Continuous Influence Maximization: What Discounts Should We Offer to Social Network Users?

Yu Yang
Simon Fraser University
Burnaby, Canada
yya119@sfu.ca

Xiangbo Mao
Simon Fraser University
Burnaby, Canada
xiangbom@sfu.ca

Jian Pei
Simon Fraser University
Burnaby, Canada
jpei@cs.sfu.ca

Xiaofei He
Zhejiang University
Hangzhou, China
xiaofeihe@gmail.com

ABSTRACT

Imagine we are introducing a new product through a social network, where we know for each user in the network the purchase probability curve with respect to discount. Then, what discount should we offer to those social network users so that the adoption of the product is maximized in expectation under a predefined budget? Although influence maximization has been extensively explored, surprisingly, this appealing practical problem still cannot be answered by the existing influence maximization methods. In this paper, we tackle the problem systematically. We formulate the general continuous influence maximization problem, investigate the essential properties, and develop a general coordinate descent algorithm as well as the engineering techniques for practical implementation. Our investigation does not assume any specific influence model and thus is general and principled. At the same time, using the most popularly adopted independent influence model as a concrete example, we demonstrate that more efficient methods are feasible under specific influence models. Our extensive empirical study on four benchmark real world networks with synthesized purchase probability curves clearly illustrates that continuous influence maximization can improve influence spread significantly with very moderate extra running time comparing to the classical influence maximization methods.

1. INTRODUCTION

Influence maximization [8, 13] is a critical technique in many social network applications, such as viral marketing. The intuition is that, by targeting on only a small number of nodes (called seed nodes), it is possible to trigger a large cascade of information spreading in a social network. Technically, in a social network, influence maximization tries to identify a set of nodes such that if the selected nodes are committed to spread a piece of information to their neighbors, such as adopting a product, the expected spread in the social network is maximized. There have been abundant studies on various models and computational methods for influence maximization. We will review some representative milestone studies in Section 2.

All existing works on influence maximization, except for [10], assume a node is either a seed or not. Eftekhar *et al.* [10] generalized the assumption one step further by assuming that nodes are divided into groups and each group takes a predefined probability to be part of the seeds. However, many demands in practice are still not addressed.

Imagine a company is introducing a new product through a social network by providing discounts to users in the network in the hope of maximizing the influence spread. The total discount is constrained by a budget defined by the company. It is well known that different users in a social network may have a different capability in spreading influence. Consequently, the company naturally wants to offer different users different discounts. It is reasonable to assume that the more discount a user is offered, the more likely the user may adopt the product and spread the influence to her neighbors, which is also known in marketing research as the *purchase probability curve* being monotonic with respect to discount. At the same time, different users may have different purchase probability curves. Given a budget and the users' purchase probability curves, what discounts should the company offer to the users so that the expected influence spread is maximized? Apparently, this is an interesting question that is asked again and again in various applications where influence maximization is used. At the same time, unfortunately the existing influence maximization techniques cannot answer the question.

Motivated by the practical demands, in this paper, we investigate the questions about what discounts we should offer to social network users. In general, given a social network, a budget, and, for each user in the network, the seed probability function on discount (corresponding to the purchase probability curve with respect to discount in the above motivation example), the continuous influence maximization problem is to find the optimal configuration, which consists of a discount rate for each user, that maximizes the influence spread in expectation. We make several contributions.

First, to the best of our knowledge, we are the first to identify and systematically study the problem of continuous influence maximization, which has significant applications in practice. We show that the continuous influence maximization problem is a generalization of the existing influence maximization problem, which focuses on discrete configurations. Consequently, we investigate the hardness of the problem, and analyze several essential properties of the problem. We do not assume any specific influence model, and

thus all properties explored are general.

Second, we develop a general coordinate descent framework for the general continuous influence maximization problem. Again, this algorithm does not assume any specific influence model. We analyze the practical challenges in implementation, and suggest the corresponding engineering techniques.

Third, we demonstrate that more efficient methods are feasible for specific influence models. As a concrete example, we consider the independent cascade model [13], which is the most popularly used influence model in literature. We develop two simple yet effective algorithms based on the latest, state-of-the-art polling-based framework [1].

Last, we report an extensive empirical evaluation using four benchmark real social network data sets with synthesized purchase probability curves. The largest data set has almost 4 million nodes and 70 million edges. The experiment results clearly show that continuous influence maximization can significantly improve influence spread. At the same time, the extra running time remains moderate.

The rest of the paper is organized as follows. We review the related work in Section 2 and formulate the problem of continuous influence maximization in Section 3. In Section 4, we investigate the properties of the expectation of influence spread. We present the general coordinate descent framework in Section 5. We study in Section 6 the relationship between continuous influence maximization developed in this paper and the existing influence maximization problem. In Section 7, we examine the challenges of implementing the coordinate descent algorithm in practice, and provide the corresponding engineering techniques. In Section 8, we develop two algorithms under the independent cascade model. We report an extensive empirical evaluation in Section 9, and conclude the paper in Section 10.

For the interest of space, all mathematical proofs are given in Appendix.

2. RELATED WORK

Domingos *et al.* [8] proposed to take advantage of peer influence between users in social networks for marketing. The essential idea is that, by targeting on only a small number of users (called seed users), it is possible to trigger a large cascade of users purchasing a specific product through a social network. Consequently, the technical challenge is to find a small set of users who can trigger the largest cascade in the network. Kempe *et al.* [13] formulated the problem as a discrete optimization problem, which is well known as the influence maximization problem. Since then, influence maximization has drawn much attention from both academia and industry [4, 5, 6, 12, 9, 23, 22].

Most influence maximization algorithms are designed for triggering models [13]. Among these algorithms, a polling-based method [1] has the lowest worst-case time complexity, $O((k+l)(n+m)\log^2 n/\varepsilon^3)$. Tang *et al.* [23, 22] further improved the method to make it run in $O((k+l)(n+m)\log n/\varepsilon^2)$ expected time. The empirical studies showed that their improved algorithms are orders of magnitude faster than the other influence maximization algorithms. Lei *et al.* [14] proposed a method that learns the propagation probabilities while running the viral marketing campaigns. Another line of algorithmic viral marketing research is budgeted influence maximization [25, 19]. Under such problem settings, every user in a social network is assigned with a threshold value that indicates the amount of money a company needs to spend to persuade her/him to be an initial adopter. One key problem of this setting is how to obtain users' threshold values. Singer [20] proposed a mechanism that can elicit users' true threshold values if they are rational agents. Chen *et al.* [3] provided a comprehensive survey on influence maximization algorithms.

Farajtabar *et al.* [11] modeled social events using multivariate Hawkes processes, and developed a convex optimization framework to determine the incentives on users for stimulating a social event of a desired activity level. Although the objective function in [11] is flexible since it only requires that the objective is a concave utility function, both the properties explored in [11] and the algorithm proposed are only suitable for multivariate Hawkes processes rather than a general influence model. Descriptive influence models, such as the independent cascade model and the linear threshold model [13], the two most widely used models, cannot fit in the framework in [11].

Eftekhar *et al.* [10] discovered that sometimes instead of targeting on very few individual users, persuasion attempts on groups of users, for example, displaying advertisements to them, may lead to wider range cascades in social networks. The motivation of persuasion on groups is that by spending less money on a targeted individual a company can target on much more users and, as a result, in expectation such a strategy may bring more initial adopters [24]. Eftekhar *et al.* [10] assumed that the probability that a user is persuaded to be a seed user is given and fixed, if she/he is targeted. A more realistic strategy is that we can adjust the resource spent on a specific individual to manipulate the probability she/he becomes a seed user, which is the subject studied in this paper.

3. PROBLEM DEFINITION

A social network is a graph $G = \langle V, E\rangle$, where V is a set of users and E is a set of relationships between users. Denote by $n = |V|$ the number of users and $m = |E|$ the number of relationships, that is, edges.

An influence cascading model (influence model for short) describes the process of how influence is cascaded in a social network. Two most widely used influence models are the independent cascade model and the linear threshold model [13]. In an influence cascade process, a cascade is started by a small number of users, whom we call *seed users* (or *seeds* for short). We call the set of seed users the *seed set*, denoted by S. Every influence model has an *influence function* $I : 2^V \to R$, where $I(S)$ is the expected size of the cascade triggered by the seed set S and is also called the *influence spread* of S. Usually, $I(S)$ is assumed monotonic and submodular [13, 18], which capture the intuition about influence spreading.

In this paper, we are interested in customizing a discount for every user in a social network to maximize influence cascading. With a discount of 0%, a user has to pay the full price. With a discount of 100%, the product is free for the user. Please note that the notion of discount here can also be used to model in general the cost that we would like to pay to a user to turn the user into a seed.

Technically, a user $u \in V$ is associated with a *seed probability function* $p_u : [0,1] \to [0,1]$, which models the probabilistic distribution that u is attracted to become a seed user given a discount between 0% to 100%. Denote by c_u the discount we offer to u. Then, $p_u(c_u)$ is the probability that u becomes a seed user given such a discount. In this paper, we assume that a seed probability function $p_u(\cdot)$ has the following properties: (1) $p_u(0) = 0$; (2) $p_u(1) = 1$; (3) $p_u(c_u)$ is monotonic with respect to c_u; and (4) $p_u(c_u)$ is continuously differentiable. Conditions (1) and (2) are also assumed in the classical influence maximization problem.

The existing marketing research [2, 21] estimates user purchase probability. Most of the existing work focuses on the adoption rate of the whole population rather than each individual, and estimations are on specific goods. In reality, a user's purchasing behavior may change over time and on different types of goods [7]. Thus, the best way to decide a user's seed probability function (purchase proba-

bility curve) is to learn from data. Since seed probability functions can take many different forms, it is important to design a general marketing method that can handle all kinds of such functions.

We assume that different users become seed users independently. Denote by an n-dimensional vector $C = (c_1, c_2, ..., c_n)$ a *configuration* of discounts assigned to all users in G. It is clear that, unlike the situation in the influence maximization problem, the seed set S in our problem setting is probabilistic. Given a social network $G = \langle V, E \rangle$ and a configuration C, the probability that a subset $S \subseteq V$ of users is the seed set is

$$Pr(S;V,C) = \prod_{u \in S} p_u(c_u) \prod_{v \in V - S} (1 - p_v(c_v)) \quad (1)$$

For a specific influence model with an influence function $I(S)$, the *expected influence spread* is

$$UI(C) = \sum_{S \in 2^V} Pr(S;V,C) I(S) \quad (2)$$

Now we define the *continuous influence maximization* problem (*CIM* for short) studied in this paper as follows. Given a social network $G = \langle V, E \rangle$, a budget B, a seed probability function $p_u(c_u)$ for every user u, and an influence model with an influence function $I(S)$, find the configuration C that is the optimal solution to the following continuous optimization problem.

$$\begin{aligned} \text{maximize} \quad & UI(C) \\ \text{s.t.} \quad & 0 \le c_u \le 1, \forall u \in V \\ & \sum_{u \in V} c_u \le B \end{aligned} \quad (3)$$

We call a configuration satisfying the constraints in Eq. 3 a *feasible* configuration. Please note that the budget B here is a safe budget in general. When discounts here are used to model the costs committed to each user, it models the total cost. When discounts here are explained as discount rates, the budget is the worst-case budget. We leave the exploration of other forms of the budget constraint to future work, such as the expected budget under the discount rate explanation.

The classical influence maximization problem is a special case of the continuous influence maximization problem, since it can be written in a similar way as follows.

$$\begin{aligned} \text{maximize} \quad & UI(C) \\ \text{s.t.} \quad & c_u = 0 \text{ or } c_u = 1, \forall u \in V \\ & \sum_{u \in V} c_u \le B \end{aligned} \quad (4)$$

We call a configuration satisfying the constraints in Eq. 4 an *integer* configuration. Apparently, an integer configuration is also a feasible configuration.

In the rest of the paper, for the sake of clarity, we also call the classical influence maximization problem *discrete influence maximization* (*DIM* for short). Table 1 summarizes the frequently used notations.

Notation	Description
$G = \langle V, E \rangle$	Social network G, where V is the set of users and E is the set of relationships
$n = \|V\|$	The number of nodes in G
$m = \|E\|$	The number of edges in G
$p_u(c_u)$	The probability that u becomes a seed user if u is offered a discount c_u
$C = (c_1, c_2, ..., c_n)$	A configuration, where c_u $(0 \le c_u \le 1)$ is the discount offered to user u
$Pr(S;V,C)$	The probability of a subset of users $S \subseteq V$ is the seed set S (Eq. 1)
$UI(C)$	The expected influence spread caused by configuration C (Eq. 2)

Table 1: Frequently used notations.

4. EXPECTED INFLUENCE SPREAD

The CIM problem is to optimize the expected influence spread $UI(C)$. In this section, we discuss the computation of $UI(C)$ and the monotonicity of $UI(C)$, which prepare us for the solution development in the next sections.

4.1 Computing $UI(C)$

Given $G = \langle V, E \rangle$, an influence function $I(S)$, and a seed probability function $p_u(c_u)$ for every $u \in V$, how can we obtain $UI(C)$? It is known that, for many popular influence models, computing $I(S)$ is #P-hard [4, 6]. What is the hardness of computing $UI(C)$?

THEOREM 1 (COMPLEXITY). *Given a configuration C, computing $UI(C) = \sum_{S \in 2^V} Pr(S;V,C) I(S)$ is #P-hard if computing $I(S)$ is #P-hard.*

Since $UI(C)$ is the expectation of $I(S)$ over the random variable S, we can use Monte Carlo simulations to estimate $UI(C)$. Because every user becomes a seed user independently, randomly generating a seed set S according to $Pr(S;V,C)$ is equivalent to simply adding each user u to S independently with probability $p_u(c_u)$. We have the following result.

THEOREM 2 ((ε, δ) ESTIMATION). *Suppose we have an influence spread oracle that can return the influence spread $I(S)$ of a given seed set S. By calling the influence spread oracle $\frac{n^2 \ln \frac{2}{\delta}}{2\varepsilon^2 (\sum_{u \in V} p_c(c_u))^2}$ times, we can have an (ε, δ) estimation [17] of $UI(C)$.*

As mentioned before, computing $I(S)$ is #P-hard for some influence functions. The good news is that there exists a FPRAS[1] (Fully Polynomial Randomized Approximation Scheme) [17, 16] for estimating $I(S)$. We prove that if $I(S)$ can be estimated efficiently, so is $UI(C)$. Similar to influence maximization where the number of seeds B is assumed to be $\Omega(1)$, we assume that the expected number of seeds $\sum_{u \in V} p_u(c_u)$ is also $\Omega(1)$.

THEOREM 3 (FPRAS ESTIMATION). *For an influence function $I(\cdot)$, if there is a FPRAS for estimating $I(\cdot)$, there is a FPRAS for estimating $UI(\cdot)$.*

For the two most widely used influence models, namely the independent cascade model and the linear threshold model [13], we have the following upper bound on the time complexity of Monte Carlo simulations needed for obtaining an (ε, δ) approximation of $UI(C)$.

THEOREM 4 (APPROXIMATION RUNTIME). *Under the independent cascade model and the linear threshold model, we can use $O(\frac{mn^2 \ln \frac{1}{\delta}}{2\varepsilon^2 (\sum_{u \in V} p_u(c_u))^2})$ time to obtain an (ε, δ) approximation of $UI(C)$.*

In summary, the results in this subsection establish that, as long as $I(S)$ can be computed efficiently (that is, in polynomial time),

[1] An FPRAS is an algorithm which returns an (ε, δ) estimation of the desired value in time polynomial to n (size of input), $\frac{1}{\varepsilon}$ and $\ln \frac{1}{\delta}$.

$UI(C)$ can also be computed efficiently. This strong relation makes comparing two different configurations C_1 and C_2 computationally feasible, since we can efficiently estimate $UI(C_1)$ and $UI(C_2)$ accurately.

4.2 Monotonicity of $UI(C)$

Eq. 3 contains an inequality constraint $\sum_{u\in V} c_u \le B$. According to the assumption that $p_u(c_u)$ is monotonic with respect to c_u, we can prove that the inequality constraint $\sum_{u\in V} c_u \le B$ can be substituted by an equation constraint $\sum_{u\in V} c_u = B$.

LEMMA 1. *Given configurations $C_1 = (c_1^1, c_2^1, ..., c_n^1)$ and $C_2 = (c_1^2, c_2^2, ..., c_n^2)$, if there exists u $(1 \le u \le n)$ such that $c_u^1 \ge c_u^2$, and $\forall v \in V - \{u\}$, $c_v^1 = c_v^2$, then $UI(C_1) \ge UI(C_2)$.*

For two configurations $C_1 = (c_1^1, c_2^1, ..., c_n^1)$ and $C_2 = (c_1^2, c_2^2, ..., c_n^2)$, we write $C_1 \succeq C_2$ if $\forall u$, $c_u^1 \ge c_u^2$. By the transitivity of $\ge$ and Lemma 1, we have the following immediately.

THEOREM 5 (MONOTONICITY). *If $C_1 \succeq C_2$, then $UI(C_1) \ge UI(C_2)$.*

According to Theorem 5, it is obvious that the optimal C for the continuous influence maximization problem must use up the budget B. Thus, the continuous influence maximization problem can be rewritten as follows.

$$\begin{aligned} \text{maximize} \quad & UI(C) \\ \text{s.t.} \quad & 0 \le c_u \le 1, \forall u \in V \\ & \sum_{u\in V} c_u = B \end{aligned} \tag{5}$$

5. A GENERAL COORDINATE DESCENT FRAMEWORK

In this section, we develop a coordinate descent algorithm to solve the continuous influence maximization problem. Our algorithm is a general framework, since we do not compose any constraints on the specific form of the influence function $I(S)$ and the seed probability function $p_u(c_u)$. We only assume that $p_u(c_u)$ is monotonic and continuously differentiable.

5.1 Gradient

For a node $u \in V$, we can rewrite $UI(C)$ as follows.

$$\begin{aligned} & UI(C) \\ =& \sum_{S\in 2^V \wedge u\in S} Pr(S;V,C)I(S) + \sum_{S\in 2^V \wedge u\notin S} Pr(S;V,C)I(S) \\ =& \sum_{S\in 2^{V-\{u\}}} Pr(S;V-\{u\},C)I(S)[1-p_u(c_u)] \\ & + \sum_{S\in 2^{V-\{u\}}} Pr(S;V-\{u\},C)I(S\cup\{u\})p_u(c_u) \\ =& p_u(c_u)\cdot \sum_{S\in 2^{V-\{u\}}} Pr(S;V-\{u\},C)[I(S\cup\{u\}) - I(S)] + const \end{aligned}$$

where *const* is a constant with respect to c_u. Given a graph $G = \langle V, E\rangle$ and influence function $I(S)$, for a node $u \in V$, $Pr(S;V-\{u\},C)$, $I(S)$ and $I(S\cup\{u\})$ are constants with respect to c_u. Therefore, using a node $u \in V$ we can rewrite the objective function $UI(C)$ into a linear function of $p_u(c_u)$, where c_u is the only variable.

Assuming $p_u(c_u)$ is continuously differentiable, we can take the partial derivative of $UI(C)$ with respect to c_u, that is,

$$\frac{\partial UI(C)}{\partial c_u} = p_u'(c_u) \sum_{S\in 2^{V-\{u\}}} Pr(S;V-\{u\},C)[I(S\cup\{u\}) - I(S)] \tag{6}$$

In this way, we can compute the gradient of $UI(C)$ with respect to a specific configuration C. The gradient information will be used in the coordinate descent algorithm to be developed next.

5.2 A Coordinate Descent Algorithm

The coordinate descent algorithm is an iterative algorithm. In each iteration, we pick only two variables c_i and c_j, and fix the rest $n-2$ variables. We try to increase the value of the objective function $UI(C)$ by changing only the values of c_i and c_j.

As stated in Eq. 5, $\sum_{u\in V} c_u = B$. Thus, when we fix c_u for all $u \in V - \{i,j\}$, $\sum_{u\in V-\{i,j\}} c_u$ is a constant. Let $B' = B - \sum_{u\in V-\{i,j\}} c_u = c_i + c_j$. In other words, $c_j = B' - c_i$. Combining the other constraints $0 \le c_i \le 1$ and $0 \le c_j \le 1$ in Eq. 5, we have an equivalent constraint $max(0, B'-1) \le c_i \le min(1, B')$.

Thus, in each iteration, increasing $UI(C)$ can be achieved by solving the following optimization problem.

$$\begin{aligned} \text{maximize} \quad & UI(C) \text{ w.r.t. } c_i \text{ and } c_j = B' - c_i \\ \text{s.t.} \quad & max(0, B'-1) \le c_i \le min(1, B') \end{aligned} \tag{7}$$

To solve the above optimization problem, we further rewrite $UI(C)$ by fixing c_u for all $u \in V - \{i,j\}$ and setting $c_j = B' - c_i$. That is,

$$\begin{aligned} UI(C) = & \sum_{S\in 2^{V-\{i,j\}}} Pr(S;V-\{i,j\},C)\Big\{ \\ & [1-p_i(c_i)][1-p_j(B'-c_i)]I(S) \\ & + [1-p_i(c_i)]p_j(B'-c_i)I(S\cup\{j\}) \\ & + [1-p_j(B'-c_i)]p_i(c_i)I(S\cup\{i\}) \\ & + p_i(c_i)p_j(B'-c_i)I(S\cup\{i,j\})\Big\} \end{aligned} \tag{8}$$

In Eq. 8, except for $p_i(c_i)$ and $p_j(B'-c_i)$, all terms can be regarded as constants. Therefore, we have a new form of $UI(C)$ with respect to c_i and $c_j = B' - c_i$ as follows.

$$\begin{aligned} UI(C) = & (A_1 + A_2 - A_3 - A_4)p_i(c_i)p_j(B'-c_i) \\ & + (A_3 - A_1)p_i(c_i) + (A_4 - A_1)p_j(B'-c_i) + const, \end{aligned} \tag{9}$$

where

$$\begin{aligned} A_1 &= \sum_{S\in 2^{V-\{i,j\}}} Pr(S;V-\{i,j\},C)I(S) \\ A_2 &= \sum_{S\in 2^{V-\{i,j\}}} Pr(S;V-\{i,j\},C)I(S\cup\{i,j\}) \\ A_3 &= \sum_{S\in 2^{V-\{i,j\}}} Pr(S;V-\{i,j\},C)I(S\cup\{i\}) \\ A_4 &= \sum_{S\in 2^{V-\{i,j\}}} Pr(S;V-\{i,j\},C)I(S\cup\{j\}) \end{aligned} \tag{10}$$

In Eq. 9, $UI(C)$ only has one variable c_i. We take the derivative of $UI(C)$, that is,

$$\begin{aligned} \frac{\mathrm{d}UI(C)}{\mathrm{d}c_i} = & (A_1 + A_2 - A_3 - A_4)[p_i'(c_i)p_j(B'-c_i) \\ & - p_i(c_i)p_j'(B'-c_i)] + (A_3 - A_1)p_i'(c_i) \\ & - (A_4 - A_1)p_j'(B'-c_i) \end{aligned} \tag{11}$$

Since $p_i(c_i)$ and $p_j(B'-c_i)$ are both continuously differentiable on $(max(0,B'-1), min(B',1))$, the value $c_i \in [max(0,B'-1), min(B',1)]$ that maximizes the objective function in Eq. 7 must be in one of the following three situations: (1) $c_i = max(0,B'-1)$; (2) $c_i = min(B',1)$; or (3) $c_i = x$, where $x \in (max(0,B'-1), min(B',1))$ and $\frac{\mathrm{d}UI(C)}{\mathrm{d}c_i}|_{c_i=x} = 0$. Thus, we only need to check

Input: Budget B, social network G, seed probability function $p_u(c_u)$, $\forall u \in V$, and influence function (influence model) $I(S)$
Output: Configuration C

1: Initialize C such that C satisfies constraints in Eq. 5
2: **while** not converge **do**
3: Pick two variables c_i and c_j
4: $B' \leftarrow c_i + c_j$
5: Find all $x \in (max(0, B'-1), min(B', 1))$ that $\frac{\mathrm{d}UI(C)}{\mathrm{d}c_i}|_{c_i=x} = 0$
6: $c_i \leftarrow \mathrm{argmax}_{c_i \in \{x, max(0,B'-1), min(1,B')\}} UI(C)$
7: $c_j \leftarrow B' - c_i$
8: **end while**
9: **return** C

Algorithm 1: The Coordinate Descent Algorithm

these three types of points and set c_i to the one that results in the maximum value of $UI(C)$ with respect to c_i and $c_j = B' - c_i$.

Based on the above discussion, the pseudo-code of the coordinate descent algorithm for solving the continuous influence maximization problem is given in Algorithm 1.

We do not assume any specific seed probability function $p_u(c_u)$ and influence function $I(S)$. Thus, Algorithm 1 is a general framework for solving the continuous influence maximization problem.

In Line 3 of Algorithm 1, we do not specify which c_i and c_j should be picked. One heuristic that may help here is to use the partial derivative $\frac{\partial UI(C)}{\partial c_u}$ as a heuristic. For example, we can pick a variable with a large partial derivative and another variable that has a small partial derivative. Limited by space, we leave the exploration of effective heuristics to future work.

The convergence of Algorithm 1 is guaranteed by the following observations. First, $UI(C) \leq n$, where n is the number of nodes in the social network. Second, after each iteration in Algorithm 1, the updated configuration C ensures that the value of $UI(C)$ is at least as good as the previous one, that is, the value of $UI(C)$ is non-decreasing.

Algorithm 1 approaches a stationary configuration as the limit, which is a necessary condition for finding local optima. Since the objective function $UI(C)$ is not necessarily convex or concave, even when the stationary point is a local optima, it may not be the global optima. At the same time, because in each iteration the value of our objective cannot be decreased, when taking a configuration C and applying Algorithm 1, we can always find a configuration C' no worse than C.

6. CIM AND DIM

In this section, we examine the relation between the continuous influence maximization problem (CIM) studied in this paper and the classical and well studied (discrete) influence maximization problem (DIM).

Our first result is that, when the influence function $I(S)$ and the seed probability function $p_u(c_u)$ satisfy certain conditions, the continuous influence maximization problem and the discrete influence maximization problem share the same optimal solution.

THEOREM 6 (CIM AND DIM). *Given an influence function $I(S)$ that is monotonic and submodular, a budget B that is a positive integer, and a seed probability function $p_u(\cdot)$ for every node such that $\forall u \in V, \forall c_u \in [0,1]$, $p_u(c_u) \leq c_u$, the optimal objectives of CIM and DIM are equivalent.*

COROLLARY 1. *Given an influence function $I(S)$ that is monotonic and submodular, an integer budget B, if $\forall u \in V, \forall c_u \in [0,1]$, $p_u(c_u) \leq c_u$, then there exists an integer configuration C that is optimal to CIM.*

Theorem 6 also immediately indicates the complexity of the continuous influence maximization problem.

COROLLARY 2. *If maximizing $I(S)$ is NP-hard, and $I(S)$ is monotonic and submodular, then given a social network $G = \langle V, E \rangle$, a budget B and the seed probability function $p_u(c_u)$ for each $u \in V$, maximizing $UI(C)$ over C is also NP-hard.*

To further understand the significance of Theorem 6, we notice that the seed probability function $p_u(c_u)$ represents how user u is sensitive to discount c_u. If $p_u(c_u) \leq c_u$, user u is insensitive to discount. Theorem 6 indicates that, if all users in the network are insensitive to discount, then we would better give free products to some seed users, that is, setting $c_u = 1$, to trigger a sizeable cascade propagation.

Although under some conditions, CIM and DIM share the same optimal objectives, it can be shown that in some situations it is not the case, particularly when some users are sensitive to discount.

EXAMPLE 1. *Consider a social network $G = \langle V, E \rangle$ where $E = \emptyset$. In other words, G is a graph of n isolated nodes. In the independent influence model or the linear cascade model, if the budget $B = 1$, and for each node u, the seed probability function $p_u(c_u) = \sqrt{c_u}$, then the optimal solution for discrete influence maximization is to pick an arbitrary node u and the optimal influence spread is 1. However, the optimal solution to continuous influence maximization is to assign $\frac{1}{n}$ discount to each node, and the optimal influence spread is $\sqrt{n}$. Thus, not only the optimal solution to discrete influence maximization is not optimal to continuous influence maximization in some cases, but also such solutions to DIM can be arbitrarily bad for CIM as the size of the network becomes very large.*

It can be easily shown that CIM can always achieve influence spread no smaller than DIM. Given a budget B, one can first run a DIM algorithm to find a seed set of $\lfloor B \rfloor$ seeds. Then, by taking the corresponding integer configuration C of the seed set as the initial configuration, after applying the coordinate descent algorithm, a configuration C' that $UI(C') \geq UI(C)$ can be found.

7. PRACTICAL CIM SOLUTIONS

In this section we first discuss some challenges that the coordinate descent algorithm faces in practical implementation. Then, we introduce a practical strategy. Although the strategy cannot guarantee a global or local optimal solution to the continuous influence maximization problem, it can always find a solution no worse than a solution to the discrete influence maximization problem that a known method can produce.

7.1 Practical Challenges for the Coordinate Descent Algorithm

In the coordinate descent algorithm, we need to compute 3 coefficients, $(A_1 + A_2 - A_3 - A_4)$, $A_3 - A_1$, and $A_4 - A_1$. Similar to $UI(C)$, these three coefficients can only be estimated by sampling techniques if computing $I(S)$ is #P-hard. All the three coefficients can be very close to 0. Consequently, estimating them may be very challenging.

To tackle the challenge, a practical trick is not to solve $\frac{\mathrm{d}UI(C)}{\mathrm{d}c_i}|_{c_i=x} = 0$. Instead, we can estimate $UI(C)$ directly by trying

all possible values of $c_i \in [max(0,B'-1), min(B',1)]$. This makes good sense in practice since a budget typically carries a minimum unit. For example, if we want the absolute error of C to be up to 0.01, at most we only need to try 101 different values of c_i because $min(B',1) - max(0,B'-1) \leq 1$.

However, even the above trick may face serious challenges in some situations. A key observation is that the gain obtained in an iteration of the coordinate descent algorithm is limited.

THEOREM 7 (GAIN IN AN ITERATION). *In the coordinate descent algorithm, let the configuration before an iteration and that after the iteration be C and C_1, respectively. Then, after the iteration, we have*

$$UI(C_1) - UI(C) \leq \operatorname{argmax}_{u \in V} \sum_{S \in 2^{V-\{u\}}} Pr(S; V-\{u\}, C)\big(I(S \cup \{u\}) - I(S)\big)$$

Theorem 7 indicates that sometimes the gain after one iteration in the coordinate descent algorithm can be very small, because $I(S \cup \{u\}) - I(S)$ can be close to 0. While very often we can only calculate $UI(C)$ by Monte Carlo simulations, such sampling techniques may fail to detect a very small difference between two highly similar configurations.

7.2 Unified Discount Configuration

A practical engineering strategy to design discounts is to offer a unified discount to some users in a social network. That means, for each node u in G, c_u is either a predefined value c or 0. Thus, for finding a good configuration C, we can optimize over the unified discount c. Although c is continuous, because we have to use up all budget due to the monotonicity of $UI(C)$, to optimize over c, we only need to consider situations when $c = \frac{B}{\lceil B \rceil}, \frac{B}{\lceil B \rceil + 1}, \ldots, \frac{B}{n}$. Now, the problem becomes how we can find the optimal configuration when c is fixed.

When c is fixed, finding the optimal configuration C is to find the optimal set of users to offer each of them discount c. Suppose we choose the set S of users to offer discounts, denote by $Pr(S'; S, c)$ the probability of generating a seed set S' when the unified discount is c, that is,

$$Pr(S'; S, c) = \prod_{u \in S'} p_u(c) \prod_{v \in S - S'} (1 - p_v(c)) \tag{12}$$

We define $UI(S; c)$ as the expected influence spread when we offer each user in S with a unified discount c. That is,

$$UI(S; c) = \sum_{S' \in 2^S} Pr(S'; S, c) I(S') \tag{13}$$

We observe the following nice property of $UI(S; c)$ when c is fixed.

THEOREM 8 (MONOTONICITY AND SUBMODULARITY). *If $I(S)$ is monotonic and submodular with respect to S, then $UI(S; c)$ is also monotonic and submodular with respect to S.*

The monotonicity and submodularity of $UI(S; c)$ with respect to S imply that, when c is fixed, we can apply a greedy algorithm to find a set of users S to offer discounts which can cause influence spread at least $(1 - \frac{1}{e})$ times of the influence spread caused by the optimal set of users S^*. In such a case, when the influence model and seed probability function for each user are given, some efficient influence maximization algorithms [15, 23, 22] can be applied here.

8. CIM ALGORITHMS UNDER THE INDEPENDENT CASCADE MODEL

In this paper, we demonstrate how we can develop specific algorithms for continuous influence maximization under some specific influence model. Since the independent cascade model is the most popularly used influence model [5, 4, 13, 23], we design two simple yet effective algorithms to solve the continuous influence maximization problem under the independent cascade model. Note that our main contribution in this paper is the general framework, the discussion in this subsection is for providing an example that how a specific influence model is plugged into our framework.

Recently, a polling-based algorithmic framework [1, 23, 22] was proposed for triggering models [13] like the IC model and was shown the most efficient influence maximization algorithm so far. Our algorithms are also polling-based algorithms.

Let us first briefly review how a polling method works for influence maximization.

Given a graph $G = \langle V, E \rangle$, a poll is conducted as follows: a node $v \in V$ is picked in random and then we try to find out which nodes are likely to influence v. We run a Monte Carlo simulation of IC propagation from v on the transpose graph [2] G^T. Note that the propagation probability of an edge (v,u) in G^T is pp_{uv}, the propagation probability of edge (u,v) in G. Such "reverse" propagation process from v is used for finding v's potential "influencers". Suppose the set of nodes reached in this poll is h. We call h a random hyper-edge. The intuition of the poll process is that if a node u has high influence, then the probability that u appears in a random hyper-edge h is high.

Then the polling method consists of two major steps.

1. Random Hyper-graph Construction. Generate a random hyper-graph H by generating a certain number of random hyper-edges. Note that nodes in H are still nodes in G. A node u and a hyper-edge $h \in H$ are incident if $u \in h$. Denote by $deg_H(S)$ the degree of the set of nodes S, which is the number of hyper-edges in H incident to at least one node of S. The greater $deg_H(S)$ is, the more likely the influence spread of S is high.

2. Maximum Coverage on Radom Hyper-graph. Greedily choose B nodes to add to the seed set S such that the selected node can increase $deg_H(S)$ the most.

One key property that makes the polling method work well is that when the number of random hyper-edges mH is fixed, $\frac{deg_H(S) * n}{mH}$ is an unbiased estimation of $I(S)$, the influence spread of S in G [1]. Thus, as long as mH is sufficiently large, the seed set returned by the polling method has good quality guarantees. Tang *et al.* [23, 22] illustrated how to estimate the lower bound of mH that makes the result of the polling method a $(1 - 1/e - \varepsilon)$-approximation with probability at least $1 - \frac{1}{n}$ given ε.

To solve the continuous influence maximization problem, we can employ a similar method that also constructs a random hyper-graph.

THEOREM 9. *Given a graph $G = \langle V, E \rangle$ and seed probability functions of all nodes, a random hyper-graph H with mH hyper-edges generated according to G and a configuration C, $\frac{n * \sum_{h \in H} [1 - \prod_{u \in h} (1 - p_u(c_u))]}{mH}$ is an unbiased estimation of $UI(C)$.*

According to Theorem 9, we build a random hyper-graph H by simply setting mH to a predefined number, usually in $O(n \log n)$.

[2] $G^T = \langle V, E^T \rangle$ is the transpose graph of $G = \langle V, E \rangle$ if $\forall (u,v) \in E$, $(v,u) \in E^T$

Two continuous influence maximization algorithms are devised based on H.

The first algorithm is called *Unified Discount* (UD). In Section 7.2, we discussed that in reality, instead of offering different users different discounts, a company can select a set of users and offer each of them the same discount c. We prove that, under such a situation, $UI(S;c)$ is monotonic and submodular if $I(S)$ is monotonic and submodular, which is true under the IC model. One can also easily prove that $\sum_{h\in H}[1-\prod_{u\in h}(1-p_u(c_u))]$ is also monotonic and submodular when $c_u = c$ or 0. Thus, when c is fixed, a simple greedy algorithm with the CELF pruning technique [15] can be applied to find a set S that is at least $(1-1/e)$ optimal with respect to maximizing $\sum_{h\in H}[1-\prod_{u\in h}(1-p_u(c_u))]$. Obviously the configuration C returned satisfies that $c_u = c$ if $u \in S$, otherwise $c_u = 0$. For finding the best discount c, we adopt a simple heuristic that is based on the fact that normally discount offered by companies is a multiple of 5%. Therefore, we simply run an exhaustive search of c by setting $c = 5\%, 10\%, ..., 95\%, 100\%$. One can choose some other step interval as needed.

The second algorithm, called Coordinate Descent (CD), takes the configuration returned by the Unified Discount algorithm as the initial values and a coordinate descent procedure is conducted to further improve $UI(C)$. Like Unified Discount, this algorithm is also run on the random hyper-graph H. Thus, the formulation that the optimization Coordinate Descent tries to solve is

$$\begin{aligned} \text{maximize} \quad & \sum_{h\in H}[1-\prod_{u\in h}(1-p_u(c_u))] \\ \text{s.t.} \quad & 0 \le c_u \le 1, \forall u \in V \\ & \sum_{u\in V} c_u \le B \end{aligned} \tag{14}$$

In our implementation, in each iteration, instead of picking a pair of coordinates c_i and c_j from $O(n^2)$ potential pairs in Algorithm 1, we only pick c_i and c_j from coordinates that have non-zero values in the initial configuration C. Note that the initial configuration C has at most $\frac{B}{5\%} = O(B)$ non-zero entries, and $B \ll n$. Efficiency is the major reason to do this. In the experiments, we run iterations in total 100 rounds. In each round, every pair of non-zero c_i and c_j are picked and optimized over.

EXAMPLE 2. *Fig. 1 is a toy example illustrating the differences between integer configuration, unified discount configuration, and continuous configuration. In Figure 1, the propagation model is the IC model and the propagation probabilities along edges are all* 0.1. *Suppose the seed probability functions for the nodes in this graph are all $p_u(c_u) = 2c_u - c_u^2$, that is, the users are sensitive to discount. When $B = 1$, the optimal seeding strategy for DIM is to choose node $v1$ as the single seed, which leads to the best integer configuration is $C_1 = (1,0,0,0,0)$ and $UI(C_1) = 1.4$. If we apply the unified discount strategy, the best unified discount value is* 0.2. *Correspondingly the best unified discount configuration is $C_2 = (0.2,0.2,0.2,0.2,0.2)$ and $UI(C_2) = 1.89216$. If we apply the coordinate descent algorithm and set $C_2 = (0.2,0.2,0.2,0.2,0.2)$ as the initial value of configuration, we get a better continuous configuration $C_3 = (0.38312,0.15422,0.15422,0.15422,0.15422)$ and $UI(C_3) = 1.93533$.*

9. EMPIRICAL EVALUATION

To examine the effectiveness and efficiency of our methods, in this section, we report experiments on four real networks with synthesized seed probability functions to test our proposed methods.

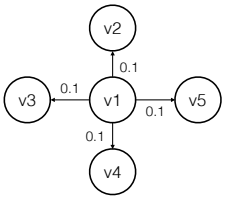


Figure 1: An example illustrating the differences among integer configuration, unified discount configuration and continuous configuration.

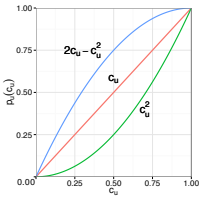


Figure 2: The seed probability functions used in the experiments.

The experiment results show that the continuous influence maximization strategy can significantly improve influence spread without incurring dramatic extra overheads compared to discrete influence maximization.

Network	n	m	Average Degree	mH
wiki-Vote	7,115	103,689	14.6	0.25M
ca-AstroPh	18,772	396,220	21.1	1M
com-dblp	317,080	2,099,732	6.6	20M
com-LiveJornal	3,997,962	69,362,378	17.3	40M

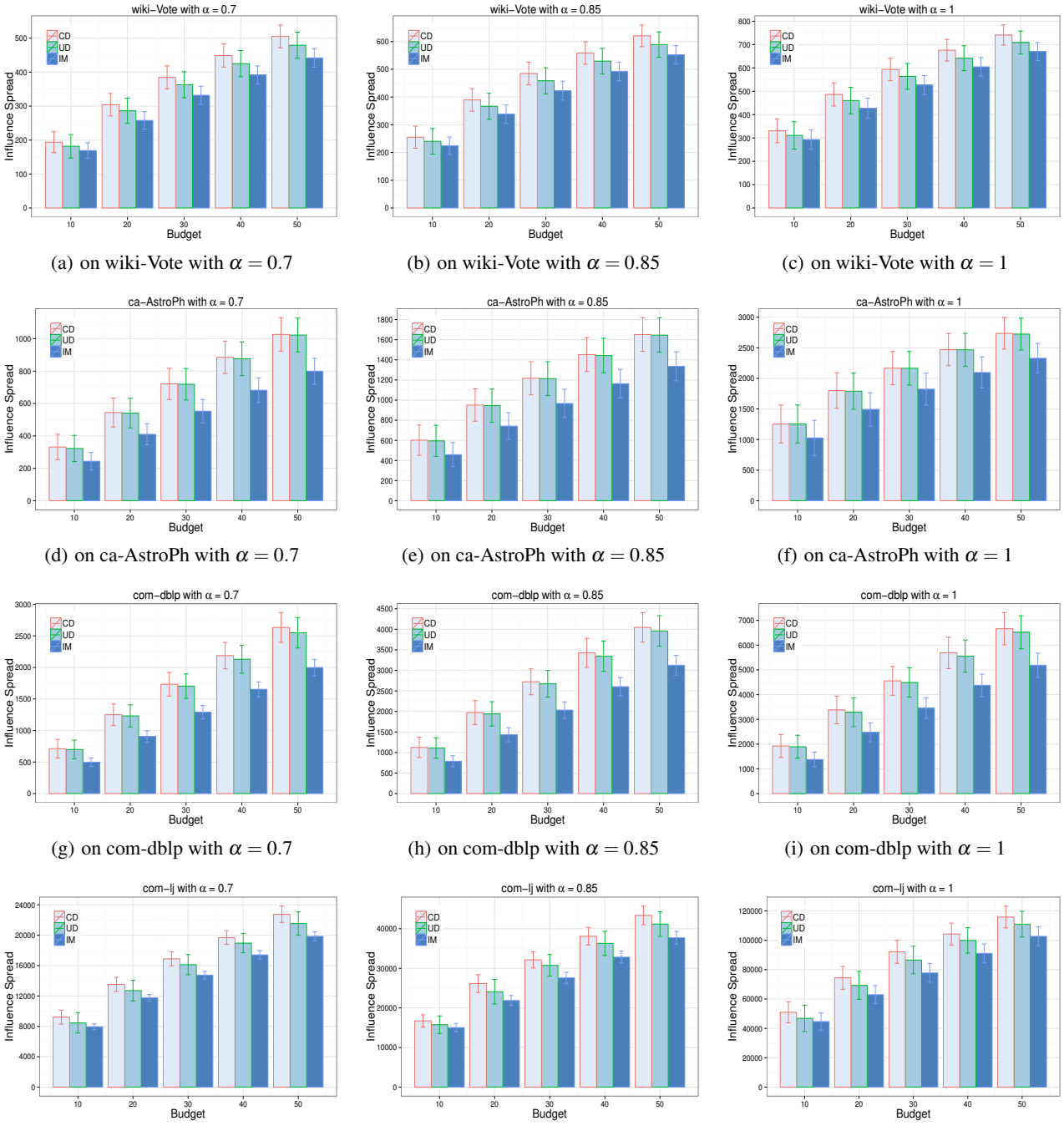
Table 2: Datasets

9.1 Expertmental Settings

We ran our experiments on four real network data sets that are publicly available in SNAP (`http://snap.stanford.edu/data/index.html`). Table 2 shows the details of the four data sets. All networks are treated as directed graphs, which means if a network is undirected, every undirected edge (u,v) is processed as two directed edges (u,v) and (v,u).

In our experiments, we adopted the Independent Cascade (IC) model as the influence model, which is the most widely used in literature [5, 4, 13, 23]. Following the most popular settings of the IC model [5, 4, 13, 23], we set the propagation probability of a directed edge (u,v) to $\frac{\alpha}{in-degree(v)}$, where $\alpha \in \{0.7, 0.85, 1\}$.

For seed probability functions, unfortunately we do not have access to any such real data sets for the purpose of experiments. Thus, we used synthesized seed probability functions. Given a network $G = \langle V, E\rangle$, we randomly picked 5% nodes to assign $p_u(c_u) = c_u^2$, and 10% nodes to assign $p_u(c_u) = c_u$ as their seed probability functions. These users (nodes) were insensitive to discount. The rest 85% nodes were assigned with $p_u(c_u) = 2c_u - c_u^2$, which means that those users are sensitive to discount. When c_u is close to 0, the probability that u becomes a seed is roughly twice of c_u. As c_u increases, the ratio $\frac{2c_u - c_u^2}{c_u}$ decreases gradually. Figure 2 shows the curves of the two seed probability functions as well as the function $p_u(c_u) = c_u$ as the reference.



(a) on wiki-Vote with $\alpha = 0.7$ (b) on wiki-Vote with $\alpha = 0.85$ (c) on wiki-Vote with $\alpha = 1$

(d) on ca-AstroPh with $\alpha = 0.7$ (e) on ca-AstroPh with $\alpha = 0.85$ (f) on ca-AstroPh with $\alpha = 1$

(g) on com-dblp with $\alpha = 0.7$ (h) on com-dblp with $\alpha = 0.85$ (i) on com-dblp with $\alpha = 1$

(j) on com-LiveJournal with $\alpha = 0.7$ (k) on com-LiveJournal with $\alpha = 0.85$ (l) on com-LiveJournal with $\alpha = 1$

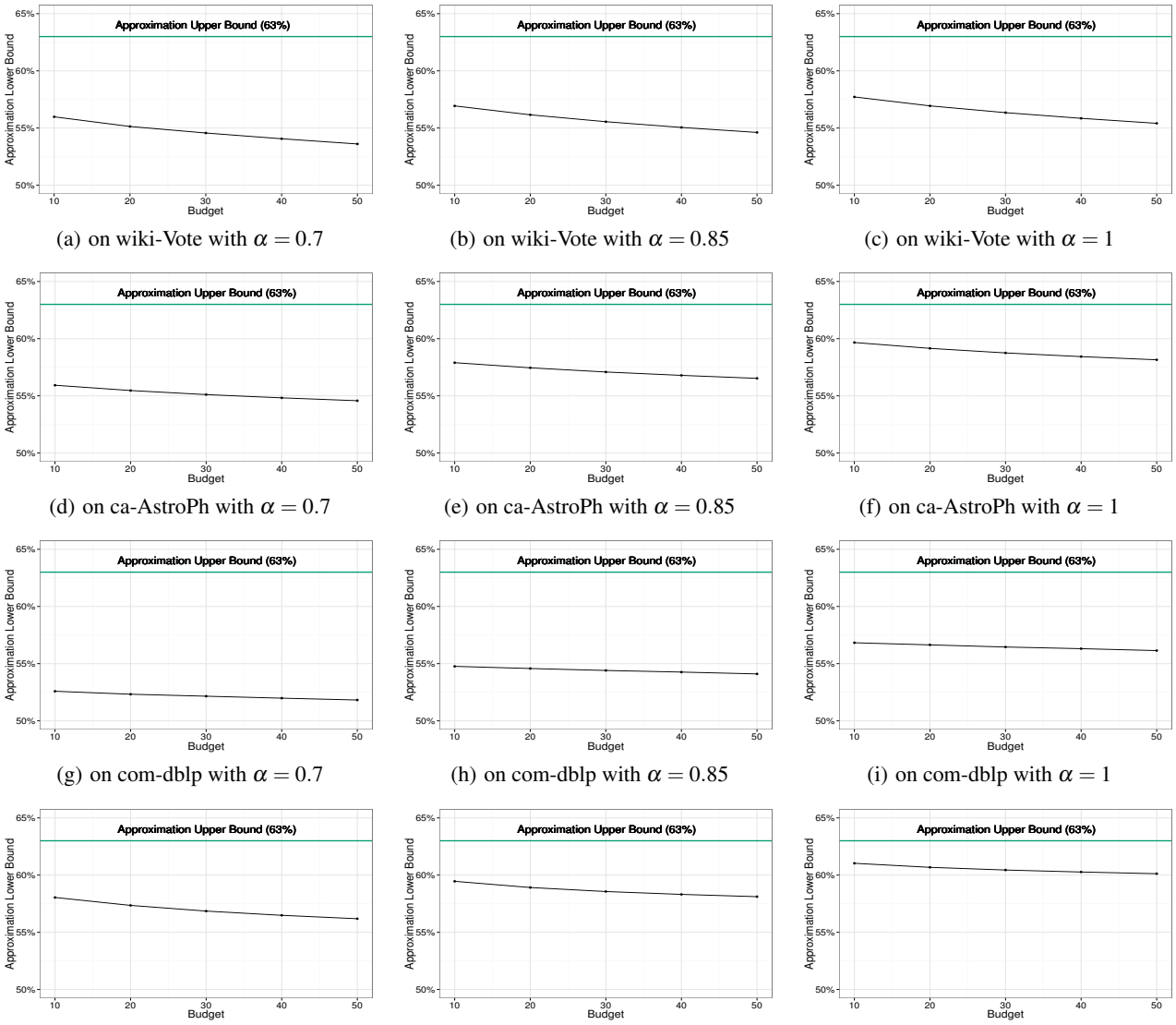
Figure 3: Influence spread

To compare the continuous influence maximization strategy and the existing discrete one, we compare the results of our algorithms to the result of discrete influence maximization. Since the IC model was used in our experiments, all algorithms implemented are based on the IC model. Therefore, in our experiments, all algorithms were implemented based on the polling framework. All algorithms were implemented in C# and ran on an Apple Mac Pro (Late 2013) computer with Intel Xeon 3.70GHz CPU, 64GB main memory. In the algorithms developed in Section 8, we need to set a parameter mH. The value for each dataset is also listed in Table 2.

9.2 Experimental Results

9.2.1 *Effectiveness*

Fig. 3 shows the influence spread of each algorithm under different settings of parameters. Since our methods introduce extra uncertainty in the seed set, we also report one standard deviation interval of influence spread in Fig. 3. All influence spreads and



(a) on wiki-Vote with $\alpha = 0.7$ (b) on wiki-Vote with $\alpha = 0.85$ (c) on wiki-Vote with $\alpha = 1$

(d) on ca-AstroPh with $\alpha = 0.7$ (e) on ca-AstroPh with $\alpha = 0.85$ (f) on ca-AstroPh with $\alpha = 1$

(g) on com-dblp with $\alpha = 0.7$ (h) on com-dblp with $\alpha = 0.85$ (i) on com-dblp with $\alpha = 1$

(j) on com-LiveJournal with $\alpha = 0.7$ (k) on com-LiveJournal with $\alpha = 0.85$ (l) on com-LiveJournal with $\alpha = 1$

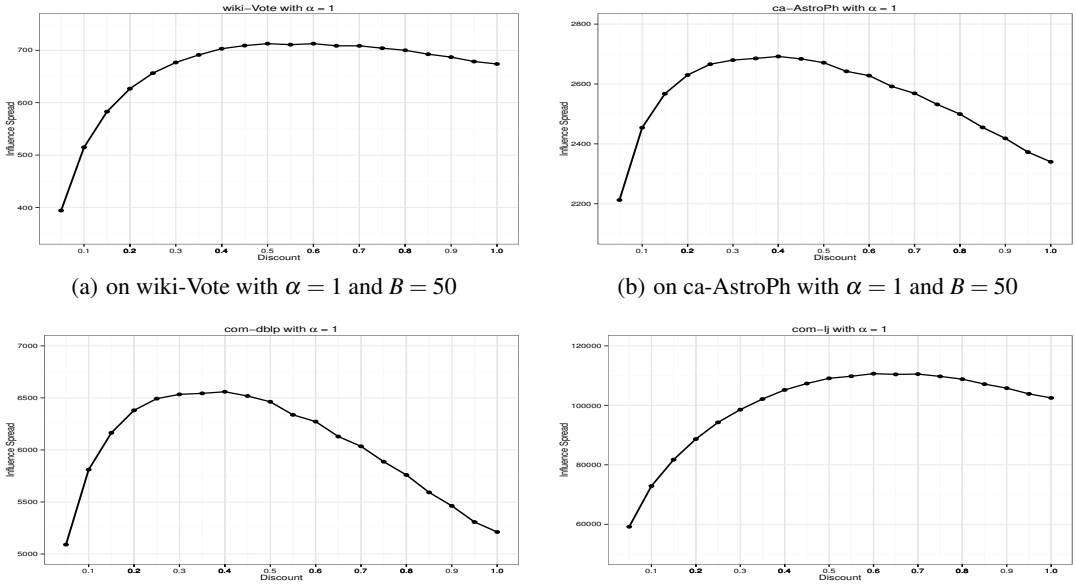
Figure 4: Approximation Lower Bound

standard deviations are obtained by 20,000 Monte Carlo simulations. From the results we find that both the Unified Discount (UD) algorithm and the Coordinate Descent (CD) algorithm can significantly increase the expected influence spread compared to discrete influence maximization (IM). Although the standard deviations of UD and CD algorithms are larger than that of IM, the gap is not large. In most cases the ratio of the first two to the last one is no more than 1.5. The CD algorithm can always achieve a higher influence spread and a lower standard deviation than the UD algorithm. Worth noting that 100 rounds of iterations for the CD algorithm are enough on datasets used in experiments. The CD algorithm converges within 100 rounds in all cases in our experiments.

Tang *et al.* [22] pointed out that, for the IM algorithm, to achieve a $(1-\frac{1}{e}-\varepsilon)$-approximation of the optimal solution with at least $1-\frac{1}{n}$ probability, the number of hyper edges mH should be set to at least $\frac{2n(1-\frac{1}{e})(\log\binom{n}{k}+\log n+\log 2)}{OPT\varepsilon^2}$. Therefore, when mH is fixed and the influence spread of the solution of the IM algorithm is known, we can figure out ε accordingly, and know how good the solution is. Thus, we also report the approximation lower bound of the influence maximization algorithm. Here we use the influence spread of the seed set returned by the IM algorithm as a lower bound of OPT. Fig. 4 shows that the approximation lower bound of the IM algorithm under our experimental settings is greater than 55% usually. Note that $(1-\frac{1}{e}) \approx 63\%$ is the approximation upper bound of any polynomial time algorithm if $P \neq NP$. This means our implementation of the baseline algorithm (IM) in the experiments is fairly good.

Fig. 5 shows how the influence spread varies with respect to the unified discount c in the Unified Discount algorithm. Limited by space, we only report the influence values when $\alpha = 1$ and $B = 50$ on each dataset. From Fig. 5 we can see that finding a best unified discount is necessary because different values of c can result in very different influence spreads.

We also tested the effectiveness of the search step 5% we chose in the Unified Discount algorithm. Table 3 shows the difference of



(a) on wiki-Vote with $\alpha = 1$ and $B = 50$

(b) on ca-AstroPh with $\alpha = 1$ and $B = 50$

(c) on com-dblp with $\alpha = 1$ and $B = 50$

(d) on com-LiveJournal with $\alpha = 1$ and $B = 50$

Figure 5: Influence Spread w.r.t. unified discount c

Dataset	B	1% Search Step	5% Search Step	Reduction Percentage
Wiki-Vote	10	311	311	0%
	20	463	460	0.7%
	30	562	562	0%
	40	650	642	1.2%
	50	719	710	1.3%
Ca-Astro	10	1241	1241	0%
	20	1760	1760	0%
	30	2120	2120	0%
	40	2422	2422	0%
	50	2680	2680	0%
Com-Dblp	10	1896	1896	0%
	20	3290	3290	0%
	30	4489	4489	0%
	40	5578	5563	0.3%
	50	6557	6530	0.4%
Com-LiveJournal	10	46906	46845	0.1%
	20	69185	69126	0.1%
	30	86469	86469	0%
	40	100027	99948	0.1%
	50	111086	110967	0.1%

Table 3: Effect of the parameter search step in calculating the best unified discount c.

using 1% and 5% as the search step when finding the best unified discount c. Here α was set to 1. The column "Reduction Percentage" means how much influence the best spread is decreased if we change the search step from 1% to 5%. From Table 3 we find that the reduction is tiny. In other words, the Unified Discount algorithm is insensitive to this parameter.

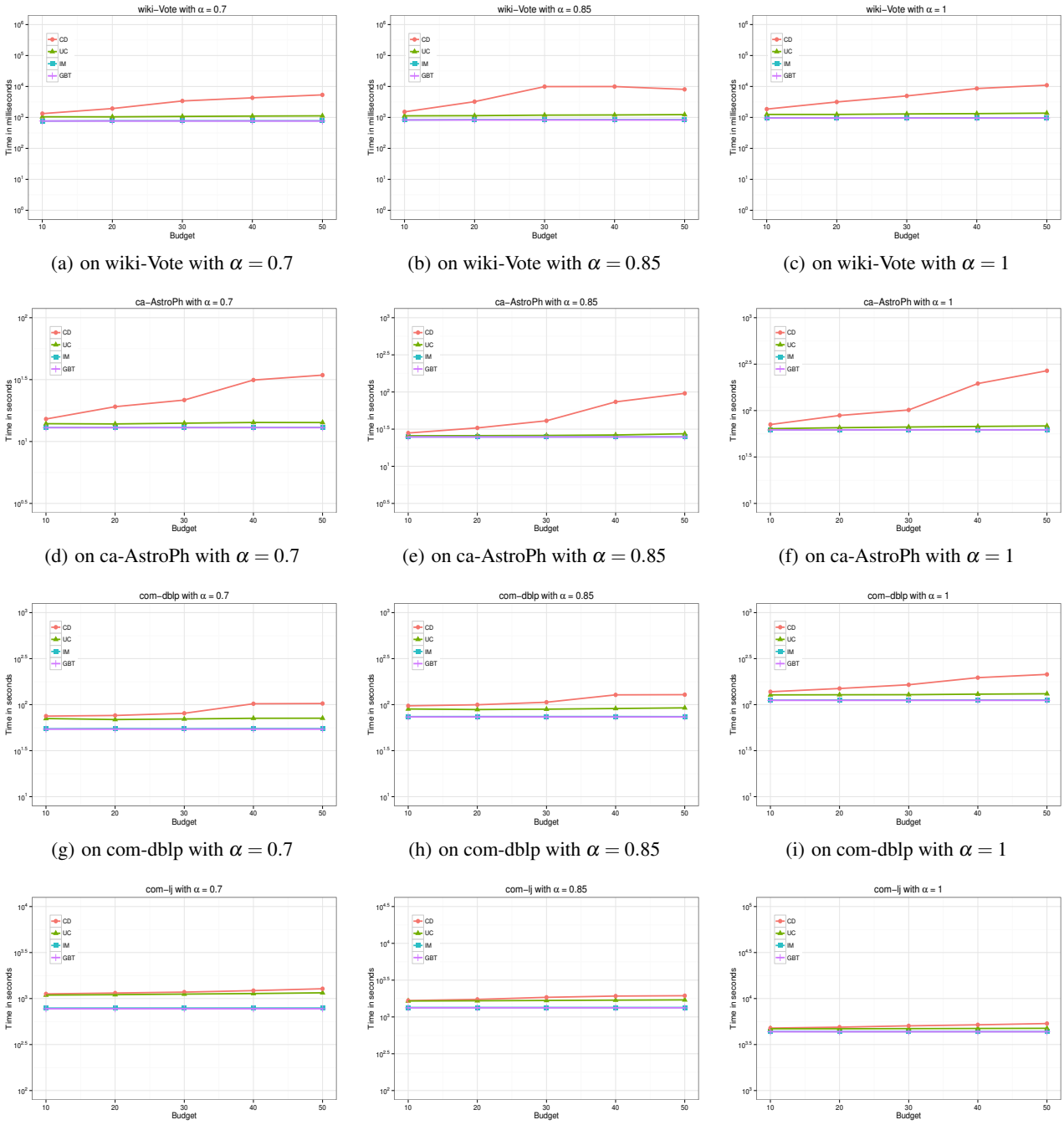
9.2.2 Sensitivity to User Purchase Probability Curves

We also tested the sensitivity of our methods with respect to user purchase probability curves. We still used the three purchase probability curves mentioned in Section 9.1, since they are representative and stand for users being sensitive to discounts, benchmark sensitivity to discounts and user being insensitive to discounts, respectively. To test the effect of purchase probability curves on the results, we reduced the portion of sensitive users and correspondingly increased the number of insensitive users. Specifically, for each social network, we randomly generated another two instances of it where different populations were assigned to those purchase probability curves. In the first instance, the portions of the three purchase probability curves were changed to 75%, 15% and 10%, respectively. In the second instance, 65%, 20% and 15%, respectively. We ran our methods on those new instances by setting $\alpha = 1$. Then we compared the new results with the original result where there are 85% sensitive users.

Table 4 shows the changes. Please note that different assignments of purchase probability curves to users may lead to different results. Moreover, it is possible that influence spread is higher when the portion of sensitive users is smaller. This is because the purchase probability curves were randomly assigned to users. Thus, even when the portion of sensitive users is smaller, it is possible that more influential users are assigned with sensitive purchase probability curves. We ran a good number of random assignments on the four data sets. The results are consistent in trend. From Table 4 we can see that the influence spread only decreases slightly after we reduced the number of sensitive users.

9.2.3 Scalability

We also tested the scalability of the algorithm and report the results in Fig. 6. GBT is the time of building the hypergraph. According to Fig 6, as the scale of network increases, the gap on the running time between our algorithm and the of the IM algorithm decreases. On the smallest dataset wiki-Vote, the running time of the Coordinate Descent algorithm is about 10 times of that of IM. The gap is reduced to no more than 4 times on the ca-AstroPh dataset and less than twice on the com-dblp dataset. On the largest dataset com-LiveJournal, the running time of the CD algorithm is only 1.5 times of that of IM. The reason is that the computational cost of building the hypergraph is high, while the configuration computation phase is relatively efficient. In addition, we only ran at most 100 rounds of iterations in CD in all experiments, and in each round we only need to optimize over $O(k^2)$ pairs of c_i and c_j, if the initial value of CD has k non-zero entries. Thus, in smaller networks, the



(a) on wiki-Vote with $\alpha = 0.7$ (b) on wiki-Vote with $\alpha = 0.85$ (c) on wiki-Vote with $\alpha = 1$

(d) on ca-AstroPh with $\alpha = 0.7$ (e) on ca-AstroPh with $\alpha = 0.85$ (f) on ca-AstroPh with $\alpha = 1$

(g) on com-dblp with $\alpha = 0.7$ (h) on com-dblp with $\alpha = 0.85$ (i) on com-dblp with $\alpha = 1$

(j) on com-LiveJournal with $\alpha = 0.7$ (k) on com-LiveJournal with $\alpha = 0.85$ (l) on com-LiveJournal with $\alpha = 1$

Figure 6: Running time

portion of iterations in total running time is higher because the cost of CD iterations is relatively stable while the cost of building hyper graph is proportional to the scale of the network.

10. CONCLUSIONS

In this paper, we proposed to offer users in social networks discounts rather than free products to trigger social cascades. We model the continuous influence maximization problem. Some key properties of the continuous influence maximization problem were studied and a coordinate descent framework was devised. Based on this framework, we proved that under certain conditions the continuous influence maximization problem and the original influence maximization problem share the same optimal solutions. We then pointed out the key challenges in practically implementing the coordinate descent framework and proposed feasible engineering techniques that work in practice. The experiment results demonstrated that our methods can improve influence spread significantly, while the extra running time over the classical discrete influence

Dataset	B	Unified Discount					Coordinate Descent				
		85% Sensitive	75% Sensitive	Reduction Percentage	65% Sensitive	Reduction Percentage	85% Sensitive	75% Sensitive	Reduction Percentage	65% Sensitive	Reduction Percentage
Wiki-Vote	10	311	319	-2.6%	319	-2.6%	332	332	0%	337	-1.5%
	20	460	463	-0.7%	455	1.1%	487	480	1.4%	473	2.9%
	30	562	564	-0.4%	551	2.0%	593	586	1.2%	575	3.0%
	40	642	648	-1.0%	631	1.7%	675	671	0.6%	656	2.8%
	50	710	715	-0.7%	696	2.0%	745	742	0.4%	725	2.7%
Ca-Astro	10	1241	1251	-0.8%	1173	5.5%	1252	1269	-1.4%	1184	5.4%
	20	1760	1754	0.3%	1663	5.5%	1785	1785	0%	1698	4.9%
	30	2120	2111	0.4%	2027	4.4%	2159	2147	0.6%	2065	4.4%
	40	2422	2405	0.7%	2328	3.9%	2461	2445	0.7%	2370	3.7%
	50	2680	2656	0.9%	2577	3.8%	2720	2702	0.9%	2626	3.5%
Com-Dblp	10	1896	1861	1.9%	1827	3.8%	1925	1917	0.4%	1882	2.2%
	20	3290	3210	2.4%	3158	4.0%	3338	3271	2.0%	3220	3.5%
	30	4489	4356	3.0%	4284	4.6%	4650	4449	4.3%	4379	5.8%
	40	5563	5409	2.8%	5331	4.2%	5694	5564	2.3%	5471	4.0%
	50	6530	6371	2.4%	6273	4.0%	6675	6522	2.3%	6429	3.7%
Com-LiveJournal	10	46845	46973	-0.3%	46722	0.3%	51500	48423	6.0%	49630	3.6%
	20	69126	68605	0.8%	67763	2.0%	74309	72675	2.2%	71909	3.2%
	30	86469	85536	1.1%	83237	3.7%	92201	90524	1.8%	85653	7.1%
	40	99948	98766	1.2%	95897	4.1%	104267	102080	2.1%	100541	3.6%
	50	110967	109660	1.2%	106268	4.2%	115870	113364	2.2%	111568	3.7%

Table 4: Sensitivity to user purchase probability curves.

maximization algorithm remains moderate.

We believe that this work opens a new direction for future work. For example, it is imperative to investigate whether our objective function $UI(C)$ is convex or concave, and how the form of $p_u(c_u)$ may affect the convexity/concavity of $UI(C)$? Moreover, the coordinate descent algorithm is only guaranteed to converge to a stationary point. Under what conditions can it converge to a local optima? Efficient continuous influence maximization algorithms for specific influence models are essential for real life applications. For example, in the experiments, we designed two simple algorithms Unified Discount (UD) and Coordinate Descent (CD) that work well in practice. In UD, we conduct a brute force search for finding the optimal discount c. Is there a better algorithm for searching c? For both UD and CD, how does the number of hyper-edges mH affect the quality of the solutions?

Another interesting direction to investigate is minimizing the budget of our continuous seeding strategy for covering a given portion of users in a social network. While minimizing budget under integer seeding strategy can be easily obtained by slightly modifying the greedy algorithm for influence maximization, it is far from trivial to design a new algorithm for our continuous seeding strategy.

11. REFERENCES

[1] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 946–957. SIAM, 2014.

[2] M. Brennan. Constructing demand curves from purchase probability data: an application of the juster scale. *Marketing Bulletin*, 6(May):51–58, 1995.

[3] 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.

[4] 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.

[5] 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.

[6] W. Chen, Y. Yuan, and L. Zhang. Scalable influence maximization in social networks under the linear threshold model. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 88–97. IEEE, 2010.

[7] Y. B. Cho, Y. H. Cho, and S. H. Kim. Mining changes in customer buying behavior for collaborative recommendations. *Expert Systems with Applications*, 28(2):359–369, 2005.

[8] 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.

[9] N. Du, L. Song, M. Gomez-Rodriguez, and H. Zha. Scalable influence estimation in continuous-time diffusion networks. In *Advances in Neural Information Processing Systems*, pages 3147–3155, 2013.

[10] M. Eftekhar, Y. Ganjali, and N. Koudas. Information cascade at group scale. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 401–409. ACM, 2013.

[11] M. Farajtabar, N. Du, M. Gomez-Rodriguez, I. Valera, H. Zha, and L. Song. Shaping social activity by incentivizing users. In *NIPS*, 2014.

[12] 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.

[13] 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.

[14] S. Lei, S. Maniu, L. Mo, R. Cheng, and P. Senellart. Online influence maximization. In *Proceedings of the 21st ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 645–654. ACM, 2015.

[15] 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.

[16] C. Long and R.-W. Wong. Minimizing seed set for viral marketing. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 427–436. IEEE, 2011.

[17] M. Mitzenmacher and E. Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[18] E. Mossel and S. Roch. On the submodularity of influence in social networks. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 128–134. ACM, 2007.

[19] H. Nguyen and R. Zheng. On budgeted influence maximization in social networks. *Selected Areas in Communications, IEEE Journal on*, 31(6):1084–1094, 2013.

[20] Y. Singer. How to win friends and influence people, truthfully: influence maximization mechanisms for social networks. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 733–742. ACM, 2012.

[21] E. Suh, S. Lim, H. Hwang, and S. Kim. A prediction model for the purchase probability of anonymous customers to support real time web marketing: a case study. *Expert Systems with Applications*, 27(2):245–255, 2004.

[22] Y. Tang, Y. Shi, and X. Xiao. Influence maximization in near-linear time: A martingale approach. In *Proceedings of the 2015 ACM SIGMOD international conference on Management of data*. ACM, 2015.

[23] Y. Tang, X. Xiao, and Y. Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 75–86. ACM, 2014.

[24] G. Ventures. Business advertising. `http://www.gaebler.com/Business-Advertising.htm`.

[25] Y. Wang, W. Huang, L. Zong, T. Wang, and D. Yang. Influence maximization with limit cost in social network. *Science China Information Sciences*, 56(7):1–14, 2013.

Appendix

Proof of Theorem 1 We prove by a simple reduction from computing $I(S)$. For any S, we can make a configuration C such that $c_u = 1$ if $u \in S$ and $c_u = 0$ otherwise. Clearly we have $UI(C) = I(S)$. Thus, if computing $I(S)$ is #P-hard, so is computing $UI(C)$. ∎

Proof of Theorem 2 Recall that $UI(C) = \sum_{S\in 2^V} Pr(S;V,C)I(S)$. $\forall S \in 2^V$, we have $0 \le I(S) \le n$. We can estimate $UI(C)$ by a Monte Carlo method. By applying the Hoeffding bound, we have

$$Pr(|\widehat{UI}(C) - UI(C)| \ge \varepsilon UI(C)) \le 2e^{-\frac{2R^2\varepsilon^2 UI^2(C)}{Rn^2}},$$

where R is the number of MC simulations. Since $UI(C) \ge \sum_{u\in V} p_u(c_u)$, to achieve the goal that $Pr(|\widehat{UI}(C) - UI(C)| \ge \varepsilon UI(C)) \le \delta$, we can set $R \ge \frac{n^2 \ln \frac{2}{\delta}}{2\varepsilon^2(\sum_{u\in V} p_u(c_u))^2}$. ∎

Proof of Theorem 3 If there is a FPRAS for estimating $I(S)$, in $O(\text{POLY}(\frac{n}{\varepsilon'}\ln\frac{1}{\delta'}))$ time we can obtain an (ε',δ') approximation of $I(S)$. If we set $\delta' = \frac{\delta_1}{O(\text{POLY}(\frac{n}{\varepsilon'}\ln\frac{1}{\delta_1}))}$, the time we need is $O(\text{POLY}(\frac{n}{\varepsilon'}\ln\frac{O(\text{POLY}(\frac{n}{\varepsilon'}\ln\frac{1}{\delta_1}))}{\delta_1})) = O(\text{POLY}(\frac{n}{\varepsilon'}\ln\frac{1}{\delta_1}))$.

According to Theorem 2, if we can access the value of $I(S)$, we can have an (ε',δ_1) approximation of $UI(C)$ by calling the oracle $O(\text{POLY}(\frac{n}{\varepsilon'}\ln\frac{1}{\delta_1}))$ times. Suppose $R = O(\text{POLY}(\frac{n}{\varepsilon'}\ln\frac{1}{\delta_1}))$, and we have R randomly generated seed sets $\{S_1,S_2,...,S_R\}$. Let $\widehat{UI}(C) = \sum_{i=1}^R I(S_i)$, $\widehat{I}(S_i)$ be the estimated influence spread of S_i using the FPRAS in $O(\text{POLY}(\frac{n}{\varepsilon'}\ln\frac{1}{\delta_1}))$ time, and $\widehat{\widehat{UI}}(C) = \sum_{i=1}^R \widehat{I}(S_i)$. Clearly, $Pr\{|\widehat{UI}(C) - UI(C)| > \varepsilon' UI(C)\} < \delta_1$, and $Pr\{|\widehat{\widehat{UI}}(C) - \widehat{UI}(C)| > \varepsilon'\widehat{UI}(C)\} < \delta_1$ (by Union Bound). Applying union bound, we have $Pr\{|\widehat{\widehat{UI}}(C) - UI(C)| < \varepsilon'(2+\varepsilon')UI(C)\} > 1-2\delta_1$.

Setting $\varepsilon'(2+\varepsilon') = \varepsilon$ and $2\delta_1 = \delta$, which means $\varepsilon' = \sqrt{1+\varepsilon}-1$ and $\delta_1 = \frac{\delta}{2}$, we can obtain an (ε,δ) approximation of $UI(C)$ in $O(\text{POLY}(\frac{n}{\varepsilon'}\ln\frac{1}{\delta_1})) \times O(\text{POLY}(\frac{n}{\varepsilon'}\ln\frac{1}{\delta_1})) = O(\text{POLY}(\frac{n}{\varepsilon'}\ln\frac{1}{\delta_1}))$ time. Note that $\frac{1}{\varepsilon'} = \frac{1}{\sqrt{1+\varepsilon}-1} = \frac{\sqrt{\varepsilon+1}+1}{\varepsilon} \le \frac{3}{\varepsilon} = O(\frac{1}{\varepsilon})$, and $\ln\frac{1}{\delta_1} = O(\ln\frac{1}{\delta})$. So in $O(\text{POLY}(\frac{n}{\varepsilon}\ln\frac{1}{\delta}))$ time we can have an (ε,δ) approximation of $UI(C)$. ∎

Proof of Theorem 4 We firstly prove $UI(C)$ equals the influence spread of a fixed seed sets on a gadget graph. Given a graph $G = \langle V,E\rangle$ and a configuration C, we create a gadget graph $G' = \langle V',E'\rangle$ by adding a gadget node u' for each node u in G, and adding a direct edge (u',u) with propagation probability $p_u(c_u)$. Using the equivalent live edges random process of independent cascade model or linear threshold model [13], it can be easily proved that $UI(C) = I(V'-V) - n$. So we can use the same Monte Carlo simulation for computing influence spread of a given seed set to estimate $I(V'-V)$ and then obtain $UI(C)$. Applying a simple Hoeffding bound to obtain an (ε,δ) approximation of $UI(C)$, we obtain the number of Monte Carlo simulations needed as $O(\frac{n^2\ln\frac{1}{\delta}}{2\varepsilon^2(\sum_{u\in V} p_u(c_u))^2})$. Since each Monte Carlo simulation takes $O(m)$ time. in total we can have an (ε,δ) approximation of $UI(C)$ in $O(\frac{mn^2\ln\frac{1}{\delta}}{2\varepsilon^2(\sum_{u\in V} p_u(c_u))^2})$ time. ∎

Proof of Lemma 1 Because $p_u(c_u)$ is monotonic with respect to c_u, we have $p_u(c_u^1) \ge p_u(c_u^2)$. Thus, $p_u(c_u^1) - p_u(c_u^2) = \alpha \ge 0$. We have

$$\begin{aligned} UI(C) = &\sum_{S\in 2^{V-\{u\}}} Pr(S;V-\{u\},C)I(S)[1-p_u(c_u)]+ \\ &\sum_{S\in 2^{V-\{u\}}} Pr(S;V-\{u\},C)I(S\cup\{u\})p_u(c_u) \end{aligned}$$

Therefore,

$$\begin{aligned} &UI(C_1) - UI(C_2) \\ =& \sum_{S\in 2^{V-\{u\}}} Pr(S;V-\{u\},C_1)I(S\cup\{u\})\alpha \\ -& \sum_{S\in 2^{V-\{u\}}} Pr(S;V-\{u\},C_2)I(S)\alpha \\ =& \sum_{S\in 2^{V-\{u\}}} \alpha Pr(S;V-\{u\},C_2)[I(S\cup\{u\}) - I(S)] \end{aligned}$$

Due to the monotonicity of $I(S)$, $I(S\cup\{u\}) - I(S) \ge 0$ and $UI(C_1) - UI(C_2) \ge 0$. Thus, we have $UI(C_1) \ge UI(C_2)$. ∎

Proof of Theorem 6 The major idea of our proof is to show that, for an arbitrary feasible configuration C, there is an integer configuration C' such that $UI(C') \ge UI(C)$. As $p_u(c_u) \le c_u$ stated in the

theorem, we consider two cases.

First, we consider the situation where $\forall u \in V, p_u(c_u) = c_u$. For any feasible configuration C, in Line 3 of Algorithm 1, if C contains a component c_i that is not an integer, since B is an integer, C must contains another component c_j ($i \neq j$) such that c_j is not an integer, either. Therefore, we can always pick non-integers c_i and c_j. Then, we optimize over c_i and c_j by solving the optimization problem in Eq. 7. Due to Eq. 9, we have

$$\begin{aligned} UI(C) =& (A_1 + A_2 - A_3 - A_4)c_i(B' - c_i) + \\ & (A_3 - A_1)c_i + (A_4 - A_1)(B' - c_i) \end{aligned}$$

which is a quadratic form of c_i and the coefficient of the quadratic term is $-(A_1 + A_2 - A_3 - A_4)$. Since $I(S)$ is submodular, we have $I(S \cup \{i,j\}) - I(S \cup \{j\}) \leq I(S \cup \{i\}) - I(S)$. Thus, $(A_1 + A_2 - A_3 - A_4) \leq 0$, which means $UI(C)$ is a convex function with respect to c_i. Therefore, the value of x that makes $\frac{\mathrm{d}UI(C)}{\mathrm{d}c_i}|_{c_i = x} = 0$ is the global minimum. Since we are interested in only the maximum, we can ignore the root of $\frac{\mathrm{d}UI(C)}{\mathrm{d}c_i} = 0$ completely. This means that, after optimization, c_i must be either $max(0, B' - 1)$ or $min(B', 1)$.

Let us examine the value of c_i and c_j after optimization under different situations of B'. There are two possible cases.

- If $B' \geq 1$, then $B' - 1 \leq c_i \leq 1$. After optimization over c_i and c_j, if $c_i = B' - 1$, then $c_j = 1$. If $c_i = 1$, then $c_j = B' - 1$.
- If $B' < 1$, then $0 \leq c_i \leq B'$. After optimization over c_i and c_j, if $c_i = 0$, then $c_j = B'$. If $c_i = B'$, then $c_j = 0$.

Thus, after optimization over c_i and c_j, at least one variable of c_i and c_j takes an integer value. In other words, after one iteration we eliminate at least one non-integer c_u. Apparently, after at most n iterations we can make all c_u's integers. Since in every iteration the objective function does not decrease, the final integer configuration C' can achieve an influence spread no smaller than the initial configuration. Therefore, we only need to consider integer configurations, which means CIM degenerates into DIM.

Second, we consider the situation when there exists at least one node u such that $p_u(c_u) < c_u$. In other words, we consider for each u, $p_u(c_u) \leq c_u$. Let $\bar{p}_u(c_u)$ be the seed probability function such that for each u, $\bar{p}_u(c_u) = c_u$. Denote by $\overline{UI}(C)$ be the influence spread using $\bar{p}_u(c_u)$ with respect to configuration C. For each u, due to the assumption that $\bar{p}_u(\cdot)$ is continuous, we have $\bar{c}_u$ such that $p_u(c_u) = \bar{p}_u(\bar{c}_u) \leq c_u = \bar{p}_u(c_u)$. Consider two configurations $C = (c_1, \ldots, c_n)$ and $\overline{C} = (\bar{c}_1, \ldots, \bar{c}_n)$. Clearly, $\overline{C} \preceq C$. Due to the monotonicity of $UI(C)$, we have $UI(C) = \overline{UI}(\overline{C}) \leq \overline{UI}(C)$. As the first case, we already prove that $\overline{UI}(C)$ has the same objectives in CIM and DIM when C is an integer configuration. Note that for any integer configuration C, $UI(C) = \overline{UI}(C)$. Thus, the theorem holds in this general case. ∎

Proof of Theorem 7 Suppose in this iteration c_i and c_j are picked for optimization. Clearly after optimization over c_i and c_j, if c_i and c_j change, then one of them increases and the other decreases, because the sum of c_i and c_j remains a constant. Without loss of generality, assume c_i increases by α and $p_i(c_i + \alpha) - p_i(c_i) = \beta$. Apparently, since $0 \leq p_i(c_i) \leq 1$ and $p_i(c_i)$ is monotonic, $\beta \leq 1$.

Let C_2 be a new configuration such that $C_2(j) = C_1(j) + \alpha$ and all other entries of C_2 are identical to those in C_1. Due to the monotonicity of $UI(C)$, since $C_2 \succeq C_1$, we have $UI(C_2) \geq UI(C_1)$. Moreover,

$$UI(C_2) - UI(C) = \beta \sum_{S \in 2^{V - \{i\}}} Pr(S; V - \{i\}, C)\big(I(S \cup \{u\}) - I(S)\big).$$

Therefore,

$$\begin{aligned} & UI(C_1) - UI(C) \leq UI(C_2) - UI(C) \\ \leq & \sum_{S \in 2^{V - \{i\}}} Pr(S; V - \{i\}, C)\big(I(S \cup \{u\}) - I(S)\big) \\ \leq & \operatorname*{argmax}_{u \in V} \sum_{S \in 2^{V - \{u\}}} Pr(S; V - \{u\}, C)\big(I(S \cup \{u\}) - I(S)\big) \end{aligned}$$

∎

Proof of Theorem 8 The monotonicity can be immediately proved using Theorem 5. Next, we show the submodularity of $UI(S; c)$.

Suppose we have two sets S_1 and S_2 such that $S_1 \cup \{v\} = S_2$. Let u be a node such that $u \notin S_2$. Then,

$$\begin{aligned} & UI(S_1 \cup \{u\}; c) - UI(S_1; c) \\ = & \sum_{S \in 2^{S_1}} Pr(S; S_1, c)\Big(p_u(c)I(S \cup \{u\}) + (1 - p_u(c))I(S)\Big) - \\ & \sum_{S \in 2^{S_1}} Pr(S; S_1, c)I(S) \\ = & \, p_u(c) \sum_{S \in 2^{S_1}} Pr(S; S_1, c)\Big(I(S \cup \{u\}) - I(S)\Big) \end{aligned}$$

We also have

$$\begin{aligned} & UI(S_2 \cup \{u\}; c) - UI(S_2; c) \\ = & \sum_{S \in 2^{S_1}} Pr(S; S_1, c)\Big(p_v(c)p_u(c)I(S \cup \{u, v\}) + \\ & \qquad p_v(c)(1 - p_u(c))I(S \cup \{v\}) + \\ & \qquad p_u(c)(1 - p_v(c))I(S \cup \{u\}) + \\ & \qquad (1 - p_u(c))(1 - p_v(c))I(S)\Big) - \\ & \sum_{S \in 2^{S_1}} Pr(S; S_1, c)\Big(p_v(c)I(S \cup \{v\}) + (1 - p_v(c))I(S)\Big) \\ = & \sum_{S \in 2^{S_1}} Pr(S; S_1, c)\Bigg(p_u(c)p_v(c)\Big(I(S \cup \{u, v\}) - I(S \cup \{v\})\Big) + \\ & \qquad p_u(c)(1 - p_v(c))\Big(I(S \cup \{u\}) - I(S)\Big)\Bigg) \\ \leq & \sum_{S \in 2^{S_1}} Pr(S; S_1, c)\Bigg(p_u(c)p_v(c)\Big(I(S \cup \{u\}) - I(S)\Big) + \\ & \qquad p_u(c)(1 - p_v(c))\Big(I(S \cup \{u\}) - I(S)\Big)\Bigg) \\ = & \, p_u(c) \sum_{S \in 2^{S_1}} Pr(S; S_1, c)\Big(I(S \cup \{u\}) - I(S)\Big) \\ = & \, UI(S_1 \cup \{u\}; c) - UI(S_1; c) \end{aligned}$$

That is, we prove

$$UI(S_1 \cup \{u\}; c) - UI(S_1; c) \geq UI(S_2 \cup \{u\}; c) - UI(S_2; c).$$

By a simple induction, we can show that, if $S \subseteq T$ and $u \notin T$, $UI(S \cup \{u\}; c) - UI(S; c) \geq UI(T \cup \{u\}; c) - UI(T; c)$, which means $UI(S; c)$ is submodular with respect to S. ∎

Proof of Theorem 9 Recall that in the proof of Theorem 4 we illustrated that $UI(C) = I(V' - V) - n$ on the gadget graph $G' = \langle V', E' \rangle$. Consider a poll process on G' similar to it on G: we firstly randomly pick a node $v \in V$, then starting from v we run the random reverse propagation to generate a random hyper-edge h' on G'. According to the equivalent living edges process of IC model, this random

reverse propagation process can be divided into two steps. (1) We randomly reversely "propagate" to nodes in V; and (2) we reversely "propagate" to nodes in $V' - V$. Clearly ,the first step is equivalent to generating a random hyper edge on G. Denoted by $h'^G = h' \cap V$ the set polled in the first step.

Denote by $Pr(h'^G = h)$ the probability that in the first step the set of nodes h is polled and $Pr(h' \cap (V - V') \neq \varnothing)$ the probability that at least one node in $V' - V$ is reversely propagated to. Similar to [1], $Pr\{h' \cap (V - V') \neq \varnothing\} = \frac{I(V'-V)-n}{n}$. According to the equivalent living edges process of the IC model, given $h'^G = h$, the conditional probability $Pr\{h' \cap (V - V') \neq \varnothing \mid h'^G = h\} = 1 - \prod_{u \in h}(1 - p_u(c_u))$.

Now we consider for a random hyper-edge h the expectation of $1 - \prod_{u \in h}(1 - p_u(c_u))$,

$$
\begin{aligned}
& E[1 - \prod_{u \in h}(1 - p_u(c_u))] \\
&= \sum_{S \in 2^V} Pr(h = S)[1 - \prod_{u \in S}(1 - p_u(c_u))] \\
&= \sum_{S \in 2^V} Pr(h'^G = S) Pr\{h' \cap (V - V') \neq \varnothing \mid h'^G = S\} \\
&= Pr\{h' \cap (V - V') \neq \varnothing\} \\
&= \frac{I(V' - V) - n}{n}
\end{aligned}
$$

Apparently,

$$E[\sum_{h \in H}[1 - \prod_{u \in h}(1 - p_u(c_u))]] = mH * E[1 - \prod_{u \in h}(1 - p_u(c_u))]$$

Since $UI(C) = I(V' - V) - n$, $\frac{n * \sum_{h \in H}[1 - \prod_{u \in h}(1 - p_u(c_u))]}{mH}$ is an unbiased estimation of $UI(C)$. ■