Optimal Budget Allocation: Theoretical Guarantee and Efficient Algorithm

Tasuku Soma TASUKU_SOMA@MIST.I.U-TOKYO.AC.JP
Graduate School of Information Science and Technology, The University of Tokyo, Tokyo, 113-8656, and JST, ERATO, Kawarabayashi Large Graph Project, Tokyo, 101-8430

Naonori Kakimura KAKIMURA@GLOBAL.C.U-TOKYO.AC.JP
College of Arts and Sciences, The University of Tokyo, Tokyo, 153-8902

Kazuhiro Inaba KINABA@GOOGLE.COM
Google Inc.

Ken-ichi Kawarabayashi K_KENITI@NII.AC.JP
National Institute of Informatics, JST, ERATO, Kawarabayashi Project, Tokyo, 101-8430

Abstract

We consider the budget allocation problem over bipartite influence model proposed by Alon et al. (Alon et al., 2012). This problem can be viewed as the well-known influence maximization problem with budget constraints.

We first show that this problem and its much more general form fall into a general setting; namely the *monotone submodular function maximization over integer lattice subject to a knapsack constraint*. Our framework includes Alon et al.'s model, even with a competitor and with cost.

We then give a $(1 - 1/e)$-approximation algorithm for this more general problem. Furthermore, when influence probabilities are non-increasing, we obtain a faster $(1 - 1/e)$-approximation algorithm, which runs essentially in linear time in the number of nodes. This allows us to implement our algorithm up to almost 10M edges (indeed, our experiments tell us that we can implement our algorithm up to 1 billion edges. It would approximately take us only 500 seconds.).

1. Introduction

Domingos and Richardson (Domingos & Richardson, 2001; Richardson & Domingos, 2002) introduced *viral marketing*, which is a cost-effective marketing strategy that promotes products by giving "free" or "discounted" items to a selected group of highly influential individuals, in the hope that through the word-of-mouth effects, a large number of product adoption will occur. In the same papers, Domingos and Richardson (Domingos & Richardson, 2001; Richardson & Domingos, 2002) considered the following problem: Suppose we have data on a social network, with estimates for the extent to which individuals influence one another, and we would like to market a new product that hopefully will be adopted by a large fraction of the network. How should we choose a few "influential" members of the network that can "trigger" a cascade of influence? This problem, called *influence maximization*, is to find a small set of the most influential individuals (which is sometimes called a *seed* node set) in a social network so that their aggregated influence in the network is maximized. The seminal work by Kempe, Kleinberg and Tardos (Kempe et al., 2003) provides the first systematic study of influence maximization as a combinatorial optimization problem. The influence maximization problem further motivated the research community to conduct extensive studies on various aspects of the influence maximization problems (e.g., (Chen et al., 2010; 2009)).

Previous work mentioned so far has only focused on the selection of a subset of influencing seeds. But there is one more important factor we have to consider in the context of an algorithmic marketing approach; namely *Budgets*. Certainly one of the major decisions in a marketing plan deals with the allocation of a given budget among media channels such as TV, newspaper, and webs, in order to maximize the impact on a set of potential customers.

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

This prompts Alon et al. (Alon et al., 2012) to consider the following influence maximization problem with budget constraints (which is called a *(source-node) bipartite influence model*). First, we may model as a bipartite graph in which one side is the set of possible marketing channels, and the other is the population of customers. An edge between a channel i and a customer j indicates that i may influence j with some probability that depends on the budget allocated to i. Formally, the model consists of a bipartite graph $G = (S, T; E)$, where S and T are collections of source nodes and target nodes, respectively, and $E \subseteq S \times T$ is an edge set. Each source node s has a capacity $c(s) \in \mathbf{Z}_+$ and probabilities $p_s^{(i)} \in [0, 1]$ for $i = 1, \ldots, c(s)$. Each source node s will be allocated a budget $b(s) \in \{0, 1, \ldots, c(s)\}$ such that $\sum_{s \in S} b(s) \leq B$, where $B \in \mathbf{Z}_+$ denotes a *total budget capacity*. If a source node s is allocated a budget of $b(s)$, the node s makes $b(s)$ independent trials to activate each neighboring target node t. The probability that t is activated by s in the ith trial is $p_s^{(i)}$. That is, the probability that t becomes active is

$$1 - \prod_{s \in \Gamma(t)} \prod_{i=1}^{b(s)} (1 - p_s^{(i)}), \tag{1}$$

where $\Gamma(t)$ denotes the set of source nodes that is adjacent to t. The objective of this model is to distribute the budget among the source nodes respecting the capacities of nodes, and to maximize the expected number of target nodes that become active. We call this problem the *budget allocation problem over bipartite influence model*.

Alon et al. (Alon et al., 2012) devised a $(1 - 1/e)$-approximation algorithm for this problem. Here, for a positive number $\alpha \leq 1$, an α-approximation algorithm is one returning a feasible solution whose value is at least α times the optimal value. Their algorithm is essentially greedy with some preprocess (i.e., enumerating all the possible allocations with only three source nodes). The algorithm motivates us to consider the following issues.

1. The *scalability*. Their algorithm adapts a greedy procedure. Therefore, the time complexity is expensive, because in each iteration, the greedy procedure searches all the nodes in the graph as a potential candidate for the next seed node. As a result, the algorithm entails at least quadratic number of steps in terms of the number of nodes. In fact, not only due to this issue, but also due to the expensive preprocess, their algorithm would not work for more than 5K edges, say[1].

2. *Submodularity*. For a finite set S, we say that a set function $f : 2^S \to \mathbf{R}$ is *submodular* if it satisfies $f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y)$ for all $X, Y \subseteq S$. A set function f is *monotone* if $f(X) \leq f(Y)$ for all X, Y with $X \subseteq Y$. Submodularity often arises in analyzing influence maximization in a social network (e.g., (Chen et al., 2010; 2009; Kempe et al., 2003)), and guarantees why a greedy algorithm finds a nearly optimal solution (Fisher et al., 1978). In the budget allocation problem, one may select a node multiple times, and hence the objective function is not a set function. As suggested by Alon et al., however, their algorithm is similar to the one for maximizing a monotone submodular set function subject to a knapsack constraint (Sviridenko, 2004), and hence it seems to us that submodularity can explain the problem setting.

3. *Extending the framework*. The budget allocation problem needs to be extended to adopt more complicated real settings. Alon et al. only considered the case when cost to allocate a budget is the same for all sources. Since each media channel such as TV, newspaper, and webs has different scales, it would be more natural to consider different costs for each channel. Moreover, a market may often have a competitor against us, and we have to change our allocation according to the competitor's decision.

[1] Indeed, as far as we are aware, nobody has implemented their algorithm.

Our Contribution

In this paper, we clarify all the issues above. Let us first consider the second and the third issues. To deal with choosing a node multiple times in the model, we extend a set function to a function over the integer lattice, i.e., the set of the integer vectors in the Euclidean space. We define a *submodular function over integer lattice*, that is, a function f satisfying $f(x) + f(y) \geq f(x \vee y) + f(x \wedge y)$ for all integer vectors x and y, where $x \vee y$ and $x \wedge y$ denote the coordinate-wise maxima and minima, respectively. This definition generalizes submodularity of a set function, because if we restrict the domain to the unit hypercube then the submodularity can be identified with one for a set function. Note that a function f over integer lattice is usually supposed to be given as an *evaluation oracle*, a black box that computes $f(x)$ for any integer vector x.

A submodular set function has been studied extensively in combinatorial optimization. There are combinatorial polynomial-time algorithms for minimizing a submodular set function (see (Fujishige, 2005; Iwata, 2008) for details). It is known in (Fujishige, 2005) that these minimization algorithms can cope with submodular functions over integer lattice if bounded, which can be identified with a ring family on some ground set with pseudo-polynomial size, and hence the minimization problem can be solved in pseudo-polynomial time. On the other hand, maximizing a submodular function over integer lattice has been done for only special classes with applications to multi-unit combinato-

rial auctions (Shioura, 2009).

Having defined submodular functions over integer lattice, our main theoretical contributions in this paper are the following.

- We introduce a general framework using a submodular function over integer lattice. We consider the problem of maximizing a monotone submodular function subject to a knapsack constraint, which is a natural extension of maximizing a submodular set function with constraints (Nemhauser et al., 1978; Sviridenko, 2004).

  We then devise a $(1-1/e)$-approximation algorithm for this problem. Since the budget allocation problem is shown to fall into the problem, this gives a solution of the second issue.

- We show that our framework includes more generalized models in which the source costs are nonuniform and there is a competitor. Our first result implies that the model can also be solved in polynomial time within $(1-1/e)$ approximation factor. The result can be compared to some results in algorithmic game theory (Bharathi et al., 2007; Borodin et al., 2010; Budak et al., 2011).

In the *competitor model*, we aim at allocating our budget against a competitor's allocation, which is known in advance. Such a situation is practical when we consider advertising (i.e., imagine television commercial messages. Companies can prepare their strategies based on competitors' information). This competitor model also considers the case when the competitor spreads information in advance (i.e., the competitor spreads the information even before we start spreading our information). This means that we can take "start-up delay" against the competitor into account.

It should be noted that our submodular framework admits a potential applicability to other problems than the budget allocation problem. In fact, classical combinatorial optimization problems such as the maximum coverage and facility location can be naturally generalized to our framework over integer lattice. Such problems can be applied to *text summarization* and *sensor placement* in machine learning. These applications will be discussed in Section 3.3.

Let us now consider the first issue. Relatively little work has been done on improving the quadratic nature of the greedy algorithm, even for the influence maximization problem without the budget constraint. Perhaps the most notable work is (Leskovec et al., 2007), where submodularity is exploited to develop an efficient heuristic algorithm, based on "lazy-forward" optimization in selecting seeds. The idea is that the marginal gain of a node in the current iteration cannot be much better than its marginal gain in the previous iterations. Then this algorithm exploits submodularity of influence spread function to reduce the number of Monte-Carlo simulations. Empirical results show 700 times faster than the original greedy algorithm, however it still takes a few hours to compute a solution for graphs with tens of thousands of vertices. Subsequently, various heuristics (Chen et al., 2009; 2010; Jiang et al., 2011) have been proposed to avoid using Monte-Carlo simulations, however, these algorithms do not guarantee any theoretical guarantee. Recently, a theoretically fast algorithm is proposed by (Borgs et al., 2014).

In this paper, we deal with the budget allocation problem with nonincreasing influence probabilities. Let us observe that this problem setting often appears in the real world. Indeed, the impact of advertisement is supposed to be nonincreasing. Using the submodularity of the budget allocation problem, we obtain a faster $(1-1/e)$-approximation algorithm, which runs essentially in linear time in the number of nodes. This allows us to implement our algorithm up to almost 10M edges (indeed, our experiments tell us that we can implement our algorithm up to 1 billion edges. It would approximately take us only 500 seconds.). As far as we are aware, this is perhaps the first case that the greedy $(1-1/e)$-approximation algorithm for the submodularity scales up to this large size.

The rest of the paper is organized as follows. In Section 2, we design a $(1-1/e)$-approximation algorithm for the monotone submodular function maximization over integer lattice subject to a knapsack constraint. In Section 3, we apply it to the budget allocation problems and other problems. Section 4 describes faster algorithms for the budget allocation problem assuming that influence probabilities are nonincreasing. Numerical experiments are presented in Section 5. Finally, Section 6 concludes the paper.

2. Submodular Maximization over Integer Lattice subject to Knapsack Constraint

Let S be a finite set. We say that a function f over integer lattice is *submodular* if it satisfies $f(x)+f(y) \geq f(x \vee y)+f(x \wedge y)$ for all x and y, where $x \vee y$ and $x \wedge y$ denote the coordinate-wise maxima and minima, respectively, i.e., $(x \vee y)(s)=\max\{x(s), y(s)\}$ and $(x \wedge y)(s)=\min\{x(s), y(s)\}$. A function f is *monotone* if $f(x) \leq f(y)$ for all x and y with $x \leq y$. We assume that $f(0)=0$ without loss of generality.

Let $c \in \mathbf{Z}_+^S$, $w \in \mathbf{R}_+^S$ and B be a nonnegative integer. Consider the problem to find $b \in \mathbf{Z}_+^S$ maximizing a monotone submodular function f subject to $0 \leq b \leq c$ and $w \cdot b := \sum_{s \in S} w(s)b(s) \leq B$. We call this problem the *monotone submodular function maximization over integer lattice subject to a knapsack constraint*. We may assume

without loss of generality that $w(s) > 0$ for each s in S.

This problem is a generalization of the problem of maximizing a monotone submodular set function subject to a knapsack constraint. Indeed, if c is the all-one vector, then f can be identified with a monotone submodular set function. In this case, we can find a $(1 - 1/e)$-approximate solution in polynomial time (Sviridenko, 2004). The main result of this section is the following for a general case.

Theorem 2.1. *For the monotone submodular function maximization over integer lattice subject to a knapsack constraint, we can find a $(1-1/e)$-approximate solution in $O(B^5|S|^4\theta)$ time, where θ is the running time of an evaluation oracle for f.*

Note that the complexity is also bounded by using $c_{\max}$, the maximum entry of c, instead of B. The bound is $O(c_{\max}^4|S|^5\theta)$, and this corresponds to the bound by Sviridenko (Sviridenko, 2004) for submodular set function maximization subject to a knapsack constraint, where $c_{\max} = 1$.

We here summarize basic properties for submodular functions over integer lattice. The *marginal return* with respect to a feasible solution b, $s \in S$ and $k \in \mathbf{Z}_+$ is the value $f(b+k\chi_s)-f(b)$, where χ_s is the characteristic vector of s. We denote this value by $\Delta(b, s, k)$. The *average marginal return* with respect to a feasible solution $b \in \mathbf{Z}_+^S$, $s \in S$ and $k \in \mathbf{Z}_+$ is the value $\Delta(b, s, k)/(w(s)k)$, which we denote by $\delta(b, s, k)$.

It is well-known that for a set function f, submodularity is equivalent to *the diminishing marginal return property*, i.e., $f(X\cup\{s\})-f(X) \geq f(Y\cup\{s\})-f(Y)$ for all $X \subseteq Y \subseteq S$ and $s \notin Y$. However, submodularity over integer lattice does not imply the *diminishing marginal return property*:

$$f(b + \chi_s) - f(b) \geq f(b + 2\chi_s) - f(b + \chi_s) \quad (2)$$

for arbitrary b and $s \in S$. On the other hand, a weaker version of this inequality does hold, whose proof can be found in the appendix.

Lemma 2.2. *Let f be a monotone submodular function over integer lattice. For arbitrary $s \in S$, $k \in \mathbf{Z}_+$, x and y with $x \leq y$, we have*

$$f(x \vee k\chi_s) - f(x) \geq f(y \vee k\chi_s) - f(y). \quad (3)$$

The *(positive) support* of b, denoted by $\mathrm{supp}^+(b)$, is the set of elements s in S such that $b(s) > 0$.

Lemma 2.3. *Let f be a submodular function. For arbitrary x and y, we have*

$$f(x \vee y) \leq f(x) + \sum_{s \in \mathrm{supp}^+(y-x)} (f(x \vee y(s)\chi_s) - f(x)). \quad (4)$$

2.1. $(1 - 1/e)$-Approximation Algorithm

We next describe our approximation algorithm. In our algorithm, we first enumerate every feasible solution b_0 such that $|\mathrm{supp}^+(b_0)| \leq 3$. The number of such solutions is $O(B^3|S|^3)$. For each feasible solution b_0, the algorithm increases each component of b_0 in a greedy way using GREEDYPROCEDURE, whose description is presented in Algorithm 1, and obtains a feasible solution b. Finally, the algorithm returns the best one among the obtained solutions. Since Algorithm 1 requires $O(B^2|S|\theta)$ time, the total complexity is $O(B^5|S|^4\theta)$. Note that the correctness of our algorithm is omitted due to the space limitation, which can be found in the appendix.

Algorithm 1 GREEDYPROCEDURE

Input: a feasible solution b_0

Let $b := b_0$.
Let $Q := \{(s, k) : s \in S \text{ and } 1 \leq k \leq c(s) - b(s)\}$.
while $w \cdot b < B$ and $Q \neq \emptyset$ **do**
  Find s and k maximizing the average marginal return $\delta(b, s, k)$ among $(s, k) \in Q$.
  if $w \cdot b + w(s)k \leq B$ **then**
    Let $b(s) := b(s) + k$.
    Remove all pairs (s, l) such that $b(s) + l > c(s)$ from Q.
  else
    Remove (s, k) from Q.
  end if
end while

Let us leave some remarks on our algorithm. Although the worst case complexity of our algorithm is quite expensive, Algorithm 1 can be accelerated using a "lazy evaluation" technique given in (Minoux, 1978), and thus it runs more efficiently in practice. Also, there is another approach to reduce the time complexity at the expense of a good approximation ratio. We can apply ideas similar to those for maximizing a submodular set function to make practical simpler algorithms with somewhat worse approximation factors (Lin & Bilmes, 2010). For example, as in (Alon et al., 2012), if we enumerate solutions with only one positive support before performing GREEDYPROCEDURE, then we can find a $1/2(1 - 1/e)$-approximate solution in $O(B^2|S|^2\theta)$ time.

2.2. Faster Algorithm under Diminishing Marginal Return Property

We conclude this section with designing a faster $(1 - 1/e)$-approximation algorithm if f satisfies the diminishing marginal return property (2) and a cost w is *uniform*, i.e., $w(s) = 1$ for any $s \in S$. Note that even if f admits the properties, the problem is still NP-hard and $(1 - 1/e)$-approximation is best possible unless P=NP (Feige, 1998).

We execute GREEDYPROCEDURE with $b_0 := 0$. By (2), we have $\delta(b, s, k) \leq \delta(b, s, 1)$ for arbitrary b, s and k. Hence GREEDYPROCEDURE always chooses $k = 1$ for each iteration. Furthermore, the procedure never fails to increase the tentative solution because w is uniform. Therefore, by following a similar argument to the proof of Theorem 2.1, we can show that the output is a $(1 - 1/e)$-approximate solution. The running time follows from the fact that b increases B times and each iteration requires $O(|S|\theta)$ time to find s maximizing $\delta(s, b, 1)$. Thus the following holds.

Theorem 2.4. *For the submodular function maximization over integer lattice, if f admits the diminishing marginal return property and w is uniform, then we can find a $(1 - 1/e)$-approximate solution in $O(B|S|\theta)$ time.*

3. Applications of Submodular Maximization

In this section, we focus on the *budget allocation problem* (as defined in Section 1), and we show that more generalized models fall into our framework. We also mention applications to other problems such as text summarization and sensor placement in Section 3.3.

3.1. Budget Allocation Problem with Nonuniform Costs

We extend to the bipartite influence model with *nonuniform costs*. That is, in addition to an instance of the budget allocation problem, each source node s has a cost $w(s) > 0$. The objective is to distribute the budget respecting the capacities of nodes and $\sum_{s\in S} w(s)b(s) \leq B$ such that the expected number of activated nodes is maximized. We call this problem the *budget allocation problem over bipartite influence model with nonuniform costs*.

In the model, the probability that t becomes active is equal to (1). Let f_t be a function of an allocation vector b defined as (1). We note that we define $f_t(b) = -\infty$ if b does not satisfy $0 \leq b \leq c$. It is easy to see that the function f_t is monotone for each $t \in T$. In addition, the following lemma asserts that f_t is submodular.

Lemma 3.1. *For any $t \in T$, the function f_t is submodular.*

Proof. It suffices to show $g_t(b) := \prod_{s\in\Gamma(t)} \prod_{i=1}^{b(s)} (1-p_s^{(i)})$ is supermodular. Let $\alpha := g_t(x \wedge y)$, $\beta := g_t(x)/\alpha$ and $\gamma := g_t(y)/\alpha$. We can easily check that $\alpha, \beta, \gamma \in [0, 1]$ and $g_t(x \vee y) = \alpha\beta\gamma$. Then we obtain $g_t(x \vee y) + g_t(x \wedge y) - g_t(x) - g_t(y) = \alpha(\beta\gamma + 1 - \beta - \gamma) = \alpha(1 - \beta)(1 - \gamma) \geq 0$. □

Since the expected number of activated nodes for an allocation b is equal to $f(b) := \sum_{t\in T} f_t(b)$, the expected number is also submodular as a function of b. The following result is now immediate from Theorem 2.1, since $\theta = O(B|S||T|)$.

Theorem 3.2. *A $(1 - 1/e)$-approximate solution for the budget allocation problem with nonuniform costs can be found in $O(B^6|S|^5|T|)$ time.*

3.2. Budget Allocation Problem with a Competitor

We extend the budget allocation problem with nonuniform costs to the two-player case. In the model, there is a competitor against an advertiser. The competitor allocates his/her budget to S in advance, and will try to influence target nodes at the same time as the advertiser's trials. Under this situation, the advertiser aims at allocating a budget to source nodes to maximize the expected number of target nodes influenced by his/her trials. We suppose that the trials of the two players are made in a discrete time step: At each time step i, first the competitor makes the ith trial, and then the advertiser makes the ith trial. The trials will be repeated until both budget allocations run out.

Thus each target node t of T has the following three states: inactive, positively active (influenced by an advertiser), and negatively active (influenced by a competitor). Since an advertiser is a follower, we assume that it has a chance to activate positively both an inactive node and a negatively activated node. Note that when a node is already influenced by the competitor, the probability to activate it should be smaller than one to activate an inactive node. In contrast, we suppose that the competitor can activate an inactive node, but not a positively active node. Thus the model is progressive with respect to positively active nodes.

More specifically, the model is defined as follows. For a bipartite graph $G = (S, T; E)$, an advertiser is given a capacity $c(s)$, a cost $w(s)$ and two probabilities $p_s^{(i)}$ and $q_s^{(i)}$ with $q_s^{(i)} \leq p_s^{(i)}$ for $i = 1, \ldots, c(s)$ for each $s \in S$. In addition, a competitor has already allocated his budget to source nodes, which is denoted by $\tilde{b}(s)$ for each source node $s \in S$. The competitor also has probabilities $\tilde{p}_s^{(i)}$ for $i = 1, \ldots, \tilde{b}(s)$ for each s in S. The probability that t is activated by s in the ith trial depends on the status of t as follows. If t is inactive, then the probabilities that t is positively/negatively activated are $p_s^{(i)}$ and $\tilde{p}_s^{(i)}$, respectively. If t is already negatively active, then the probability that t is positively activated is $q_s^{(i)}$. In this setting, we aim at maximizing the expected number of positively active nodes. For $t \in T$ and k with $1 \leq k \leq 1 + \max_{s\in S} \tilde{b}(s)$, let $E_{t,k}$ be the event that t becomes negatively active in the kth trial (when $k = 1 + \max_{s\in\Gamma(t)} \tilde{b}(s)$, $E_{t,k}$ means the event that t never become negatively active). Conditioned on $E_{t,k}$, the probability that t becomes positively active is

$$f_{t,k}(b) = 1 - \prod_{s\in\Gamma(t)} \prod_{i=1}^{\min\{b(s),k-1\}} \left(1 - p_s^{(i)}\right) \prod_{i=k}^{b(s)} \left(1 - q_s^{(i)}\right).$$

The probability that t becomes positively active equals

$\sum_k \lambda_{t,k} f_{t,k}(b)$, where $\lambda_{t,k} := \Pr(E_{t,k})$. Therefore, the expected number of positively active nodes, denoted by $f(b)$, is equal to $\sum_{t\in T}\sum_k \lambda_{t,k} f_{t,k}(b)$. Similarly to Lemma 3.1, $f_{t,k}$ is shown to be monotone and submodular for any t and k. Since f is a nonnegative linear combination of $f_{t,k}$'s, f is also monotone and submodular. Thus this problem can be reduced to the monotone submodular function maximization over integer lattice.

Theorem 3.3. *For the budget allocation problem with a competitor, the objective function is monotone and submodular. Thus we can find a $(1-1/e)$-approximate solution in polynomial time.*

3.3. Other Applications

In this section, we present applications of our framework to problems other than the budget allocation problems. Submodular set functions arise in several applications (see, e.g., (Krause & Golovin, 2014)). Such applications can naturaly be extended to our framework allowing multiple choices.

Maximum Coverage Let us first see that our framework includes a generalization of the well-known maximum coverage problem. In the *maximum coverage problem*, we are given a finite set V and a family of subsets $C_1,\dots,C_m$ in V, and the objective is to choose k subsets from $C_1,\dots,C_m$ so that the number of the covered elements in V is maximized. Here consider covering functions $p_j : \mathbf{Z}_+ \to 2^V (j = 1,\dots,m)$, each of which is monotone, and we would like to maximize the number of the elements covered by p_j's, i.e., $|\cup_{j=1}^m p_j(b_j)|$, subject to $\sum_{j=1}^m b_j \le k$. Note that the covering function p_j corresponds to the situation where choosing j multiple times makes the covered set larger. This problem clearly generalizes the maximum coverage problem, that corresponds to the case when we can only choose each p_j at most once. It is not difficult to see that the objective function is a monotone submodular function over integer lattice, and therefore a $(1-1/e)$-approximate solution can be found in polynomial time by Theorem 2.1.

Facility Location We are given a set V of facilities, and we aim at deciding how large facilities are opened up in order to serve a set of m customers, where we represent scale of facilities as integers $0,1,\dots,c$ ("0" means we do not open a facility). If we open up a facility j of scale b_j, then it provides service of value $p_{i,j}(b_j)$ to customer i, where $p_{i,j} : \mathbf{Z}_+ \to \mathbf{R} (j = 1,\dots,m)$ is a given monotone function. We suppose that each customer chooses the opened facility with highest value. That is, when we assign b_j to each facility j, the total value provided to all customers is given by $f(b) = \sum_{i=1}^m \max_{j\in S} p_{i,j}(b_j)$. It turns out f is monotone and submodular over integer lattice. Thus a $(1-1/e)$-approximate solution can be found in polynomial time by Theorem 2.1.

Sensor Placement In a standard sensor placement scenario, we can put at most one sensor in each spot to maximize the area covered by sensors. Here we can consider a more general problem where to place sensors with some specified power in a field subject to budget constraint, where the area covered by a sensor depends on its power setting. This problem is almost equivalent to the maximum coverage problem with multiple choices as discussed before, and hence we can apply our approximation algorithm for the problem. A similar sensor placement problem where we can put two kinds of sensors in each spot is studied in (Singh et al., 2012).

Text Summarization The objective of text summarization is to find a small set of words that summarizes the feature of a given text as well as possible. Recently, it is shown that a variety of objective functions used in summarization admits submodularity and that submodular-based approaches outperform in practice (Lin & Bilmes, 2011). For example, in the *concept-based summarization* (Filatova & Hatzivassiloglou, 2004), we are to find a subset S of sentences maximizing the total credit of concepts covered by S, i.e., maximize $\sum_{i\in\Gamma(S)} c_i$, where $\Gamma(S)$ is the set of the concepts covered by S and $c_i \in \mathbf{R}_+$ is the credit of a concept i. Indeed, this objective function is submodular. We now extend the submodular summarization model to the one that incorporates "confidence" of sentences, i.e., we can choose a sentence in various confidence level like "low", "mid" or "high" rather than just choose or not. As in the maximum coverage, let us introduce a monotone covering function p_j for each sentence j. Then the objective function of the extended model is defined to be the total credit $f(b) = \sum c_i$, where the sum is taken over concept i covered by $\cup_j p_j(b_j)$. Again, this objective function is a monotone submodular function over integer lattice, and hence we can use our approximation algorithm for the extended model.

4. Faster Algorithm for Budget Allocation with Nonincreasing Influence Probabilities

In this section, we assume the budget allocation problem has *nonincreasing influence probabilities*, i.e., for each $s \in S$, we have $p_s^{(i-1)} \ge p_s^{(i)}$ for $i = 2,\dots,c(s)$. The property captures the real-world phenomena of marketing. In our model, multiple trials to activate target nodes can be viewed as discrete-time steps. Thus it is natural to decrease effectiveness of an advertisement with time.

Lemma 4.1. *Under the nonincreasing influence probabilities, the objective function f admits the diminishing marginal return property.*

By Theorem 2.4, performing GREEDYPROCEDURE with

$b_0 := 0$ yields a $(1-1/e)$-approximate solution. Algorithm 2 describes an implementation of the algorithm in Theorem 2.4 with an additional trick to compute the value of f efficiently. Here we use auxiliary variables $\phi(s)$ and $\phi(t)$ for each $s \in S$ and $t \in T$. During execution of the algorithm, $\phi(t)$ traces the value $\prod_{s \in \Gamma(t)} \prod_{i=1}^{b(s)} (1 - p_s^{(i)})$ and $\phi(s)$ is always equal to $\sum_{t \in \Gamma(s)} \phi(t)$. Thus $f(b + \chi_s) - f(b) = \sum_{t \in \Gamma(s)} (\phi(t) - (1 - p_s^{(b(s)+1)})\phi(t)) = p_s^{(b(s)+1)} \phi(s)$. Evidently the algorithm runs in $O(B(|S| + |T| + |E|))$ time. Summarizing the arguments, we have the following theorem.

Theorem 4.2. *For the budget allocation problem with nonincreasing influence probabilities, we can find a $(1 - 1/e)$-approximate solution in $O(B(|S| + |T| + |E|))$ time.*

Similarly to Algorithm 1, we can apply the lazy evaluation technique to Algorithm 2 to accelerate in practice. We use this speeding up technique in our experiments.

Algorithm 2 SIMPLEGREEDYPROCEDURE

1: Let $b := 0$, and let $\phi(t) := 1$ for each $t \in T$ and $\phi(s) := d(s)$ for each $s \in S$.
2: **for** $i = 1$ to B **do**
3: Choose s maximizing $p_s^{(b(s)+1)} \phi(s)$.
4: Let $b(s) := b(s) + 1$.
5: Let $\phi(t) := (1 - p_s^{(b(s))})\phi(t)$ for each $t \in \Gamma(s)$ and let $\phi(s) := \sum_{t \in \Gamma(s)} \phi(t)$ for each $s \in S$
6: **end for**
7: **return** b

In addition, under a similar assumption, we design a faster algorithm for the budget allocation problem with a competitor. Let us assume that $p_s^{(i-1)} \geq p_s^{(i)}$ and $q_s^{(i-1)} \geq q_s^{(i)}$ for arbitrary $s \in S$ and i with $2 \leq i \leq c(s)$. Then $f_{t,k}$ admits the diminishing marginal return property for each t and k, and hence so does f. Therefore, a greedy algorithm similar to Algorithm 2 computes a $(1 - 1/e)$-approximate solution. A pseudocode description of the greedy algorithm that runs in $O(B^2|E|)$ time is presented in the appendix.

Theorem 4.3. *If $p_s^{(i)}$ and $q_s^{(i)}$ are nonincreasing and the cost is uniform, then we can find a $(1 - 1/e)$-approximate solution for the budget allocation problem with a competitor in $O(B^2|E|)$ time.*

5. Experiments

We now verify our theoretical results by implementing Algorithm 2 for the budget allocation problem over bipartite influence model with nonincreasing influence probabilities.

To demonstrate that the greedy choice indeed performs well, the expected number $f(b)$ of activated nodes by the greedy choice is compared to that by other strategies. The

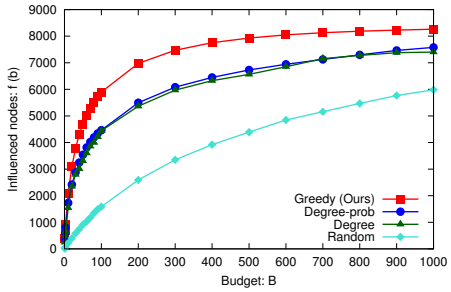

Figure 1. P=1.0, Yahoo! Bidding Data

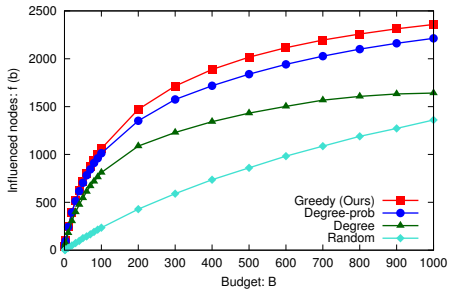

Figure 2. P=0.1, Yahoo! Bidding Data

"degree" strategy chooses the top B highest degree nodes. This strategy is based on the intuition that the high degree nodes should have high influence. In many variants of influence maximization problems, this high-degree-selection strategy is reported not to outperform the greedy choice (Kempe et al., 2003; Budak et al., 2011). The "degree-prob" strategy takes the influence probability into account, in addition to the degree condition. It chooses the top B nodes with the largest expected number of influences, i.e., degree multiplied with the probability. The "random" strategy is a baseline choice that uniformly randomly chooses B nodes.

Figures 1 and 2 are our simulation results on Yahoo! Search Marketing Advertiser Bidding Data (Yah). It is a bipartite graph between 1,000 search keywords and 10,475 accounts, where each edge represents one "bid" to advertisement on the keyword. Note that this data originally does not represent influence relationship, but it encodes the information "who is interested in what" relationship in the real world. The motivation of the experiment is to find a good choice of "what" set that maximizes the influenced "who"s, that is, a set of keywords that is maximally associated to advertisers. This would be useful for the publisher to promote keywords to advertisers. For the purpose, we only use the graph, although the original data contains bid prices or time data. The graph has 52,567 edges (representing 'who bid at least once to what' relation). The probability $p_s^{(i)}$ of influence is assigned as follows. Let P be a parameter. For each s, we have set $p_s^{(1)}$ a random value

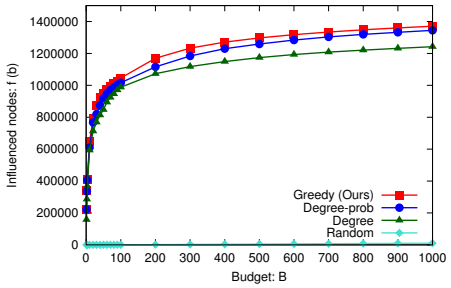

Figure 3. P=1.0, Synthesized Graph

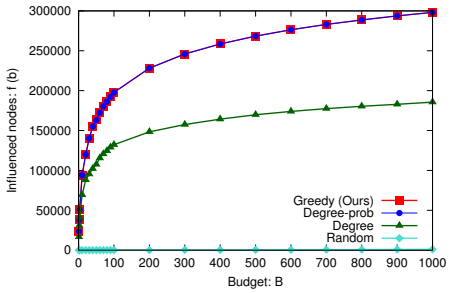

Figure 4. P=0.1, Synthesized Graph

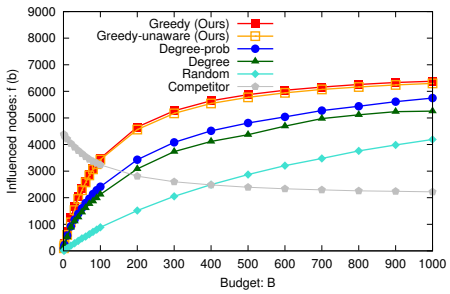

Figure 5. P=1.0, against a competitor

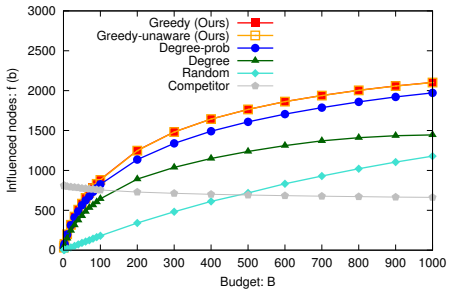

Figure 6. P=0.1, against a competitor

between 0 and P. Then, $p_s^{(i+1)}$ is set to $p_s^{(i)} \cdot X$, where X is a random value between 0 and 1, generated for each i and s. Figures 1 and 2 show the results on $P = 1$ and $P = 0.1$ cases.

In order to implement our algorithms for larger scale graphs, we have also run our experiments on artificially generated graphs. Figures 3 and 4 are simulation results on synthesized graphs with $|S| = 200,000$ and $|T| = 2,000,000$ nodes with around $8,000,000$ edges between them. The graphs are generated so that the degree distribution of the source nodes obey the *power law* of $\gamma = 2.0$. After assigning the degrees to each source node s, it is connected to $deg(s)$ nodes uniformly chosen from T. Allocation of $1,000$ budgets in the greedy algorithm took 3.36 seconds (including the graph generation time) in average on a machine with Xeon E5-2690 2.9GHz CPU and 64GB memory.

We have also implemented the algorithm when a competitor exists. We have started with the situation that the competitor already allocated 100 budgets by the "degree" strategy. The probability $q_s^{(i)}$ to turn over a node already influenced by the competitor is set to $0.2 \cdot p_s^{(i)}$. Figures 5 and 6 summarize the results in the competitive setting on the Yahoo! Bidding data graph. The "competitor" line in the figures shows the number of nodes kept influenced by the competitor against our greedy algorithm.

In all the configurations, the greedy algorithm outperformed degree-based strategies. "Degree-prob" also showed a comparable result on the synthesized graphs. Interestingly, the "greedy-unaware" strategy in the competitive setting, which is to allocate budgets greedily without taking the existence of the competitor into account at all, is showing a very close performance to the competitor-aware one. This may be explained as follows; except for the extreme situation that $p_s^{(i)}$ is very high and $q_s^{(i)}$ is very low, a highly influential set of source nodes would be taken by the advertiser, even after some part of it is grabbed by the competitor.

6. Conclusion

In this paper, we have defined a submodular function over integer lattice, extending a submodular set function, and introduced the problem of maximizing a monotone submodular function subject to knapsack constraint. This problem includes the budget allocation problem (Alon et al., 2012) and more general budget allocation problems with nonuniform costs and a competitor. Also, this problem has applications to sensor placement and text summarization. We have devised a $(1 - 1/e)$-approximation polynomial-time algorithm for the problem. In addition, based on the diminishing marginal return property, we have devised a faster algorithm for the budget allocation problem under some natural scenario, and have carried out numerical experiments.

References

Yahoo! webscope dataset: ydata-ysm-advertiser-bids-v1_0. http://research.yahoo.com/Academic_Relations.

Alon, N., Gamzu, I., and Tennenholtz, M. Optimizing budget allocation among channels and influencers. In *Proceedings of the 21st International Conference on World Wide Web*, pp. 381–388, 2012.

Bharathi, S., Kempe, D., and Salek, M. Competitive influence maximization in social networks. In *Proceedings of the 3rd International Conference on Internet and Network Economics*, WINE'07, pp. 306–311, 2007.

Borgs, C., Brautbar, M., Chayes, J., and Lucier, B. Maximizing social influence in nearly optimal time. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, SODA'14, 2014.

Borodin, A., Filmus, Y., and Oren, J. Threshold models for competitive influence in social networks. In *Proceedings of the 6th International Conference on Internet and Network Economics*, WINE'10, pp. 539–550, 2010.

Budak, C., Agrawal, D., and El Abbadi, A. Limiting the spread of misinformation in social networks. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pp. 665–674, 2011.

Chen, W., Wang, Y., and Yang, S. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 199–208. ACM, 2009.

Chen, W., Wang, C., and Wang, Y. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1029–1038. ACM, 2010.

Domingos, P. and Richardson, M. Mining the network value of customers. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 57–66. ACM, 2001.

Feige, U. A threshold of ln n for approximating set cover. *Journal of ACM*, 45:634–652, 1998.

Filatova, E. and Hatzivassiloglou, V. Event-based extractive summarization. In *Proceedings of ACL Workshop on Summarization*, 2004.

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

Fujishige, S. *Submodular Functions and Optimizations*. Number 58 in Annals of Discrete Mathematics. Elsevier, 2nd edition, 2005.

Iwata, S. Submodular function minimization. *Math. Program.*, 112(1):45–64, 2008.

Jiang, Q., Song, G., Gao, C., Wang, Y., Si, W., and Xie, K. Simulated annealing based influence maximazation in social networks. In *Proceedings of the Twenty-fifth Conference on Artificial Intelligence*, pp. 127–132. AAAI, 2011.

Kempe, D., Kleinberg, J., and Tardos, E. Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 137–146. ACM, 2003.

Krause, A. and Golovin, D. Submodular function maximization. In Bordeaux, L., Hamadi, Y., and Kohli, P. (eds.), *Tractability: Practical Approaches to Hard Problems*, to appear. Cambridge University Press, 2014.

Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., and Glance, N. S. Cost-effective outbreak detetion in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 420–429. ACM, 2007.

Lin, H. and Bilmes, J. Multi-document summarization via budgeted maximization of submodular functions. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 912–920, 2010.

Lin, H. and Bilmes, J. A class of submodular functions for document summarization. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 510–520, 2011.

Minoux, M. Accelerated greedy algorithms for maximizing submodular set functions. *Optimization Techniques, Lecture Notes in Control and Information Sciences*, 7:234–243, 1978.

Nemhauser, G., Wolsey, L., and Fisher, M. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 1978.

Richardson, M. and Domingos, P. Mining knowledge-sharing sites for viral marcketing. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 61–70. ACM, 2002.

Shioura, A. On the pipage rounding algorithm for submodular function maximization — a view from discrete convex analysis. *Discrete Math., Alg. and Appl.*, 1(1):1–24, 2009.

Singh, AP, Guillory, A., and Bilmes, J. On bisubmodular maximization. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics*, pp. 1055–1063, 2012.

Sviridenko, M. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32:41–43, 2004.

A. Proof of Theorem 2.1

Let b^* be an optimal solution for a given instance of the problem. We first examine properties of GREEDYPROCEDURE that hold for *arbitrary* b_0. We then show that there exists *some* b_0 in the enumeration step such that GREEDYPROCEDURE returns b with $f(b) \geq (1 - 1/e)f(b^*)$, which proves Theorem 2.1.

Let us fix an initial solution b_0, and analyze behavior of GREEDYPROCEDURE with input b_0. We denote by b_i the tentative solution b at the beginning of the ith iteration and

denote by s_i and k_i s and k chosen in the ith iteration, respectively. Assume that GREEDYPROCEDURE first has not updated the tentative solution b in the Lth trial. Equivalently, let L be the minimum number such that $b_L = b_{L+1}$ and $b_i < b_{i+1}$ for $i = 1, \dots, L-1$. Note that if such a situation never happens during the execution of GREEDYPROCEDURE, define L to be the number of iterations.

Lemma A.1. *Without loss of generality, we may assume that* $b(s_L) + k_L \leq b^*(s_L)$.

Proof. Suppose that $b(s_L) + k_L > b^*(s_L)$. Let us consider a modified instance in which the capacity of s_L is decreased to $b(s_L) + k_L - 1$. The optimal value is unchanged by this modification because b^* is still feasible and optimal. Furthermore, GREEDYPROCEDURE returns the same solution (with respect to same b_0). Thus it suffices to analyze the algorithm in the modified instance. Repeating this argument completes the proof of this lemma. □

Consider the ith iteration of the algorithm. For simplicity, we denote $\Delta(b_i, s_i, k_i)$ by Δ_i, $\delta(b_i, s_i, k_i)$ by δ_i, and $w(s_i)$ by w_i. Note that $f(b_i) = f(b_0) + \sum_{j=1}^{i-1} \Delta_i$ for $i = 1, \dots, L$. Let $B' := B - w \cdot b_0$.

Lemma A.2. *For* $i = 1, \dots, L$, *we have* $\Delta_i \geq \frac{w_i k_i}{B'} \left(f(b^*) - f(b_i)\right)$.

Proof. Let us denote $b_i \vee b^* = b_i + \sum \alpha_s \chi_s$, where the sum is taken over s in $\mathrm{supp}^+(b^* - b_i)$ and $\alpha_s := b^*(s) - b_i(s)$. Since $b_i \vee b^*(s)\chi_s = b_i + \alpha_s \chi_s$ for each $s \in \mathrm{supp}^+(b^* - b_i)$, (4) implies

$$\begin{aligned} f(b_i \vee b^*) &\leq f(b_i) + \sum_{s \in \mathrm{supp}^+(b^* - b_i)} \left(f(b_i + \alpha_s \chi_s) - f(b_i)\right) \\ &\leq f(b_i) + \sum_{s \in \mathrm{supp}^+(b^* - b_i)} w(s) \alpha_s \delta_i, \end{aligned}$$

where the last inequality follows from the fact that $(f(b_i + k\chi_s) - f(b_i))/(w(s)k) \leq \delta_i$ for all $s \in S$ and k. Since $\sum w(s)\alpha_s$ is at most B' and $f(b^*) \leq f(b_i \vee b^*)$ by the monotonicity of f, it holds that $f(b^*) \leq f(b_i) + \delta_i B'$. Therefore, we obtain $\delta_i \geq (f(b^*) - f(b_i))/B'$. □

Applying Lemma A.2 repeatedly, we have the following lemmas.

Lemma A.3. *For* $i = 1, \dots, L$, *we have* $f(b^*) - f(b_i) \leq (f(b^*) - f(b_0)) \cdot \prod_{j=1}^{i} (1 - w_j k_j / B')$.

Proof. We prove this lemma by induction on i. For $i = 1$, then the inequality holds because $\Delta_1 \geq w_1 k_1 (f(b^*) - f(b_0))/B'$ by Lemma A.2. For $i > 1$, we have

$$\begin{aligned} &f(b^*) - f(b_0) - \sum_{j=1}^{i} \Delta_j = f(b^*) - f(b_0) - \sum_{j=1}^{i-1} \Delta_j - \Delta_i \\ &\leq f(b^*) - f(b_0) - \sum_{j=1}^{i-1} \Delta_j - \frac{w_i k_i}{B'} \left(f(b^*) - f(b_0) - \sum_{j=1}^{i-1} \Delta_j \right) \\ &\qquad \text{(by Lemma A.2)} \\ &= \left(1 - \frac{w_i k_i}{B'}\right) \left(f(b^*) - f(b_0) - \sum_{j=1}^{i-1} \Delta_j \right) \\ &\leq \left(1 - \frac{w_i k_i}{B'}\right) \cdot (f(b^*) - f(b_0)) \cdot \prod_{j=1}^{i-1} \left(1 - \frac{w_j k_j}{B'}\right), \\ &\qquad \text{(by the induction hypothesis)} \end{aligned}$$

which completes the proof. □

Lemma A.4. *It holds that* $f(b^*) - f(b_i) \leq (f(b^*) - f(b_0))/e$.

Proof. Let $\varphi(x) = \ln(1 - x)$. Note that φ is concave in $[0, 1)$. By Jensen's inequality, we have $\sum_{i=1}^{L} \varphi(x_i)/L \leq \varphi(\sum_{i=1}^{L} x_i / L)$ for $x_1, \dots, x_L \in [0, 1)$. Putting $x_i := w_i k_i / B'$, we obtain

$$\frac{1}{L} \sum_{i=1}^{L} \ln \left(1 - \frac{w_i k_i}{B'}\right) \leq \ln \left(1 - \frac{1}{L} \sum_{i=1}^{L} \frac{w_i k_i}{B'}\right) \leq \ln \left(1 - \frac{1}{L}\right), \tag{5}$$

where the last inequality follows since $\sum_{i=1}^{L} w_i k_i \geq B'$ and φ is a monotonically decreasing function. Thus we have

$$\prod_{i=1}^{L} \left(1 - \frac{w_i k_i}{B'}\right) \leq \left(1 - \frac{1}{L}\right)^L \leq \frac{1}{e}. \tag{6}$$

Combining this fact and Lemma A.3 completes the proof. □

We now move to proving that the greedy procedure returns a $(1 - 1/e)$-approximate solution for some b_0 in the enumeration step. If the optimal solution b^* satisfies $|\mathrm{supp}^+(b^*)| \leq 3$, it can be found in the enumeration step. Therefore, we assume that all the optimal solutions have more than three positive components. Let s_1^*, s_2^*, s_3^* be elements in S satisfying:

$$s_i^* \in \operatorname*{argmax}_{s \in S \setminus \{s_1^*, \dots, s_{i-1}^*\}} \Delta \left(\vee_{j=1}^{i-1} b^*(s_j^*) \chi_{s_j^*}, s, b^*(s) \right)$$

for $i = 1, 2, 3$. Since the size of the support of b^* is more than three, such s_1^*, s_2^* and s_3^* clearly exist.

Lemma A.5. *For the feasible solution b_0 with the support $\{s_1^*, s_2^*, s_3^*\}$ satisfying $b_0(s_i^*) = b^*(s_i^*)$ for $1 \le i \le 3$, we have $\Delta_L \le f(b_0)/3$.*

Proof. It follows that $\Delta_L = f(b_L + k_L\chi_{s_L}) - f(b_L) \le f(b_L \vee b^*(s_L)\chi_{s_L}) - f(b_L)$ since $b_L(s_L) + k_L \le b^*(s_L)$. By Lemma 2.2, Δ_L is at most $f(b^*(s_L)\chi_{s_L}) - f(0) = f(b^*(s_L)\chi_{s_L})$, and hence $\Delta_L \le f(b^*(s_1^*)\chi_{s_1^*}) = \Delta(0, s_1^*, b^*(s_1^*))$ by the choice of s_1^*. Similarly, by the choices of s_2^* and s_3^*, we have $\Delta_L \le \Delta(b^*(s_1^*)\chi_{s_1^*}, s_2^*, b^*(s_2^*))$ and $\Delta_L \le \Delta(b^*(s_1^*)\chi_{s_1^*} \vee b^*(s_2^*)\chi_{s_2^*}, s_3^*, b^*(s_3^*))$. Adding these inequalities, we obtain $\Delta_L \le f(b_0)/3$. □

We are now ready to prove Theorem 2.1. Since $f(b) \ge f(b_0) + \sum_{i=1}^{L} \Delta_i - \Delta_L$, Lemmas A.4 and A.5 imply that

$$\begin{aligned} f(b) &\ge (1 - 1/3)f(b_0) + (1 - 1/e)(f(b^*) - f(b_0)) \\ &\ge (1 - 1/e)f(b^*) \end{aligned}$$

for the initial solution b_0 described in Lemma A.5. This completes the proof of Theorem 2.1.

B. Proof of Lemmas

B.1. Proof of Lemma 2.2

Proof. If $k \le x(s)$ then both sides are 0. If $x(s) < k \le y(s)$ then the inequality (3) is equivalent to $f(x \vee k\chi_s) - f(x) \ge 0$, which is valid by monotonicity. Lastly, if $k > y(s)$ then we have $f(x \vee k\chi_s) + f(y) \ge f(y \vee k\chi_s) + f(x)$ by submodularity, which directly implies (3). □

B.2. Proof of Lemma 2.3

Proof. We prove this lemma by induction on the size of $\mathrm{supp}^+(y - x)$. If $|\mathrm{supp}^+(y - x)| = 0$, that is, $x \vee y = x$, then (4) is trivial. Suppose that there is an index $s \in \mathrm{supp}^+(y - x)$. We define $y_1 = y - y(s)\chi_s$. Then $y = y_1 \vee y(s)\chi_s$ and $y_1 \wedge y(s)\chi_s = 0$ hold. By submodularity of $x \vee y_1$ and $x \vee y(s)\chi_s$, we have

$$f(x \vee y_1) + f(x \vee y(s)\chi_s) \ge f(x \vee y) + f((x \vee y_1) \wedge (x \vee y(s)\chi_s)).$$

Since it holds that

$$(x \vee y_1) \wedge (x \vee y(s)\chi_s) = x \vee (y_1 \wedge y(s)\chi_s) = x \vee 0 = x,$$

the above inequality implies that

$$f(x \vee y_1) + f(x \vee y(s)\chi_s) - f(x) \ge f(x \vee y).$$

Therefore, applying the induction hypothesis to x and y_1, we obtain (4). Thus the statement holds. □

B.3. Proof of Lemma 4.1

Proof. By simple algebra, we can easily check that

$$\begin{aligned} &f(b + \chi_s) - f(b) \\ &= p_s^{(b(s)+1)} \sum_{t \in \Gamma(s)} \prod_{v \in \Gamma(t)} \prod_{i=1}^{b(v)} (1 - p_v^{(i)}), \end{aligned} \tag{7}$$

$$\begin{aligned} &f(b + 2\chi_s) - f(b + \chi_s) \\ &= (1 - p_s^{(b(s)+1)}) p_s^{(b(s)+2)} \sum_{t \in \Gamma(s)} \prod_{v \in \Gamma(t)} \prod_{i=1}^{b(v)} (1 - p_v^{(i)}). \end{aligned} \tag{8}$$

Let $\alpha := \sum_{t \in \Gamma(s)} \prod_{v \in \Gamma(t)} \prod_{i=1}^{b(v)} (1 - p_v^{(i)})$ for simplicity of notations. Then we have $f(b + \chi_s) - f(b) - (f(b + 2\chi_s) - f(b + \chi_s)) = p_s^{(b(s)+1)}\alpha - p_s^{(b(s)+2)}(1 - p_s^{(b(s)+1)})\alpha \ge \alpha(p_s^{(b(s)+2)} - p_s^{(b(s)+2)}(1 - p_s^{(b(s)+2)})) = \alpha(p_s^{(b(s)+2)})^2 \ge 0$. □

C. Pseudocode of Algorithm

Algorithm 3 SIMPLEGREEDYPROCEDUREFORCOMPETITORMODEL

for each $t \in T$ **do**
  Let $\lambda := 1$
  for $k = 1$ to $1 + \max_{s \in \Gamma(t)} \tilde{b}(s)$ **do**
    Let $\phi_k(t) := 1$ and $\lambda_{t,k} := \lambda \left(1 - \prod_{s \in \Gamma(t): \tilde{b}(s) \ge k} (1 - \tilde{p}_s^{(k)})\right)$.
    Let $\lambda := \lambda \prod_{s \in \Gamma(t): \tilde{b}(s) \ge k} (1 - \tilde{p}_s^{(k)})$.
  end for
end for
Let $b := 0$.
for $i = 1$ to B **do**
  Choose s maximizing $\sum_{t \in \Gamma(s)} \sum_k \lambda_{t,k} r_{s,k}^{(b(s)+1)} \phi_k(t)$.
  Let $b(s) := b(s) + 1$.
  for $t \in \Gamma(s)$ **do**
    for $k = 1$ to $1 + \max_{s \in \Gamma(t)} \tilde{b}(s)$ **do**
      Let $\phi_k(t) := (1 - r_{s,k}^{(b(s))})\phi_k(t)$.
    end for
  end for
end for
return b