Influence Maximization in Bandit and Adaptive Settings

by

Sharan Vaswani

B.E. Computer Science, Birla Institute of Technology and Science, Pilani, 2012

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Computer Science)

The University of British Columbia
(Vancouver)

August 2015

Abstract

The objective of viral marketing is to leverage a social network to spread awareness about a specific product in the market through information propagation via word-of-mouth. A closely related problem is that of influence maximization which aims to identify the 'best' set of 'influential' users (seeds) to give discounts or free products to, such that awareness about the product is maximized. We study two relatively unexplored variants of influence maximization (IM) in social networks.

Conventional IM algorithms assume the structure of the social network and edge weights to be known. Though the structure of the network is usually known, edge weights need to be estimated from past data. In the first part of this thesis, we tackle the real but difficult problems of (i) learning these edge weights online and (ii) maximizing influence in the network when no past data is available as input. We adopt a combinatorial multi-armed bandit (CMAB) paradigm and formulate the above problems respectively as (i) network exploration, i.e. incrementally minimizing the error in learned edge weights and (ii) regret minimization i.e. minimizing the loss in influence from choosing suboptimal seed sets over multiple attempts.

Most previous work on influence maximization in social networks is limited to the non-adaptive setting in which the marketer is supposed to select all of the seed users up front. A disadvantage of this setting is that she is forced to select all the seeds based solely on a diffusion model. If the selected seeds do not perform well, there is no opportunity to course-correct. A more practical setting is the adaptive setting in which the marketer initially selects a batch of users and observes how seeding those users leads to a diffusion of product adoptions. Based on this market feedback, she formulates a policy for choosing the remaining seeds. We study adaptive offline strategies for two problems: (a) MAXSPREAD - given a budget on number of seeds and a time horizon, maximize the spread of influence and (b) MINTSS - given a time horizon and an expected number of target users, minimize the number of required seeds.

Preface

This thesis is submitted in partial fulfilment of the requirements for a Master of Science Degree in Computer Science. All the work presented in this dissertation are original and independent work of the author, performed under the supervision of Prof. Laks.V.S.Lakshmanan.

Table of Contents

List of Tables

List of Figures

List of Symbols

G	Graph underlying the social network
V	Set of vertices in G
E	Set of edges in G
P	Set of influence probabilities(edge weights) in G
$N_{out}(u)$	Set of out-neighbours of node u
$N_{in}(u)$	Set of in-neighbours of node u
D	Length of the longest path of G
k	Seed budget
S	Seed set
W	Possible world of the diffusion process
$\sigma(\mathbb{S})$	Expected spread for the seed set $\mathbb{S}$
m	Number of base arms in the bandit framework
π	Seed selection policy for adaptive seeding
H	Time horizon for viral marketing campaign
T	Number of rounds in the bandit game
$X_{i,s}$	Reward of the i^{th} in round s
$T_{i,s}$	Number of times arm i has been triggered until round s
$T^{s}_{(u,v)}$	Number of times edge (u,v) has been triggered until round s
μ_i	Mean of the reward distribution for the i^{th} arm
A	Superarm (subset of base arms)
$\hat{\mu}_i$	Mean estimate for arm i
p^{i}_{A}	Triggering probability of arm i if the superarm A is played
O	Influence maximization oracle
ρ	Failure probability of the Frequentist approach for node level feedback

$p_{u,v}$	Influence probability for the edge (u,v)
p_{min}	Minimum influence probability in the network
p_{max}	Maximum influence probability in the network
F	Bounded convex set
c^s	Cost function in round s in Zinkevich's framework
x^s	Point selected in round s in Zinkevich's framework
η_s	Step size in round s for gradient descent in Zinkevich's framework
$r_{\vec{\mu}}(A)$	Reward obtained on playing the superarm A when the mean rewards is given by $\vec{\mu}$
$Reg^{\mathscr{A}}_{\vec{\mu},\alpha,\beta}$	Regret for algorithm $\mathscr{A}$ when the mean rewards are given by $\vec{\mu}$ calculated according to an (α,β) oracle
C	Set of available cascades
M	Feedback mechanism
$\overline{\mu_i}$	Upper confidence bound for the i^{th} arm
ω	Initial exploration rate for the ε-greedy algorithm
ζ	Fraction of exploration rounds for the Initial Exploration algorithm
γ	Correlation decay in the network
α	Multiplicative error in the marginal gain estimaton for adaptive policies
Q	Expected target spread for MINTSS
δ, β	Shortfall in achieving the target spread for MINTSS
$\pi_{GA,k}$	Greedy Adaptive policy constrained to select k seeds
$\pi_{OA,k}$	Optimal Adaptive policy constrained to select k seeds
$\pi_{GNA,k}$	Greedy Non-adaptive policy constrained to select k seeds
$\pi_{ONA,k}$	Optimal Non-adaptive policy constrained to select k seeds
$\sigma(\pi_{strategy,k})$	Expected spread for a $\pi_{strategy}$ policy constrained to select k seeds

Acknowledgements

I am indebted to my supervisor Professor Laks.V.S.Lakshmanan for being so patient and for all his help - technical and otherwise. I am grateful to all the professors with whom I got an opportunity to interact with. In particular, I am grateful to Nick Harvey and Mark Schmidt for taking time off their busy schedules to give me helpful advice. I would like to thank my lab-mates and collaborators for all the useful discussions and ideas. I would like to thank my friends at UBC for keeping me sane during these two years. Last but not the least, I am thankful to my family for their constant love and support.

To my family

1

Introduction

> They say 90% of the promotion of a book comes through word of mouth. But you've somehow got to get your book into the hands of those mouths first!.
>
> *– Claudia Osmond*

Recently, there has been tremendous interest in the study of influence propagation in social and information networks, motivated by applications such as the study of spread of infections and innovations, viral marketing, and feed ranking to name a few (e.g.,see [18, 31, 48, 49]). In these applications, the network is given by a directed graph $G = (V, E, P)$, with nodes V, edges E, and edge weights $P : E \rightarrow [0, 1]$. The nodes denote users of the social network, the edges correspond to relations between users of the network (e.g. friendship in Facebook, followers in Twitter). The edge weights (also referred to as influence probabilities) characterize the strength of these relations. Usually, they denote the probabilities of influence between adjacent nodes and govern how information spreads from a node to its neighbours. Information propagation in these applications is modelled with the aid of a stochastic diffusion model.

We address the problem of viral marketing [38, 43] the aim of which is to leverage the social network to spread awareness about a specific product in the market through information propagation via word-of-mouth. Specifically, the marketer aims to select a fixed number of 'influential' users (called seeds) to give free products or discounts to. The marketer assumes that these users will influence others through the social network. This will result in information propagating through the network with increasing number of users adopting or becoming aware of the product. Since there is a budget on the number of free samples / discounts that can be given, the marketer must strategically

choose seeds so that the maximum number of people in the network become aware of the product. This can be formalized as the *influence maximization* (IM) problem [33]. Under an assumed model of information diffusion, IM aims to identify the 'best' set of seeds which when given free products / discounts will result in the maximum number of people adopting or becoming aware of the product. We refer to a user adopting or becoming aware of the product as being 'activated'. More precisely, if k is the budget on the number of seeds that can be selected by the marketer, IM aims to find a set S of k seed nodes, such that activating them in the beginning leads to the maximum *expected spread* σ, i.e., expected number of activated nodes according to a given diffusion model.

The IM problem can be formally formulated as:

$$S = argmax_{|\mathbb{S}| \leq k} \sigma(\mathbb{S}) \tag{1.1}$$

In their seminal paper, Kempe, Kleinberg and Tardös [33] formalized IM as a discrete optimization problem. They studied several discrete-time diffusion models originally pioneered in mathematical sociology. These models include the independent cascade and linear threshold model (details in Section 2). They showed that IM under these models is NP-hard but the expected spread function satisfies the nice properties of monotonicity and submodularity. Monotonicity means adding more seeds cannot lead to a lower expected spread whereas submodularity intuitively corresponds to the property of diminishing returns (formal definitions in Chapter 2). By exploiting these properties and early results by Nemhauser et al.[39], they showed that a simple greedy algorithm achieves a $(1-1/e)$-approximation to the optimum. There has been an explosion of research activity around this problem, including development of scalable heuristics, alternative diffusion models, and scalable approximation algorithms (e.g., see [10] [51] [35] [28] [27] [50]). We refer the reader to [13] for a more detailed survey.

Most work on IM assumes that both the network and the set of influence probabilities is available as input. However, although the network structure might be known in practice, influence probabilities are *not* generally available. To overcome this, the initial series of papers following [33] simply *assigned* influence probabilities: e.g., assign a fixed small probability such as 0.01 to all edges, assign values drawn at random from a fixed set such as $\{0.1, 0.01, 0.001\}$, or assign an equal probability of $\frac{1}{N^{in}(v)}$ to all incoming edges of a node v, where $N^{in}(v)$ is the set of in-neighbours of v. However, there is no basis for such assignment. A useful kind of data sometimes available is the set of diffusions or cascades that actually happened in the past, specified in the form of a log of

actions by network users. There is a growing body of work devoted to learning the true influence probabilities from past data [17, 26, 40, 45, 46]. Details of this body of work are covered in Section 2.

Irrespective of the differences in the approaches proposed, diffusion models used, or parameters learned, *a common characteristic of all these works is that they all critically depend on actual diffusion cascades being available.* We notice that in several real datasets, only the social network data is available and there is no available information about cascades. *This unfortunately renders the previous approaches for learning influence probabilities inapplicable.* This raises the following questions. (1) When only a social network is available, how can we learn influence probabilities, and do so in an efficient manner? (2) More generally, how can we solve the IM problem when no cascade information or influence probabilities is available? We solve these problems in an online fashion i.e. we perform IM in multiple rounds and improve our seed selection and knowledge about the influence probabilities incrementally across rounds. We refer to the first question as the *network exploration* problem i.e. how to perform our seed selections so that we can learn about the influence probabilities in the network quickly. The second question can be cast as a *regret minimization* problem (formal definitions given in chapter 3).

Formally, we adopt a multi-armed bandits(MAB) paradigm for solving the problems of learning influence probabilities and influence maximization, when only the social network is available as input. The inspiration for our work comes from recent work by Chen et al. [14, 15] on combinatorial MAB that includes influence maximization as an application. The stochastic multi-armed bandit setting [34] has been used to address problems in clinical trials [44], adaptive routing [5] and advertisement placement on websites [42] and recommender systems [36]. The combinatorial MAB framework [2, 14] is an extension to the multi-armed bandit setting and has proved to be important in solving tasks which have an associated combinatorial structure. Adoption of a bandit paradigm in turn raises the following additional questions. (3) How should we update our estimates of the influence probabilities based on observations made from previous seed selections? (4) How many rounds of seed selections are needed before influence probabilities can be learned in an accurate enough manner? Another natural question is: (5) How does the learning "rate" of bandit based algorithms compare with that of the batch algorithms (which assume the availability of a batch of cascades) ? (6) While seed selection does improve our knowledge of influence probabilities, we are paying a price for selecting seeds

under imperfect knowledge, in the form of reduced spread compared with the spread we *could* have achieved if we had perfect knowledge of the true influence probabilities. How can we quantify and how can we minimize this loss in spread or *regret*? This question is particularly important to a viral marketer. Intuitively, a new marketer who has just entered the market and has no knowledge about the network (besides its structure) will try and learn about it through a trial and error procedure (the 'exploration' phase). These exploration rounds are therefore like an investment and the knowledge gained in this phase can be used to generate revenue (by achieving a high spread) in the subsequent rounds (the 'exploitation' phase). The regret minimization objective corresponds to minimizing the loss in revenue while learning about the market whereas the network exploration objective corresponds to learning about the network in a reasonable amount of time. We describe the basics of MAB and CMAB in chapter 2. In chapter 3, we show how to address the above questions by mapping our IM problem to the CMAB framework and using ideas and algorithms from bandit theory.

In the second part of this thesis, we focus on IM in the adaptive setting. The majority of work in influence maximization has confined itself to a *non-adaptive* setting where, in viral marketing terms, the marketer must commit to choosing all the k seeds up front. We refer to conventional IM as the MAXSPREAD problem. Instead of maximizing the spread, the marketer may have a certain expected spread as the *target* that she wants to achieve. This target may be derived from the desired sales volume for the product. A natural problem is to find the minimum number of seeds needed to achieve the target. This problem, called *minimum targeted seed selection* (MINTSS for short), has been studied in the non-adaptive setting [29]. It was shown that the classic greedy algorithm leads to a bi-criteria approximation to the optimal solution. Precisely, if the target spread is Q and a shortfall of $\beta > 0$ is allowed, then the greedy algorithm will achieve an expected spread $\geq (Q - \beta)$ using no more than $OPT(1 + \ln\lceil Q/\beta \rceil)$ seeds, where OPT is the optimal number of seeds.

For both the MAXSPREAD and MINTSS problems, a non-adaptive setting implies that the choice of every single seed is driven completely by the diffusion model used for capturing the propagation phenomena. In practice, it may happen that the *actual spread* resulting from the seeds chosen may fall short of the expected spread predicted by the diffusion model. Recent work by Goyal et al. [27] shows that most diffusion models tend to over-predict the actual spread. Thus, committing to the choice of all k seeds in one shot can result in a sub-optimal performance in reality. A more realistic setting

is one where the marketer chooses a subset of seeds and activates them. She monitors how their activation spreads through the network and observes the actual or 'true' spread thus far. She can then take into account this market feedback in making subsequent seed selections. We call this setting an *adaptive* setting, as choices of subsequent seeds are adapted to observations made so far about the actual spread achieved by previous selections. Hence, the adaptive setting introduces a *policy* π which specifies which node(s) to seed at a given time.

Adaptive seed selection raises several major challenges. For instance, in the adaptive setting, in practice, there is a finite time horizon H within which the marketer wishes to conduct her viral marketing campaign. The marketer must then consider the following questions. How many seeds to select at a given time, that is, what is the *batch* size ? Which nodes should be selected in each intervention ? How long should she wait between seeding successive batches i.e. what should be the inter-intervention time ? If H is sufficiently long, it seems intuitive that selecting one seed at a time and waiting until the diffusion completes, before choosing the next seed, should lead to the maximum spread. The reason is that we do not commit any more seeds than necessary to start a fresh diffusion and every seed selection takes full advantage of market feedback. We refer to the above case as *unbounded time horizon*. Another natural question is, what if the time horizon H is not long enough to allow many small batches to be chosen and/or diffusions to be observed in full. In this case, which we call *bounded time horizon*, the marketer has to choose a strategy in which the budget k is spent within the time horizon H and every seed selection benefits from as much feedback as possible.

It is very intuitive that for the MAXSPREAD problem, adaptive seed selection should lead to a higher actual spread compared to non-adaptive seed selection, since it benefits from market feedback and tailors seed selections accordingly. For MINTSS, an interesting question is to what extent can an adaptive seed selection strategy cut down on the number of seeds required to reach a given target spread. We answer the above questions theoretically and empirically in chapter 4.

2

Background and Related Work

If I have seen farther it is by standing on the shoulders of Giants.

– Sir Isaac Newton

2.1 Conventional Influence Maximization

We first describe the necessary background for solving the conventional IM problem.

2.1.1 Diffusion Model

Information propagation is modelled using stochastic diffusion models such as Independent Cascade (IC) and Linear Threshold (LT) [33]. For both these models, time proceeds in discrete steps. In this thesis, we focus exclusively on the IC model which we briefly describe now. All our results can be easily adapted for the LT model. In the IC model, the seed nodes are active at time $t = 0$. Each active user at time t gets one chance to influence/activate her out-neighbours at the next time step $t + 1$. This activation attempt succeeds with the corresponding influence probability between the two users. If the attempt succeeds, the neighbour will become active at time $t + 1$. An edge along which an activation attempt succeeded is said to be 'live' whereas the other edges are said to be 'dead'. At a given instant of time t, an inactive node v may have multiple in-neighbours capable of activating it. We refer to the in-neighbours of v as its parents and to its in-neighbours which became active at $t - 1$ (set of in-neighbours which can activate v at t) as its active parents. The diffusion process is said to have ended if there are no more nodes which can be activated using the above procedure. The number of nodes activated

at the end of the diffusion is referred to as the spread $\bar{\sigma}$. Since the IC diffusion model is stochastic, it can result in a large number of 'possible worlds'. Note that each edge can be either 'live' or 'dead' in a possible world W depending on its influence probability. Hence there can be $2^{|E|}$ possible worlds where some worlds are more probable than others. The expected spread (σ) can be trivially calculated by taking a weighted average over $\bar{\sigma}$ in these possible worlds.

2.1.2 Algorithm

As was mentioned in chapter 1, IM is NP-hard under these common discrete time models. However, under both the IC and LT models, the expected spread σ is monotonic and submodular.

Definition 1. [Monotonicity and Submodularity] Given the universe of elements U, a real-valued set function $f : 2^U \rightarrow R$ is monotone if $f(S) \leq f(S'), \forall S \subset S' \subseteq U$. It is submodular if $\forall S \subset S' \subset U$ and $x \in U \setminus S'$, $f(S' \cup \{x\}) - f(S') \leq f(S \cup \{x\}) - f(S)$. □

The difference $f(S \cup \{x\}) - f(S)$ is the marginal gain of adding element x to the set S. Submodularity hence implies that the marginal gain of any element x decreases as the size of the set increases. Submodularity enables a simple greedy algorithm [39] to provide a $(1 - 1/e)$-approximation to the optimal solution. For the greedy algorithm, we build the set S incrementally. At each point, we add the element (in our case, seed) which maximizes the marginal gain in expected spread. Computing the expected spread of a given set (and hence marginal gain) is #P-hard for both IC and LT models [11, 12]. Kempe et al. [33] advocated using MCMC simulations to estimate marginal gains. Using MCMC estimation of the marginal gain, the greedy algorithm yields a $(1 - 1/e - \varepsilon)$-approximation to the optimum, where $\varepsilon > 0$ is the error because of the marginal gain estimation. Another technique for improving the speed of marginal gain computation is that of lazy evaluation [35]. Tang et al. [50] proposed a near-optimal (w.r.t. time complexity) randomized greedy $(1 - 1/e - \varepsilon)$-approximation algorithm for IM. We note that it is currently the state of the art for MAXSPREAD and has been shown to scale to a billion-edge network. We use the approach in [50] for solving the conventional IM problem.

2.1.3 Learning probabilities

A number of methods have addressed the problem of learning probabilities from past diffusions or cascades available in the form of logs. Goyal et al. [27] showed empirically that learning the true influence probabilities from available data is critical for ensuring the quality of seeds chosen and the spread achieved. Saito et al. [46] develop a likelihood maximization approach for learning influence probabilities from available cascades under the IC model and propose an expectation maximization algorithm. Goyal et al. [26] develop a frequentist approach for learning influence probabilities under the generalized threshold model. Netrapalli and Sanghavi [40] present a likelihood maximization formulation for learning both the graph structure and influence probabilities using the IC model. In continuous time diffusion models, in lieu of influence probabilities, one associates transfer rates of diffusion with network edges. Rodriguez et al. [17, 24, 45] develop algorithms to infer these rates from available cascades under continuous time diffusion models.

2.1.4 Feedback

In both the variants addressed in this thesis, we need some feedback from either the partial or complete diffusion process. In the case when the influence probabilities are unknown, we need feedback about the state of the network at the end of the diffusion process to refine our estimates of the influence probabilities so that we can do better on the next round of seed selection. For adaptive influence maximization, we need feedback from the partial diffusion process so that we can select the next seed. The state of the network can be characterized by identifying either the activated nodes (i.e. users who became aware / adopted the product) or the 'live' edges (i.e. edges along which a successful activation attempt has occurred). We refer to the former as *node level feedback* and to the latter as *edge level feedback*. We note that it is more feasible to identify users who have adopted a product than to gain information about each edge in the network. Thus, node level feedback is more practical than edge level feedback and we use it throughout.

2.2 Multi-armed Bandits

As mentioned in chapter 1, we can cast influence maximization with unknown influence probabilities as a combinatorial multiarmed bandit problem. The stochastic multi-armed bandit (MAB) problem was first addressed in [34]. In the traditional framework, there are

m arms each of which has an unknown reward distribution. The bandit game proceeds in rounds and in every round s, an arm is played, the environment samples the reward distribution for that arm and a corresponding reward is generated. This game continues for a fixed number of rounds T. Some of the arms will result in higher rewards whereas others have underlying distributions that lead to smaller reward values. Initially, the reward distribution of every arm is unknown to the player. The aim of the bandit game is to minimize the regret obtained by playing suboptimal arms across rounds (*regret minimization*). This results in an exploration (trying arms we don't know much about in hope of a higher rewards) vs exploitation (pulling the arm which we have learnt gives high rewards) tradeoff. For the regret minimization setting, [1, 34] proposed algorithms which can achieve the optimal regret of $\mathcal{O}(log(T))$ over T rounds. Another objective of the bandit game can be to learn the properties (such as mean) of the underlying reward distributions (*pure exploration*). This problem has been studied in the past in [7, 19, 37].

The Combinatorial Multiarmed Bandit problem is an extension to the MAB framework in which we can pull a set of arms (or superarm) together [14, 21] or simultaneously play k among the m arms [2, 8, 20]. Clearly, triggering a super arm triggers all the arms it contains and the resulting reward is some combination of the individual rewards. Previous literature [14, 21] consider a CMAB framework with an approximation oracle to find the best superarm to be played in each round. In [21] however, the authors only consider cases where the resulting reward on pulling the superarm is some linear combination of the rewards for individual arms in the superarm. This was generalized to any non-linear combination in [14]. For the regret minimization case, [14] proposed a UCB based algorithm Combinatorial UCB (CUCB) to solve the CMAB problem and obtain an optimal regret of $\mathcal{O}(log(T))$. [25] proposes a Thompson Sampling based algorithm to solve the CMAB problem. Pure exploration case in the CMAB framework has been addressed in [9]. In [15], the authors introduced the notion of probabilistically triggered arms which are triggered when a superarm is played, in addition to the arms contained in the superarm. In this paper, Chen at al. target both ad placement on web pages and viral marketing application under semi-bandit feedback [4] i.e. we can observe the rewards of arms which were triggered when the superarm was pulled. The latter is of special interest to us, and we discuss this in more detail in Section 3.1.

2.3 Adaptive Influence Maximization

Adaptive influence maximization has been proposed previously in [16, 22, 30]. In the adaptive setting, batches of nodes are seeded at different intervals. When a batch is seeded, an *actual* diffusion (called realization in [22]) unfolds as per the classical IC model. The next batch is chosen based on the partially observed diffusion/cascade. We wish to choose a policy that maximizes such an objective function in the adaptive setting.

Golovin and Krause [22] extend the definitions of submodularity and monotonicity to the adaptive setting. An objective function is *adaptive monotone* and *adaptive submodular* if the marginal gain of every element is non-negative and non-increasing in every possible realization, as the size of the set (alternatively length of the policy) increases. As before, the greedy policy consists of selecting the node with the maximum marginal gain. Golovin and Krause [22] derive average case bounds on the performance of greedy adaptive policies. They also prove bounds on the greedy adaptive policy for adaptive submodular functions under matroid constraints [23].

They assume an edge level feedback mechanism under the IC model and show that the expected spread is adaptive monotone and adaptive submodular, guaranteeing an approximation algorithm. Guillory et al. [30] study the problem of submodular set cover in the adaptive setting in which the objective is to minimize the total number of sets required to cover a certain target set and prove worst case bounds for the greedy adaptive policy. They briefly describe how their framework can be used for influence maximization in a social network with hidden information (e.g., hidden preferences of users). The problem involves simultaneously learning about the hidden preferences of users and maximizing the influence among users who prefer a particular product. We consider the more traditional influence maximization problem and assume that users do not have any hidden preferences. We establish average case guarantees similar to [22]. Finally, [16] addresses the adaptive MINTSS problem and shows that under certain conditions, the batch-greedy adaptive policy, in which the seeds are chosen in batches in a greedy manner, is competitive not only against the sequential greedy policy (choosing one seed at a time) but also against the optimal adaptive policy.

These papers assume the unrealistic edge level feedback described above. The experiments conducted in these papers (if at all) are on small toy networks with 1000 nodes and they do not clarify the *practical* benefits of going adaptive for real large networks. All previous studies are confined to the setting of *unbounded time horizon*, which means the horizon is long enough for the diffusion started by each batch to complete. In prac-

tice, the horizon may be bounded and not leave enough time for successive diffusions to complete. The theoretical results in these papers bound the performance of the greedy adaptive policy compared to the optimal adaptive policy. Notice that the optimal (adaptive) policy cannot be computed in polynomial time. The only practical options for both non-adaptive and adaptive settings are greedy approximations (possibly with techniques for scaling up to large datasets). Thus, a real question of practical interest is what do we gain by going adaptive, i.e., what is the gain in peformance of the greedy approximation algorithm when it is made adaptive? In contrast, we study MAXSPREAD and MINTSS under both unbounded and bounded time horizon and quantify the benefits of going adaptive with reference to the greedy approximation algorithm, as opposed to the optimal algorithm. Furthermore, we use the more realistic node level feedback model throughout and perform experiments on large real world networks.

3

Influence Maximization with Bandits

> Good judgment comes from experience, and experience comes from bad judgment
>
> *– Rita Mae Brown*

3.1 Theory

3.1.1 Combinatorial Multiarmed Bandit Framework

We briefly review the combinatorial multi-armed bandit (CMAB) framework proposed in [14, 15] and adapt it to the problem of IM. A CMAB problem consists of m base arms. Each arm i is associated with a random variable $X_{i,s}$ which denotes the outcome or reward of triggering the i^{th} arm on the s^{th} trial. The reward $X_{i,s}$ is bounded on the support $[0,1]$ and is independently and identically distributed according to some unknown distribution with mean μ_i. The CMAB game is played in T rounds, for some fixed $T \geq 1$. In each round s, a superarm A (a set of base arms) is played, which triggers all arms in A. In addition, other arms may get probabilistically triggered. Let p_A^i denote the triggering probability of arm i if the superarm A is played. Clearly, $\forall i \in A, p_A^i = 1$. The reward obtained in each round s can be a (possibly non-linear) function of the rewards $X_{i,s}$ for each arm i that gets triggered in that round. Let $T_{i,s}$ denote the number of times a base arm i has been triggered *until* round s. For the special case $s = T$, we define $T_i := T_{i,T}$,

as the number of times arm i is triggered in the game. Each time an arm i is triggered, we update its mean estimate $\hat{\mu}_i$ based on the observed reward. Using the current reward distribution estimates, the best superarm (which is expected to lead to the highest reward) to be played in each round is selected by an oracle O, which takes as input the vector of current mean estimates $\vec{\hat{\mu}} = (\hat{\mu}_1, ..., \hat{\mu}_m)$ and outputs an appropriate superarm A. In order to accommodate intractable problems, the framework of [14, 15] assumes that the oracle provides an (α, β)-approximation to the optimal solution, i.e., the oracle outputs with probability β a superarm A which attains a reward within a factor α of the optimal solution, based on the current mean estimate $\vec{\hat{\mu}}$.

3.1.2 Adaptation to IM

We can map the IM problem with unknown influence probabilities to the CMAB framework as follows: the edges E in the network G map to arms. Each arm is characterized by a Bernoulli distribution, i.e., the edge is either live or dead in the "true" possible world. Thus, the rewards $X_{i,s}$ are binary. The mean of this distribution for an arm i is precisely the influence probability for the corresponding edge. A set of seed nodes S corresponds to a superarm defined by the set of all edges E_S that are outgoing from the nodes in S. Specifically, given a seed budget k, in each round, k seeds are selected and the corresponding superarm is played. Once the superarm is played, information diffuses in the network and a subset of network edges become live, and $X_{i,s}$ for these edges is 1. By definition, when an arm/edge (u, v) is made live, we say that the target node v is *active*. The total reward for each round is the spread $\bar{\sigma}(S)$, i.e., the number of active nodes at the end of the diffusion process. Note that $\bar{\sigma(S)}$ is a non-linear function of the rewards of the triggered arms. The input to the influence maximization oracle is the graph G along with the current estimates of influence probabilities $\vec{\hat{\mu}}$. It constitutes a $(1 - \frac{1}{e}, 1 - \frac{1}{|E|})$-approximation oracle and outputs a seed set S under the cardinality constraint k. Notice that the well-known greedy algorithm with MC simulations used for influence maximization can serve as such an oracle. The mean estimate $\vec{\hat{\mu}}$ vector is updated based on the observed rewards. In this context, the notion of *feedback mechanism* plays an important role. It characterizes the information available after a superarm is played. This information is used to update the model of the world to improve the mean estimates of the underlying arm reward distributions. The entire framework is summarized in figure 3.1.

In the specific context of our application of bandits to IM, there are two basic types of feedback we can formulate.

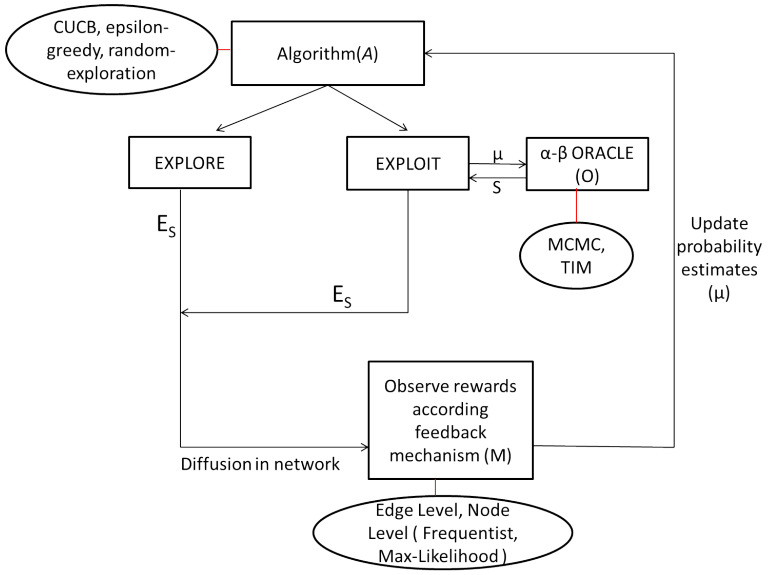


Figure 3.1: Summary of bandit framework for influence maximization

3.1.2.1 Edge Level Feedback

In the feedback mechanism proposed in [14], which we call *edge level feedback*, one assumes we know the status (live or dead) of each triggered edge in the "true" possible world generated by the diffusion in the network. The mean estimates of the arms distributions can then be updated using a simple frequentist formula, i.e., for the i^{th} arm, $\hat{\mu}_i = \frac{\sum_{j=1}^{s} X_{i,j}}{T_{i,s}}$. Assuming edge level feedback amounts to assuming that for each activated node in the network, we know exactly which of its active parents succeeded in activating the node. *A key point is that success/failure of activation attempts in general is not observable and thus edge level feedback assumption is not realistic.* This motivates us to propose node level feedback, made precise below.

3.1.2.2 Node Level Feedback

Unlike the status of edges, we can always observe the status of each node, i.e., whether a user bought/adopted the marketed product and when. We call this *node level feedback*. Clearly, compared to edge level feedback, node level feedback makes a weaker but more realistic assumption. The flip side is that update to the mean estimates is more challenging under node level feedback.

To see this, consider a given node v which becomes active at time t. Suppose v

has more than one parent and the parents $u_1, ..., u_K$ became active at time $t-1$ (see Figure 3.2).

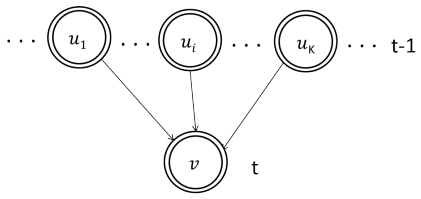


Figure 3.2: Example for node level feedback.

Under edge level feedback, we assume that we know the status of each edge (u_j, v), $1 \leq j \leq K$ and use it to update mean estimates. Under node level feedback, any of the active parents may be responsible for activating a node v and we don't know which one is. There are two ways of overcoming this problem.

3.1.2.2.1 Frequentist Approach This is an adaptation of the frequentist approach for edge level feedback. Specifically, we propose a scheme whereby we choose one of the active neighbors of v, say u_i, uniformly at random, and assign the credit of activating v to u_i (see Figure 3.2). The probability of assigning credit to any one of the K nodes is $\frac{1}{K}$. Here, edge (u_i, v) is given a reward of 1 whereas the edges (u_j, v) corresponding to all the other active parents $u_j, j \neq i$, are assigned a zero reward. We then follow the same frequentist update formula as described for the edge level feedback model.

Owing to the inherent uncertainty in node level feedback, note that we may make mistakes in credit assignment: *we may infer an edge to be live while it is dead in the true world or vice versa.* We term the probability of such faulty inference, the *failure probability* under node level feedback. An important question is whether we can bound this probability. This is important since failure probability could ultimately affect the achievable regret and the error in the learned probabilities. The following result settles this question. As a preview, in Section 3.4, we verify that the failure probability is low for real networks, thus establishing the effectiveness of learning influence probabilities under node level feedback.

Theorem 1. *Suppose a network node v became active at time t and $u_1, ..., u_K$ are the parents of v that became active at $t-1$. Let i denote the arm corresponding to edge (u_i, v). Let p_{min} and p_{max} be the minimum and maximum true influence probabilities in the network. The failure probability ρ under node level feedback is then characterized*

by:

$$\rho \leq \frac{1}{K}(1-p_{min})(1-\prod_{k=1,k\neq i}^{K}[1-p_{max}])+(1-\frac{1}{K})p_{max}. \tag{3.1}$$

Let $\hat{\mu}_i^E$ and $\hat{\mu}_i^N$ be the inferred influence probabilities for edge (u_i,v) using edge level and node level feedback respectively. Then the relative error in the learned influence probability is given by:

$$\frac{\left|\hat{\mu}_i^N-\hat{\mu}_i^E\right|}{\hat{\mu}_i^E}=\rho\left|(\frac{1}{\hat{\mu}_i^E}-2)\right| \tag{3.2}$$

Proof. Consider the network in Figure 3.2. Consider updating the influence probability on the edge (u_i,v) that is being learned. For the situation in Figure 3.2, we may infer the edge (u_i,v) to be live or dead. Our credit assignment makes an error when the edge is live and inferred to be dead and vice versa. Recall that all probabilities are conditioned on the fact that the node v is active at time t and K of its parents $(u_1,...,u_i,...,u_K)$ are active at time $t-1$. Let $\mathscr{E}_d$ ($\mathscr{E}_l$) be the event that the edge (u_i,v) is dead (resp., live) in the true world. Hence we can characterize the failure probability as follows:

$$\begin{aligned}\rho &= \Pr[(u_i,v) \text{ is inferred to be live}]\times\Pr[\mathscr{E}_d \mid v \text{ is active at time t}]\\ &+\Pr[(u_i,v) \text{ is inferred to be dead}]\times\Pr[\mathscr{E}_l \mid v \text{ is active at time t}]\end{aligned}$$

If (u_i,v) is live in the true world, then node v will be active at time t irrespective of the status of the edges $(u_j,v), j\in[K], j\neq i$. Hence,

$$\Pr[E_l \mid v \text{ is active at time t}]=\Pr[\mathscr{E}_l]$$

By definition of independent cascade model, the statuses of edges are independent of each other. Hence,

$$\begin{aligned}&\Pr[\mathscr{E}_d \mid v \text{ is activeat time t}]=\Pr[\mathscr{E}_d \wedge \text{ at least one of the edges } (u_j,v), j\neq i, \text{ is live}]\\ &\rho=\Pr[(u_i,v) \text{ is inferred to be live}]\times\Pr[\mathscr{E}_d \wedge \text{ at least one of the edges } (u_j,v), j\neq i, \text{ is live}]\\ &\quad+\Pr[(u_i,v) \text{ is inferred to be dead}]\times\Pr[\mathscr{E}_l]\end{aligned}$$

Let $p_{u_j,v}$ be the true influence probability for the edge (u_j,v), $j\in[K]$. We have the

following:

$$\Pr[\mathscr{E}_l] = p_{u_i,v}$$

$$\Pr[\mathscr{E}_d \wedge \text{at least one of the edges } (u_j,v), j \neq i, \text{ is live}] = (1-\Pr[\mathscr{E}_l])\big[1-\prod_{j=1,j\neq i}^{K}[1-p_{u_j,v}]\big]$$

Since one of the active nodes is chosen at random and assigned credit,

$$\Pr[\text{choosing } u_i \text{ for credit}] = \frac{1}{K} \Rightarrow \Pr[(u_i,v) \text{ is inferred to be live}] = \frac{1}{K}$$

We thus obtain:

$$\rho = \frac{1}{K}(1-p_{u_i,v})[1-\prod_{k=1,k\neq i}^{K}\big[1-p_{u_k,v}\big]] + (1-\frac{1}{K})p_{u_i,v} \tag{3.3}$$

Let p_{min} (p_{max}) denote the minimum (resp. maximum) true influence probability of any edge in the network. Plugging these in Eq. (3.3) gives us the upper bound in Eq. (3.1),the first part of the theorem.

Let $\hat{\mu}_i^N$ and $\hat{\mu}_i^E$ denote the mean estimates using node level and edge level feedback respectively. That is, they are the influence probabilities of edge (u_i,v) learned under node and edge level feedback. We next quantify the error in $\hat{\mu}_i^N$ relative to $\hat{\mu}_i^E$. Let $X_{i,s}^N$ be the status of the edge corresponding to arm i inferred using our credit assignment scheme described above, at round s. Recall that under both edge level and node level feedback, the mean is estimated using the frequentist approach. That is, $\hat{\mu}_i^N = \sum_{s=1}^{T}\frac{X_{i,s}^N}{T_i}$ (similarly for edge level feedback). Note that $X_{i,s}$ denotes the true reward (for edge level feedback) whereas $X_{i,s}^N$ denotes the inferred reward under node level feedback, using the credit assignment scheme described earlier. Thus, for each successful true activation of arm i (i.e., $X_{i,s}=1$) we obtain $X_{i,s}^N = 1$ with probability $1-\rho$ and for each unsuccessful true activation, we obtain $X_{i,s}^N = 1$ with probability ρ. Let S_i denote the number of rounds in which the true reward $X_{i,s}=1$. Hence, we have:

$$\hat{\mu}_i^E = \frac{S_i}{T_i} \tag{3.4}$$

$$\hat{\mu}_i^N = \frac{S_i(1-\rho) + (T_i - S_i)(\rho)}{T_i} \tag{3.5}$$

The second part of the theorem, Eq.(3.2), follows from Eq.(3.4) and (3.5) using simple algebra. □

In Section 3.4, we empirically find typical values of p_{max}, p_{min}, and K on real datasets and verify that the failure probability is indeed small. We also find that the proposed node level feedback achieves competitive performance compared to edge level feedback.

3.1.2.2.2 Maximum Likelihood Method As mentioned before, the papers [40, 46] describe an *offline* method for learning influence probabilities, where a *batch* of cascades is given as input. We adapt the approach of [40] into an *online* method, employing the CMAB paradigm. In an online setting, cascades stream in and only one cascade is available at a time. Each cascade can be viewed as resulting from choosing a set of seeds in a round of a CMAB game. Before describing our extension, we briefly review the approach of Netrapalli and Sanghavi [40]. They take as input a batch of diffusion cascades and infer the influence probabilities and the neighborhood structure for each node in the graph. They formulate a function $L^c(\theta)$ which models the likelihood of observing a particular cascade c given the current estimated neighborhood and influence probabilities. They show that the likelihood function is concave and can be decoupled over the nodes, i.e., it can be maximized independently for each node. For our use case, we assume that the graph structure (the true neighborhood) is known for each node. We next describe the likelihood function used in [40] and then propose a method for making the likelihood optimization online.

Offline MLE Learning: Let $\theta_{uv} = -\ln(1 - p_{u,v})$ where $p_{u,v}$ is the influence probability of the edge (u,v). We can characterize the likelihood in terms of $\vec{\theta}$, the vector of "θ values". The overall likelihood function $L(\vec{\theta})$ is a sum of the likelihood functions for individual cascades i.e., $L(\vec{\theta}) = \sum_{c=1}^{C} L^c(\vec{\theta})$, C being the number of cascades. The likelihood function L decomposes over the network nodes, i.e., $L(\vec{\theta}) = \sum_{c=1}^{C} \sum_{v \in V} L_v^c(\vec{\theta})$ where $L_v^c(\theta)$ models the likelihood of observing the cascade c w.r.t. node v. Let t_u^c denote the timestep at which node u becomes active in cascade c. Then $L_v^c(\vec{\theta})$ can be written as

follows:

$$L_v^c(\vec{\theta}) = -\sum_{u:t_u^c \leq t_v^c - 2} \theta_{uv} + \ln(1 - exp(-\sum_{u:t_u^c = t_v^c - 1} \theta_{uv})) \tag{3.6}$$

The first term corresponds to unsuccessful attempts by u in activating node v, whereas the second term corresponds to the successful activation attempts. Notice that $p_{u,v}$ is 0 whenever u is not a parent of v. Since the likelihood function is concave, we can maximize it for each node v and each cascade c separately, using gradient descent, using

$$\vec{\theta}_{*v}^{q+1} = \vec{\theta}_{*v}^{q} - \eta_q \nabla(-L_v^c(.)) \tag{3.7}$$

where $\nabla(-L_v^c(.))$ is the gradient of $-L_v^c(.)$ and $\vec{\theta}_{*v}$ is the vector containing the θ_{uv} from all parent nodes u of v. Here, q corresponds to the gradient descent iteration and η_q is the step size used in iteration q.

Online MLE Learning: We use a result from online optimization of convex functions, to maximize the likelihood function above in an online streaming setting. Zinkevich [52] develops a method for online optimization of convex functions over a convex set. In Zinkevich's framework, there is a fixed convex set F of n-dimensional points, known in advance. A series of convex functions $c^s : F \to R$ stream in, with c^s being revealed at time s. This framework makes no assumption about the distribution of the c^s functions. At each timestep s, the online algorithm must choose a point $x^s \in F$ *before* seeing the convex function c^s. The objective is to minimize the total cost across T timesteps, i.e., $Cost_{on}(T) = \sum_{s\in[T]} c^s(x^s)$, for a given finite T. Consider an alternative offline setting where the cost functions c^s are revealed in advance $s \in [T]$, for T timesteps. The offline algorithm is required to choose a *single* point $x \in F$ such that $Cost_{off}(T) = \sum_{s\in[T]} c^s(x)$ is minimized. Zinkevich defines the *loss*[1] of the online algorithm compared to the offline algorithm as $Loss(T) = Cost_{on}(T) - Cost_{off}(T)$. Notice that the loss depends on the number of steps T and it can also be measured in an average sense, i.e., as $Loss(T)/T$. Zinkevich [52] proposes a greedy algorithm for updating the estimates x^s i.e., $x^{s+1} = x^s - \eta_s \nabla(c^s(x^s))$.

For our IM application, the cost functions c^s correspond to the *negative* likelihood function $-L_v^s(.)$ for the cascade generated in round s of the CMAB game. Notice that these functions are *convex*. Thus, the online MLE method works as follows: at each round, we pick a seed set at random, observe the resulting cascade, and update the "θ

[1]He used the term regret which we rename to loss, to avoid confusion with the regret studied in Section 3.3.

values" using Eq.(3.7), the exact same update equation used in the offline MLE algorithm. The only difference with the offline method lies in not requiring all cascades up front and in using *only* the last cascade observed (resulting from the last seed set chosen) to perform the update.

Performance of Online MLE Learning: In our online algorithm, we seek to minimize the negative likelihood function across T rounds of the CMAB game and estimate the true influence probabilities. Let F be the convex set of $\vec{\theta}$ vectors corresponding to possible values for network influence probabilities. Let $dia(F)$ be the maximum Euclidean distance d between any two points $x, y \in F$, i.e., $dia(F) := max_{x,y \in F} d(x,y)$. The objective is to minimize the total "cost" incurred until round T. We evaluate the online method of learning influence probabilities by comparing it with the offline algorithm in [40]. Let $\vec{\theta}_{batch}$ denote the parameters learned by their offline algorithm. Let $\vec{\theta}^s$ denote the parameters learned by our online algorithm at the end of round s. (Recall that $\theta_{uv} := -\ln(1-p_{u,v})$.) We can show:

Theorem 2. *Let $\vec{\theta}_{batch}$ be the set of parameters learned offline with the cascades available in batch, and $\vec{\theta}^s$ be the estimate for the parameters in round s of the CMAB framework. Let L_v^s be the likelihood function at round s for the node v under consideration, d_v be the in-degree of v, T be the total number of rounds and $G = max_{s \in [T]} ||\nabla(-L^s(\vec{\theta}_s))||$ be the maximum L2-norm of the gradient of the negative likelihood function over all rounds. Suppose we perform greedy updates to the estimates $\vec{\theta}^s$ using Eq.(3.7) with η_s decreasing as $\frac{1}{\sqrt{s}}$. Then we have the following:*

$$\sum_{s=1}^{T} (L_v^s(\vec{\theta}_{batch}) - L_v^s(\vec{\theta}^s) \leq \frac{d_v \theta_{max}^2 \sqrt{T}}{2} + (\sqrt{T} - \frac{1}{2}) G^2. \tag{3.8}$$

Proof. The proof is an adaptation of the loss result from [52], reproduced below:

$$Loss(T) \leq \frac{dia(F)^2 \sqrt{T}}{2} + (\sqrt{T} - \frac{1}{2}) ||\nabla(c_{max})||^2 \tag{3.9}$$

where $||\nabla c_{max}||$ is the maximum gradient obtained across the T rounds in the framework of [52]. Let the true influence probabilities lie in the range $(0, p_{max}]$, for some p_{max}. Then the θ values for various edges lie in the range $(0, \theta_{max})$ where $\theta_{max} = -\ln(1-p_{max})$. Our optimization variables are $\vec{\theta}$ and the cost function c^s in our setting is $-L_v^s$ where $1 \leq s \leq T$. Furthermore, in our case, $dia(F) = \sqrt{d_v}\theta_{max}$ since this is the maximum distance between any two "θ-vectors" and $Loss(T) = \sum_{s=1}^{T} (L_v^s(\vec{\theta}_{batch}) - L_v^s(\vec{\theta}^s)$, where

$\vec{\theta}_{batch}$ is the optimal value of $\vec{\theta}$, given all the cascades as a batch. Substituting these values in Eq. 3.9, we obtain Eq. 3.8, proving the theorem. □

The significance of the theorem can be best understood by considering the average loss over T rounds. The average loss $\frac{Loss(T)}{T}$ can be seen to approach 0 as T increases. This shows that with sufficiently many rounds T, the parameters learned by the online MLE algorithm are nearly as likely as those learned by the offline algorithm of [40]. As the number of cascades (rounds) becomes sufficiently large, $\vec{\theta}_{batch}$ approaches the "true" parameters and hence by the above result, as T increases, the parameters $\vec{\theta}_s$ tend to the true ones, on an average. Hence, the online approach for maximumizing the likelihood results in a consistent estimator of the influence probabilities. Before closing, we remark that the framework of [52] on which this result is based, makes no distributional assumptions on the cascades being generated and hence the above result does not depend on the network exploration algorithm used. For instance, the result holds whether successive seed sets are chosen at random or using any criteria.

3.2 Network Exploration

The objective of network exploration is to obtain good estimates of the network's influence probabilities, regardless of the loss in spread in each round and it thus requires pure exploration of the arms. We seek to minimize the error in the learned (i.e., estimated) influence probabilities $\vec{\hat{\mu}}$ w.r.t. the true influence probabilities $\vec{\mu}$ i.e. minimize $||\vec{\hat{\mu}} - \vec{\mu}||_2$. We study two exploration strategies – *random exploration*, which chooses a random superarm at each round and *strategic exploration*, which chooses the superarm which leads to the triggering of a maximum number of edges which haven't been sampled sufficiently often.

3.2.1 Random Exploration

The idea of random exploration is simple: at each round of the CMAB game, pick a seed set of size k at random. This starts a diffusion and based on the observations, we can update the current influence probabilities according to the desired feedback model. The pseudocode is given in Algorithm 1.

Frequentist Feedback: We next consider random exploration with a frequentist feedback used for estimating the influence probabilities. We already discussed the frequentist update method in Section 3.1 (see Algorithm 4 and Sections 3.1.2.1 and 3.1.2.2.1). In

Algorithm 1: RANDOM EXPLORATION(budget k, Feedback mechanism M)

for $s = 1 \rightarrow t$ **do**
  $E_S = Explore(G,k)$;
  Play the superarm E_S and observe the diffusion cascade c ;
  $\hat{\hat{\mu}}$ = UPDATE(c,M) ;

this section, we mainly focus on its performance. Specifically, we address the important question, *how many CMAB rounds, equiv., number of cascades, are required in order to learn the influence probabilities accurately?* We settle this question below for random exploration, assuming frequentist estimation of probabilities under edge level feedback.

Thereto, we resort to the *correlation decay* assumption commonly made for random processes over large graphs [40]. This assumption intuitively says that diffusion from each seed does not travel far. More formally, we assume that there is a number $\gamma \in (0,1)$ which bounds the sum of incoming probabilities of edges to any node: precisely, $\forall v \in V : \sum_{(u,v)\in E} p_{u,v} \leq (1-\gamma)$. From a straightforward application of Lemma 1 in [40], we can show that the probability that any node v in V gets active is bounded by $\frac{p_{init}}{\gamma}$, where p_{init} is the probability that node v is a seed node. When the seed budget is k and the seeds are chosen at random, we have $p_{init} = \frac{k}{|V|}$. We have the following result.

Theorem 3. *Consider an edge (u_i,v) in the network with true probability $p_i := p_{u_i,v}$. Suppose that random exploration with frequentist estimation under edge level feedback is used to learn the influence probabilities and that the number of available cascades C satisfies the inequality*

$$C \geq \frac{3\gamma|V|\ln(\frac{1}{\delta})}{\varepsilon^2 p_i k} \tag{3.10}$$

where k is the seed budget, γ is the correlation decay bound, and $|V|$ is the number of nodes in the network. Then the influence probability of the edge (u_i,v) can be learned to within a relative error of ε with probability greater than $1-\delta$.

Proof. Consider the edge (u_i,v) which corresponds to the arm i in the CMAB framework. Suppose the number of CMAB rounds that are played in all is C and of these, the number of rounds in which arm i (edge (u_i,v)) is played is C_i. By the semantics of the IC model, this is equivalent to the number of cascades in which node u_i gets activated. We establish a bound on the number of cascades C_i needed to learn the influence probability for edge

(u_i, v) to within a relative error of ε and use it to bound the total number of cascades C needed. Arm i follows a Bernoulli distribution with mean as the true probability $p_i := p_{u_i,v}$. Let the random variable $X_{i,s}$ denote the outcome of whether the edge (u_i, v) became live in the s^{th} cascade, i.e., in the s^{th} round in the CMAB game. Define $X = \sum_{s=1}^{C_i} X_{i,s}$. Using Chernoff bounds, we get

$$\Pr[X \geq (1+\varepsilon)C_i p_{u_i,v}] \leq exp(\frac{-\varepsilon^2 C_i p_i}{3})$$

Rewriting, we get

$$\Pr[\frac{\hat{\mu}_i^E - p_i}{p_i}] \geq \varepsilon] \leq exp(\frac{-\varepsilon^2 C_i p_i}{3}) \tag{3.11}$$

Bounding by δ the probability that the relative error exceeds ε, we get

$$exp(\frac{-\varepsilon^2 C_i p_i}{3}) \leq \delta \tag{3.12}$$

$$C_i \geq \frac{3\ln(1/\delta)}{p_i \varepsilon^2} \tag{3.13}$$

This gives a bound on the number of times arm i needs to be sampled in order to learn the probabilities to within a relative error of ε. If C is the total number of cascades, then we have

$$\Pr[\text{node} u_i \text{ becomes active}] \leq \frac{p_{init}}{\gamma} \tag{3.14}$$

$$C_i \leq \frac{C p_{init}}{\gamma} \tag{3.15}$$

From Eq. (3.13) and (3.15),

$$C \geq \frac{3\gamma \ln(\frac{1}{\delta})}{p_{init} \varepsilon^2 p_i} \tag{3.16}$$

$$\tag{3.17}$$

If the seeds are chosen randomly and the seed budget is k, then $p_{init} = \frac{k}{|V|}$. Plugging this into Eq.(3.17), the theorem follows. □

This result gives the minimum number of cascades required in order to learn the

influence probabilities within a given relative error, with high probability. The is also the number of rounds of CMAB needed to learn those probabilities. Thus, this result applies for both offline and online learning. The result is couched in terms of edge level feedback. We can combine the result from Theorem 1 to bound the number of cascades needed in case of node level feedback. We leave this for future work.

3.2.2 Strategic Exploration

Random exploration doesn't use information from previous rounds to to select seeds and explore the network. On the other hand, a pure exploitation strategy selects a seed set according to the estimated probabilities in every round. This leads to selection of a seed set which results in a high spread and consequently triggers a large set of edges. However, after some rounds, it stabilizes choosing the same/similar seed set in each round. Thus a large part of the network may remain unexplored. We combine ideas from these two extremes, and propose a strategic exploration algorithm: in each round s, select a seed set which will trigger the maximum number of edges that have not been explored sufficiently often until this round. We instantiate this intuition below.

Recall T_i is the number of times arm i (edge (u_i, v)) has been triggered, equivalently, number of times u_i was active in the T cascades. Writing this in explicit notation, let $T^s_{(u,v)}$ be the number of times the edge (u,v) has been triggered in the cascades 1 through s, $s \in [T]$. Define $value(u) := \sum_{v \in N^{out}(u)} \frac{1}{T_{(u,v)}+1}$. Higher the value of a node, the more unexplored (or less frequently explored) out-edges it has. Define *value-spread* of a set $S \subset V$ exactly as the expected spread $\sigma(S)$ but instead of counting activated nodes, we add up their values. Then, we can choose seeds with the maximum marginal value-spread gain w.r.t. previously chosen seeds. It is intuitively clear that this strategy will choose seeds which will result in a large number of unexplored (or less often explored) edges to be explored in the next round. We call this strategic exploration (SE). It should be noted that the value of each node is dynamically updated by SE across rounds so it effectively should result in maximizing the amount of exploration across the network. We leave deriving formal bounds for strategic exploration for future work.

3.3 Regret Minimization

We first formally define the notion of regret, tailored to the IM problem. Intuitively, given a seed budget k, we select seed sets S_s of size k in each round s, each corresponding to

Algorithm 2: EXPLORE(Graph $G = (V, E, \vec{\mu})$, budget k)

//Returns superarm for an exploration round ;
Select k nodes at random from V to form seed set S ;
Output E_S

Algorithm 3: EXPLOIT(Graph $G = (V, E, \vec{\mu})$, Oracle O, budget k)

//Returns superarm for an exploitation round ;
$S = O(G, k)$ //Oracle returns optimal seed set using current $\vec{\mu}$ estimates ;
Output E_S

a superarm E_{S_s}. By selecting seeds under imperfect knowledge, we achieve suboptimal spread. Let $\vec{\mu}$ denote the vector of true influence probabilities of the network edges. Let $r_{\vec{\mu}}(E_S)$ denote the reward from playing the superarm E_S, under the true means $\vec{\mu}$. For IM, $r_{\vec{\mu}}(E_S) = \bar{\sigma}(S)$, i.e., spread of S under the IC model, using the true influence probabilities. Let $opt_{\vec{\mu}} = max_{S:|S|=k}\sigma(S)$, i.e., the optimal spread.

If the true influence probabilities are known, using the oracle, we can achieve a spread of $\alpha \cdot \beta \cdot opt_{\vec{\mu}}$. Let $\mathscr{A}$ be any strategy used for choosing seeds in successive rounds. Let $S_s^{\mathscr{A}}$ be the seed set chosen by $\mathscr{A}$ in round s. There is a clear tradeoff between exploring and exploiting: exploiting improves the quality of seeds selected, but only relative to the current knowledge of influence probabilities, which may be imperfect; exploring improves the knowledge of influence probabilities, but at the expense of choosing seeds of lower quality. The regret incurred by $\mathscr{A}$ can be quantified as

$$Reg_{\mu,\alpha,\beta}^{\mathscr{A}} = T \cdot \alpha \cdot \beta \cdot opt_{\mu} - \mathbb{E}_S\big[\sum_{s=1}^{T} r_{\vec{\mu}}(S_s^{\mathscr{A}})\big],$$

where the expectation is over the randomness in the seed sets output by the oracle. We now discuss several strategies for seed selection over the rounds. The strategies, given as Algorithms 5-7, invoke the subroutines in Algorithms 2-4. Given a network G and budget k, Algorithm 2 outputs a random subset of size k from V as the seed set. Algorithm 3, on the same input, consults an oracle and outputs the seed set of size k that maximizes the spread according to current mean estimates. Algorithm 4 examines the latest cascade c and updates the parameters using the desired feedback mechanism M.

Combinatorial Upper Confidence Bound: The Combinatorial Upper Confidence Bound (CUCB) algorithm was proposed in [14] and shown to achieve logarithmic regret

Algorithm 4: UPDATE(Cascade c, Feedback mechanism M)

if M == *'Edge Level'* **then**
  $\forall i$ update $\hat{\mu}_i$ according to scheme in section 3.1.2.1 ;
else if M == *'Node Level Frequentist'* **then**
  $\forall i$ update $\hat{\mu}_i$ according to scheme in section 3.1.2.2.1 ;
else
  $\forall i$ update $\hat{\mu}_i$ according to scheme in section 3.1.2.2.2 ;
Output $\vec{\hat{\mu}}$

Algorithm 5: CUCB(budget k, Feedback mechanism M)

Initialization
Initialize $\vec{\hat{\mu}}$;
$\forall i$ initialize T_i ;
for $s = 1 \rightarrow T$ **do**
  E_S = Exploit(G,O,k) ;
  Play the superarm E_S and observe the diffusion cascade c ;
  $\vec{\hat{\mu}}$ = Update(c,M) ;
  $\forall$ arm i that was triggered, increment values of $T_{i,s}$;
  $\forall\ i\ \hat{\mu}_i = \hat{\mu}_i + \sqrt{\frac{3\ln(s)}{2T_i}}$;

in the number of rounds (T). It is a variant of the popular upper confidence bound algorithm. The algorithm assumes that each edge in the network has been triggered at least once initially and maintains an overestimate $\overline{\mu_i}$ of the mean estimates $\hat{\mu}_i$. $\overline{\mu_i} = \hat{\mu}_i + \sqrt{\frac{3\ln(t)}{2T_i}}$. Pure exploitation, i.e., finding the best seed set using the oracle O with the $\overline{\mu_i}$ values as input leads to implicit exploration and is able to achieve optimal regret [14]. The pseudocode appears in Algorithm 5.

ε-Greedy: Another strategy proposed for the CMAB problem in [14] is the ε-Greedy strategy. In each round s, this strategy involves exploration – select k seeds at random with probability ε_s and exploitation – with probability $1 - \varepsilon_s$ use the influence maximization oracle with the mean estimates $\hat{\mu}_i$ as input to select the seed set. Chen et al. [14] show that that if ε_s is annealed as $1/s$, logarithmic regret can be achieved. The pseudocode appears in Algorithm 6.

As mentioned above, both the CUCB and ε-greedy approaches have been shown to achieve optimal regret under edge level feedback [14]. The regret proofs for both these algorithms rely on the mean estimates. To characterize the regret for node level feedback,

Algorithm 6: ε-GREEDY(budget k, Feedback mechanism M, parameter ω)

Initialize $\hat{\vec{\mu}}$;
for $s = 1 \rightarrow T$ **do**
    $\varepsilon_s = \frac{\omega}{s}$;
    With probability $1 - \varepsilon_s$, E_S = Exploit(G,O,k) ;
    Else E_S = Explore(G, k) ;
    Play the superarm E_S and observe the diffusion cascade c ;
    $\hat{\vec{\mu}}$ = Update(c,M) ;

Algorithm 7: INITIAL EXPLORATION(budget k, Feedback mechanism M, parameter ζ)

Initialize $\vec{\hat{\mu}}$;
for $s = 1 \rightarrow T$ **do**
    if $s \leq \zeta T$ **then**
        E_S = Explore(G, k) ;
    else
        E_S = Exploit(G,O,k)
    ;
    Play the superarm E_S and observe the diffusion cascade c ;
    $\vec{\hat{\mu}}$ = Update(c,M) ;

we can use the mean estimates for node level feedback from Theorem 1 and essentially repeat the proofs to characterize the regret. For lack of space, we suppress these results and refer the reader to [14].

Initial Exploration: In this strategy, we explore for the first ζ fraction of rounds and then exploit. The pseudocode in Algorithm 7 is self-explanatory.

Pure Exploitation: This strategy exploits in every round. Even if we have no knowledge about the initial probabilities, it results in implicit exploration: the initially picked seed sets produce cascades which are used to learn the probabilities. This amounts to invoking Algorithm 3 for T rounds.

3.4 Experiments

Goals: Our main goal is to test the various algorithms and evaluate the exploration algorithms on the error in learning the influence probabilities, and regret minimization algorithms on regret achieved. Our main goal is to benchmark the proposed algorithms

on the regret minimization and network exploration tasks.

Datasets: We use 3 real datasets – NetHept, Epinions and Flickr, whose characteristics are summarized in Table 3.1. The "true" probabilities for these datasets are not available, so we had to synthetically generate them: we set them according to the weighted cascade model [33]: for an edge (u,v), the influence probability is set to $p_{u,v} = \frac{1}{|N^{in}(v)|}$. The probabilities to be learned are initialized to 0.

Dataset	| V |	| E |	Av.Degree	Max.Degree
NetHEPT	15K	31K	4.12	64
Epinions	76K	509K	13.4	3079
Flickr	105K	2.3M	43.742	5425

Table 3.1: Dataset characteristics

Experimental Setup: We simulate the diffusion in the network by sampling a deterministic world from the probabilistic graph G: for purposes of our experiments, we assume that the diffusion cascade in the real world happened according to this deterministic graph. For this, we sample each edge in the graph according to its "true" probability. We vary the size of the seed set or budget k from 10 to 100. For our influence maximization oracle, we use the TIM algorithm proposed by Tang et al. [50], which is based on the notion of reverse reachable (RR) sets. For lack of space, we refer the reader to [50] for details of the algorithm but note that (i) it obtains a $(1-\frac{1}{e}-\varepsilon)$-approximation solution in near optimal time, (ii) it is much faster than using Greedy algorithm with MC simulations, and (iii) it readily serves as a $(1-\frac{1}{e}, 1-\frac{1}{|E|})$-approximation oracle for IM. Owing to runtime constraints, we use a value of $\varepsilon = 0.5$. We have verified that smaller, more accurate values of ε give us qualitatively similar results. We run the bandit algorithm for a total of 1000 rounds. For each round, we find the "actual" spread obtained using the diffusion according to the deterministic graph generated. We find this value with seed sets obtained using both the learned probabilities as well as the "true" probabilities. Thus, we are able to calculate the regret (difference in spread) in each round of the process. We also plot the relative $L2$ error between the true and the learned probabilities. In addition to this, we find the fraction of edges f within a relative error of $p = 10\%$ and plot f as p is varied. All our results are those obtained by averaging across 3 runs.

Algorithms Compared: (abbreviations used in plots are in brackets) Regret Minimization: CUCB, ε-greedy(EG), Initial Exploration(IE), Pure Exploitation(PE). Network Exploration: Random Exploration(R), Strategic Exploration(S). Feedback Mechanisms:

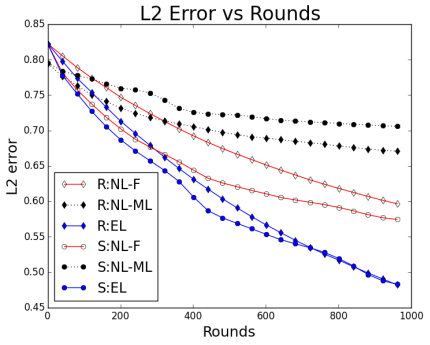
Edge Level(EL), Node Level Frequentist (NLF), Node Level Maximum Likelihood (NL-ML). Using algorithm A with feedback M is abbreviated as A-M.

In order to characterize the runtime for the various algorithms, we characterize the runtime for each of the basic components - EXPLORE, EXPLOIT and UPDATE (under all three feedback mechanisms) for all three datasets. 3.2.

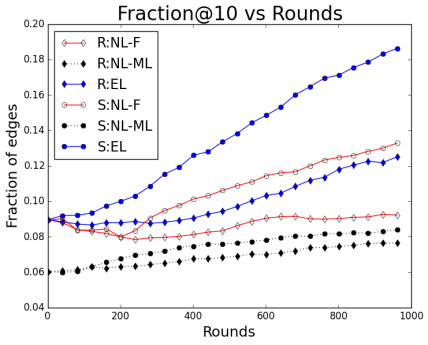
Dataset	Exploit	Explore	Update		
			EL	NL-F	NL-ML
NetHEPT	2.682	0.0002	0.026	0.0275	1.148
Epinions	245.225	0.0014	0.214	0.224	15.156
Flickr	97.808	0.0019	1.452	2.479	690.103

Table 3.2: Subroutine runtimes (in sec/round)

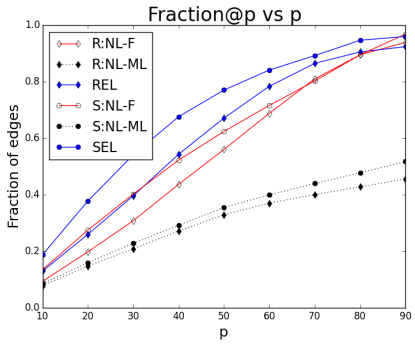
Network Exploration: Figure 3.3(a) shows the L2 error obtained by using Random Exploration and Strategic Exploration strategies, coupled with Edge level feedback and both the frequentist and maximum likelihood based node level feedback mechanisms. First, we can see that strategic exploration is better than just choosing nodes at random because it incorporates feedback from the previous rounds and explicitly tries to avoid those edges which have been sampled (often). We observe that the decrease in error for frequentist approaches is much quicker as compared to the maximum likelihood approach. As expected, edge level feedback shows the fastest decrease in error. The frequentist node level feedback takes more aggressive steps as compared to the maximum likelihood approach which is more principled and moderate towards distributing credit. However, since the failure probability is low in typical networks, the aggressiveness of the frequentist approach pays off leading to a very quick decrease in error. In Figure 3.3(b), we plot the fraction of edges which are within a relative error of 10% of their true probabilities. Figure 3.3(c) depicts an ROC-type curve which shows how the fraction of edges whose probabilities are learned within a given precision. We observe that there is a quick decrease in $L2$ error and in just 1000 rounds, our network exploration algorithm is able to learn a large percentage of edges lie within 20% error. Since we have the flexibility to generate cascades to learn about the hitherto unexplored parts of the network, our network exploration algorithms can lead to a far lesser sample complexity as compared to algorithms which try to learn the probabilities from a given set of cascades.
Regret Minimization: For our regret minimization experiments, we compare CUCB, ε-Greedy, Initial Exploration, Pure Exploitation. We explored random seed selection



(a) L2 Error



(b) Fraction of edges within 10% Rel Err

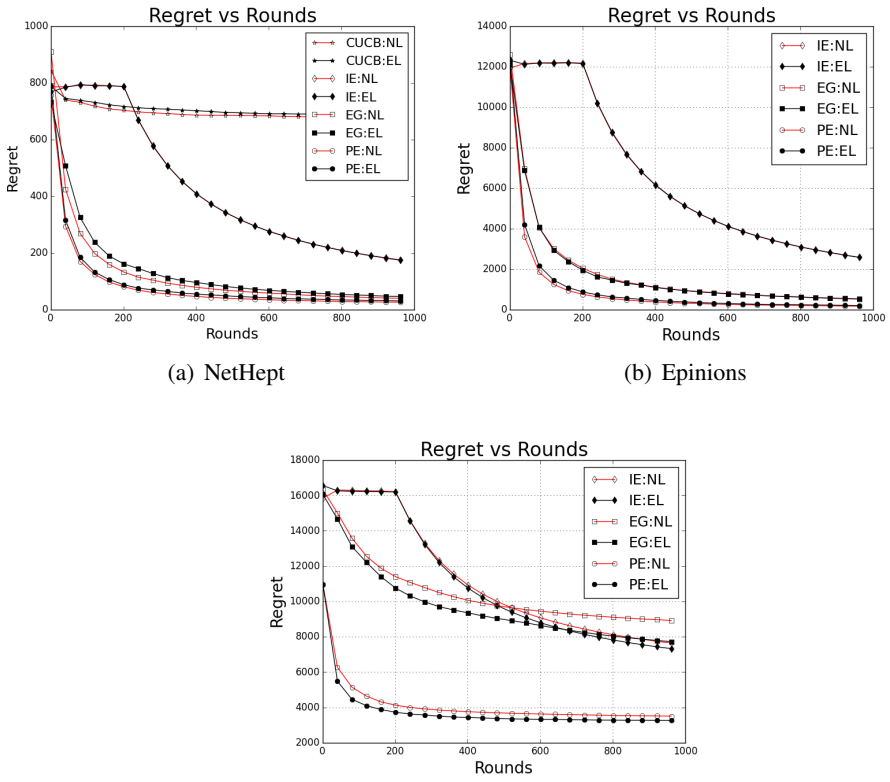


(c) Fraction of edges with p% Rel Err

Figure 3.3: NetHept,k = 50: Network Exploration

as a baseline but the regret it achieved is much worse than that of all other algorithms and it barely decreases over rounds, so we omit it completely from the plots. Since the maximum likelihood approach is not able to learn the probabilities as effectively as the frequentist approach and also has a high update runtime 3.2, we use just the frequentist approaches in the regret minimization setting. For ε-Greedy, we set the parameter $\omega = 5$ and for initial exploration we set $\zeta = 0.2$. For CUCB, if the update results in any μ_i exceeding 1, we reset it back to 1, following [14]. All the probabilities are initialized to 0. We show the results for all 3 datasets for $k = 50$ for both the node level and edge level feedback. Results for other k are similar.

Figure 3.4(a) shows the decrease in average regret as the number of rounds increases. We see that convergence for the CUCB is slow. This is because a UCB-based algorithm is biased towards exploring edges which have not been triggered often (or even once). Since



(a) NetHept (b) Epinions

(c) Flickr

Figure 3.4: $k = 50$:Regret vs Number of Rounds

typical networks consist of a large number of edges, the rate of decrease in regret for CUCB is slow. This behaviour is observed for other datasets as well, so we omit CUCB from further plots. For algorithms such as ε-Greedy and Initial Exploration, we can control the amount of exploration using the parameters ω and ζ. For these algorithms, it can be seen from Figures 3.4(a)-(c), that the regret decreases at a much faster rate, becoming very low in 1000 rounds. This implies that at the end of 1000 rounds, the probabilities are estimated well enough to lead to a comparable spread as against the known probabilities. For Initial Exploration, the regret remains almost the same during the exploration phase in the first 200 rounds, after which its regret steeply decreases. Pure Exploitation achieves the best average regret at the end of 1000 rounds. This is not uncommon for cases where the rewards are noisy. Initially, with unknown probabilities, rewards are noisy in our case, so exploiting and greedily choosing the best superarm

often leads to very good results. We also observe that the edge level and node level feedback achieve almost the same regret in several cases and this leads us to conjecture that the credit distribution in node level feedback does not lead to much error and hence the failure probability ρ is indeed low for typical social networks. In fact, we can use Eq. (3.1) to find the failure probability. For the NetHept dataset, we find that the average number of active parents K for a node is 1.175. Hence credit distribution is not a big issue. Previous work has shown that the probabilities learned from diffusion cascades are generally small [26, 40, 47]. E.g., if $p_{min} = 0$ and p_{max} varies from 0.01 to 0.2, the failure probability ρ varies from 0.0115 to 0.2261. Hence the failure probability of the proposed node level feedback is small. We obtain similar results on the Epinions and Flickr datasets as well.

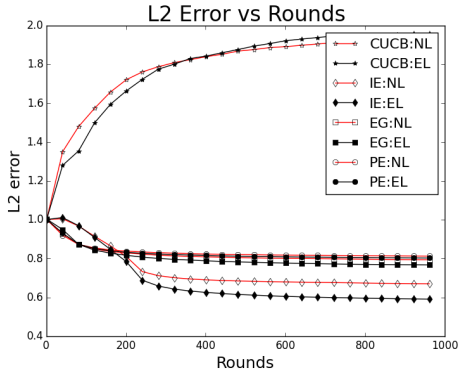


Figure 3.5: NetHept, $k = 50$: L2 error vs Number of Rounds

In order to better understand the behaviour of the algorithms, we also plot the L2 error on influence probabilities. We show the results for NetHept in Figure 3.5. As the rounds progress, the mean estimates improve and the relative L2 error goes down. This leads to better estimates of expected spread, the quality of the chosen seeds improves, the true spread increases and hence the average regret goes down (see Figure 3.4). We can see that although pure exploitation led to the best regret, it narrowed down on the seed set to be chosen quickly and did not learn about the other edges in the network. ε-Greedy and Initial Exploration however did a fair bit of exploration and hence have a lower L2 error. However, for the larger datasets, pure exploitation explores the network more before settling on a seed set. Hence, the L2 error is as small as the other algorithms. For brevity, we omit the plots. We also observe that the L2 error increases for the CUCB algorithm. This is because of the resetting of mean estimates above 1. Since the influence

probabilities are small, resetting them to 1 leads to a large error.

k	EG-NL spread	EG-EL spread
10	6549.04	6599.06
20	8696.23	8616.90
50	11998.53	11986.96
100	14770.16	14730.80

Table 3.3: Epinions: Spread vs k

We show the effect of changing k in Table 3.3. We calculate the average spread obtained using the learned probabilities across the rounds. We observe that the spread obtained using node level feedback is always close to that using edge level feedback and hence failure probability is small.

4

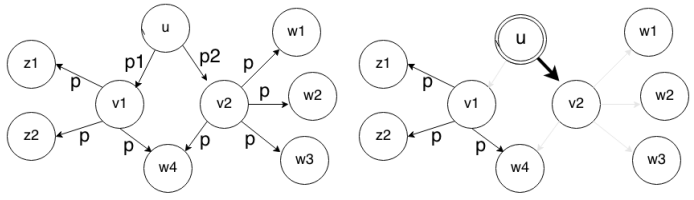
Adaptive Influence Maximization

The measure of intelligence is the ability to change.

– Albert Einstein

4.1 Theory

We introduced the idea of adaptive influence maximization in chapter 1 and hypothesized that following an adaptive strategy by reacting to market feedback can potentially lead to large gains for both the MAXSPREAD and MINTSS problems. In this chapter, we give algorithms for adaptive influence maximization and formalize the benefit of going adaptive. The toy example below provides basic intuition about the advantage of adaptive seed selection over non-adaptive seed selection.



(a) Original probabilistic network (b) Network after diffusion for one seed

Figure 4.1: Example for adaptive versus non-adaptive seed selection

Example 1 (Adaptive Seed Selection)**.** *Figure 4.1(a) shows a probabilistic network. The*

influence probabilities (edge weights) corresponding to $(u, v1)$ *and* $(u, v2)$ *are* $p1$ *and* $p2$ *respectively with* $p2 < p1$*. All remaining edges have a probability p. We need to select* $k = 2$ *seeds for this network. It is easy to see that node u has the maximum marginal gain, i.e. activating it will result in maximum number of nodes being activated in expectation. Given that u has been selected as a seed node, we observe that* $v2$ *has a higher marginal gain since it has lesser probability of being activated by u than* $v1$ *and activating it will lead to a greater number of additional nodes being influenced (*$4p$ *as compared to* $3p$ *for* $v1$*, in expectation). Hence the non-adaptive strategy will select nodes u and* $v2$ *as the seed nodes. Suppose the true world consists of 4 links –* $(u, v2)$*,* $(v1, z1)$*,* $(v1, z2)$ *and* $(v1, wz)$*. Hence the non-adaptive strategy will result in an actual spread of 2 in this true world. An adaptive strategy selects the first seed node, observes the diffusion and then selects the second seed node. It selects the first seed node as u, then observes the network feedback. The precise feedback mechanism is described later. Figure 4.1(b) shows the observed network after seeding the first node. The double-circle for u means it has already been seeded. Links shown by a bold arc (*$(u, v2)$*) are no longer probabilistic and are known to exist in the true world. Similarly, links shown by a grey arc (*$(u, v1)$ *and all links outgoing from* $v2$*) are known not to exist. The other links are still probabilistic, i.e., they have not yet been revealed in the feedback. Given this, the adaptive strategy will select node* $v1$ *as the second seed, resulting in an actual spread of 6 in this true world. This example gives an intuition on why we expect adaptive strategies to do better than non-adaptive seed selection.* □

Recall, we say time horizon is unbounded if $H \geq kD$, where k is the seed budget, D is the length of the longest path of G, and H is the time horizon. We consider unbounded time horizon up to Section 4.1.2. Bounded time horizon is addressed in subsection 4.1.3. Our first result is that our spread function based on node level feedback is adaptive monotone and adaptive submodular, thus affording an efficient approximation algorithm.

Theorem 4. *For unbounded time horizon, if the diffusion process is allowed to complete after every intervention, node level feedback is equivalent to edge level feedback w.r.t. marginal gain computation and therefore the expected spread function is adaptive submodular and adaptive monotone under node level feedback.*

Proof. We will show that node level feedback is equivalent to edge level feedback from the perspective of marginal gain computation. In [22], the authors show that the expected spread function under edge level feedback is adaptive monotone and adaptive submod-

ular. The above theorem will follow from this. Specifically, we prove that (a) for every edge level feedback based network state, there is a corresponding state based on node level feedback, which preserves marginal gains of nodes, and (b) vice versa.

Given edge level feedback, we clearly know which nodes are active. These are precisely nodes reachable from the seeds via live edge paths in the revealed network. In the rest of the proof, we show that for each node level feedback state, there is a corresponding edge level feedback state that preserves marginal gains. Let S_0 be the set of seeds chosen at time $t = 0$. Given node level feedback, we can infer the corresponding edge level feedback based network state using the following rules. Consider an edge from an active node u to node v. Notice that the status of an edge leaving an inactive node is unknown in either feedback model.

Rule 1: If node u is active, v is inactive, and there is an edge from u to v, then infer that edge (u,v) is dead.

Rule 2: If nodes u and v are both active and u is the only in-neighbor of v, then conclude that the edge (u,v) is live.

Rule 3: If nodes u and v are both active and u has more than one in-neighbor, arbitrarily set the status of the edge (u,v) to be live or dead.

We now show that the way edge status is chosen to be live or dead in Rule 3 plays no role in determining the marginal gains of nodes. We make the observation that if the diffusion process is allowed to complete after each intervention, the only extra information about the network that is observed using edge level feedback over node level feedback is the status of edges between 2 active nodes. Given that the node u is active, we need to calculate the marginal gain of every other node in the network for the next intervention. Next we show that the status of edges between 2 active nodes does not matter in the marginal gain computation for any node.

For the rest of the argument, we consider the both u and v are active and that v has multiple active in-neighbours, i.e., the case that is addressed by Rule 3. Consider an arbitrary node w the marginal gain of which we need to calculate. There maybe multiple paths from w to a node reachable from w. These paths can be classified into those which involve the edge (u,v) and ones which don't. The marginal gain computation involving the latter paths is independent of the status of the edge (u,v). Since the diffusion process is allowed to complete, all nodes which can be reached (in the "true" possible world) from w through (u,v) have already been activated. Hence paths going through (u,v) do not contribute to the marginal gain for w. Thus, the status of the edge (u,v) does

not matter. Since w is any arbitrary node, we can conclude that the marginal gain of every node remains the same under states based on both feedback models. Adaptive monotonicity and submodularity are properties of marginal gains. Since marginal gains are preserved between edge level and node level feedback, it follows that these properties carry over to our node level feedback model. □

4.1.1 MAXSPREAD

There are four types of policies – the greedy non-adaptive policy (abbreviated GNA), the greedy adaptive policy (GA), the optimal non-adaptive policy (ONA) and the optimal adaptive policy (OA). We use $\pi_{GA,k}$ to denote the greedy adaptive policy constrained to select k seeds and $\sigma(\pi_{GA,k})$ to refer to the expected spread for this policy. While previous results bound the performance of greedy (adaptive) policies in relation to optimal adaptive policies, they do not shed light on practically implementable policies under either setting. These previous results do not quite answer the question "What do we gain in practice by going adaptive?" since both the optimal non-adaptive or optimal adaptive policies are intractable. We establish relations between two key practical (and hence implementable!) kinds of policies – the greedy non-adaptive policy and the greedy adaptive policy – for both MAXSPREAD and MINTSS. These relations quantify the average "adaptivity gain", i.e., the average benefit one can obtain by going adaptive.

We first restate Theorem 7 from [16]. This theorem gives a relation between the spreads obtained using a batch greedy adaptive policy which is constrained to select seeds in batches of size b and the optimal adaptive policy.

Fact 1. *If $\sigma(\pi_{GA,lb})$ is the average spread obtained by using a greedy batch policy with a batchsize b and $\sigma(\pi_{OA,mb})$ is the spread using an optimal sequential policy (the optimal policy if we are able to select one seed per intervention) constrained to selecting a number of seeds divisible by the batchsize b, then*

$$\sigma(\pi_{GA,lb}) > (1 - e^{\frac{-l}{\alpha\gamma m}})\sigma(\pi_{OA,mb}) \tag{4.1}$$

where α is the multiplicative error in calculating the marginal gains. gamma is a constant and equal to $(\frac{e}{e-1})^2$.

Proposition 1. *Let the horizon be unbounded. Let $\pi_{GA,n_{GA}}$ be the greedy batch policy that select n_{GA} seeds overall in batches of size b_{GA}, and let $\pi_{OA,n_{OA}}$ be the optimal adaptive*

policy that selects n_{OA} seeds overall in batches of size b_{OA}. Then

$$\sigma(\pi_{GA,n_{GA}}) \geq \left[1 - \exp\left(-\frac{\lceil \frac{n_{GA}}{b_{GA}} \rceil}{\alpha\gamma\lceil \frac{n_{OA}}{b_{OA}} \rceil}\right)\right]\sigma(\pi_{OA,n_{OA}}) \tag{4.2}$$

where $\alpha \geq 1$ is the multiplicative error in calculating the marginal gains and $\gamma = (\frac{e}{e-1})^2$ is a constant.

Proof. Fact 4.1 gives a relation between the spreads obtained by a batch greedy adaptive policy constrained to select lb seeds and the optimal adaptive policy constrained to select mb seeds. Both these policies are constrained to select seeds in batches of size b. The relation is in terms of the number of batches used by the policies. Let l and m be the number of batches for the greedy and optimal policies respectively. We make the following observations. First, the two policies can be constrained to select seeds in different batchsizes, b_{GA} and b_{OA} respectively. Next, the number of seeds selected by the policies need not be divisible by the batchsizes. We can follow a similar proof procedure as Theorem 7 in [16] and replace l by $\lceil \frac{n_{GA}}{b_{GA}} \rceil$ and m by $\lceil \frac{n_{OA}}{b_{OA}} \rceil$. □

Theorem 5. *Let $\pi_{GNA,k}$ be a greedy non-adaptive policy, $\pi_{GA,k}$ and $\pi_{OA,k}$ be the greedy and optimal adaptive policies respectively with batch-sizes equal to one i.e. the adaptive policies are sequential. All policies are constrained to select k seeds. Then we have the following relations:*

$$\sigma(\pi_{GA,k}) \geq (1 - e^{-1/\alpha\gamma})\sigma(\pi_{OA,k}) \tag{4.3}$$

$$\sigma(\pi_{GNA,k}) \geq (1 - \frac{1}{e} - \varepsilon)^2\sigma(\pi_{OA,k}) \tag{4.4}$$

Proof. Proposition 1 gives us bounds on the ratio of the spread achieved by batch-greedy adaptive policy and that achieved by the optimal adaptive policy. We set $n_{OA} = k$ and $b_{GA} = b_{OA} = 1$ and obtain equation 4.3 of the theorem.

Theorem 2 of [3] states that for a submodular monotone function, there exists a non-adaptive policy which obtains $(1 - 1/e - \varepsilon)$ fraction of the value of the optimal adaptive policy. In our context, this implies that the spread due to an optimal non-adaptive policy constrained to select n_{ONA} seeds is within a $(1 - e^{-n_{ONA}/n_{OA}} - \varepsilon)$ factor of the spread of an optimal adaptive policy selecting n_{OA} seeds. More precisely,

$$\sigma(\pi_{ONA,n_{ONA}}) \geq (1 - e^{-n_{ONA}/n_{OA}} - \varepsilon)\sigma(\pi_{OA,n_{OA}}) \tag{4.5}$$

The classical result from Nemhauser [39] states that the greedy non-adaptive algorithm obtains a $(1-1/e-\varepsilon)$ fraction of the value of the optimal non-adaptive algorithm, where ε is the additive error made in the marginal gain computation. Moreover if the greedy non-adaptive policy is constrained to select n_{GNA} seeds and the optimal non-adaptive policy selects n_{ONA} seeds we have the following:

$$\sigma(\pi_{GNA,n_{GNA}}) \geq (1-e^{-n_{GNA}/n_{ONA}}-\varepsilon)\sigma(\pi_{ONA,n_{ONA}}) \tag{4.6}$$

Combining equations 4.5 and 4.6, we obtain the following result

$$\sigma(\pi_{GNA,n_{GNA}}) \geq (1-e^{-n_{GNA}/n_{OA}}-\varepsilon)(1-e^{-n_{GNA}/n_{OA}}-\varepsilon)\sigma(\pi_{OA,n_{OA}}) \tag{4.7}$$

Setting $n_{GNA} = n_{GA} = n_{OA} = k$, we obtain equation 4.4 of the theorem. □

Discussion: To clarify what this theorem implies, lets assume that we can estimate the marginal gains perfectly. Let's set $\varepsilon = 0$ and $\alpha = 1$. We thus obtain the following relations: $\sigma(\pi_{GA,k}) \geq (1-e^{-1/\gamma})\sigma(\pi_{OA,k})$ and $\sigma(\pi_{GNA,k}) \geq (1-\frac{1}{e})^2\sigma(\pi_{OA,k})$. These two factors are almost equal (in fact non-adaptive is slightly better) and in the case of perfect marginal estimation, there is not much gain in going adaptive. This intuition is confirmed by our experiments in section 4.3.

4.1.2 MINTSS

Given that it takes the optimal adaptive policy n_{OA} seeds to achieve a spread of Q, we seek to find the number of seeds that it will take the greedy adaptive and traditional greedy non-adaptive policy to achieve the same spread. Since the non-adaptive policy can be guaranteed to achieve the target spread only in expectation, we allow it to have a small shortfall β_{ONA}. In addition, we allow both the greedy policies to have a small shortfall against their optimal variants. We formalize these notions in the following theorem.

Theorem 6. *Let the target spread to be achieved by the optimal adaptive policy be Q. Let the allowable shortfall for the optimal non-adaptive policy over the optimal adaptive policy be β_{ONA}. Let β_{GA} and β_{GNA} be the shortfall for the greedy adaptive and non-adaptive policies over their optimal variants. Let the number of seeds required by the four policies - OA, ONA, GA and GNA be n_{OA}, n_{ONA}, n_{GA} and n_{GNA}. Then we have the following relations*

$$n_{GA} \leq n_{OA}(\alpha\gamma\ln(Q/\beta_{GA})) \tag{4.8}$$

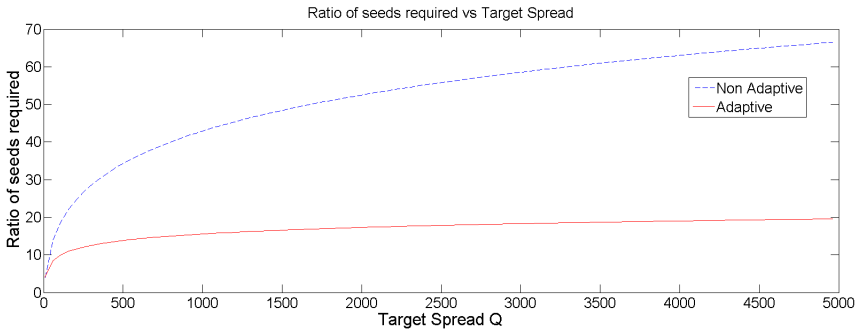


Figure 4.2: Theoretical comparison of adaptive and non-adaptive strategies

$$n_{GNA} \leq n_{OA} \ln\left(\frac{Q}{\beta_{ONA} - Q\varepsilon}\right) \ln\left(\frac{Q - \beta_{ONA}}{\beta_{GNA} - \varepsilon(Q - \beta_{ONA})}\right) \tag{4.9}$$

$$n_{GNA} \leq n_{OA} \ln\left(\frac{Q}{\beta_{GA} - \beta_{GNA} - Q\varepsilon}\right) \ln\left(\frac{Q - \beta_{GA} + \beta_{GNA}}{\beta_{GNA} - \varepsilon(Q - \beta_{GA} + \beta_{GNA})}\right) \tag{4.10}$$

Proof. If in proposition 1, we set $b_{GA} = b_{OA} = 1$, $\sigma(\pi_{OA,n_{OA}}) = Q$ and $\sigma(\pi_{GA,n_{GA}}) = Q - \beta_{GA}$, after some algebraic manipulation we can obtain equation 4.8 of the theorem. Setting $\sigma(\pi_{ONA,n_{ONA}}) = Q - \beta_{ONA}$, $\sigma(\pi_{OA,n_{OA}}) = Q$ in equation 4.5, we obtain the intermediate relation 4.11.

$$n_{ONA} \leq n_{OA} \ln(\frac{Q}{Q - \beta_{ONA}}) \tag{4.11}$$

Setting $\sigma(\pi_{ONA,n_{ONA}}) = Q - \beta_{ONA}$ and $\sigma(\pi_{GNA,n_{GNA}}) = Q - \beta_{ONA} - \beta_{GNA}$, we obtain the following relation.

$$n_{GNA} \leq n_{ONA} \ln(\frac{Q - \beta_{ONA}}{\beta_{GNA} - \varepsilon(Q - \beta_{ONA})}) \tag{4.12}$$

We constrain the spreads for the greedy adaptive and greedy non-adaptive policies to be the same. Hence, $Q - \beta_{GA} = Q - \beta_{ONA} - \beta_{GNA}$. Hence $\beta_{ONA} = \beta_{GA} - \beta_{GNA}$. By combining equations 4.11 and 4.12 and substituting β_{ONA} as $\beta_{GA} - \beta_{GNA}$, we obtain equation 4.10 of the theorem. □

Discussion: To understand the implications of this theorem, set $\alpha = 1$, $\varepsilon = 0$. Let the $\beta_{GNA} = 2$ and $\beta_{GA} = 1$, thus allowing for a shortfall of only 2 nodes in the spread. We obtain the following relations: $n_{GA} \leq n_{OA} \gamma \ln(Q/2)$ and $n_{GNA} \leq n_{OA} \ln(Q) \ln((Q-1)/2)$.

Figure 4.2 shows the growth of these functions with Q. We can see that as Q increases, the ratio $\frac{n_{GA}}{n_{OA}}$ grows much slower than $\frac{n_{GNA}}{n_{OA}}$. Hence, for the MINTSSproblem, there is clearly an advantage on going adaptive. This is confirmed by our experiments in

section 4.3.

4.1.3 Bounded Time Horizon

In discrete diffusion models (e.g., IC), each time-step represents one hop in the graph, so the time needed for a diffusion to complete is bounded by D, the longest simple path in the network. In networks where this length is small [41], most diffusions complete within a short time. This is also helped by the fact that in practice, influence probabilities are small. However, if we are given a very short time horizon, the diffusion process may not complete. In this case, seed selection is forced to be based on observations of incomplete diffusions. We show that the spread function in this case is no longer adaptive submodular.

Theorem 7. *The spread function with the IC diffusion model is not adaptive submodular if the diffusion process after each intervention is not allowed to complete.*

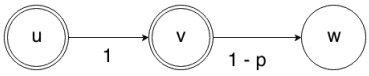


Figure 4.3: Counterexample to show that the spread is not adaptive submodular under incomplete diffusion

Proof. We give a counterexample. Consider the network shown in Figure 4.3 and the true world, where the edge (u,v) is live and (v,w) is dead. Let $H = 2$, $k = 2$. Suppose at $t = 0$, we choose the seed set $S = \{u\}$, so the next intervention must be made at time $t = 1$. Based on the true world, we observe that nodes u and v are active at time $t = 1$. Hence we infer the edge (u,v) to be live. We do not know the status of edge (v,w). Even though w is reachable in the network G, there is incomplete information in the realization revealed at $t = 1$ to decide if the node w is active or not, since the observed diffusion is incomplete. Thus, the expected spreads w.r.t. the realization above are as follows: $\sigma(S) = 2 + (1-p)$ and $\sigma(S \cup \{w\}) = 3$. Let $S' = \{u,v\}$. Then $\sigma(S') = 2$ and $\sigma(S' \cup \{w\}) = 3$. This is because w is one hop away from $v \in S'$ and the realization tells us that w is not active. Thus, we have $\sigma(S \cup \{w\}) - \sigma(S) < \sigma(S' \cup \{w\}) - \sigma(S')$. This was to be shown. □

What are our options, given that the spread under bounded time horizon is in general not adaptive submodular? Theorem 24 in [22] shows that if a function is not adaptive submodular, no polynomial algorithm can approximate the optimal expected spread

within any reasonable factor. Thus, we may continue to use adaptive greedy policy, but without any guarantees in general. In our experiments 4.3, we use a novel Sequential Model Based Optimization (SMBO) approach for finding a reasonably good policy when the time horizon is bounded.

4.2 Algorithms

To obtain a greedy adaptive policy, we need to repeatedly select nodes with the maximum marginal gain at every intervention. This implies that we need to run the greedy influence maximization algorithm to compute the marginal gain over the entire network multiple times. Fortunately, this can be done efficiently by exploiting the recent work [50] which describes a near-optimal and efficient greedy algorithm – Two-phase Influence Maximization (TIM) for non-adaptive influence maximization. We first review TIM and describe the modifications we made to it for the adaptive case.

4.2.1 Two phase Influence Maximization

Overview of TIM: Given a budget of k seeds, a network with m edges and n nodes and an appropriate diffusion model such as IC, TIM obtains a $(1-1/e-\varepsilon)$ fraction of the optimal spread in the non-adaptive case, incurring a near-optimal runtime complexity of $\mathcal{O}(k+l)(n+m)logn/\varepsilon^2$. TIM operates by generating a large number of random Reverse Reachable (RR) sets. An RR set is defined for a particular node v and a possible world W of the network. It consists of the set of nodes that can reach the node v in the possible world W. Given enough number (see [50] for an explicit bound) of RR sets, the nodes which cover a large number of RR sets are chosen as the seed nodes: the node u which appears in the maximum number of RR sets is chosen as the first seed. Once a node u is selected as a seed, we remove all the RR sets containing u and the next seed is the node which covers the maximum of the remaining RR sets and so on until a seed set S with k nodes is chosen. Tang et al. [50] show that this simple strategy is enough to guarantee a $(1-1/e-\varepsilon)$-approximation factor in near optimal time.

Adaptive TIM: In a greedy adaptive policy, we need to select seed nodes in every intervention. After each intervention, a certain number of nodes are influenced and become active. These already active nodes should not be selected as seeds. To ensure this, we eliminate all RR sets covered by any of these active nodes. If the number of nodes which became active is large, it brings the number of remaining RR sets below the required

bound, which in turn can invalidate the theoretical guarantees of TIM, as the marginal gain of seeds selected in the next intervention may not be estimated accurately. Hence after each intervention, we need to re-generate the RR sets to effectively select seeds for the next intervention. To avoid this expensive repeated RR set generation, we instead eliminate all active nodes from the original network, by making all the incoming and outgoing edges have a zero probability, and generating the required number of RR sets for the new modified network. This guarantees that the resulting RR sets do not contain the already active nodes. This is equivalent to running the greedy non-adaptive algorithm multiple times on modified networks and results in retaining preserves the theoretical guarantees of TIM. For the unbounded time horizon, the optimal policy consists of selecting one seed per intervention and letting the diffusion complete. For the IC model, the diffusion can take a maximum of D time steps.

4.2.2 Sequential Model Based Optimization

In the case of bounded time horizon (i.e., $H < kD$), as discussed at the end of Section 4.1.3, there is no straightforward strategy to find or approximate the optimal policy. The policy depends on the precise values of time horizon H and properties of the network. For MAXSPREAD the two extreme cases are the non-adaptive policy and the completely sequential greedy policy. The non-adaptive policy does not take any feedback into account and is hence suboptimal. For a sequential policy, the inter-intervention time H/k will be less than D. Hence the completely sequential policy will result in incomplete diffusions and from 7 will be suboptimal. A similar reasoning applies for MINTSSḞor both problems, we are either forced to seed more than one node per intervention or wait for less than D time-steps between interventions, or both. We split the problem of finding the optimal policy into two parts - finding the intervention times and the number of nodes to be seeded at each intervention and which nodes need to be seeded at each intervention. Using the logic in 7, we solve the latter problem by using the adaptive TIM algorithm described above. For the former problem, we resort to a heuristic approach since the expected spread function we need to optimize does not have any nice algebraic properties w.r.t. time. In order to find the best offline policy, we need to calculate σ for each candidate policy. Calculating σ across all the candidate possible worlds is expensive. Thus we need to maximize an expensive function without any nice mathematical properties. Hence we resort to a bayesian optimization technique known as sequential model based optimization (SMBO)[32]. The SMBO approach narrows down on the promising

configurations (in our case, policies) to optimize a certain function. It iterates between fitting models and using them to make choices about which configurations to investigate.

We now show the above problems can be encoded for solving these problems using SMBO. Consider MAXSPREADẆe have a maximum of k interventions. Some of these interventions may seed multiple nodes whereas other might not seed any. There are another $k-1$ variables corresponding to the inter-intervention times. Since the number of variables is $2k-1$, SMBO techniques will slow down as k increases. It is also non-trivial to add the constraint that the sum of seeds across all interventions will add to k. Since this leads to an unmanageable number of variables for large k, we introduce a parameter p which we refer to as the policy complexity. Essentially, p encodes the number of degrees of freedom a policy can have. For every $i < p$, we have a variable s_i which is the number of nodes to be seeded at a particular intervention. We have also have a variable t_i which encodes waiting time before making the next intervention. For example, if $p = 2$ and $s1 = 2, t1 = 5, s2 = 3, t2 = 7$ we initially seed 2 nodes, wait for 5 time-steps, then seed 3 nodes, wait for 7 time-steps before the next intervention. In the next intervention, we repeat the above procedure, until we run out of time, i.e., reach H or get too close (within $s1$ or $s2$) to the budget of k seeds. In the latter case, the last intervention just consists of using the remaining seeds. We use the same strategy to encode policies for MINTSSİn this case, however, we stop if the time reaches H or if $\geq Q$ nodes become active. Since we have a manageable number of parameters, we can easily use SMBO techniques to optimize over these parameters. The objective function for the first problem is to maximize the spread. The constraint is covered by the encoding. For the second problem, the objective function is to minimize the seeds to achieve a spread of Q. This can be modelled by introducing penalty parameters λ_1 and λ_2. The function can be written as,

$$\textit{minimize } g(x) + \lambda_1(Q - f(x)) + \lambda_2(f(x) - Q) \tag{4.13}$$

where x is the parameter vector, $g(x)$ is the number of seeds, $f(x)$ is the spread, Q is the target spread. The parameter λ_1 penalizes not achieving the target spread whereas λ_2 penalizes over-shooting the target spread. λ_1 encodes the hard constraint whereas λ_2 is used to direct the search.

4.3 Experiments

4.3.1 Datasets

We run all our experiments on 3 real datasets – the author collaboration network NetHEPT (15k nodes and 62k edges), the trust network Epinions (75k nodes and 500k edges) and Flixster. On NetHEPT and Epinions where real influence probabilities are not available, we set the probability of an edge into a node v to $1/indegree(v)$, following the popular approach [10, 11, 51]. We use the Flixster network under the topic-aware independent cascade model of diffusion [6] for which the authors learned the probabilities using Expectation Maximization. Their processed network has 29k nodes and 10 topics. We choose the topic which results in the maximum number of non-zero edge weights. The resulting sub-network of Flixster consists of 29k nodes and 200k edges.

4.3.2 Experimental Setup

As mentioned earlier, we consider only the IC model of diffusion. We compare between greedy non-adaptive, greedy sequential adaptive and the batch-greedy adaptive policies. Since the actual true world is not known, we sample each edge in the network according to its influence probability and generate multiple true worlds. Since we are interested in the performance of a policy on an average, we randomly generate generate 100 true worlds and average our results across them. For either problem, the seeds selected by the non-adaptive policy is based on expected case calculations and remain the same irrespective of the true world. Only the performance of the policy is affected by the true possible world. Also note that for MINTSS in some true worlds the spread of the non-adaptive policy might be less than the target Q. The shortfall can be modelled by the factor β introduced in Section 4.1.

4.3.3 Sequential Model Based Optimization

We use Sequential Model-Based Optimization for General Algorithm Configuration (SMAC) [32]. SMAC is the state of the art tool used for automatic algorithm configuration for hard problems including SAT and Integer Linear Programming. Based on the performance of a configuration on certain kinds of benchmark instances characterized by problem specific properties, SMAC creates a random forest model and uses it to choose promising candidate configurations to be evaluated. SMAC uses the model's predictive distribution

to compute its expected positive improvement over the the incumbent (current best solution). This approach automatically trades exploitation for exploration. SMAC can easily handle both numerical and categorical parameters.

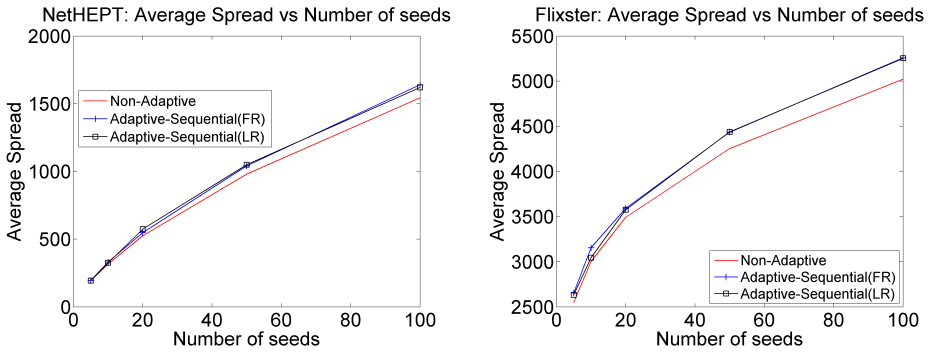
For our case, we need to optimize an expensive black-box function (as defined in the previous section) over $2p$ configurations where p is the policy complexity. Because the function is hard to evaluate a simple brute-force grid search over the parameter space is not feasible. SMAC is implicitly able to leverage the structure present in the problem and come up with promising solutions to the problem.

The benchmark instances consist of seeds for the random process generating 10 true worlds at a time. Hence, the evaluation of each configuration on each instance involves running the algorithm 10 times. We use a training set of 1000 such instances and a separate test set of 50 instances to evaluate the policies found by SMAC. We restrict the number of function evaluations SMAC can make to 500 and set the tuner timeout (the maximum time that can be spent by SMAC in building the random forest model and deciding which configuration to evaluate next) is set to 100 seconds.

4.3.4 MAXSPREAD

4.3.4.1 Unbounded time horizon

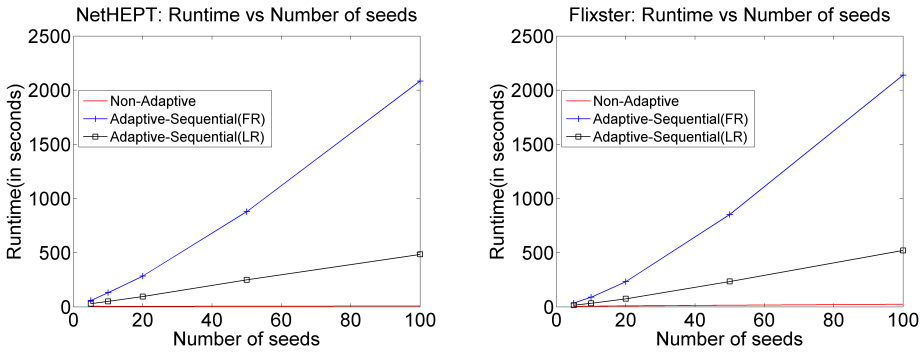
For MAXSPREAD we vary the number of seeds k over $\{1, 10, 20, 50, 100\}$. For the unbounded horizon, we compute the spread obtained using the greedy non-adaptive and the greedy adaptive sequential policies. We set $\varepsilon = 0.1$.



(a) NetHEPT (b) Flixster

Figure 4.4: Average Spread vs Number of seeds

Figures 4.4(a) and 4.4(b) show the average spread σ across 100 possible true worlds as the number of seeds is varied in the given range. We quantify the the effect of adaptivity by the ratio $\frac{\sigma(\pi_{GA})}{\sigma(\pi_{GNA})}$, which we call the average adaptivity gain. We see that the average adaptivity gain remains constant as the number of seeds are varied. We obtain similar results even with higher (100 to 500) values of k. This finding is consistent with the observations made in Section 4.1.

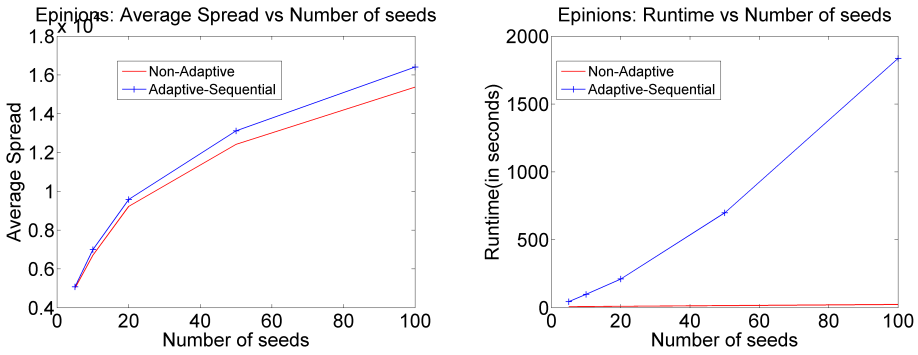


(a) NetHEPT (b) Flixster

Figure 4.5: Runtime vs Number of seeds

For the adaptive greedy sequential strategy in which we select one seed at a time, we generate RR sets for $k = 1$ and regenerate the RR sets between each pair of interventions. The run-time graphs are shown in Figures 4.5(a) and 4.5(b). As can be seen, although this method scales linearly with the number of seeds, it is much slower than the non-adaptive case and will prohibitive for larger datasets. Instead we can generate a large number of RR sets upfront and use these sets to select seeds for the first few interventions. The RR sets are regenerated as soon as the change in the number of active nodes becomes greater than a certain threshold (the regeneration threshold θ). The intuition is that if the number of active nodes has not increased much in a few interventions, the number of RR sets does not decline significantly and they still represent information about the state of the network well. We call this optimization trick lazy RR(LR) set regeneration to contrast it with the full RR(FR) set regeneration. We observe that because of submodularity, the frequency of RR set (re)generation decreases as the number of seeds (and time) increases. For our experiments, we empirically set θ equal to 10. Higher values of θ lead to lower runtimes but to a smaller average adaptivity gain.

We use this strategy to find the spread for both NetHEPT and Flixster. As can be seen



(a) Average Spread vs Number of seeds

(b) Runtime vs Number of seeds

Figure 4.6: Epinions:Adaptive vs Non-adaptive for MAXSPREAD

from the runtime graphs and average spread graphs, this strategy does not decrease the spread much but leads to significant computational savings. After verifying this strategy, we use it to compare the 2 policies on the larger Epinions dataset, where the same trend is observed – see Figures 4.6(a) and 4.6(b). The average adaptivity gain is small even for the greedy adaptive sequential policy in case of unbounded time horizon.

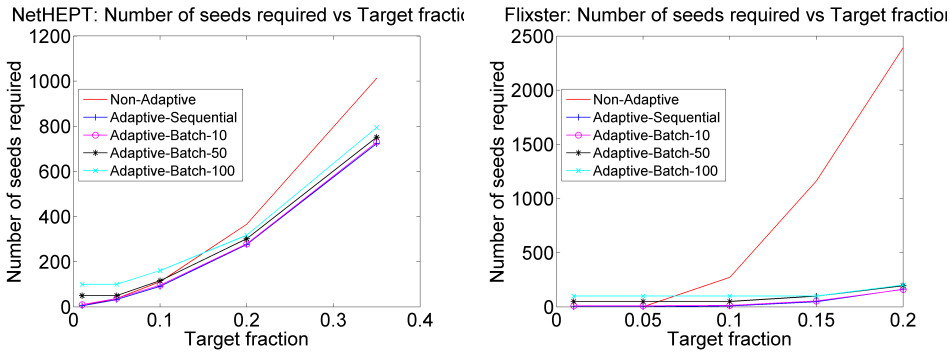
4.3.4.2 Bounded time horizon

For the bounded time horizon, the policy will be forced to group sets of seeds together to form a batch. From Fact 4.1, we know that the average spread for such a policy will be lower and hence the average adaptivity gain will further decrease. To verify this, we conduct an experiment on the NetHEPT dataset in which we decrease the time horizon T from a large value (corresponding to unbounded time horizon) to low values of the order of the length of the longest path in the network. We vary the policy complexity p to be 1 or 2 in this case. We aim to find the best configuration by varying the batch-size in the range 1 to 100 and the inter-intervention time between 1 and the D of the network. Since the difference between the spreads for the non-adaptive policy vs. the greedy adaptive sequential policy is so small, for the bounded time horizon, SMAC is unable to find a unique optimal policy. Different runs of SMAC yield different policies for the same number of seeds, sometimes converging to the non-adaptive policy even for reasonably large time horizons! A higher configuration time for SMAC might lead to stable results or alternatively we might need to encode the problem differently. We leave this for future work.

4.3.5 MINTSS

4.3.5.1 Unbounded time horizon

For all 3 datasets, for the unbounded time horizon, we compare the greedy non-adaptive and greedy adaptive policies with different batch sizes in the range $\{1, 10, 50, 100\}$. Because a large number of seeds may be required to saturate a certain fraction of the network, we use the lazy RR set generation approximation explained above and set ε to 0.2. Figures 4.7(a), 4.7(b) show the comparison between the non-adaptive and various adaptive greedy policies for the NetHEPT and Flixster datasets. Epinions shows a similar trend.



(a) NetHEPT (b) Flixster

Figure 4.7: Number of seeds required vs Target fraction

As can be seen, the non-adaptive policy is competitive for smaller number of target nodes. But as the target fraction increases, the adaptive policies are better able to exploit the market feedback mechanism and lead to large savings in the number of seeds. This again agrees with our theoretical results which showed that the adaptivity gain increases as the number of target nodes increases. As the size of the network increases, the estimated spread calculation in the non-adaptive case is averaged across greater number of true worlds and hence becomes less efficient. We observed that in many cases, the final true spread for the non-adaptive policy either overshoots the target spread or misses the target spread by a large amount. We conclude that adaptive policies are particularly useful if we want to influence a significant fraction of the network.

We give some intuition for the difference in the adaptivity gains for the two problems. For adaptive policies, the rate of increase in the expected spread is fast in the

beginning before the effects of submodularity take over. Hence adaptive policies require fewer seeds than non-adaptive to reach a comparable target spread. However, once submodularity kicks in, the additional seeds added contribute relatively little to the spread. Hence for MINTSS, where the objective is to reach a target spread with minimum seeds, the adaptivity gain is higher. However for MAXSPREAD, even though the adaptive policy reaches a high spread with fewer seeds, the remaining seeds in the budget don't add much more to the true spread.

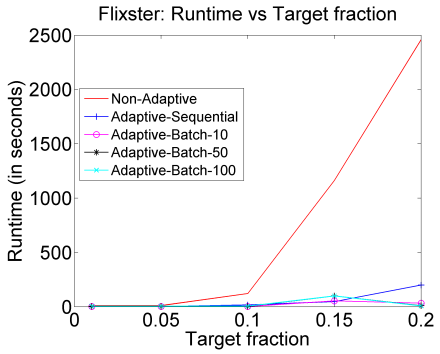


Figure 4.8: Flixster: Runtime vs Target fraction

We also plot the runtime graph for the Flixster dataset. The non-adaptive time dominates because it needs to choose a larger number of seeds. Since the batch-greedy policies select batches of seeds and consider feedback less often, they have a lower running time which decreases as the batch-size increases. Figure 4.8 shows the runtime variation for Flixster. Results on other datasets show a similar trend.

4.3.5.2 Bounded time horizon

T	10	50	100	1000
ShortFall (β)	709	174.98	10.54	0
Number of seeds	200	177.33	171.11	168
Objective function	7290	1927.1	276.51	168
Policy(s,t)	(100,6)	(28,8)	(20,11)	(3,12)

Table 4.1: Policies of $p = 1$ recovered by SMAC for varying time horizons(T) for Flixster with Q = 5800

We now consider the important question, how good is the effect of adaptivity for a bounded time horizon for the MINTSS problem. For this, we vary the time horizon T

from 10 to 1000 and the policy complexity p is set to either 1 or 2. We use the Flixster dataset and fix the target fraction of nodes to 0.2. As in the previous problem, we aim to find the best configuration by varying the batch size in the range 1-100 and the inter-intervention time between 1 and the D of the network. Since each configuration run involves solving MINTSS 500 times, to save computation time we use a relatively high $\varepsilon = 0.5$. We verified that similar results hold for smaller values of ε. The optimal policy returned by SMAC is evaluated on a different set of instances (possible true worlds) averaging the results over 50 such instances.

Table 4.1 shows the results for this experiment. For both $p = 1, 2$, as the time horizon increases, the shortfall goes to zero and the objective function is just the number of seeds required. We see that even for a low time horizon, SMAC is able to find a policy for which the number of seeds is close to the policy (which uses 163 seeds) for an unbounded time horizon. It is still much better than the non-adaptive version of the policy which uses a large number of seeds even for unbounded time horizon. As T increases, in the policy found by SMAC, the number of seeds/interventions decreases and inter-intervention time increases. In fact, for $T = 1000$ the $p = 1$, the policy found by SMAC seeded 3 nodes per intervention and had a inter-intervention time equal to 12 (which is greater than D of the graph). For extremely small T, the policy found by SMAC had 100 nodes per intervention and a very short inter-intervention time of 3. We observe similar behaviour even for $p = 2$ and with the NetHEPT dataset as well. Note that as long as $T > D$, the non-adaptive version will require the same number of seeds it needs for the unbounded horizon case. This shows us the benefit of adaptivity even when the time horizon is severely constrained. These experiments show the effectiveness of SMAC in finding reasonably good policies for any time horizon for MINTSS.

5

Conclusion

A conclusion is simply the place where you got tired of thinking.

– Dan Chaon

In this thesis, we studied two important variants of the influence maximization problem - namely in the bandit and adaptive settings.

We studied the important, but under-researched problem of influence maximization *when no influence probabilities or diffusion cascades are available.* Adopting a combinatorial multi-armed bandit paradigm, we formulated two interesting technical problems: minimizing error in the learned influence probabilities and minimizing regret from reduced spread as a result of choosing suboptimal seed sets across rounds of a CMAB game. We proposed various algorithms from the bandits literature for these problems and proved bounds on their performance. We also evaluated their empirical performance comprehensively on three real datasets. It would be interesting to consider Thompson sampling based algorithms for these problems. It is important to extend the results and algorithms to continuous time diffusion models. How to learn on the fly, not just influence probabilities, but the graph structure as well, is another interesting question.

For the second part of the thesis, we relaxed the assumption that all seeds need to be selected in the beginning of the diffusion process by adopting an adaptive approach. We gave algorithms for finding guaranteed approximations to the optimal policy for both the MAXSPREAD and MINTSS problems. We quantified the gain of an adaptive policy over a non-adaptive strategy for both these problems. We studied these problems under an unbounded as well as a bounded time horizon. We performed experiments on three real world datasets to verify our theoretical findings. In the future, we plan to evaluate

the performance of the greedy adaptive policy under a bounded time horizon from both a theoretical and empirical perspective.

Investigating a combination of both problems i.e. considering adaptive strategies while learning influence probabilities in the CMAB framework is another interesting future research direction.

Bibliography

[1] R. Agrawal. Sample mean based index policies with o (log n) regret for the multi-armed bandit problem. *Advances in Applied Probability*, pages 1054–1078, 1995. → pages 9

[2] V. Anantharam, P. Varaiya, and J. Walrand. Asymptotically efficient allocation rules for the multiarmed bandit problem with multiple plays-part i: Iid rewards. *Automatic Control, IEEE Transactions on*, 32(11):968–976, 1987. → pages 3, 9

[3] A. Asadpour, H. Nazerzadeh, and A. Saberi. Maximizing stochastic monotone submodular functions. *arXiv preprint arXiv:0908.2788*, 2009. → pages 38

[4] J.-Y. Audibert, S. Bubeck, and G. Lugosi. Minimax policies for combinatorial prediction games. *arXiv preprint arXiv:1105.4871*, 2011. → pages 9

[5] I. C. Avramopoulos, J. Rexford, and R. E. Schapire. From optimization to regret minimization and back again. In *SysML*, 2008. → pages 3

[6] N. Barbieri, F. Bonchi, and G. Manco. Topic-aware social influence propagation models. *Knowledge and information systems*, 37(3):555–584, 2013. → pages 45

[7] S. Bubeck, R. Munos, and G. Stoltz. Pure exploration in finitely-armed and continuous-armed bandits. *Theoretical Computer Science*, 412(19):1832–1852, 2011. → pages 9

[8] F. Caro and J. Gallien. Dynamic assortment with demand learning for seasonal consumer goods. *Management Science*, 53(2):276–292, 2007. → pages 9

[9] S. Chen, T. Lin, I. King, M. R. Lyu, and W. Chen. Combinatorial pure exploration of multi-armed bandits. In *Advances in Neural Information Processing Systems*, pages 379–387, 2014. → pages 9

[10] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 199–208. ACM, 2009. → pages 2, 45

[11] W. Chen, C. Wang, and Y. Wang. 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*, pages 1029–1038. ACM, 2010. → pages 7, 45

[12] W. Chen, Y. Yuan, and L. Zhang. Scalable influence maximization in social networks under the linear reshold model. In *Proc. 2010 IEEE Int. Conf. on Data Mining*, pages 88–97, 2010. → pages 7

[13] W. Chen, L. V. Lakshmanan, and C. Castillo. Information and influence propagation in social networks. *Synthesis Lectures on Data Management*, 5(4): 1–177, 2013. → pages 2

[14] W. Chen, Y. Wang, and Y. Yuan. Combinatorial multi-armed bandit: General framework and applications. In *Proceedings of the 30th International Conference on Machine Learning*, pages 151–159, 2013. → pages 3, 9, 12, 13, 14, 25, 26, 27, 30

[15] W. Chen, Y. Wang, and Y. Yuan. Combinatorial multi-armed bandit and its extension to probabilistically triggered arms. *arXiv preprint arXiv:1407.8339*, 2014. → pages 3, 9, 12, 13

[16] Y. Chen and A. Krause. Near-optimal batch mode active learning and adaptive submodular optimization. In *Proceedings of The 30th International Conference on Machine Learning*, pages 160–168, 2013. → pages 10, 37, 38

[17] H. Daneshmand, M. Gomez-Rodriguez, L. Song, and B. Schoelkopf. Estimating diffusion network structures: Recovery conditions, sample complexity & soft-thresholding algorithm. *arXiv preprint arXiv:1405.2936*, 2014. → pages 3, 8

[18] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 57–66. ACM, 2001. → pages 1

[19] V. Gabillon, M. Ghavamzadeh, and A. Lazaric. Best arm identification: A unified approach to fixed budget and fixed confidence. In *Advances in Neural Information Processing Systems*, pages 3212–3220, 2012. → pages 9

[20] Y. Gai, B. Krishnamachari, and R. Jain. Learning multiuser channel allocations in cognitive radio networks: A combinatorial multi-armed bandit formulation. In *New Frontiers in Dynamic Spectrum, 2010 IEEE Symposium on*, pages 1–9. IEEE, 2010. → pages 9

[21] Y. Gai, B. Krishnamachari, and R. Jain. Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual

observations. *IEEE/ACM Transactions on Networking (TON)*, 20(5):1466–1478, 2012. → pages 9

[22] D. Golovin and A. Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42(1):427–486, 2011. → pages 10, 35, 41

[23] D. Golovin and A. Krause. Adaptive Submodular Optimization under Matroid Constraints. *Computing Research Repository*, abs/1101.4, 2011. → pages 10

[24] M. Gomez Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1019–1028. ACM, 2010. → pages 8

[25] A. Gopalan, S. Mannor, and Y. Mansour. Thompson sampling for complex online problems. In *Proceedings of The 31st International Conference on Machine Learning*, pages 100–108, 2014. → pages 9

[26] A. Goyal, F. Bonchi, and L. V. Lakshmanan. Learning influence probabilities in social networks. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 241–250. ACM, 2010. → pages 3, 8, 32

[27] A. Goyal, F. Bonchi, and L. V. Lakshmanan. A data-based approach to social influence maximization. *Proceedings of the VLDB Endowment*, 5(1):73–84, 2011. → pages 2, 4, 8

[28] A. Goyal, W. Lu, and L. V. Lakshmanan. Simpath: An efficient algorithm for influence maximization under the linear threshold model. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 211–220. IEEE, 2011. → pages 2

[29] A. Goyal, F. Bonchi, L. Lakshmanan, and S. Venkatasubramanian. On minimizing budget and time in influence propagation over social networks. *Social Network Analysis and Mining*, 3(2):179–192, 2013. ISSN 1869-5450. doi:10.1007/s13278-012-0062-z. URL http://dx.doi.org/10.1007/s13278-012-0062-z. → pages 4

[30] A. Guillory and J. Bilmes. Interactive submodular set cover. In *ICML*, pages 415–422, 2010. → pages 10

[31] H. W. Hethcote. The mathematics of infectious diseases. *SIAM review*, 42(4): 599–653, 2000. → pages 1

[32] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*, pages 507–523. Springer, 2011. → pages 43, 45

[33] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003. → pages 2, 6, 7, 28

[34] T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985. → pages 3, 8, 9

[35] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 420–429. ACM, 2007. → pages 2, 7

[36] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM, 2010. → pages 3

[37] S. Mannor and J. N. Tsitsiklis. The sample complexity of exploration in the multi-armed bandit problem. *The Journal of Machine Learning Research*, 5: 623–648, 2004. → pages 9

[38] A. L. Montgomery. Applying quantitative marketing techniques to the internet. *Interfaces*, 31(2):90–108, 2001. → pages 1

[39] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1): 265–294, 1978. → pages 2, 7, 39

[40] P. Netrapalli and S. Sanghavi. Learning the graph of epidemic cascades. In *ACM SIGMETRICS Performance Evaluation Review*, volume 40, pages 211–222. ACM, 2012. → pages 3, 8, 18, 20, 21, 22, 32

[41] M. E. Newman. The structure and function of complex networks. *SIAM review*, 45 (2):167–256, 2003. → pages 41

[42] S. Pandey and C. Olston. Handling advertisements of unknown quality in search advertising. In *Advances in neural information processing systems*, pages 1065–1072, 2006. → pages 3

[43] J. Rayport. The virus of marketing. *Fast Company*, 6(1996):68, 1996. → pages 1

[44] H. Robbins. Some aspects of the sequential design of experiments. In *Herbert Robbins Selected Papers*, pages 169–177. Springer, 1985. → pages 3

[45] M. G. Rodriguez, D. Balduzzi, and B. Schölkopf. Uncovering the temporal dynamics of diffusion networks. *arXiv preprint arXiv:1105.0697*, 2011. → pages 3, 8

[46] K. Saito, R. Nakano, and M. Kimura. Prediction of information diffusion probabilities for independent cascade model. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 67–75. Springer, 2008. → pages 3, 8, 18

[47] K. Saito, K. Ohara, Y. Yamagishi, M. Kimura, and H. Motoda. Learning diffusion probability based on node attributes in social networks. In *Foundations of Intelligent Systems*, pages 153–162. Springer, 2011. → pages 32

[48] J. J. Samper, P. A. Castillo, L. Araujo, J. Merelo, O. Cordon, and F. Tricas. Nectarss, an intelligent rss feed reader. *Journal of Network and Computer Applications*, 31(4):793–806, 2008. → pages 1

[49] X. Song, Y. Chi, K. Hino, and B. L. Tseng. Information flow modeling based on diffusion rate for prediction and ranking. In *Proceedings of the 16th international conference on World Wide Web*, pages 191–200. ACM, 2007. → pages 1

[50] Y. Tang, X. Xiao, and S. Yanchen. Influence maximization: Near-optimal time complexity meets practical efficiency. 2014. → pages 2, 7, 28, 42

[51] C. Wang, W. Chen, and Y. Wang. Scalable influence maximization for independent cascade model in large-scale social networks. *Data Mining and Knowledge Discovery*, 25(3):545–576, 2012. → pages 2, 45

[52] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 928–936, 2003. URL http://www.aaai.org/Library/ICML/2003/icml03-120.php. → pages 19, 20, 21