since the assumptions stated above are quite general, model complexity enables us to parametrize the model in terms of the profit/long-term investment tradeoff (property 1) and “nice” myopic prices (property 2). Specific values of model complexity for instances of well-known cascade models are also provided below.

Definition 1 *The “model complexity” of a model M is defined to be $\frac{L}{(1-f)cq}$.*

The following corollary is immediately obtained from the proof of lemma 1:

Corollary 1. *A model M that satisfies all the above stated assumptions has constant model complexity.*

While it may not be obvious that all the above assumptions are met in general, we will show that they are for both the Independent Cascade and Linear Threshold models, and indeed, the arguments there extend naturally to many other cases as well.

In the traditional Independent Cascade model, there is a fixed probability p that a node will purchase a product each time it is recommended to them. These decisions are made independently for each recommendation, but each node will buy the product at most once. To generalize this to multiple prices, it is natural to make p a function $[0, 1] \rightarrow [0, 1]$ where $p(x)$ represents the probability that a node will buy the product at price x . We can express this in terms of our cumulative distribution function as: $F_{k,d}(x) = 1 - (1 - p(x))^k$. Again, we can work with the inverse of $F_{k,d}(x)$. In fact, since $F_{k,d}(\cdot)$ has a *decoupled* form we find it easier to work with the inverse of p which we call C' . Our general conditions on the model reduce to setting $C'(0) = 1$ and $C'(1) = 0$ in this case. The last two assumptions can now be stated along with the model as follows:

Definition 2 *Fix a cost function $C' : [0, 1] \rightarrow [0, 1]$ with $C'(0) = 1, C'(1) = 0$ and with C' differentiable at 0 and 1. We define the Independent Cascade Model ICM_C as follows:*

Every time a node receives a recommendation at price $C'(x)$, it buys the product with probability x and does nothing otherwise. If a node receives multiple recommendations, it performs this check independently for each recommendation but it never purchases the product more than once.

The following lemma follows immediately from the proof of lemma 1.

Lemma 2. *Fix a cost function C' . Then:*

1. ICM_C has bounded (model) complexity.
2. If C' has maximum slope m (i.e. $|C'(x) - C'(y)| \leq m|x - y|$ for all x, y), then ICM_C has $O(m^2)$ complexity.

3. If C' is a step function with n regularly spaced steps (i.e. $C'(x) = C'(y)$ if $\lfloor \frac{x}{n} \rfloor = \lfloor \frac{y}{n} \rfloor$), then ICM_C has $O(n^2)$ complexity.

In the traditional Linear Threshold model, there are fixed influences $b_{v,w}$ on each directed edge (v, w) in the network. Each node independently chooses a threshold θ uniformly at random from $[0, 1]$, and then purchases the product if and when the total influence on it from nodes that have recommended the product exceeds θ .

To generalize this to multiple prices, it is natural to make $b_{v,w}$ a function $[0, 1] \rightarrow [0, 1]$ where $b_{v,w}(x)$ indicates the influence v exerts on w as a result of recommending the product at price x . To simplify the exposition, we will focus on the case where a node is equally influenced by all its neighbors. (This is not strictly necessary but removing this assumptions requires rephrasing the definition of f to be a *weighted* fraction of a node's neighbors.) Finally, we assume for all v, w that $b_{v,w}(0) = 1$ to satisfy the second general condition for models.

Definition 3 Fix a max influence function $B : (0, 1] \rightarrow [0, 1]$, not uniformly 0. We define the Linear Threshold Model LTM_B as follows: Every node independently chose a threshold θ uniformly at random from $[0, 1]$. A node will buy the product at price $x > 0$ only if $B(x) \geq \frac{\theta}{\alpha}$ where α denotes the fraction of the node's neighbors that have recommended the product. A node will always accept a recommendation if the product is offered for free.

Lemma 3. Fix a max influence function B and let $K = \max_x x \cdot B(x)$. Then LTM_B has complexity $O(\frac{1}{K})$.

We omit the proof since it is similar to that of Lemma 1. In fact, it is simpler since, on a line graph, a node either gets the product for free or it has probability at most $\frac{1}{2}$ of buying the product and passing on a recommendation.

3 Approximating the Optimal Revenue

In this section, we present our main theoretical contribution: a non-adaptive seller strategy that achieves expected revenue within a constant factor of the revenue from the optimal *adaptive* strategy. We show the problem of finding the exact optimal strategy is NP-hard (see section 7.1 in the appendix), so this kind of result is the best we can hope for. Note that our approximation guarantee is against the strongest possible optimum, which is perhaps surprising: it is unclear a priori whether such a strategy should even exist.

The strategy we propose is based on computing a *maximum-leaf spanning tree* (MAXLEAF) of the underlying social network graph, i.e., computing a spanning tree of the graph with the maximum number of leaf nodes. The MAXLEAF problem is known to be NP-Hard, and it is in fact also MAX SNP-Complete, but there are several constant-factor approximation algorithms known for the problem [3, 7, 10, 15]. In particular, one of these is nearly linear-time [10], making it practical to apply on large online social network graphs. The seller strategy we

attain through this is an *influence-and-exploit* strategy that offers the product to all of the interior nodes of the spanning tree for free, and charges a fixed price from the leaves. Note that this strategy works for all the buyer decision models discussed above, including multi-price generalizations of both Independent Cascade and Linear Threshold.

We consider the setting of Problem 1, where we are given an undirected social network graph $G(V, E)$, a seed set $S^0 \subseteq V$ and a buyer decision model M . Throughout this section, we will let L , f , c and q denote the quantities that parametrize the model complexity, as described in Section 2.1. To simplify the exposition, we will assume that the seed set is a singleton node (i.e., $|S^0| = 1$). If this is not the case, the seed nodes can be merged into a single node, and we can make much the same argument in that case. We will also ignore cashbacks for now, and return to address them at the end of the section.

The exact algorithm we will use is stated below:

- Use the MAXLEAF algorithm [10] to compute an approximate max-leaf spanning tree T for G that is rooted at S_0 .
- Offer the product to each internal node of T for free.
- For each leaf of T (excluding S_0), independently flip a biased coin. With probability $\frac{1+f}{2}$, offer the product to the node for free. With probability $\frac{1-f}{2}$, offer the product to the node at cost c .

We henceforth refer to this strategy as STRATEGYMAXLEAF.

Our analysis will revolve around what we term as “good” vertices, defined formally as follows:

Definition 4 *Given a graph $G(V, E)$, we define the good vertices to be the vertices with degree at least 3 and their neighbors.*

On the one hand, we show that if G has g good vertices, then the MAXLEAF algorithm will find a spanning tree with $\Omega(g)$ leaves. We then show that each leaf of this tree leads to $\Omega(1)$ revenue, implying STRATEGYMAXLEAF gives $\Omega(g)$ revenue overall. Conversely, we can decompose G into at most g line-graphs joining high-degree vertices, and the total revenue from these is bounded by $gL = O(g)$ for all policies, which gives the constant-factor approximation we need.

We begin by bounding the number of leaves in a max-leaf spanning tree. For dense graphs, we can rely on the following fact [7, 10]:

Fact 1 *The max-leaf spanning tree of a graph with minimum degree at least 3 has at least $n/4 + 2$ leaves [7, 10].*

In general graphs, we cannot apply this result directly. However, we can make any graph have minimum degree 3 by replacing degree-1 vertices with small, complete graphs and by contracting along edges to remove degree-2 vertices. We can then apply Fact 1 to analyze this auxiliary graph, which leads to the following result (all the proofs for this section are in the appendix):

Lemma 4. *Suppose a connected graph G has n_3 vertices with degree at least 3. Then G has a spanning tree with at least $\frac{n_3}{8} + 1$ leaves.*

We must further extend this to be in terms of the number of good vertices g , rather than being in terms of n_3 :

Lemma 5. *Given an undirected graph G with g good vertices, the MAXLEAF algorithm [10] will construct a spanning tree with $\max(\frac{g}{50} + 0.5, 2)$ leaves.*

We can now use this to prove a guarantee on the performance of STRATEGYMAXLEAF in terms of the number of good vertices on an arbitrary graph:

Lemma 6. *Given a social network G with g good vertices, STRATEGYMAXLEAF guarantees an expected revenue of $\Omega((1-f)cq \cdot g)$.*

Now that we have computed the expected revenue from STRATEGYMAXLEAF , we need to characterize the optimal revenue to bound the approximation ratio. This bound is given by the following lemma.

Lemma 7. *The maximum expected revenue achievable by any strategy (adaptive or not) on a social network G with g good vertices is $O(L \cdot g)$.*

Now, we can combine the above lemmas to state the main theorem of the paper, which states that STRATEGYMAXLEAF provides a constant factor approximation guarantee for the revenue.

Theorem 1. *Let K denote the complexity of our buyer decision model M . Then, the expected revenue generated by STRATEGYMAXLEAF on an arbitrary social network is $O(K)$ -competitive with the expected revenue generated by the optimal (adaptive or not) strategy.*

Proof. This follows immediately from Lemmas 6 and 7, as well as the fact that $K = \frac{L}{(1-f)cq}$.

As a corollary, we get the fact that the adaptivity gap is also constant:

Corollary 2. *Let K denote the complexity of our buyer decision model M . Then the adaptivity gap is $O(K)$.*

Now we briefly address the issue of cashbacks that were ignored in this exposition. We set the cashback r to be a small fraction of our expected revenue from each individual r_0 , i.e. $r = z \cdot r_0$, where $z < 1$. Then, our total profit will be $n \cdot r_0 \cdot (1 - z)$. Adding this cashback decreases our total profit by a constant factor that depends on z , but otherwise the argument now carries through as before, and nodes now have a positive incentive to pass on recommendations.

In light of Corollary 2, one might ask whether the adaptivity gap is not just 1. In other words, is there any benefit at all to be gained from using non-adaptive strategies? In fact, there is. For example, consider a social network consisting of 4 nodes $\{v_1, v_2, v_3, v_4\}$ in a cycle, with v_3 connected to two other isolated vertices. Suppose furthermore that a node will accept a recommendation with probability 0.5 unless the price is 0, in which case the node will accept it with

probability 1. On this network, with seed set $S^0 = \{v_1\}$, the optimal adaptive strategy is to always demand full price unless exactly one of v_2 and v_4 purchases the product initially, in which case v_3 should be offered the product for free. This beats the optimal non-adaptive strategy by a factor of 1.0625.

4 Conclusions

In this work, we discussed pricing strategies for sellers distributing a product over social networks through viral marketing. We show that computing the optimal (one that maximizes expected revenue) non-adaptive strategy for a seller is NP-Hard. In a positive result, we show that there exists a non-adaptive strategy for the seller which generates expected revenue that is within a constant factor of the expected revenue generated by the optimal adaptive strategy. This strategy is based on an *influence-and-exploit* policy which computes a max-leaf spanning tree of the graph, and offers the product to the interior nodes of the spanning tree for free, later on exploiting this influence by extracting its profit from the leaf nodes of the tree. The approximation guarantee of the strategy holds for fairly general conditions on the probability function.

5 Open Questions

The added dimension of pricing to influence maximization models poses a host of interesting questions, many of which are open. An obvious open question from this work is the characterization of non-trivial adaptive strategies. In particular, is there an adaptive strategy which achieves optimal revenue? It is also unclear whether the assumptions on the function $C(\cdot)$ are the minimal set that is required, and it would be interesting to remove the assumption that there exists a price at which the probability of acceptance is 1. A different direction of research would be to consider the game-theoretic issues involved in a practical system. Namely, in the model presented here, we think of each buyer as just sending the recommendations to all its friends and ignore the issue of any “cost” involved in doing so, thereby assuming all the nodes to be non-strategic. It would be very interesting to model a system where the nodes were allowed to behave strategically, trying to maximize their payoff, and characterize the optimal seller strategy (especially w.r.t. the cashback) in such a setting.

6 Acknowledgments

Supported in part by NSF Grant ITR-0331640, TRUST (NSF award number CCF-0424422), and grants from Cisco, Google, KAUST, Lightspeed, and Microsoft. The third author is grateful to Jason Hartline and Mukund Sundararajan for useful discussions.

References

1. David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.
2. P. Domingos and M. Richardson. Mining the network value of customers. *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 57–66, 2001.
3. M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. WH Freeman & Co. New York, NY, USA, 1979.
4. J. Hartline, V. Mirrokni, and M. Sundararajan. Optimal Marketing Strategies over Social Networks. *Proceedings of the 17th international conference on World Wide Web*, 2008.
5. D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146, 2003.
6. J. Kleinberg. Cascading Behavior in Networks: Algorithmic and Economic Issues. In N. Nisan, T. Roughgarden, E. Tardos, and V.V. Vazirani, editors, *Algorithmic Game Theory*. Cambridge University Press New York, NY, USA, 2007.
7. D.J. Kleitman and D.B. West. Spanning Trees with Many Leaves. *SIAM Journal on Discrete Mathematics*, 4:99, 1991.
8. J. Leskovec, A. Singh, and J. Kleinberg. Patterns of influence in a recommendation network. *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2006.
9. Jure Leskovec, Lada A. Adamic, and Bernardo A. Huberman. The dynamics of viral marketing. *ACM Trans. Web*, 1(1):5, 2007.
10. H.I. Lu and R. Ravi. Approximating Maximum Leaf Spanning Trees in Almost Linear Time. *Journal of Algorithms*, 29(1):132–141, 1998.
11. A. Mislove, M. Marcon, K.P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 29–42. ACM New York, NY, USA, 2007.
12. MEJ Newman, DJ Watts, and SH Strogatz. Random graph models of social networks, 2002.
13. BBC News. Facebook valued at \$15 billion. <http://news.bbc.co.uk/2/hi/business/7061042.stm>, 2007.
14. Erick Schonfeld. Amiando makes tickets go viral and widgetizes event management. <http://www.techcrunch.com/2008/07/17/amiando-makes-tickets-go-viral-and-widgetizes-event-management-200-discount-for-techcrunch-readers/>, 2008.
15. R. Solis-Oba. 2-Approximation Algorithm for finding a Spanning Tree with Maximum Number of leaves. *Proceedings of the Sixth European Symposium on Algorithms*, pages 441–452, 1998.
16. Wikipedia. The beacon advertisement system. http://en.wikipedia.org/wiki/Facebook_Beacon, 2008.
17. Wikipedia. Facebook revenue in 2008. <http://en.wikipedia.org/wiki/Facebook>, 2008.

7 Appendix

7.1 Hardness of finding the optimal strategy

In this section, we show that Problem 1 is NP-hard even for a very simple buyer model \mathbf{M} by a reduction from vertex cover with bounded degree (see [3] for the hardness of bounded-degree vertex cover). Letting d denote the degree bound, and letting $p = \frac{1}{4d}$, we will use an Independent Cascade Model ICM_C with:

$$C(x) = \begin{cases} 1 & \text{if } x < p, \\ 0 & \text{if } x \geq p \end{cases}$$

Intuitively, the seller has to partition the nodes into “free” nodes and “full-price” nodes. In the former case, nodes are offered the product for free, and they accept it with probability 1 as soon as they receive a recommendation. In the latter case, nodes are offered the product for price 1, and they accept each recommendation with probability p . (Note that the seller is *allowed* to use other prices between 0 and 1 but a price of 1 is always better.)

We are going to use a special family of graphs illustrated in Figure 1. The graph consists of four layers:

- A singleton node s , which we will use as the only initially active node (i.e., $S^0 = \{s\}$);
- s links to a set of n nodes, denoted by V_1 ;
- Nodes in V_1 also link to another set of nodes, denoted by V_2 . Each node in V_1 will be adjacent to d nodes in V_2 , and each node in V_2 will be adjacent to 2 nodes in V_1 (so $|V_2| = dn/2$);
- Each node $v \in V_2$ also links to $k = 20d$ new nodes, denoted by W_v ; these k nodes do not link to any other nodes. The union of all W_v 's is denoted by V_3 .

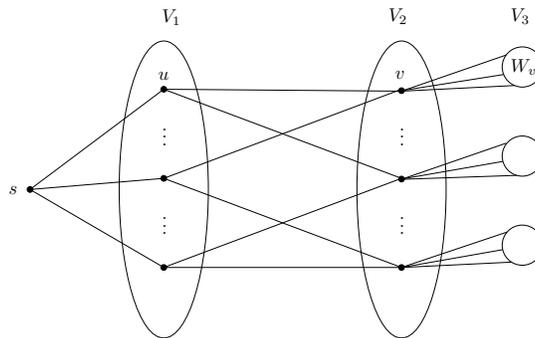


Fig. 1. Reducing Bounded-Degree Vertex Cover to Optimal Network Pricing

We first sketch the idea of the hardness proof. The connection between V_1 and V_2 will be decided by the vertex cover instance: given a vertex cover instance $G'(V, E)$ with bounded degree d , we construct a graph G as above where $V_1 = V$ and $V_2 = E$, adding an edge between V_1 and V_2 if the corresponding vertex is incident to the corresponding edge in G' . The key lemma is that, in the optimal pricing strategy for G , the subset of nodes in V_1 that are given the product for free is the minimum set that covers V_2 (i.e., a minimum vertex cover of G').

To formalize this, first note that, in an optimal strategy, all nodes in V_3 should be full-price. Giving the product to them for free gets 0 immediate revenue, and offers no long-term benefit since nodes in V_3 cannot recommend the product to anyone else. If the nodes are full-price, on the other hand, there is at least a chance at some revenue.

On the other hand, we show the optimal strategy must also ensure each vertex in V_2 eventually becomes active with probability 1.

Lemma 8. *In an optimal strategy, every node $v \in V_2$ is free, and can be reached from s by passing through free nodes.*

Proof. Suppose, by way of contradiction, that the optimal strategy has a node $v \in V_2$ that does not satisfy these conditions. Let u_1 and u_2 be the two neighbors of v in V_1 , and let q denote the probability that v eventually becomes active.

We first claim that $q < 2dp$. Indeed, if v is full-price, then even if u_1 and u_2 become active, the probability that v becomes active is $1 - (1 - p)^2 < 2p$. Otherwise, u_1 and u_2 are both full-price. Since u_1 and u_2 connect to at most $2d$ edges other than v , the probability that one of them becomes active before v is at most $1 - (1 - p)^{2d} < 2dp$. Thus, $q < 2dp$.

It follows that the total revenue that this strategy can achieve from u_1 , v , and W_v is $2 + kqp < 2 + 2kdp^2 = 4.5$. Conversely, if we make u_1 and v free, we can achieve $kp = 5$ revenue from the same buyers. Furthermore, doing this cannot possibly lose revenue elsewhere, which contradicts the assumption that our original strategy was optimal. \square

It follows that, in an optimal strategy, all of V_3 is full-price, all of V_2 is free, and every node in V_2 is adjacent to a free node in V_1 . It remains only to determine C , the nodes in V_1 , that an optimal strategy should make free. At this point, it should be intuitively clear that C should correspond to a minimum vertex-cover of V_2 . We now formalize this.

Lemma 9. *Let C denote the set of free nodes in V_1 , as chosen by an optimal strategy. Then C corresponds to a minimum vertex cover of G' .*

Proof. As noted above, every node in V_2 must be adjacent to a node in C , which implies C does indeed correspond to a vertex cover in G' .

Now we know an optimal strategy makes every node in V_2 free, and every node in V_3 full-price. Once we know C , the strategy is determined completely. Let x_C denote the expected revenue obtained by this strategy. Since all nodes in V_2 are free and are activated with probability 1, we know the strategy achieves 0 revenue from V_2 and $p|V_3|$ expected revenue from V_3 .

Among nodes in V_1 , the strategy achieves 0 revenue for free nodes, and exactly $1 - (1 - p)^{d+1}$ expected revenue for each full-price node. This is because each full-price node is adjacent to exactly $d+1$ other nodes, and each of these nodes is activated with probability 1. Therefore, $x_C = (|V_1| - |C|)(1 - (1 - p)^{d+1}) + p|V_3|$, which is clearly minimized when C is a minimum-vertex cover. \square

Therefore, optimal pricing, even in this limited scenario, can be used to calculate the minimum-vertex cover of any bounded-degree graph, from which NP-hardness follows.

Theorem 2. *Two Coupon Optimal Strategy Problem is NP-Hard.*

7.2 Proofs

Proof (of Lemma 1). Fix k and d such that the parameters m and $1/\epsilon$ (defined shortly) are maximized. For the remainder of this proof, we drop the subscripts and denote $C_{k,d}(\cdot)$ by C . We show that the complexity of \mathbb{M} can be bounded in terms of the maximum slope of C near 0 and 1. Recall that if C is differentiable at 0, then, by definition, there exists $\epsilon > 0$ so that $\frac{|C(x) - C(0)|}{x} \leq |C'(0)| + 1$ for $x < \epsilon$. A similar argument can be made for $x = 1$, and thus we can say formally that there exist m and $\epsilon > 0$ such that:

$$\begin{aligned} C(x) &\geq 1 - mx \quad \text{for } x \leq \epsilon, \text{ and} \\ C(x) &\leq m(1 - x) \quad \text{for } x \geq 1 - \epsilon. \end{aligned}$$

We begin by analyzing L_n , the maximum expected revenue that can be achieved on a path of length n if *one* of the endpoints is a seed. Note that $L \leq 2 \max_n L_n$ since selling a product on a line graph with two seeds can be thought of as two independent sales, each with one seed, that are cut short if the sales ever meet. Now we have:

$$L_n = \max_x x(C(x) + L_{n-1}).$$

This is because offering the product at cost $C(x)$ will lead to a purchase with probability x , and in that case, we get $C(x)$ revenue immediately and L_{n-1} expected revenue in the future. Since L_n is obviously increasing in n , this can be simplified further:

$$\begin{aligned} L_n &\leq \max_x x(C(x) + L_n) \\ \implies L &\leq 2L_n \leq \max_{0 < x < 1} \frac{2x \cdot C(x)}{1 - x} \end{aligned}$$

For $x \geq 1 - \epsilon$, we have $\frac{2x \cdot C(x)}{1 - x} \leq \frac{2x \cdot m(1 - x)}{1 - x} \leq 2m$, and for $x < 1 - \epsilon$, we have $\frac{2x \cdot C(x)}{1 - x} \leq \frac{2}{\epsilon}$. Either way, $L \leq 2 \max(\frac{1}{\epsilon}, m)$.

It remains to show that there exists a choice of f , c and q that satisfies the second condition in the lemma. We use $f = 0$, $q = \min(\epsilon, \frac{1}{2m})$ and $c =$

$C(q) \geq \frac{1}{2}$. Indeed, if a node has more than 0 active neighbors, it will accept a recommendation at cost $C(q)$ with probability q . We note that $\frac{L}{(1-f)cq} \leq 8 \max(\frac{1}{\epsilon}, m)^2$.

□

Proof (of Lemma 4). Let n_1 and n_2 denote the number of vertices of degree 1 and 2 respectively, and let M denote the number of leaves in a max-leaf spanning tree of G . If $n_1 = n_2 = 0$, the result follows from Fact 1.

Now, suppose $n_2 = 0$ but $n_1 > 0$. Clearly, every spanning tree has at least n_1 leaves, so the result is obvious if $n_1 \geq \frac{n_3}{8} + 1$. Otherwise, we replace each degree-1 vertex with a copy of K_4 (the complete graph on 4 vertices), one of whose vertices connects back to the rest of the graph. Let G' denote the resulting graph. Then G' has $4n_1 + n_3$ vertices, and they are all at least degree 3, so G' has a spanning tree T' with at least $n_1 + \frac{n_3}{4} + 2$ leaves.

We can transform this into a spanning tree T on G by contracting each copy of K_4 down to a single point. Each contraction could transform up to 3 leaves into a single leaf, but it will not affect other leaves. Since there are exactly n_1 contractions that need to be done altogether, T has at least $n_1 + \frac{n_3}{4} + 2 - 2n_1 \geq \frac{n_3}{8} + 1$ leaves, as required.

We now prove the result holds in general by induction on n_2 . We have already shown the base case ($n_2 = 0$). For the inductive step, we will define an auxiliary graph G' with n'_2, n'_3 and M' defined as for G . We will then show $n'_2 < n_2, n'_3 \geq n_3$, and for every spanning tree T' on G' , there is a spanning tree T on G with at least as many leaves. This implies $M' \leq M$, and using the inductive hypothesis, it follows that $M \geq M' \geq \frac{n'_3}{8} + 1 \geq \frac{n_3}{8} + 1$, which will complete the proof.

Towards that end, suppose v is a degree-2 vertex in G , and let its neighbors be u and w . If u and w are not adjacent, we let G' be the graph attained by contracting along the edge (u, v) . Then $n'_2 = n_2 - 1$ and $n'_3 = n_3$. Any spanning tree T' on G' can be extended back to a spanning tree T on G by uncontracting the edge (u, v) and adding it to T . This does not decrease the number of leaves in the tree, so we are done.

Next, suppose instead that u and w are adjacent. We cannot contract (u, v) here since it will create a duplicate edge in G' . However, a different construction can be used. If the entire graph is just these 3 vertices, the lemma is trivial. Otherwise, let G' be the graph attained by adding a degree-1 vertex x adjacent to v . Then $n'_2 = n_2 - 1$ and $n'_3 = n_3 + 1$. Now consider a spanning tree T' of G' . We can transform this into a spanning tree T on G by removing the edge (v, x) that must be in T' . This removes the leaf x but if v has degree 2 in T' , it makes v a leaf. In this case, T and T' have the same number of leaves, so we are done.

Otherwise, (u, v) and (v, w) are also in T' , and since G was assumed to have more than 3 vertices, u and w cannot both be leaves in T' . Assume without loss of generality that u is not a leaf. We then further modify T by replacing (v, w) with (u, w) . Now, v is a leaf in T and the only vertex whose degree has changed is u , which is not a leaf in either T or T' . Therefore, T and T' again have the same number of leaves, and we are once again done.

The result now follows from induction, as discussed above.

Proof (of Lemma 5). If $g = 0$, the result is trivial. Otherwise, let n_3 denote the number of vertices in G with degree at least 3, and let M denote the number of leaves in a max-leaf spanning tree of G . By Lemma 4, we know $M \geq \frac{n_3}{8} + 1$.

Now consider constructing a spanning tree as follows:

1. Let A denote the set of vertices in G with degree at least 3.
2. Set T to be a minimal subtree of G that connects all vertices in A .
3. Add all remaining vertices in G to T one at a time. If a vertex v could be connected to T in multiple ways, connect it to a vertex in A if possible.

To analyze this, note that $G - A$ can be decomposed into a collection of “primitive” paths. Given a primitive path P , let g_P denote the number of good vertices on P and let l_P denote the number of leaves T has on P .

In Step 2 of the algorithm above, exactly $n_3 - 1$ of these paths are added to T . For each such path P , we have $g_P \leq 2$ and $l_P = 0$. On the remaining paths, we have $g_P = l_P$. Therefore, the total number of leaves on T is at least

$$\sum_P l_P = (g - n_3) + \sum_P (l_P - g_P) \geq (g - n_3) - 2(n_3 - 1).$$

Thus,

$$\begin{aligned} M &\geq \max\left(\frac{n_3}{8} + 1, g - 3n_3 + 1\right) \\ &\geq \frac{24}{25} \cdot \left(\frac{n_3}{8} + 1\right) + \frac{1}{25} \cdot (g - 3n_3 + 1) = \frac{g}{25} + 1 \end{aligned}$$

The result now follows from the fact that the MAXLEAF algorithm gives a 2-approximation for the max-leaf spanning tree, and that every non-degenerate tree has at least two leaves.

Proof (of Lemma 6). Let T denote the spanning tree found by the MAXLEAF algorithm. Let U denote the set of interior nodes of T , and let V denote the leaves of T (excluding S_0). Since we assumed $|S_0| = 1$, Lemma 5 guarantees $|V| \geq \max(\frac{g}{50} - 0.5, 1) = \Omega(g)$.

Note every vertex can be reached from S_0 by passing through nodes in U , each of which is offered the product for free. These nodes are guaranteed to accept the product, and therefore, they will collectively pass on at least one recommendation to each vertex.

Now consider the expected revenue from a vertex $v \in V$. Let M be the random variable giving the fraction of v 's neighbors in V that were *not* offered the product for free. We know $E[M] = \frac{1-f}{2}$, so with probability $\frac{1}{2}$, we have $M \leq 1 - f$.

In this case, v is guaranteed to receive recommendations from a fraction f of its neighbors in V , as well as all of its neighbors in $U \cup S_0$ (of which there is at least 1). If we charge v a total of c for the product, it will then purchase the product with probability at least q , by the original definitions of f , c and q . Furthermore, independent of v 's neighbors, we will ask this price from v with probability $\frac{1-f}{2}$. Therefore, our expected revenue from v is at least $\frac{1}{2} \cdot q \cdot \frac{1-f}{2} \cdot c$.

The result now follows from linearity of expectation.

Proof (of Lemma 7). Let A denote the set of vertices in G with degree at least 3, and let $n_3 = |A|$. Clearly, no strategy can achieve more than n_3 revenue directly from the nodes in A .

As observed in the proof of Lemma 5, however, $G - A$ can be decomposed into a collection of primitive paths. Since each primitive path contains at least one unique good vertex with degree less than 3, there is at most $g - n_3$ such paths. Even if each endpoint of a path is guaranteed to recommend the product, the total revenue from the path is at most L .

Therefore, the total revenue from any strategy on such a graph is at most $n_3 + (g - n_3)L = O(L \cdot g)$.

8 Local Search

In this section, we discuss how an arbitrary seller strategy can be tweaked by the use of a local search algorithm. Taken on its own, this technique can sometimes be problematic since it can take a long time to converge to a good strategy. However, it performs very well when applied to an already good strategy, such as STRATEGYMAXLEAF. This approach of combining theoretically sound results with local search to generate strong techniques in practice is similar in spirit to the recent **k-means++** algorithm [1].

Intuitively, the local search strategy for pricing on social networks works as follows:

- Choose an arbitrary seller strategy S and an arbitrary node v to edit.
- Choose a set of prices $\{p_1, p_2, \dots, p_k\}$ to consider.
- For each price p_i , empirically estimate the expected revenue r_i that is achieved by using the price p_i for node v .
- If any revenue r_i beats the current expected revenue (also estimated empirically) by some threshold ϵ , then change S to use the price p_i for node v .
- Repeat the preceding steps for different nodes until there are no more improvements.

Henceforth, we call this the LOCALSEARCH algorithm for improving seller strategies.

To empirically estimate the revenue from a seller strategy, we can always just simulate the entire process. We know who has the product initially, we know what price each node will be offered, and we know the probability each node will purchase the product at that price after any number of recommendations. Simulating this process a number of times and taking the average revenue, we can arrive at a fair approximation at how good a strategy is in practice. In fact, we can prove that performing local search on any input policy will ensure that the seller gets at least as much revenue as the original policy with high probability. The proof of this fact holds for any simulatable input policy, and proceeds by induction on the evolution tree of the process. The proof is somewhat technical,

so we will skip it, and instead focus on the empirical question of the advantage provided by local search.

In light of the fact that local search can only improve the revenue (and never hurt it), it seems that one should always implement local search for any policy. There is an important technical detail that complicates this, however. Suppose we wish to evaluate strategies S_1 and S_2 , differing only on one node v . If we independently run simulations for each strategy, it could take thousands of trials (or more!) before the systematic change to one node becomes visible over the noise resulting from random choices made by the other nodes. It is impractical to perform these many simulations on a large network every time we want to change the strategy for a single node.

Fortunately, it is possible to circumvent this problem using an observation first noted in [5]. Let us consider the Linear Threshold model LTM_B . In this case, all randomness occurs before the process begins when each node chooses a threshold that encodes how resistant it is to buying the product. Once these thresholds have been fixed, the entire sales process is deterministic. We can now change the strategy slightly and *maintain the same thresholds* to isolate exactly what effect this strategy change had. Any model, including Independent Cascade, can be rephrased in terms of thresholds, making this technique possible.

The LOCALSEARCH algorithm relies heavily on this observation. While comparing strategies, we choose several threshold lists, and simulate each strategy against the same threshold lists. If these lists are not representative, we might still make a mistake drawing conclusions from this, but we will not lose a universally good signal or a universally bad signal under the weight of random noise.

With this implementation, empirical tests (see the next section) show the LOCALSEARCH algorithm does its job: given enough time, it will improve virtually any strategy enough to be competitive. It is not a perfect solution, however. First of all, it can still make small mistakes while doing the random estimates, possibly causing a strategy to become worse over time⁶. Secondly, it is possible to end up with a sub-optimal strategy that simply cannot be improved by any local changes. Finally, the LOCALSEARCH algorithm can often take many steps to improve a bad strategy, making it occasionally too slow to be useful in practice.

Nonetheless, these drawbacks really only become a serious problem if one begins with a bad strategy. If one begins with a relatively good strategy – for example STRATEGYMAXLEAF – the LOCALSEARCH algorithm performs well, and is almost always worth doing in practice. We justify this claim in the following section.

⁶ Note that if we choose ϵ and the number of trials carefully, we can make this possibility vanishingly small (this is also the intuition behind the local search guarantee, as we had mentioned earlier. In practice, however, it is usually better to run fewer trials and accept the possibility of regressing slightly.

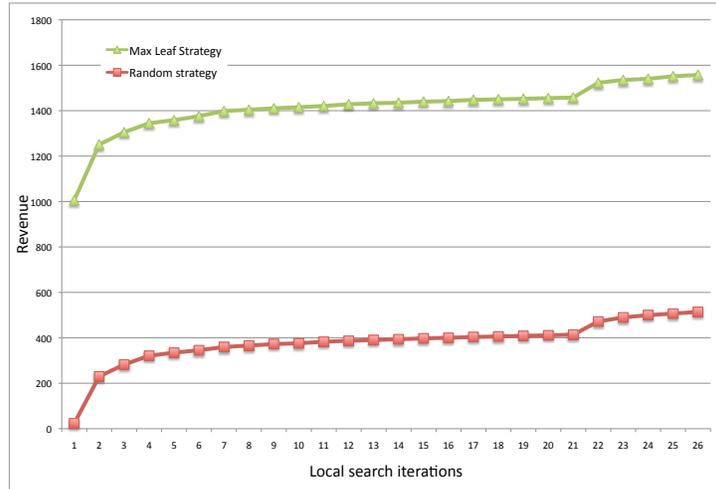
8.1 Experimental Results

In this section, we provide experimental evidence for the efficacy of the LOCALSEARCH algorithm in improving the revenue guarantee. Note that in these experiments, we need to assume a benchmark strategy as finding the optimal strategy is NP-hard (see section 7.1). We pick a very simple strategy RANDOMPRICING, which picks a random price independently for each node. The results demonstrate that even this naive strategy can be coupled with the LOCALSEARCH algorithm to do well in practice.

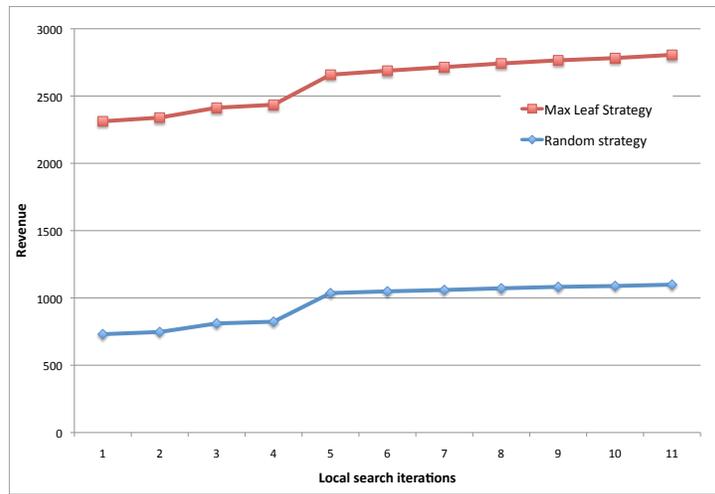
We simulate the cascading process on two kind of graphs. The first graph we study is a randomly generated graph, based on the preferential attachment model that is a popular model for representing social networks [12]. We generate a 1000 node preferential attachment graph at random, and simulate the cascading process by picking a random node as the seed in the network. The probability model we examine is a step function (see the second example given in Lemma 2) of probabilities. We note that the function is necessarily arbitrary. The result of one particular parameter settings are shown in figure 2(a), which plots average revenue obtained by the two pricing strategies: RANDOMPRICING and STRATEGYMAXLEAF. Each point on the figure is obtained by average revenue over 10 runs on the same graph but with a different (random) seed. The horizontal axis indicates the number of LOCALSEARCH iterations that were done on the graph, where each iteration consisted of simulating the process 50 times, and choosing the best value over the runs. It is clear from the graph that STRATEGYMAXLEAF does quite well even without the addition of LOCALSEARCH, although the addition of LOCALSEARCH does increase the revenue. On the other hand, the RANDOMPRICING strategy performs poorly on its own, but its revenue increases steadily with the iterations of the LOCALSEARCH algorithm. We note that the difference between the revenue from the two policies does vary (as expected) with the probability model, and the difference between the revenue is not as large in all the different runs. But the difference does persist across the runs, especially when the strategies are run without the local search improvement.

We also conduct a similar simulation with a real-world network, namely the links between users of the video-sharing site YouTube.⁷ The YouTube network has millions of nodes, and we only study a subset of 10,000 nodes of the network. We simulate the random process as earlier, and the results are shown in figure 2(b). Again, we note that STRATEGYMAXLEAF does very well on its own, easily beating the revenue of RANDOMPRICING. The RANDOMPRICING strategy does improve a lot with LOCALSEARCH, but it fails to equalize the revenue of STRATEGYMAXLEAF. The large size of the YouTube graph and the expensive nature of the LOCALSEARCH algorithm restrict the size of the experiments we can conduct with the graph, but the results from the above does experiments do offer some insights. In particular, STRATEGYMAXLEAF succeeds in extracting a good portion of the revenue from the graph, if we consider the revenue obtained from STRATEGYMAXLEAF combined with LOCALSEARCH based improvements to

⁷ The network can be freely downloaded; see [11] for details.



(a) Random preferential attachment graph



(b) Youtube subgraph

Fig. 2. The variation in revenue generated by RANDOMPRICING and STRATEGYMAX-LEAF with the iterations of the LOCALSEARCH algorithm. The data is averaged over 10 runs of a 1000 node random preferential attachment graph (a) or a 10000 node subgraph of YouTube (b), starting with a random seed each time.

be the benchmark. Further, LOCALSEARCH can improve the revenue from any strategy by a substantial margin, though it may not be able to attain enough revenue when starting with a sub-optimal strategy such as RANDOMPRICING. Finally, we observe that the combination of STRATEGYMAXLEAF and LOCAL-

SEARCH generates the best revenue among our strategies, and it is an open question as to whether this is the optimal adaptive strategy.