VARIANT INFLUENCE MAXIMIZATION: APPROXIMATION ALGORITHM AND DEEP SOLUTION

by

Tiantian Chen

APPROVED BY SUPERVISORY COMMITTEE:

Weili Wu, Chair

Ding-Zhu Du, Co-Chair

Feng Chen

Latifur Khan

This thesis is dedicated to my parents, supervisors, and friends.

VARIANT INFLUENCE MAXIMIZATION: APPROXIMATION ALGORITHM AND DEEP SOLUTION

by

TIANTIAN CHEN, BS, MS

DISSERTATION

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN

COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

May 2023

ACKNOWLEDGMENTS

Firstly, I would like to thank my advisors, Dr. Weili Wu and Dr. Ding-Zhu Du, for their invaluable and professional support and suggestions throughout my PhD journey. I first met Dr. Ding-Zhu Du at a research talk when I was still an undergraduate. I was deeply attracted by his talk and the problems considered in his works are very interesting and closely related to our daily life. After that, we met again at a graduate summer school, and I learned more about social networks and optimization theory and expressed my great interest in pursuing my PhD degree under Dr. Du's supervision. Dr. Du provided continuous expert support and belief in my abilities, which helped me finish this work.

It is also my great honor to have Dr. Wu as my co-advisor. Every time I encountered difficulties, I would go to Dr. Wu, and she always provided helpful suggestions and encouragement, which gave me the confidence to persevere through the challenges and obstacles I faced during this process. Her willingness to share her expertise, insights, and experiences with me has been instrumental in enabling me to pursue my academic goals and achieve my dreams. I am deeply grateful for her mentorship and the positive impact she has had on my academic and personal growth.

Secondly, I really appreciate Dr. Latifur Khan and Dr. Feng Chen for being part of my dissertation committee. Their valuable and helpful feedback and suggestions have greatly improved the quality of this work.

Thirdly, I would like to express my heartfelt appreciation to my master advisor, Dr. Qizhi Fang, who has led me to the research road. Without her, I would never have stepped foot on this field. Her dedication to my success has been unwavering, and I am truly grateful for the knowledge, skills, and guidance she has provided me throughout my academic journey. Also, I want to thank my colleagues, especially Jianxiong Guo, for his continuous support

during my PhD period, my collaborators, Siwen Yan, Xiao Li, Wenting Wang, and Smita Ghosh, and other labmates.

Finally, I want to thank my whole family for their unwavering support, love, and belief in me, which have been the constant source of my strength and inspiration.

April 2023

VARIANT INFLUENCE MAXIMIZATION: APPROXIMATION ALGORITHM AND DEEP SOLUTION

Tiantian Chen, PhD
The University of Texas at Dallas, 2023

Supervising Professors: Weili Wu, Chair
Ding-Zhu Du, Co-Chair

In recent two decades, online social platforms have become more and more popular, and the dissemination of information on social networks has attracted wide attention of the industries and academia. The users of these social media platforms and the relationships between them can be characterized as a social network. A large number of works have been focused on the diffusion phenomenon on social networks, including diffusion of ideas, news, adoptions of new products, etc. Influence maximization problem is one of the well-studied topics, which seeks for a small subset of nodes as seeds to maximize the expected number of influenced users under some diffusion model. However, there are some impractical assumptions of this problem, such as the uniform cost of activating nodes as seeds and profit obtained from influenced users. In this dissertation, we propose several practical and novel variants of influence maximization problem: continuous activity maximization problem, budget profit maximization with coupon advertisement problem, adaptive multi-feature budgeted profit maximization problem, and learning-based influence maximization problem. Due to their NP-hardness, we focused on designing approximation algorithms and the deep reinforcement learning model. Reverse influence sampling and deep Q-networks techniques are utilized to

overcome the #P-hardness of computing the objective functions and to solve the problems more efficiently.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The rise of online social platforms, such as Facebook, Twitter, LinkedIn, has dramatically impacted the way we consume and disseminate information. Because of "word of mouth" effects, information usually can spread rapidly on these social media platforms. Therefore, it is vital to understand the potential diffusion dynamics kinetics and quantify the consequences of information spread. One of the problems that has been extensively studied is the influence maximization problem, seeking to select a small group of users as the initial information spreader to maximize the number of users influenced. Kempe *et al.* (Kempe et al., 2003) first formulated the influence maximization problem as a combinatorial optimization problem and proved that it is NP-hard under two proposed diffusion models: Independent Cascade (IC) model and Linear Threshold (LT) model. They designed the greedy algorithm for influence maximization, which can achieve $(1 - 1/e)$-approximation ratio. The formal definitions and more details will be introduced in Chapter 2 (see p. 6).

However, it is #P-hard to compute the objective function of influence maximization problem under the IC model (Chen et al., 2010a) and LT model (Chen et al., 2010b). Kempe *et al.* utilized the Monte Carlo method to make the estimation of the objective function, which is too time-consuming. Later, a large number of works have been focused on solving influence maximization problem and its variations more efficiently, such as profit maximization and rumor blocking. In this dissertation, we consider several variants of influence maximization problem: continuous activity maximization problem, budget profit maximization with coupon advertisement problem, adaptive multi-Feature budgeted profit maximization problem, and learning-based influence maximization problem, and propose efficient and effective algorithms to solve them.

Continuous Activity Maximization. Activity maximization is a task of seeking a small subset of users in a given social network that makes the expected total activity benefit

maximized. This is a generalization of many real applications. In this paper, we extend activity maximization problem to that under the general marketing strategy $\vec{x}$, which is a d-dimensional vector from a lattice space and has probability $h_u(\vec{x})$ to activate a node u as a seed. Based on that, we propose the continuous activity maximization problem, where the domain is continuous and the seed set we select conforms to a certain probability distribution. It is a new topic to study the problem about information diffusion under the lattice constraint. Therefore, we address the problem systematically here. First, we analyze the hardness of continuous activity maximization and how to compute the objective function of continuous activity maximization accurately and effectively. We prove this objective function is monotone, but not DR-submodular and not DR-supermodular. Then, we develop a monotone and DR-submodular lower bound and upper bound of continuous activity maximization, and apply sampling techniques to design three unbiased estimators for continuous activity maximization, its lower bound and upper bound. Next, adapted from the IMM algorithm and sandwich approximation framework, we obtain a data-dependent approximation ratio. This process can be considered as a general method to solve those maximization problems on lattice but not DR-submodular. Last, we conduct experiments on three real-world datasets to evaluate the correctness and effectiveness of our proposed algorithms. Details will be introduced in Chapter 3 (see p. 9).

Budget Profit Maximization with Coupon Advertisement. Coupon advertisement is everywhere in people's daily lives, and it is a common marketing strategy adopted by merchants. A problem, Budget Profit Maximization with Coupon Advertisement, is formulated in this paper, which is a branch of classical profit maximization problem in social networks. Profit maximization has been researched intensively before, but its theoretical bound is not satisfactory usually because of NP-hardness, budget constraint and non-monotonicity. Learned from the latest results, we proposed the B-Framework, which combines the ideas of Random Greedy and Continuous Double Greedy to obtain a more

acceptable approximation ratio for this problem. For Continuous Double Greedy, it can be implemented by multilinear extension and discretized techniques. In addition, most existing researches only consider maximizing total profit. However, in real scenarios, the diffusion probability is hard to determine due to the uncertainty. Then, we study the robustness for budgeted profit maximization, which can be used as a general strategy to analyze the robustness of non-monotone submodular function. It aims to obtain the best solution maximizing the ratio between the profit of any feasible seed set and the optimal seed set. We design LU-B-Framework first, and then we apply the method of uniform sampling to improve the robustness by reducing the uncertainty. The effectiveness and correctness of our algorithms are evaluated by heavy simulation on real-world social networks eventually. Details will be introduced in Chapter 4 (see p. 40).

Adaptive Multi-Feature Budgeted Profit Maximization. Online social networks have been one of the most important platforms for viral marketing. Most existing researches about diffusion of adoptions of new products on networks are about one diffusion. That is, only one piece of information about the product is spread on the network. However, in fact, one product may have multiple features and the information about different features may spread independently in social networks. When a user would like to purchase the product, he would consider all of the features of the product comprehensively, not just consider one. Based on this, we propose a novel problem, multi-feature budgeted profit maximization problem, which first considers budgeted profit maximization under multiple features propagation of one product. Given a social network with each node having an activation cost and a profit, multi-feature budgeted profit maximization problem seeks for a seed set with expected cost no more than the budget to make the total expected profit as large as possible. We mainly consider the multi-feature budgeted profit maximization problem under the adaptive setting, where seeds are chosen iteratively and the next seed is selected according to current diffusion results. We study the adaptivemulti-feature budgeted profit

maximization problem under two models, oracle model and noise model. The oracle model assumes conditional expected marginal profit of any node could be obtained in $O(1)$ time and a $(1 - 1/e)$ expected approximation policy is proposed. Under the noise model, we estimate conditional expected marginal profit of a node by modifying the EPIC algorithm and propose an efficient policy, which could achieve a $(1 - e^{-(1-\epsilon)})$ expected approximation ratio. Several experiments are conducted on six realistic datasets to compare our proposed policies with their corresponding non-adaptive algorithms and some heuristic adaptive policies. Experimental results show efficiencies and superiorities of our policies. Details will be demonstrated in Chapter 5 (see p. 68).

Learning-based Influence Maximization. Aiming at selecting a small subset of nodes with maximum influence on networks, the influence maximization problem has been extensively studied. Since it is #P-hard to compute the influence spread given a seed set, the state-of-the-art methods, including heuristic and approximation algorithms, faced great difficulties such as theoretical guarantee, time efficiency, generalization, etc. This makes it unable to adapt to large-scale networks and more complex applications. On the other hand, with the latest achievements of deep reinforcement learning in artificial intelligence and other fields, lots of works have been focused on exploiting deep reinforcement learning to solve combinatorial optimization problems. Inspired by this, we propose a novel end-to-end deep reinforcement learning framework, ToupleGDD, to address the IM problem in this paper, which incorporates three coupled graph neural networks for network embedding and double deep Q-networks for parameters learning. Previous efforts to solve the IM problem with deep reinforcement learning trained their models on subgraphs of the whole network, and then tested on the whole graph, which makes the performance of their models unstable among different networks. However, our model is trained on several small randomly generated graphs with a small budget, and tested on completely different networks under various large budgets, which can obtain results very close to IMM and better results than OPIM-C

on several datasets, and shows strong generalization ability. Finally, we conduct a large number of experiments on synthetic and realistic datasets, and experimental results prove the effectiveness and superiority of our model. Details will be shown in Chapter 6 (see p. 100).

The rest of this dissertation proceeds as follows: Chapter 2 provides the background knowledge for the rest of the chapters. Continuous activity maximization problem, budget profit maximization with coupon advertisement problem, and adaptive multi-Feature budgeted profit maximization problem, are introduced in through Chapter 3 to Chapter 5, respectively. Chapter 6 discusses the learning-based influence maximization problem and Chapter 7 concludes this dissertation.

CHAPTER 2

PRELIMINARIES

In this chapter, some background knowledge is introduced, including social networks, diffusion models, submodular set function and notations.

2.1 Social Network

A social network is represented by a directed graph $G = (V, E)$ where V denotes the set of (nodes) users, and E denotes the set of directed edges which describe the relationship between users. Assume $|V| = n$ and $|E| = m$. For each edge $(u, v) \in E$, we say u (resp. v) is an incoming neighbor (resp. an outgoing neighbor) of v (resp. u). For each node $v \in V$, $N^-(v)$ (resp. $N^+(v)$) denotes the set of incoming neighbors (resp. outgoing neighbors) of node v, and $N(v) = N^-(v) \cup N^+(v)$. We adopt the IC model and LT model (Kempe et al., 2003), to model the influence diffusion. Each node has two possible states: active and inactive. Initially, all nodes in the seed set S are activated and all other nodes are set inactive. Then the diffusion process repeats, and terminates until no new node is activated. Once a node is activated, it remains active in the following timestamps.

2.2 Basic Diffusion Models

Definition 2.2.1 (IC model)**.** *It assumes that when a node u is activated in this round, in the next round, which has only one chance to activate those nodes v in its outgoing neighbors $N^+(u)$ with a predefined probability. Each edge (u, v) is associated with an activation probability $p_{uv} \in [0, 1]$, and the activation process of different edges or different rounds is independent. Then, the diffusion process terminates when no node becomes active in this round.*

Definition 2.2.2 (LT model)**.** *Each edge* $(u, v) \in E$ *has a weight* b_{uv}*, and each node* $v \in V$ *has a threshold* θ_v *sampled uniformly in* $[0, 1]$ *and* $\sum_{u \in N^-(v)} b_{uv} \leq 1$*. For each inactive node* v *at time step* $t-1$*, it can be activated at time step* t *if satisfying* $\sum_{u \in A_{t-1} \cup N^-(v)} b_{uv} \geq \theta_v$*, where* A_{t-1} *is the set of active nodes at time step* $t-1$*.*

The diffusion process will continue until there is no more node activated. Given a seed set S, denote by $I(S)$ the number of activated nodes when the diffusion process terminates. Let $\sigma(S)$ be the expected number of nodes that can be activated by S. That is, $\sigma(S) = \mathbb{E}[I(S)]$ and $\sigma(S)$ is called the influence spread of S.

2.3 Realization

A (full) realization $\mathrm{g} = (V, E(\mathrm{g}))$ is a subgraph of G with $E(\mathrm{g}) \subseteq E$. Each edge in $E(\mathrm{g})$ is a live edge, or else it is a blocked edge. Under the IC model, we can decide whether edge (u, v) is live or blocked with probability p_{uv}. Let $\Pr[\mathrm{g}]$ be the probability of g sampled from G based on IC model. That is,

$$\Pr[\mathrm{g}] = \prod_{e \in E(\mathrm{g})} p_e \prod_{e \in E \setminus E(\mathrm{g})} (1 - p_e)\,. \tag{2.1}$$

There are 2^m possible realization altogether under the IC model. Under the LT model, node v chooses at most one of incoming neighbors u from $N^-(v)$ such that edge (u, v) appears in $E(\mathrm{g})$. Thus, for each node $u \in N^-(v)$, (u, v) appears in $E(\mathrm{g})$ with probability b_{uv} exclusively, and there is no incoming edge of v in $E(\mathrm{g})$ with probability $1 - \sum_{u \in N^-(v)} b_{uv}$. We define $V'(\mathrm{g}) = \{v : \nexists (u, v) \in E(\mathrm{g})\}$ as the node set which has no incoming edge in realization g. Let $\Pr[\mathrm{g}]$ be the probability of g sampled from G based on LT model. That is,

$$\Pr[\mathrm{g}] = \prod_{e \in E(\mathrm{g})} b_{uv} \prod_{v \in V'} \left(1 - \sum_{u \in N^-(v)} b_{uv}\right). \tag{2.2}$$

Given a seed set S, the influence diffusion can be considered as a stochastic diffusion process on graph G under the IC/LT model, or a deterministic diffusion process on a realization

g generated from G with probability Pr[g]. Therefore, for the IC/LT model, we have

$$\sigma(S) = \sum_{\mathrm{g} \in \mathcal{G}} \Pr[\mathrm{g}] \cdot \sigma_{\mathrm{g}}(S), \tag{2.3}$$

where $\mathcal{G}$ is the set of all realizations generated from G, and $\sigma_{\mathrm{g}}(\cdot)$ is the number of nodes that can be reached from some node in S in the realization g.

2.4 Influence Maximization

Definition 2.4.1 (Influence Maximization (IM))**.** *Given a social network $G = (V, E)$, a positive integer b and a diffusion model, IM aims to find a small set S of nodes as seeds with $|S| \leq b$, which has the maximum influence spread.*

Definition 2.4.2. *A set function $f : 2^V \to \mathbb{R}$ is said to be monotone non-decreasing if $f(S) \leq f(T)$ for any $S \subseteq T \subseteq V$.*

Definition 2.4.3. *A set function f is said to be submodular when satisfying $f(S \cup \{u\}) - f(S) \geq f(T \cup \{u\}) - f(T)$ for any $S \subseteq T \subseteq V$ and $u \in V \setminus T$.*

Monotonicity and submodularity are important properties to analyze set function, because it can be optimized easily supported by existing theory, such as a $(1-1/e)$-approximation obtained by classical hill-climbing algorithm (Nemhauser et al., 1978).

Theorem 2.4.4. *(Kempe et al., 2003) The expected influence $\sigma(\cdot)$ is monotone non-decreasing and submodular under the IC/LT model.*

CHAPTER 3

CONTINUOUS ACTIVITY MAXIMIZATION [1]

Authors – Jianxiong Guo, Tiantian Chen, and Weili Wu

The Computer Science Department, EC 31

The University of Texas at Dallas

800 West Campbell Road

Richardson, Texas 75080-3021

The online social platforms, such as Twitter, WeChat, Facebook and LinkedIn, were developing quickly in recent years, and gradually become a mainstream way to communicate and make friends. More and more people share what one sees and hears, and discuss some hot issues in these social platforms. The relationships among the users in these social platforms can be represented by social networks, and information can be spread rapidly through the edges in social networks. Based on that, Influence Maximization (IM) considers the problem: selects a subset of users for an information cascade to maximize the expected follow-up adoptions (influence spread). It is a mathematical generalization of plenty of real scenarios, such as viral marketing, rumor blocking and profit maximization. In the Kempe *et al.*'s seminal work (Kempe et al., 2003), two widely accepted diffusion models were proposed, IC model and LT model, where IC model is relied on peer-to-peer communication but LT model considers the total influence from user's neighbors. Then, they showed the IM problem is NP-hard, and its objective function is monotone and submodular under the IC/LT model, and simple greedy algorithm can achieve $(1 - 1/e)$-approximation (Nemhauser et al., 1978). In order to solve its efficiency problem, there were lots of scalable IM algorithms proposed, heuristic algorithms (Chen et al., 2009) (Chen et al., 2010a) (Chen et al., 2010b) (Goyal et al., 2011b) (Jung et al., 2012) and approximate algorithms that improve the Monte-Carlo simulations (Leskovec et al., 2007) (Borgs et al., 2014) (Tang et al., 2014) (Tang et al., 2015) (Guo and Wu, 2019) (Guo et al., 2019) (Guo et al., 2020a).

Motivated by IM, more interested and real problems emerged and were studied. Wang *et al.* (Wang et al., 2017) considered to maximize the expected total activity strength about the target issue in online social networks and proposed activity maximization problem. The activity maximization aims to maximize the total activity strength (activity benefit) associated with those edges between influenced users given a seed set. Different from IM, maximized expected influenced users does not mean that total activity strength is maximized because different edges are associated with different activity strength. In addition, they have proved

the objective function of activity maximization is NP-hard, monotone, but not submodular and not supermodular (Wang et al., 2017), and gave us a sandwich approximation framework (Lu et al., 2015) to get an approximate solution by approximating its upper bound and lower bound. However, it seems unrealistic to select a deterministic seed set, because we do not know who is willing to be a seed in advance, which is stochastic.

Kempe *et al.* (Kempe et al., 2015) considered a more general case that uses a marketing strategy instead of the seed set. This marketing strategy is denoted by $\vec{x} = (x_1, x_2, \cdots, x_d)$ where each strategy j takes value x_j, and for each node u, it will be activated as a seed with probability $h_u(\vec{x})$. Thus, the seed set is not deterministic, but activated probabilistically according to a marketing strategy. In this paper, we consider the activity maximization problem under such general marketing strategy. We propose the continuous activity maximization (CAM), which is to find the optimal marketing strategy $\vec{x}^*$ such that the expected activity benefit can be maximized subject to the budget constraint $|\vec{x}| \leq k$.

Example 1. *In the real world, the companies often adopt some non-deterministic marketing strategies, such as discounts, coupons, rewards, cashback, and propagandas, each of which corresponds to a component in marketing vector* $\vec{x} = (x_1, x_2, x_3, x_4, x_5)$. *Here* $x_2 = b$ *implies we give* b *units of investment to marketing strategy "coupons". The promotion results on different individuals are random and distinct according to strategy function* $h_u(\vec{x})$. *Therefore, CAM is more realistic and generalized than before.*

In this paper, we consider the marketing strategy $\vec{x}$ taken from discretized lattice $\mathcal{X}$ with granularity t, and the hardness of CAM is discussed. We show that CAM is NP-hard under the IC/LT model. Given a marketing strategy $\vec{x}$, computing the expected activity benefit is #P-hard. Since it is not easy to compute the expected activity benefit with respect to a given marketing strategy $\vec{x}$, we provide an equivalent method to compute it by creating a constructed graph, and running Monte-Carlo simulations on this constructed

graph. Then, we show that the objective function of CAM problem is monotone, but not DR-submodular and not DR-supermodular. DR-submodularity (Soma and Yoshida, 2015) is the diminishing return property extended from set to lattice. If a function defined on lattice is DR-submodular, a $(1 - 1/e)$-approximation can be obtained by the simple greedy algorithm. In order to find a valid approximate solution, we construct a lower bound and upper bound that are monotone and DR-submodular. Similarly, we show that maximizing this lower bound and upper bound is NP-hard as well and computing their exact value is #P-hard under the IC/LT model. For IM problem, the computational cost of greedy algorithm with Monte-Carlo simulations is not acceptable, to our CAM problem, the scalability could be worse than IM because the strategy space is larger and the greedy iterative times should be k/t given a budget k and granularity t. Thus, based on reverse influence sampling (RIS) (Borgs et al., 2014) (Tang et al., 2014) (Tang et al., 2015) (Guo et al., 2020c), we obtain unbiased estimators for the CAM problem and its lower bound based on RE-sampling, for its upper bound based on RN-sampling. The adaptation of RIS to CAM is determined by the partial coverage of the collection of RE-sampling. From this, we design a general scalable algorithm to solve CAM problem, its upper bound and lower bound adapted from IMM algorithm (Tang et al., 2015) for IM problem. We obtain a data-dependent approximation ratio by combining them with the sandwich approximation framework finally. Summarizing our contributions as follows: (1) This is the first to study activity maximization problem under the general marketing strategy (lattice constraint). In this paper, a new problem, named CAM, is proposed and its objective function is proved to be monotone, but not DR-submodular and DR-supermodular; (2) To estimate the expected activity benefit with respect to marketing strategy $\vec{x}$, it could be done on a constructed graph by Monte-Carlo simulations; (3) We obtain a lower bound and upper bound of CAM, which are monotone and DR-submodular; (4) We design unbiased estimators for CAM and its lower/upper bound based on RE/RN-sampling. Adapted from IMM algorithm and sandwich approximation

framework, a data-dependent approximation ratio can be obtained. It is the first time to consider such problems on lattice constraint; (5) The effectiveness and correctness of our proposed algorithms are tested and verified by several datasets of real-word social networks.

3.1 Related Work

Viral marketing was first studied systematically by Domingos Richardson (Domingos and Richardson, 2001) (Richardson and Domingos, 2002), and they proposed the concept of customers' the value and used markov random fields to model the process of viral marketing. Kempe *et al.* (Kempe et al., 2003) formulated IM to a combinatorial optimization problem and gave us a greedy algorithm with $(1-1/e)$-approximation. Chen *et al.* followed Kempe's work, and proved it is #P-hard to compute the exact influence spread for a given seed set under the IC model (Chen et al., 2010a) and the LT model (Chen et al., 2010b). To tackle this problem, Monte-Carlo simulations were adopted as a general method, but the running time was too slow to apply to larger real networks. Subsequently, to attempt to improve the low efficiency of Monte-Carlo simulations, plenty of researchers made effort, for instance, Leskovec *et al.* proposed a CELF algorithm (Leskovec et al., 2007) implemented by a lazy forward evaluation, avoiding redundant computation by exploiting its submodularity. Adapted from CELF, CELF++ reduced its time complexity further. Until the emergence of RIS, it opened a new door for us. Brogs *et al.* (Borgs et al., 2014) proposed the concept of reverse influence sampling (RIS) firstly, which is scalable in practice and guaranteed theoretically at the same time. Then, a series of efficient randomized algorithms were arisen, such as TIM/TIM+ (Tang et al., 2014), IMM (Tang et al., 2015). They were scalable algorithms to solve the IM problem with $(1-1/e-\varepsilon)$-approximation and can be adapted to other relative problems. Other variants derived from IM in social networks are shown in (He et al., 2016) (Cai et al., 2016) (Lin et al., 2019) (Wang et al., 2019) (Chen et al., 2019).

DR-submodular maximization problem on lattice attracted more and more researchers' attention recently. Soma *et al.* (Soma and Yoshida, 2015) generalized the diminishing return property on the integer lattice firstly and solved the submodular cover problem with a bicriteria approximation algorithm. Relied on gradient methods, Hassani *et al.* (Hassani et al., 2017) addressed monotone continuous DR-submodular maximization effectively, but assumed that the function is continuous and differentiable. On integer lattice, Soma *et al.* (Soma and Yoshida, 2018) studied the problem of maximizing monotone DR-submodular exhaustively, where they designed algorithms with $(1 - 1/e - \varepsilon)$-approximation under the cardinality, polymatroid and knapsack constraint. Simultaneously, they (Soma, 2017) considered non-monotone DR-submodular maximization over the integer lattice, and presented a $1/(2 + \varepsilon)$-approximate algorithm within polynomial time. Optimal budget allocation was a typical application of the DR-submodular maximization, and was studied systematically (Soma et al., 2014) (Maehara et al., 2015) (Miyauchi et al., 2015) (Hatano et al., 2016). For social networks, Chen *et al.* (Chen et al., 2018) investigated IM problem over the lattice, whose objective function is monotone and DR-submodular. Following that, we study the activity maximization over lattice, different from IM, our objective function is monotone but not DR-submodular, which is the main contribution of this paper.

3.2 Problem Formulation

In this section, we formulate the CAM problem. Table 3.1 summarizes the frequently used notations. Given a social network $G = (V, E)$, in the activity maximization problem, there is an activity strength $A_{uv} \in \mathbb{R}_+$ associated with each edge $(u, v) \in E$. A_{uv} means that the benefit or profit between user u and user v if they are both active (Wang et al., 2017). Given a social graph $G = (V, E)$, an influence model, and seed set S, let $I(S)$ be a random variable that denotes the set of activated nodes after the diffusion terminates. Let $G[I(S)] = (I(S), E[I(S)])$ be the induced subgraph by activated node set $I(S)$, where we have $E[I(S)] =$

Table 3.1. The frequently used notation summarization

Notation	**Description**
$G = (V, E)$	a graph G with node set V and edge set E
m, n	the number of nodes and edges in G
$f_d(\cdot)$	objective function of DAM
$f_c(\cdot)$	objective function of CAM
$I(S)$	set of activated nodes after diffusion
$G[I(S)]$	subgraph induced by nodes in $I(S)$
$\vec{x}$	marketing strategy (marketing vector)
$h_u(\vec{x})$	activation probability to node u give $\vec{x}$
μ, ν	a random RE(RN)-sampling
θ	# RE(RN)-sampling in a collection
$\overline{f}, \underline{f}$	upper and lower bound of the function f

$\{(u, v) \in E : u \in I(S) \wedge v \in I(S)\}$. Given the seed set S, the activity function of the activity maximization problem (Wang et al., 2017) is

$$f_d(S) = \mathbb{E}\left[\sum_{(u,v)\in E[I(S)]} A_{uv}\right], \tag{3.1}$$

where $f_d(S)$ is the expected activity benefit of final active nodes for the diffusion starting from S. The task of activity maximization is to select at most k seed nodes to maximize the expected activity benefit, i.e., to find $S^* = \arg\max_{S\subseteq V, |S|\leq k} f_d(S)$.

In this paper, we extend the activity maximization problem with general marketing strategy (Kempe et al., 2015), which is a d-dimensional vector $\vec{x} = (x_1, x_2, ..., x_d) \in \mathbb{R}_+^d$. Each component x_i, $i \in [d] = \{1, 2, ..., d\}$, corresponds to the investment to marketing action M_i. Given a marketing strategy $\vec{x}$, the probability that node $u \in V$ is selected as a seed is denoted by strategy function $h_u(\vec{x})$, where $h_u(\vec{x}) \in [0, 1]$. Thus, different from previous definition, the seed set under the general marketing strategy is stochastic, not deterministic. Given a marketing strategy $\vec{x}$, the probability we select $S \subseteq V$ according to $\vec{x}$ as the seed

set is

$$\Pr[S|\vec{x}] = \prod_{u \in S} h_u(\vec{x}) \cdot \prod_{v \in V \setminus S} (1 - h_v(\vec{x})), \tag{3.2}$$

where $\Pr[S|\vec{x}]$ is the probability that exactly nodes in S are selected as seeds but not in S are not selected as seeds under the marketing strategy $\vec{x}$, which is because each node is select as a seed independently. Then, the activity function now is

$$f_c(\vec{x}) = \sum_{S \subseteq V} \Pr[S|\vec{x}] \cdot f_d(S) \tag{3.3}$$

$$= \sum_{S \subseteq V} f_d(S) \cdot \prod_{u \in S} h_u(\vec{x}) \cdot \prod_{v \in V \setminus S} (1 - h_v(\vec{x})). \tag{3.4}$$

Remark 1. *We can address marketing vector $\vec{x}$ in a discretized manner with granularity t, where each component x_i takes discretized value $\{0, t, 2t, \cdots\}$. These set of vectors is called lattice $\mathcal{X}$, where $\mathcal{X} = \{0, t, 2t, \cdots\}^d$.*

Now, we define the continuous activity maximization (CAM) problem as follows:

Problem 1 (Continuous Activity Maximization)**.** *Given a social network $G = (V, E)$ with a influence model, a budget k, a marketing strategy functions $h_u(\cdot)$ for each user u, CAM aims to find an optimal marketing strategy $\vec{x}$ such that the expected activity benefit can be maximized. That is,*

$$\vec{x}^* = \arg \max_{\vec{x} \in \mathcal{X}, |\vec{x}| \leq k} f_c(\vec{x}), \tag{3.5}$$

where consider the marketing strategy $\vec{x}$ under the budget constraint: $|\vec{x}| = \sum_{i \in [d]} x_i \leq k$. Here, each configuration satisfying $\vec{x} \in \mathcal{X}$ and $|\vec{x}| \leq k$ is called a feasible solution.

To make the context clear, we refer to the problem finding $S^* = \arg\max_{S \subseteq V, |S| \leq k} f_d(S)$ as discrete activity maximization (DAM).

3.3 Properties of CAM

In this section, we discuss the hardness, dr-submodularity and computability of our CAM problem.

3.3.1 Hardness

In order to show the hardness, we can start from a classical NP-hard problem, Set Cover problem, and reduce MC to our CAM problem in polynomial time.

Theorem 3.3.1. *The CAM problem is NP-hard under the IC model and the LT model.*

Proof. We assume that $\mathcal{X} = \{0,1\}^n$ and $h_v(\vec{x}) = x_v$, that is, v is selected as a seed if and only if $x_v = 1$. Now, marketing strategy $\vec{x}$ is the characteristic vector of the seed set, and CAM problem can be reduced to DAM problem trivially. It has been proved in (Wang et al., 2017) that DAM is NP-hard under the IC model and LT model by reducing from the set cover problem. Thus, CAM is more general, and it is NP-hard by inheriting the NP-hardness of DAM. □

It is known that under the IC model and LT model, computing influence spread is #P-hard (Chen et al., 2010b) (Chen et al., 2010a). Given a marketing strategy $\vec{x}$, the hardness of computing $f_c(\vec{x})$, that is

Lemma 3.3.2. *Given a marketing strategy $\vec{x}$, computing $f_c(\vec{x})$ by Equation (5) is #P-hard.*

Proof. Similar to the proof of Theorem 1, CAM can be reduced to DAM problem by setting $\mathcal{X} = \{0,1\}^n$ and $h_v(\vec{x}) = x_v$. Based on Equation (3), computing $f_d(S)$ is equivalent to compute $\mathbb{E}[I(S)]$, thus, computing $f_d(S)$ is #P-hard. Except for this special case, the computation of $f_c(S)$ is harder than $f_d(S)$, we hare computing $f_c(S)$ is #P-hard. □

Monte-Carlo simulation can be used to estimate $f_c(\vec{x})$ because it is the expectation of $f_d(\vec{x})$ over the random variable S. We need to sample S according to distribution $\vec{x}$.

Lemma 3.3.3. *Provided that we have value oracle that returns the activity benefit $f_d(S)$ given a seed set S, we can obtain a (γ, δ)-Estimation of $f_c(\vec{x})$ by sampling S according to $\vec{x}$ at least $\frac{\alpha^2 \ln(2/\delta)}{2\gamma^2\beta^2}$ times, where $\alpha = \sum_{(u,v)\in E}[A_{uv}]$ and $\beta = \sum_{e\in E}[h_u(\vec{x})h_v(\vec{x}) \cdot A_{uv}]$.*

Proof. According to Equation (5), we can estimate $f_c(\vec{x})$ with the help of Monte-Carlo simulations, denoted by $\dot{f}_c(\vec{x})$ and based on Hoeffding's inequality, we have

$$\Pr\left[\left|\dot{f}_c(\vec{x}) - f_c(\vec{x})\right| \geq \gamma f_c(\vec{x})\right] \leq 2e^{-\frac{2r\gamma^2(f_c(\vec{x}))^2}{\alpha^2}},$$

where r is the number of Monte-Carlo simulations and $f_d(S) \in [0, \alpha]$. Then, we consider the lower bound of $f_c(\vec{x})$. For each edge $(u, v) \in E$, the probability of both u and v are active is at least $h_u(\vec{x})h_v(\vec{x})$, thus, we have $f_c(\vec{x}) \geq \sum_{e\in E}[h_u(\vec{x})h_v(\vec{x}) \cdot A_{uv}]$. Therefore, we can set $r \geq \frac{\alpha^2 \ln(2/\delta)}{2\gamma^2\beta^2}$ that establishing $\Pr[|\dot{f}_c(\vec{x}) - f_c(\vec{x})| \geq \gamma f_c(\vec{x})] \leq \delta$. □

Unfortunately, it is not easy to compute the activity benefit $f_d(S)$ given a seed set S. Thus, we need to address this problem by other techniques. First, we establish an equivalent relationship between $f_d(\cdot)$ and $f_c(\cdot)$. Given a social graph $G = (V, E)$ and a marketing strategy $\vec{x}$, we create a constructed graph $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ by adding a new node $\widetilde{u}$ and a new directed edge $(\widetilde{u}, u)$ for each node u in V to G. For example, under the IC model, we set the activation probability alone edge $(\widetilde{u}, u)$ is $p_{\widetilde{u}u} = h_u(\vec{x})$. Then, we are able to estimate our objective function directly. That is,

$$f_c(\vec{x}|G) = f_d(\widetilde{V} - V|\widetilde{G}) - \sum_{u\in V}[h_u(\vec{x}) \cdot A_{\widetilde{u}u}], \tag{3.6}$$

where $f_c(\vec{x}|G)$ means computing $f_c(\vec{x})$ under the graph G. We set the activity strength $A_{\widetilde{u}u} = 0$ for each node u in V, then we have $f_c(\vec{x}|G) = f_d(\widetilde{V} - V|\widetilde{G})$. Now, we can compute $f_d(\widetilde{V} - V|\widetilde{G})$ instead of $f_c(\vec{x}|G)$ when we are required to get the value of $f_c(\vec{x}|G)$.

Theorem 3.3.4. *Given a social graph $G = (V, E)$ and a marketing strategy $\vec{x}$, the total running time to get a (γ, δ)-Estimation of $f_c(\vec{x})$ is $O\left(\frac{(m+n)\alpha^2 \ln(2/\delta)}{2\varepsilon^2\beta^2}\right)$, where $\alpha = \sum_{(u,v)\in E}[A_{uv}]$ and $\beta = \sum_{e\in E}[h_u(\vec{x})h_v(\vec{x}) \cdot A_{uv}]$.*

Proof. From the Equation (8), we have $f_c(\vec{x}|G) = f_d(\widetilde{V} - V|\widetilde{G})$. According to Equation (3), we can estimate $f_d(\widetilde{V} - V|\widetilde{G})$ by Monte-Carlo simulations. Denoted by $S' = \widetilde{V} - V$, and

based on Hoeffding's inequality, we have

$$\Pr\left[\left|\dot{f}_d(S') - f_d(S')\right| \geq \gamma f_d(S')\right] \leq 2e^{-\frac{2r\gamma^2(f_d(S'))^2}{\alpha^2}},$$

where r is the number of Monte-Carlo simulations and $\sum_{(u,v)\in\widetilde{E}[I(S')]} A_{uv} \in [0, \alpha]$. Then, we consider the lower bound of $f_d(S')$. Similar to Lemma 2, we have $f_d(S') \geq \sum_{e\in E}[h_u(\vec{x})h_v(\vec{x}) \cdot A_{uv}]$ as well. To achieve a (γ, δ)-Estimation of $f_d(S')$, the number of Monte-Carlo simulations is at least $\frac{\alpha^2 \ln(2/\delta)}{2\gamma^2\beta^2}$. Each Monte-Carlo simulation takes $O(m+n)$ running time in constructed graph $\widetilde{G}$. Thus, we have a (γ, δ)-Estimation of $f_c(\vec{x}|G)$ in $O\left(\frac{(m+n)\alpha^2 \ln(2/\delta)}{2\gamma^2\beta^2}\right)$ running time. □

Remark 2. *From the Lemma 2 and Theorem 2, we can know that computing $f_c(\vec{x})$ on G is equivalent to compute $f_d(\widetilde{V} - V)$ on constructed graph $\widetilde{G}$, which give us an efficient technique to estimate the value $f_c(\vec{x})$ by use of Monte-Carlo simulations.*

3.3.2 Modularity of Objective Functions

In order to address CAM problem, a intuitive method is to use the greedy algorithm that can obtain a constant approximation ratio depended on the diminishing return property. We say that A set function $f : 2^V \to \mathbb{R}$ is monotone if $f(S) \leq f(T)$ for all $S \subseteq T \subseteq V$, and submodular if $f(S \cup \{u\}) - f(S) \geq f(T \cup \{u\}) - f(T)$ for all $S \subseteq T \subseteq V$ and $u \in V \backslash T$. Conversely, if $f(S \cup \{u\}) - f(S) \leq f(T \cup \{u\}) - f(T)$ for all $S \subseteq T \subseteq V$ and $u \in V \backslash T$, we say f is supermodular. Soma *et al.* (Soma and Yoshida, 2015) extended the submodularity and the diminishing return property to functions defined on the lattice, that is referred to as the DR-submodular property. To our CAM problem, for two vectors $x, y \in \mathcal{X}$, a function $g : \mathcal{X} \to \mathbb{R}$ is monotone if $g(\vec{x}) \leq g(\vec{y})$ for all $\vec{x} \leq \vec{y}$, and DR-submodular if $g(\vec{x}+t\vec{e_i}) - g(\vec{x}) \geq g(\vec{y}+t\vec{e_i}) - g(\vec{y})$ for all $\vec{x} \leq \vec{y}$ and $i \in [d]$. Conversely, if $g(\vec{x}+t\vec{e_i}) - g(\vec{x}) \leq g(\vec{y}+t\vec{e_i}) - g(\vec{y})$ for all $\vec{x} \leq \vec{y}$ and $i \in [d]$, we say g is DR-supermodular. Unfortunately, the objective function of CAM problem is not DR-submodular and DR-supermodular.

Remark 3. *Here, we assume that the strategy functions $h_u(\vec{x})$ for each $u \in V$ are monotone and DR-submodular. It is obvious that the probability of a user agreeing to be a seed increases with more investment. However, Endless additional investment does not mean that the activation probability will increase endlessly as well, namely its marginal gain is non-increasing. Therefore, it is valid to consider $h_u(\cdot)$ is a DR-submodular function.*

Theorem 3.3.5. *$f_c(\cdot)$ is monotone but not DR-submodular under the IC model and the LT model.*

Proof. We prove by a counterexample, consider graph $G = (V, E)$, $V = \{v_1, v_2, v_3, v_4\}$ and $E = \{(v_1, v_2), (v_2, v_3), (v_4, v_3)\}$. By setting $\mathcal{X} = \{0, 1\}^4$ and $h_v(\vec{x}) = x_v$, we have $h_v(\vec{x})$ is monotone and DR-submodular. The activation probabilities in IC model and weights in LT model of $\{(v_1, v_2), (v_4, v_3)\}$ are set to be 1, but $\{(v_2, v_3)\}$ is 0. The activity strengths are all set to be 1. Let $\vec{x} = (0, 0, 0, 0)$ and $\vec{y} = (0, 0, 0, 1)$, we have $f_c(\vec{x}) = 0$, $f_c(\vec{x} + e_1) = 1$, $f_c(\vec{y}) = 1$ and $f_c(\vec{y} + e_1) = 3$. That is $f_c(\vec{x} + e_1) - f_c(\vec{x}) < f_c(\vec{y} + e_1) - f_c(\vec{y})$ where $\vec{x} \leq \vec{y}$. Therefore, $f_c(\cdot)$ is not DR-submodular. □

In (Wang et al., 2017), they explained the reason why $f_d(\cdot)$ is not submodular as the "combination effect" between the new activated node with existing activated node. It can be extended to $f_c(\cdot)$ naturally.

Theorem 3.3.6. *$f_c(\cdot)$ is monotone but not DR-supermodular under the IC model and the LT model.*

Proof. We prove by a counterexample, consider graph $G = (V, E)$, $V = \{v_1, v_2, v_3, v_4\}$ and $E = \{(v_2, v_1), (v_2, v_3), (v_3, v_4)\}$. By setting $\mathcal{X} = \{0, 1\}^4$ and $h_v(\vec{x}) = x_v$, we have $h_v(\vec{x})$ is monotone and DR-submodular. The activation probabilities in IC model, weights in LT model and activity strengths are all set to be 1. Let $\vec{x} = (0, 0, 0, 0)$ and $\vec{y} = (0, 0, 1, 0)$, we have $f_c(\vec{x}) = 0$, $f_c(\vec{x} + e_2) = 3$, $f_c(\vec{y}) = 1$ and $f_c(\vec{y} + e_2) = 3$. That is $f_c(\vec{x} + e_2) - f_c(\vec{x}) > f_c(\vec{y} + e_2) - f_c(\vec{y})$ where $\vec{x} \leq \vec{y}$. Therefore, $f_c(\cdot)$ is not DR-supermodular. □

3.4 Upper and Lower Bound

In this section, we design an upper bound and a lower bound for our objective function $f_c(\cdot)$, and discuss their DR-submodularity and computability.

3.4.1 Bounds Definition

According to the activity function of CAM problem, Equation (5), in order to get an upper bound and a lower bound of $f_c(\cdot)$, we firstly need to get both bounds of the objective function $f_d(\cdot)$ for DAM. Wang *et al.* (Wang et al., 2017) pointed out that the non-submodularity of $f_d(\cdot)$ is derived from the "combination effect". Thus, for a lower bound, we only consider those edges whose two endpoints can be influenced by the cascade from the same seed node. For an edge $(u, v) \in E$, in a realization, we count it only if there is at least one node in S can reach u and v simultaneously. We denote by $\underline{f_d}$ the lower bound of f_d. That is,

$$\underline{f_d}(S) = \mathbb{E}\left[\sum_{(u,v)\in\bigcup_{x\in S} E[I(x)]} A_{uv}\right], \tag{3.7}$$

where $E[I(x)]$ is the edges of induced subgraph by activated node set $I(x)$. Given a seed set S, we have $\underline{f_d}(S) \leq f_d(S)$ because it neglects those edges whose endpoints can not be activated by the different seed nodes. Then, we denote by $\overline{f_d}$ the upper bound of f_d. That is,

$$\overline{f_d}(S) = \mathbb{E}\left[\sum_{u\in V[I(S)]} \sum_{v\in N(u)} \frac{A_{uv}}{2}\right], \tag{3.8}$$

where $V[I(S)]$ is the nodes of induced subgraph by activated node set $I(S)$. Given a seed set S, we have $\overline{f_d}(S) \geq f_d(S)$ because we consider each active node contributes to half of activity strength associated to those edges connected to it. Thus, for each edge, it is not mandatory to require both of its endpoints are activated.

According to the above bounds of f_d, we can obtain the upper bound and lower bound of the activity function of CAM problem by the same way. From Equation (6), we denote

by $\underline{f_c}$ the lower bound of f_c, that is,

$$\underline{f_c}(\vec{x}) = \sum_{S \subseteq V} \underline{f_d}(S) \cdot \prod_{u \in S} h_u(\vec{x}) \cdot \prod_{v \in V \setminus S} (1 - h_v(\vec{x})) \tag{3.9}$$

denote by $\overline{f_c}$ the upper bound of f_c, that is,

$$\overline{f_c}(\vec{x}) = \sum_{S \subseteq V} \overline{f_d}(S) \cdot \prod_{u \in S} h_u(\vec{x}) \cdot \prod_{v \in V \setminus S} (1 - h_v(\vec{x})). \tag{3.10}$$

Given a marketing strategy $\vec{x}$, we have $\underline{f_c}(\vec{x}) \leq f_c(\vec{x}) \leq \overline{f_c}(\vec{x})$ because $\underline{f_c}(\vec{x})$ (resp. $\overline{f_c}(\vec{x})$) is the linear combination of $\underline{f_d}(S)$ (resp. $\overline{f_d}(S)$). Thus, we can conclude that $\underline{f_d}(S) \leq f_d(S) \leq \overline{f_d}(S)$ means $\underline{f_c}(S) \leq f_c(S) \leq \overline{f_c}(S)$.

3.4.2 Properties of the Bounds

Lu *et al.* (Lu et al., 2015) provided us with an idea where we can obtain an approximate solution of CAM problem by maximizing its the upper bound and lower bound. As we know, by setting $\mathcal{X} = \{0, 1\}^n$ and $h_v(\vec{x}) = x_v$, the CAM problem can be reduced to DAM problem. Similarly, maximizing the $\underline{f_c}(\vec{x})$ (resp. $\overline{f_c}(\vec{x})$) can also be reduced maximixing the $\underline{f_d}(S)$ (resp. $\overline{f_d}(S)$) under this special case, which inherits its NP-hardness. Because of maximizing the $\underline{f_d}(\cdot)$ and $\overline{f_d}(\cdot)$ is NP-hard (Wang et al., 2017), it is natural to have

Theorem 3.4.1. *Maximizing the lower bound $\underline{f_c}(\cdot)$ is NP-hard under the IC model and the LT model.*

Theorem 3.4.2. *Maximizing the upper bound $\overline{f_c}(\cdot)$ is NP-hard under the IC model and the LT model.*

Even though that, the lower bound $\underline{f_d}(\cdot)$ and the upper bound $\overline{f_d}(\cdot)$ of DAM are submodular.

Lemma 3.4.3 ((Wang et al., 2017))**.** *The lower bound $\underline{f_d}(\cdot)$ is monotone and submodular, but computing it given a seed set S is #P-hard under the IC model and the LT model.*

Lemma 3.4.4 ((Wang et al., 2017)). *The upper bound $\overline{f_d}(\cdot)$ is monotone and submodular, but computing it given a seed set S is #P-hard under the IC model and the LT model.*

Then, the submodularity of $\underline{f_d}(\cdot)$ (resp, $\overline{f_d}(\cdot)$) can be correlated to the DR-submodularity of $\underline{f_c}(\cdot)$ (resp, $\overline{f_c}(\cdot)$). Let us look at the following Lemma:

Lemma 3.4.5. *Given a set function $f : 2^V \to \mathbb{R}$ and a function $g : \mathcal{X} \to \mathbb{R}$, they satisfies that*

$$g(\vec{x}) = \sum_{S \subseteq V} f(S) \cdot \prod_{u \in S} h_u(\vec{x}) \cdot \prod_{v \in V \setminus S} (1 - h_v(\vec{x})). \tag{3.11}$$

When $h_u(\vec{x})$ for each $u \in V$ are monotone and DR-submodular, if $f(\cdot)$ is monotone and submodular, then $g(\cdot)$ is monotone and DR-submodular.

Proof. This lemma is an indirect corollary from the section 7 of (Kempe et al., 2015), but there is a typo over there, and we fix and rearrange here. We denote $\alpha(u) = h_u(\vec{x}+t\vec{e_j}) - h_u(\vec{x})$ and $\beta(u, S) = \prod_{i<u, i \in S} h_i(\vec{x}+t\vec{e_j}) \cdot \prod_{i<u, i \notin S}(1-h_i(\vec{x}+t\vec{e_j})) \cdot \prod_{i<u, i \in S} h_i(\vec{x}) \cdot \prod_{i<u, i \notin S}(1-h_i(\vec{x}))$. Thus, we have $g(\vec{x} + t\vec{e_i}) - g(\vec{x}) = \sum_{S \subseteq V} f(S) \cdot (\prod_{u \in S} h_u(\vec{x} + t\vec{e_j}) \cdot \prod_{u \in V \setminus S}(1 - h_u(\vec{x} + t\vec{e_j})) - \prod_{u \in S} h_u(\vec{x}) \cdot \prod_{u \in V \setminus S}(1 - h_u(\vec{x}))) = \sum_{S \subseteq V} f(S) \cdot (\sum_{u \in S} \alpha(u) \cdot \beta(u, S) - \sum_{u \in V \setminus S} \alpha(u) \cdot \beta(u, S)) = \sum_{u \in V}(\alpha(u) \cdot \sum_{S: u \in V \setminus S}(f(S \cup \{u\}) - f(S)) \cdot \beta(u, S))$. Then, we study the difference $(g(\vec{x}+t\vec{e_i}) - g(\vec{x})) - (g(\vec{y}+t\vec{e_i}) - g(\vec{y}))$ for $\vec{x} \leq \vec{y}$, and show it is non-negative, whose techniques are similar to the section 7 of (Kempe et al., 2015). □

Based on Lemma 3, Lemma 4 and Lemma 5, the following theorems can be introduced directly, that is

Theorem 3.4.6. *The lower bound $\underline{f_c}(\cdot)$ is monotone and DR-submdoualr, but computing it given a marketing strategy $\vec{x}$ is #P-hard under the IC model and the LT model.*

Theorem 3.4.7. *The upper bound $\overline{f_c}(\cdot)$ is monotone and DR-submdoualr, but computing it given a marketing strategy $\vec{x}$ is #P-hard under the IC model and the LT model.*

Algorithm 1: lattice-Greedy $(g, \mathcal{X}, k)$

Initialize: $\vec{x} = 0$ and $c = 0$;
while $c < k$ **do**
    $i^* \leftarrow \arg\max_{i \in [d]} (g(\vec{x} + t\vec{e}_i) - g(\vec{x}))$;
    $\vec{x} \leftarrow \vec{x} + t\vec{e}_{i^*}$;
    $c \leftarrow c + t$;
return $\vec{x}$;

Given a marketing strategy $\vec{x}$, how can we compute the value of $\underline{f_c}(\vec{x})$ and $\overline{f_c}(\vec{x})$ effectively. The same as before, Equation (8), we create a constructed graph $\widetilde{G} = (\widetilde{V}, \widetilde{E})$. According to Remark 2, computing $\underline{f_c}(\vec{x})$ (resp, $\overline{f_c}(\vec{x})$) is equivalent to compute $\underline{f_d}(\widetilde{V} - V|\widetilde{G})$ (resp, $\overline{f_d}(\widetilde{V} - V|\widetilde{G})$). They can be done by user of Monte-Carlo simulations.

3.5 Algorithms

Given a function g on lattice $\mathcal{X} = \{0, t, 2t, \cdots\}^d$ and a budget k, the lattice-Greedy algorithm is shown in Algorithm 1. If this function g is monotone and DR-submodular, Algorithm 1 returns a solution that achieves a $(1 - 1/e)$-approximation (Nemhauser et al., 1978). The idea of lattice-Greedy algorithm is to find the component that has the largest marginal gain, and then allocate one unit t (lattice granularity) to this coordinate until the budget is exhausted. In our CAM problem, it is #P-hard to compute the lower bound $\underline{f_c}(\vec{x})$ and the upper bound $\overline{f_c}(\vec{x})$ in IC model and LT model. Thus, Algorithm 1 can give us a $(1 - 1/e - \varepsilon)$-approximate solution by use of Monte Carlo simulations. However, the efficiency of Monte-Carlo simulations is very low, so it is not scalable. In this section, we propose a sampling technique for these objective functions such that our CAM problem is scalable based on reverse influence sampling (RIS) (Borgs et al., 2014). Then, we adapt Influence Maximization with Martingale (IMM) (Tang et al., 2015) algorithm and combine it with sandwich approximation framework to solve our lattice-based problem.

3.5.1 Sampling techniques

Given a social network $G = (V, E)$, an diffusion model (IC/LT model), and a seed set S, let $\mathrm{g} = (V, E_{\mathrm{g}})$ be a realization sampled from a distribution, Equation (1) or Equation (2), denoted by $\mathrm{g} \sim G$. We denote by $R_{\mathrm{g}}(S)$ the set of nodes that are reachable from at least one node in S through E_{g} and $R_{\mathrm{g}^T}(v)$ the reverse reachable set (RR-Set) (Tang et al., 2014) for node v in g, which is a set composed of all nodes that can reach v through E_{g}. Let (u, v) be a edge sampled from the probability distribution A_{uv}/T where $T = \sum_{(u,v)\in E} A_{uv}$, denoted by $(u, v) \sim E$. Then, a random edge sampling (RE-sampling) μ can be defined as follows:

1. Initialize $\mu = (\emptyset, \emptyset)$
2. Select an edge $(u, v) \in E$ with probability A_{uv}/T
3. Generate a realization g from G according to the IC/LT model
4. Let $N_1 = R_{\mathrm{g}^T}(u)$ and $N_2 = R_{\mathrm{g}^T}(u)$
5. Let $\mu = (N_1, N_2)$
6. Return μ

Given a marketing strategy $\vec{x}$, to estimate $f_c(\vec{x})$, we have the following results, that is,

Theorem 3.5.1. *Given $G = (V, E)$ and a marketing strategy $\vec{x} \in \mathcal{X}$, we have*

$$f_c(\vec{x}) = T \cdot \mathbb{E}_{\mu=(N_1,N_2)} \left[\mathcal{H}(N_1) \cdot \mathcal{H}(N_2)\right], \tag{3.12}$$

where μ is a RE sampling, $T = \sum_{(u,v)\in E} A_{uv}$ and $\mathcal{H}(N_1) = 1 - \prod_{s\in N_1}(1 - h_s(\vec{x}))$.

Proof. Given a marketing strategy $\vec{x} \in \mathcal{X}$, according to Equation (5), we can write $f_c(\vec{x})$ as:

$$\begin{aligned} f_c(\vec{x}) &= \mathbb{E}_{S\sim\vec{x}}[f_d(S)] \\ &= T \cdot \mathbb{E}_{S\sim\vec{x},\mu=(N_1,N_2)}[\mathbb{I}(S \cap N_1 \neq \emptyset \wedge S \cap N_2 \neq \emptyset)] \\ &= T \cdot \mathbb{E}_{\mu=(N_1,N_2)} \left[\Pr_{S\sim\vec{x}}[S \cap N_1 \neq \emptyset \wedge S \cap N_2 \neq \emptyset)]\right]. \end{aligned}$$

Here, the domain $N_1 \cup N_2$ can be considered as $(N_1 \cap N_2) \cup (N_1 \backslash N_2) \cup (N_2 \backslash N_1)$. Thus, we have

$$\begin{aligned} f_c(\vec{x}) &= T \cdot \mathbb{E}_{S\sim\vec{x},\mu=(N_1,N_2)} \Big[\Pr_{S\sim\vec{x}}[S \cap (N_1 \cap N_2) \neq \emptyset] \\ &\quad + \Pr_{S\sim\vec{x}}[S \cap (N_1 \cap N_2) = \emptyset] \cdot \Pr_{S\sim\vec{x}}[S \cap (N_1 \backslash N_2) \neq \emptyset] \\ &\quad \cdot \Pr_{S\sim\vec{x}}[S \cap (N_1 \backslash N_2) \neq \emptyset] \Big] \\ &= T \cdot \mathbb{E}_{S\sim\vec{x},\mu=(N_1,N_2)} \left[\mathcal{H}(N_1 \cap N_2) \right. \\ &\quad \left. + (1 - \mathcal{H}(N_1 \cap N_2)) \cdot \mathcal{H}(N_1 \backslash N_2) \cdot \mathcal{H}(N_2 \backslash N_1) \right], \end{aligned}$$

where $\mathbb{I}(\cdot)$ is the indicator function which is equal to 1 if $(\cdot)$ is true. Then, $\Pr_{S\sim\vec{x}}[S \cap N_1 \neq \emptyset]$ is the probability there is at least one node in N_1 activated as a seed, thus, we have $\Pr_{S\sim\vec{x}}[S \cap N_1 \neq \emptyset] = 1 - \prod_{s\in N_1}(1 - h_s(\vec{x})) = \mathcal{H}(N_1)$. □

Let $M = \{\mu_1, \mu_2, \cdots, \mu_\theta\}$ be a collection of θ independent RE-sampling, by Equation (14), we have

$$\begin{aligned} \hat{f}_c(\vec{x}) = \frac{T}{\theta} \sum_{\mu=(N_1,N_2),\mu\in M} & (\mathcal{H}(N_1 \cap N_2) \\ & + (1 - \mathcal{H}(N_1 \cap N_2)) \cdot \mathcal{H}(N_1 \backslash N_2) \cdot \mathcal{H}(N_2 \backslash N_1)) . \end{aligned} \tag{3.13}$$

According to Theorem 9, $\hat{f}_c(\vec{x})$ is an unbiased estimator of $f_c(\vec{x})$ for any fixed θ and it is not DR-submodular as well. Similarly, for the lower bound $\underline{f_c}(\vec{x})$, we have the following results, that is,

Theorem 3.5.2. *Given $G = (V, E)$ and a marketing strategy $\vec{x} \in \mathcal{X}$, we have*

$$\underline{f_c}(\vec{x}) = T \cdot \mathbb{E}_{\mu=(N_1,N_2)} \left[\mathcal{H}(N_1 \cap N_2) \right], \tag{3.14}$$

where μ is a RE sampling, $T = \sum_{(u,v)\in E} A_{uv}$ and $\mathcal{H}(N_1 \cap N_2) = 1 - \prod_{s\in N_1\cap N_2}(1 - h_s(\vec{x}))$.

Proof. Given a marketing strategy $\vec{x} \in \mathcal{X}$, according to Equation (11), we can write $\underline{f_c}(\vec{x})$ as

$$\begin{aligned}
\underline{f_c}(\vec{x}) &= \mathbb{E}_{S\sim\vec{x}}[\underline{f_d}(S)] \\
&= T \cdot \mathbb{E}_{S\sim\vec{x},\mu=(N_1,N_2)}[\mathbb{I}(S \cap (N_1 \cap N_2) \neq \emptyset)] \\
&= T \cdot \mathbb{E}_{\mu=(N_1,N_2)}\left[\Pr_{S\sim\vec{x}}[(S \cap (N_1 \cap N_2) \neq \emptyset)]\right] \\
&= T \cdot \mathbb{E}_{\mu=(N_1,N_2)}\left[\mathcal{H}(N_1 \cap N_2)\right],
\end{aligned}$$

where $\mathbb{I}(\cdot)$ is the indicator function which is equal to 1 if $(\cdot)$ is true. Then, $\Pr_{S\sim\vec{x}}[S \cap (N_1 \cap N_2) \neq \emptyset]$ is the probability there is at least one node in $N_1 \cap N_2$ activated as a seed because it requires that the endpoints of an edge can be activated by the same seed node, thus, we have $\Pr_{S\sim\vec{x}}[S \cap (N_1 \cap N_2) \neq \emptyset] = 1 - \prod_{s\in(N_1\cap N_2)}(1 - h_s(\vec{x})) = \mathcal{H}(N_1 \cap N_2)$. □

By Equation (16), we have

$$\hat{\underline{f}}_c(\vec{x}) = \frac{T}{\theta} \sum_{\mu=(N_1,N_2),\mu\in M} \left(\mathcal{H}(N_1 \cap N_2)\right). \tag{3.15}$$

For the upper bound $\overline{f_c}(\vec{x})$, the sampling technique is a little different. Shown as Equation (10), the upper bound is a weighted influence maximization on lattice. Let u be a node sampled from the probability distribution $w(u)/W$ where $w(u) = \sum_{v\in N(u)} A_{uv}/2$ and $W = \sum_{u\in V} w(u)$, denoted by $u \sim V$. Then, a random node sampling (RN-sampling) ν can be defined as follows:

1. Initialize $\nu = (\emptyset, \emptyset)$
2. Select an node $u \in V$ with probability $w(u)/W$
3. Generate a realization g from G according to the IC/LT model
4. Let $\nu = R_{\mathrm{g}^T}(u)$
5. Return ν

Given a marketing strategy $\vec{x}$, to estimate $\overline{f_c}(\vec{x})$, we have the following results, that is,

Theorem 3.5.3. *Given $G = (V, E)$ and a marketing strategy $\vec{x} \in \mathcal{X}$, we have*

$$\overline{f_c}(\vec{x}) = W \cdot \mathbb{E}_\nu \left[\mathcal{H}(\nu)\right], \tag{3.16}$$

where ν is a RN sampling, $W = \sum_{u \in V} w(u)$ and $\mathcal{H}(\nu) = 1 - \prod_{s \in \nu}(1 - h_s(\vec{x}))$.

Proof. Given a marketing strategy $\vec{x} \in \mathcal{X}$, according to Equation (18), we can write $\overline{f_c}(\vec{x})$ as $\overline{f_c}(\vec{x}) = \mathbb{E}_{S\sim\vec{x}}[\overline{f_d}(S)] = W \cdot \mathbb{E}_{S\sim\vec{x},\nu}[\mathbb{I}(S \cap \nu \neq \emptyset)] = W \cdot \mathbb{E}_\nu[\Pr_{S\sim\vec{x}}[(S \cap \nu \neq \emptyset)]] = W \cdot \mathbb{E}_\nu\left[\mathcal{H}(\nu)\right]$. Then, $W \cdot \mathbb{E}_{S\sim\vec{x},\nu}[\mathbb{I}(S \cap \nu \neq \emptyset)]$ can be inferred from the proof proposed in (Nguyen et al., 2016a) and $\Pr_{S\sim\vec{x}}[S \cap \nu \neq \emptyset]$ is the probabilty there is at least one node in ν activated as a seed, thus, we have $\Pr_{S\sim\vec{x}}[S \cap \nu \neq \emptyset] = 1 - \prod_{s\in\nu}(1 - h_s(\vec{x})) = \mathcal{H}(\nu)$. □

Let $N = \{\nu_1, \nu_2, \cdots, \nu_\theta\}$ be a collection of θ independent RN-sampling, by Equation (18), we have

$$\overline{\hat{f}_c}(\vec{x}) = \frac{W}{\theta} \sum_{\nu \in N} \left(\mathcal{H}(\nu)\right). \tag{3.17}$$

According to Theorem 10 and Theorem 11, $\underline{\hat{f}_c}(\vec{x})$ and $\overline{\hat{f}_c}(\vec{x})$ is an unbiased estimator of $\underline{f_c}(\vec{x})$ and $\overline{f_c}(\vec{x})$ for any fixed θ and they are monotone and DR-submodular. Based on that, we can design our algorithms with a valid approximation ratio.

3.5.2 Modified IMM on Lattice

The unbiased estimators of our objective functions have been obtained in last subsection, here, we extend the IMM algorithm (Tang et al., 2015), the state-of-the-art method for the IM problem, to design the solutions of lower bound and upper bound of our CAM problem. The core idea of IMM on IM problem: produce enough random reverse reachable set (Random RR-Set), where the node is selected uniformly and randomly, and then find the maximum coverage under the cardinality constraint by use of greedy algorithm. The IMM process can be divided into two stages as follows:

Algorithm 2: lattice-Greedy $(\underline{\hat{f}_c}(\overline{\hat{f}_c}), M(N), \mathcal{X}, k)$

Initialize: $\vec{x}^\circ = 0$ and $c = 0$;
while $c < k$ **do**
  $i^\circ \leftarrow \arg\max_{i\in[d]} \left(\underline{\hat{f}_c}(\overline{\hat{f}_c})(\vec{x}^\circ + t\vec{e}_i) - \underline{\hat{f}_c}(\overline{\hat{f}_c})(\vec{x}^\circ)\right)$;
  $\vec{x}^\circ \leftarrow \vec{x}^\circ + t\vec{e}_{i^\circ}$;
  $c \leftarrow c + t$;
return $\vec{x}^\circ$;

1. Sampling Random RR-Sets: This stage generates enough random RR-set iteratively and independently and put them into $\mathcal{R}$ until satisfying a certain stopping condition.

2. Node selection: This stage adopts standard greedy method to drive a size-k seed set that covers sub-maximum number of RR-Sets in $\mathcal{R}$.

Extended to our problem, we generate enough RE-sampling for lower bound or RN-sampling for upper bound first, then the lattice-greedy algorithm on these RE-sampling or RN-sampling is adopted to get the sub-optimal strategy marketing $\vec{x}$. Let us introduce the node selection first. Let $M = \{\mu_1, \mu_2, \cdots, \mu_\theta\}$ be a collection of θ independent RE-sampling and $N = \{\nu_1, \nu_2, \cdots, \nu_\theta\}$ be a collection of θ independent RN-sampling. The node selection is shown in Algorithm 2, which is a $(1-1/e)$-approximate solution to the estimator of upper and lower bound.

In the first stage, we can use the sampling procedure similar to IMM, but need some modifications. For the lower bound, these modifications are: (1) we replace the number of node n with T, where $T = \sum_{(u,v)\in E} A_{uv}$; (2) we use lattice-greedy algorithm, Algorithm 2, on RE-sampling instead of greedy algorithm on RR-set; and (3) we replace $\log\binom{n}{k}$ with $\min(kt^{-1}\log d, d\log(kt^{-1}))$ in the two parameters λ' and λ^* (Chen et al., 2018). We have

$$\alpha = \sqrt{\ell \log T + \log 2} \tag{3.18}$$

$$\beta = \sqrt{(1-1/e)(\min(kt^{-1}\log d, d\log(kt^{-1})) + \alpha^2)} \tag{3.19}$$

Algorithm 3: sampling-LB $(G, \underline{\hat{f}_c}, \mathcal{X}, k, \varepsilon, \ell)$

Initialize: $M = \emptyset$, $LB = 0$, $\varepsilon' = \sqrt{2}\varepsilon$;
Initialize: $M' = \emptyset$;
Let $\lambda' = (2 + \frac{2}{3}\varepsilon')(\min(Bt^{-1}\log d, d\log(Bt^{-1})) + \ell\log T + \log\log_2 T) \cdot T/\varepsilon'^2$;
Let $\lambda^* = 2T \cdot ((1 - \frac{1}{e}) \cdot \alpha + \beta)^2/\varepsilon^2$;
for $i = 1$ *to* $\log_2 T - 1$ **do**
    Let $y_i = T/2^i$;
    Let $\theta_i = \lambda'/y_i$, where λ' is defined above;
    while $|M| \leq \theta_i$ **do**
        $\mu \leftarrow$ RE-sampling (G);
        $M \leftarrow M \cup \{\mu\}$;
    $\vec{x}^\circ \leftarrow$ lattice-Greedy $(\underline{\hat{f}_c}, M, \mathcal{X}, k)$;
    if $\underline{\hat{f}_c}(\vec{x}^\circ) \geq (1 + \varepsilon') \cdot y_i$ **then**
        $LB \leftarrow \underline{\hat{f}_c}(\vec{x}^\circ)/(1 + \varepsilon')$;
        break;
$\theta \leftarrow \lambda^*/LB$;
while $|M'| \leq \theta$ **do**
    $\mu \leftarrow$ RE-sampling (G);
    $M' \leftarrow M' \cup \{\mu\}$;
return M';

Algorithm 4: IMM-LB $(G, \underline{\hat{f}_c}, \mathcal{X}, k, \varepsilon, \ell)$

$M' \leftarrow$ sampling-LB $(G, \underline{\hat{f}_c}, \mathcal{X}, k, \varepsilon, \ell)$;
$\vec{x}_L \leftarrow$ lattice-Greedy $(\underline{\hat{f}_c}, M', \mathcal{X}, B)$;
return $\vec{x}_L$;

Then, the sampling procedure for lower bound, sampling-LB, can be shown in Algorithm 10, where ε is accuracy and ℓ is confidence. Chen has told us that there is an issue (Chen, 2018) in original IMM algorithm (Tang et al., 2015) and gave us two workarounds (Chen et al., 2018). We adopt the first workaround, line 19 to 22 in Algorithm 3, that is more simple and straightforward. The IMM-LB algorithm is shown in Algorithm 4.

Theorem 3.5.4. *The solution $\vec{x}_L$ returned by Algorithm 4 is a $(1 - 1/e - \varepsilon)$-approximation of the upper bound of CAM problem with at least $1 - 1/T^\ell$ probability.*

Algorithm 5: sampling-UB $(G, \overline{\hat{f}_c}, \mathcal{X}, k, \varepsilon, \ell)$

Initialize: $N = \emptyset$, $LB = 0$, $\varepsilon' = \sqrt{2}\varepsilon$;
Initialize: $N' = \emptyset$;
Let $\lambda' = (2 + \frac{2}{3}\varepsilon')(\min(Bt^{-1}\log d, d\log(Bt^{-1}) + \ell\log W + \log\log_2 W) \cdot W/\varepsilon'^2$;
Let $\lambda^* = 2W \cdot ((1 - \frac{1}{e}) \cdot \alpha + \beta)^2/\varepsilon^2$;
for $i = 1$ *to* $\log_2 W - 1$ **do**
    Let $y_i = T/2^i$;
    Let $\theta_i = \lambda'/y_i$, where λ' is defined above;
    while $|N| \leq \theta_i$ **do**
        $\nu \leftarrow$ RN-sampling (G);
        $N \leftarrow N \cup \{\nu\}$;
    $\vec{x}^\circ \leftarrow$ lattice-Greedy $(\overline{\hat{f}_c}, N, \mathcal{X}, k)$;
    if $\overline{\hat{f}_c}(\vec{x}^\circ) \geq (1 + \varepsilon') \cdot y_i$ **then**
        $LB \leftarrow \overline{\hat{f}_c}(\vec{x}^\circ)/(1 + \varepsilon')$;
        break;
$\theta \leftarrow \lambda^*/LB$;
while $|N'| \leq \theta$ **do**
    $\nu \leftarrow$ RN-sampling (G);
    $N' \leftarrow N' \cup \{\nu\}$;
return N';

Algorithm 6: IMM-UB $(G, \overline{\hat{f}_c}, \mathcal{X}, k, \varepsilon, \ell)$

$N' \leftarrow$ sampling-UB $(G, \overline{\hat{f}_c}, \mathcal{X}, k, \varepsilon, \ell)$;
$\vec{x}_U \leftarrow$ lattice-Greedy $(\overline{\hat{f}_c}, N', \mathcal{X}, k)$;
return $\vec{x}_U$;

To the original problem, we have known that $\hat{f}_c(\vec{x})$ is an unbiased estimator of $f_c(\vec{x})$. Based on the collection M' generated in Algorithm 4, we can use it to get solution $\vec{x}_A$ by calling lattice-Greedy $(\hat{f}_c, M', \mathcal{X}, k)$, because they always rely on RE-sampling. Here, $\vec{x}_A$ is a heuristic solution, no any theoretical guarantee, to the CAM problem.

For the upper bound, the modifications are similar to that of lower bound, but (1) we replace the number of node n with W, where $W = \sum_{u \in V} w(u)$; and (2) we use lattice-greedy

algorithm, Algorithm 2, on RN-sampling. That is,

$$\alpha' = \sqrt{\ell \log W + \log 2} \tag{3.20}$$

$$\beta' = \sqrt{(1-1/e)(\min(kt^{-1}\log d, d\log(kt^{-1})) + \alpha'^2)} \tag{3.21}$$

Then, the sampling procedure for upper bound, sampling-UB, can be shown in Algorithm 5, where ε is accuracy and ℓ is confidence. The IMM-UB algorithm is shown in Algorithm 6 similarly.

Theorem 3.5.5. *The solution $\vec{x}_U$ returned by Algorithm 6 is a $(1-1/e-\varepsilon)$-approximation of the upper bound of CAM problem with at least $1-1/W^{\ell}$ probability.*

3.5.3 Sandwich Approximation Framework

To optimize non-submodular function, there is no universal technique to approximate it within constant approximation ratio. Lu *et al.* (Lu et al., 2015) provided a sandwich approximation framework to us, where a data-dependent approximation ratio can be obtained by approximating the upper bound and lower bound that are monotone and submodular. It can be extended to solve our monotone but not DR-submodular objective function. First, we get a $(1-1/e-\varepsilon)$-approximate solution to the lower bound by calling IMM-LB, during that, we record the immediate collection of RE-sampling M'. Then, we use this M' as the input of lattice-greedy to find a heuristic solution to the original problem. Finally, we get a $(1-1/e-\varepsilon)$-approximate solution to the upper bound by calling IMM-UB and return the best one to the original problem. It is shown in Algorithm 7. The result returned by Algorithm 7 can be guaranteed to have following approximation:

Theorem 3.5.6. *Let $\vec{x}_{sand}$ be the marketing strategy returned by Algorithm 7, then we have*

$$f_c(\vec{x}_{sand}) \geq \max\left\{\frac{f_c(\vec{x}_U)}{\overline{f_c}(\vec{x}_U)}, \frac{\underline{f_c}(\vec{x}_L^*)}{f_c(\vec{x}_A^*)}\right\}\frac{1-\gamma}{1+\gamma}\left(1-\frac{1}{e}-\varepsilon\right)f_c(\vec{x}_A^*), \tag{3.22}$$

where $\vec{x}_L^*$ is the optimal solution to maximize the lower bound and $\vec{x}_A^*$ is the optimal solution of the CAM problem.

Algorithm 7: Sandwich Approximation Framework

$\vec{x}_L \leftarrow$ IMM-LB $(G, \underline{\hat{f}_c}, \mathcal{X}, k, \varepsilon, \ell)$ // Record the M' returned by sampling-LB here;

$\vec{x}_A \leftarrow$ lattice-Greedy $(\hat{f}_c, M', \mathcal{X}, k)$;

$\vec{x}_U \leftarrow$ IMM-UB $(G, \overline{\hat{f}_c}, \mathcal{X}, k, \varepsilon, \ell)$;

$\vec{x}_{sand} \leftarrow \arg\max_{\vec{x}\in\{\vec{x}_L,\vec{x}_A,\vec{x}_U\}} \dot{f}_c(\vec{x})$, where $\dot{f}_c(\vec{x})$ can be computed by $\dot{f}_d(\widetilde{V} - V|\widetilde{G})$ on constructed graph $\widetilde{G}$ equivalently, shown as Remark 2.;

return $\vec{x}_{sand}$;

Proof. Let $\vec{x}_U^*$ be the optimal solution to maximize the upper bound. For the upper bound, we have

$$\begin{aligned} f_c(\vec{x}_U) &= \frac{f_c(\vec{x}_U)}{\overline{f_c}(\vec{x}_U)}\overline{f_c}(\vec{x}_U) \geq \frac{f_c(\vec{x}_U)}{\overline{f_c}(\vec{x}_U)}\left(1-\frac{1}{e}-\varepsilon\right)\overline{f_c}(\vec{x}_U^*) \\ &\geq \frac{f_c(\vec{x}_U)}{\overline{f_c}(\vec{x}_U)}\left(1-\frac{1}{e}-\varepsilon\right)\overline{f_c}(\vec{x}_A^*) \\ &\geq \frac{f_c(\vec{x}_U)}{\overline{f_c}(\vec{x}_U)}\left(1-\frac{1}{e}-\varepsilon\right) f_c(\vec{x}_A^*). \end{aligned}$$

For the lower bound, we have

$$\begin{aligned} f_c(\vec{x}_L) &\geq \underline{f_c}(\vec{x}_L) \geq \left(1-\frac{1}{e}-\varepsilon\right)\underline{f_c}(\vec{x}_L^*) \\ &\geq \frac{\underline{f_c}(\vec{x}_L^*)}{f_c(\vec{x}_A^*)}\left(1-\frac{1}{e}-\varepsilon\right) f_c(\vec{x}_A^*). \end{aligned}$$

Let $\vec{x}_{max} = \arg\max_{\vec{x}\in\{\vec{x}_L,\vec{x}_A,\vec{x}_U\}} f_c(\vec{x})$. That is,

$$f_c(\vec{x}_{max}) \geq \max\left\{\frac{f_c(\vec{x}_U)}{\overline{f_c}(\vec{x}_U)}, \frac{\underline{f_c}(\vec{x}_L^*)}{f_c(\vec{x}_A^*)}\right\}\left(1-\frac{1}{e}-\varepsilon\right) f_c(\vec{x}_A^*).$$

According to Theorem 2, $\dot{f}_c(\vec{x})$ is a (γ, δ)-Estimation of $f_c(\vec{x})$ given a marketing strategy $\vec{x}$. Then, $\vec{x}_{sand} = \arg\max_{\vec{x}\in\{\vec{x}_L,\vec{x}_A,\vec{x}_U\}} \dot{f}_c(\vec{x})$, if $\vec{x}_{sand} \neq \vec{x}_{max}$, we have $(1+\gamma) f_c(\vec{x}_{sand}) \geq (1-\gamma) f_c(\vec{x}_{max})$. Thus, the Inequality (24) is established. □

3.6 Experiment

In this section, we carry out several experiments on different datasets to validate the correctness and efficiency of our proposed algorithms. There are three datasets (Rossi and Ahmed, 2015) used in our experiments: (1) Dataset-1: a co-authorship network, co-authorship among scientists to publish papers about network science; (2) Dataset-2: a Wiki network, who-voteson-whom network which come from the collection Wikipedia voting; (3) Dataset-3: A collaboration netwook extracted from Arxiv General Relativity. We consider thes three real networks as directed graph, and the statistics information of the three datasets is represented in Table 3.2.

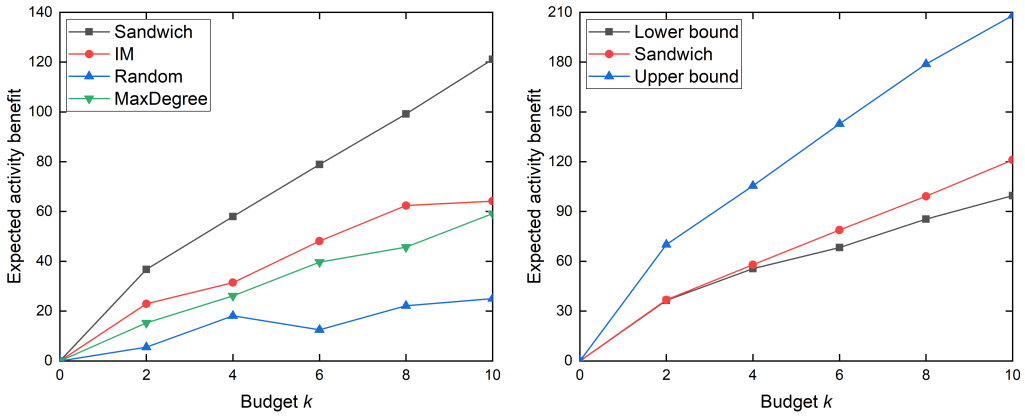
Table 3.2. The statistics of three datasets

Dataset	**n**	**m**	**Type**	**Average degree**
Dataset-1	0.4K	1.01K	directed	4
Dataset-2	1.0K	3.15K	directed	6
Dataset-3	5.2K	14.5K	directed	5

3.6.1 Experimental Settings

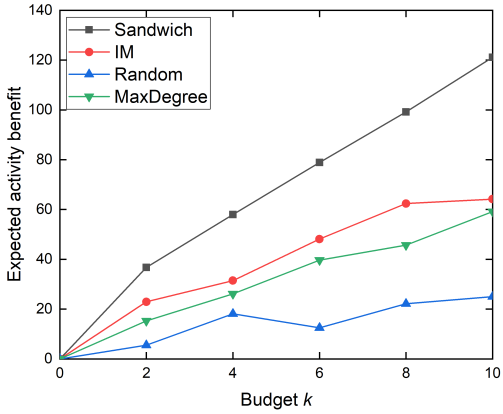
The diffusion model of our proposed experiments relies on IC model and LT model. Under the IC model, for each edge $(u, v) \in E$, the diffusion probability is set as $p_{uv} = 1/|N^-(v)|$. Under the the LT model, for each edge $e = (u, v)$, the weight is set as $b_{uv} = 1/|N^-(v)|$. This setting is widely used by prior works about influence maximization. Given a marketing strategy $\vec{x}$, for each node $u \in V$, we have a strategy function $h_u(\vec{x})$. Here, we consider the case: independent strategy activation (Chen et al., 2018), where each component $x_j \in \vec{x}$ attempts to activate u as seed independently. Then, we have

$$h_u(\vec{x}) = 1 - \prod_{j \in [d]} (1 - q_{uj}(x_j)), \tag{3.23}$$

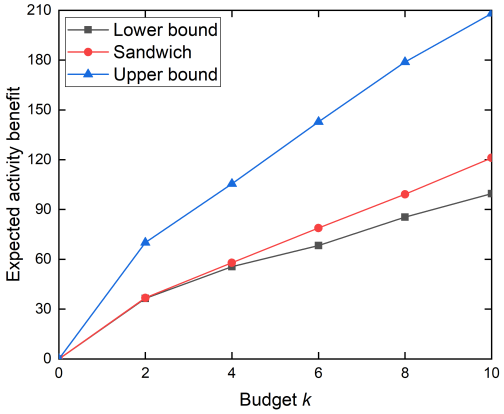


(a) Dataset-1, Performance

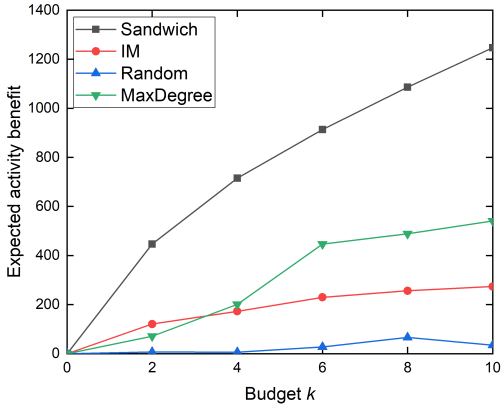
(b) Dataset-1, Sandwich



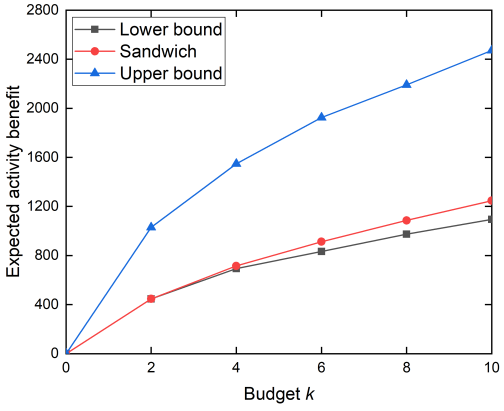
(c) Dataset-2, Performance



(d) Dataset-2, Sandwich

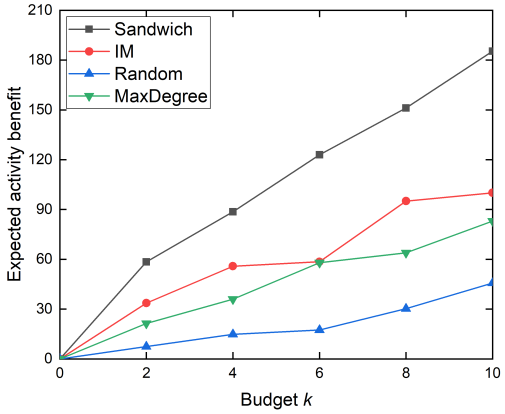


(e) Dataset-3, Performance

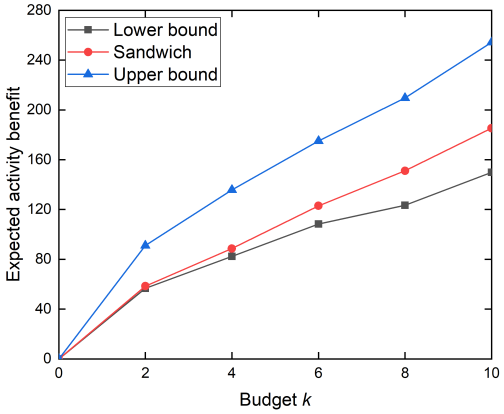


(f) Dataset-3, Sandwich

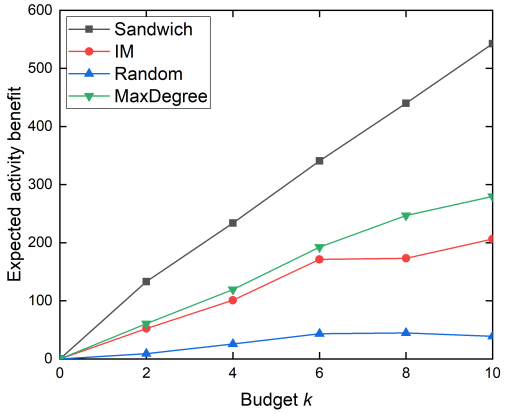
Figure 3.1. Under the IC model: left column is the performance comparison of different algorithms changes over budget k; right column is the result of sandwich approximation framework.



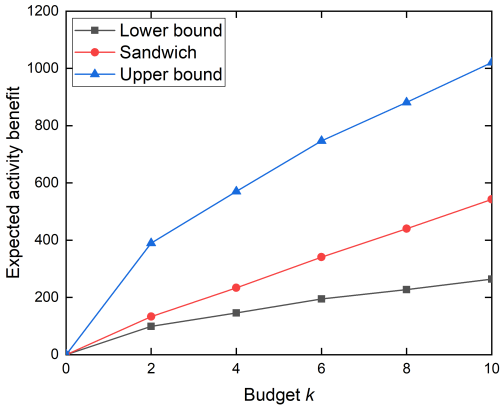
(a) Dataset-1, Performance



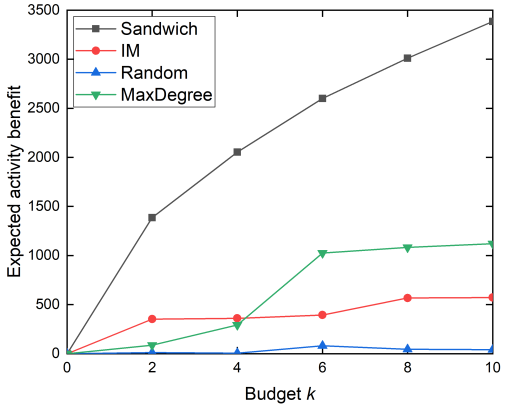
(b) Dataset-1, Sandwich



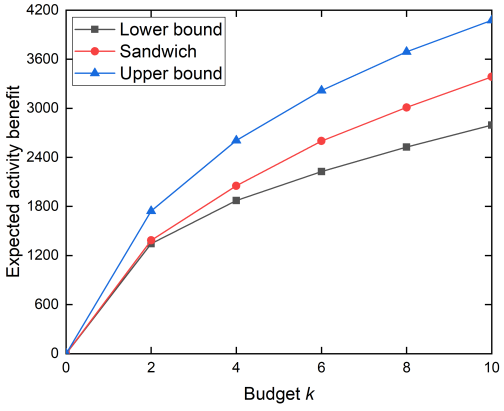
(c) Dataset-2, Performance



(d) Dataset-2, Sandwich



(e) Dataset-3, Performance



(f) Dataset-3, Sandwich

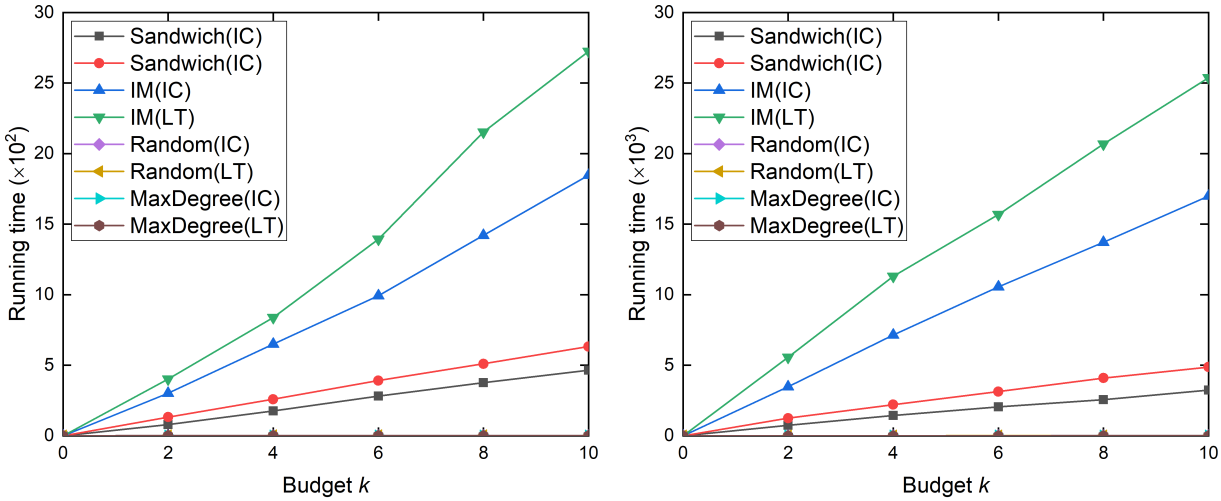
Figure 3.2. Under the LT model: left column is the performance comparison of different algorithms changes over budget k; right column is the result of sandwich approximation framework.

where strategy $j \in [d]$ activates u as seed with probability $q_{uj}(x_j)$. Chen *et al.* (Chen et al., 2018) pointed out $h_u(\vec{x})$ is monotone and DR-submodular if $q_{uj}(x_j)$ is monotone and concave for each $j \in [d]$ and each node $u \in V$. In this experiment, we test personalized marketing scenario (Yang et al., 2016), where strategy function is defined as $h_u(\vec{x}) = 2x_u - x_u^2$ and $\vec{x} = (x_1, x_2, \cdots, x_n)$. It means that the probability that activates node u as seed only depends on component x_u.

For our sandwich approximation framework, we set parameters of accuracy $\varepsilon = 0.1$, confidence $\ell = 1$ and granularity $t = 0.2$. Besides, we set activity strength $A_{uv} = 1$ for each edge $(u, v) \in E$. Then, we compare it with some commonly used baseline algorithms, which is summarized as follows: (1) IM: It returns the active nodes by lattice greedy algorithm to maximize the influence spread, and then computes the activity benefit. (2) MaxDegree: It selects the node with the highest outdegree under the budget k. (3) Random: It selects a node u randomly and increases its x_u by t until using up the budget k. Sandwich algorithm have been shown in Algorithm 7, and we implement IM algorithm by Monte-Carlo simulations, where the number of Monte-Carlo simulations is set as 200 for dataset-1, 400 for dataset-2, and 100 for dataset-3. Here, in order to test the running time of different algorithms, we do not use parallel acceleration in our implementations.

3.6.2 Experimental Results

Fig. 3.1 and Fig. 3.2 draw the performance achieved by our Sandwich method under the IC model and LT model, Algorithm 7, and other heuristic algorithms. Theoretically, our sandwich method can guarantee an approximate bound, but the others can not. From the left column of Fig. 3.1 and Fig. 3.2, the total activity benefit returned by our Sandwich method is always the best among all results returned by other algorithms. With the increasing size of dataset, the advantage of sandwich is more apparent. For IM and MaxDegree, which one is better? The answer is uncertain. For the dataset-1, IM is better than MaxDegree under



(a) Dataset-1 (b) Dataset-2

Figure 3.3. The running time comparisons among different algorithms under the IC model and the LT model.

the IC model and LT model. But for the dataset-2 and dataset-3, MaxDegree is better than IM. From the right column of Fig. 3.1 and Fig. 3.2, it is observed that the expected activity benefit returned by sandwich approximation framework lies in between its upper bound and lower bound. Until now, the correctness and effectiveness of our algorithms have been tested and validated.

As for its efficiency, let us look at Fig. 3.3. It draws the running time comparisons among different algorithms. From Fig. 3.3, the running time of Sandwich is lower than IM algorithm, which implies that our algorithm combined with sampling techniques improve time efficiency significantly. If they all adopt Monte-Carlo simulations to estimate objective value, Sandwich should be slower than IM because of its higher time complexity under the value oracle. Besides, the expected activity benefit and running time under the LT model is larger than that under the IC model, which is related to their different model features. For example, given a graph $G = (V, E)$, $V = \{v_1, v_2, v_3\}$ and $E = \{(v_2, v_1), (v_3, v_1)\}$, we assume $\{v_2, v_3\}$ are active, which attempt to activate node v_1 now. Under the IC model, the probability that activates node v_1 successfully is $1 - (1 - p_{v_2v_1})(1 - p_{v_3v_1})$, which is less

than $p_{v_2v_1} + p_{v_3v_1}$, this probability under the LT model. Then, consider the running time, Sandwich under the LT model is slower because its process of sampling RE(RN)-sampling is more time-consuming.

CHAPTER 4

BUDGET PROFIT MAXIMIZATION WITH COUPON ADVERTISEMENT [1]

Authors – Jianxiong Guo, Tiantian Chen and Weili Wu

The Computer Science Department, EC 31

The University of Texas at Dallas

800 West Campbell Road

Richardson, Texas 75080-3021

[1]

The online social platforms were developing quickly in the last decades and derived a series of technology giants, such as Facebook, Twitter, LinkedIn and Tencent. There are billion of people sharing their emotions and discussing current affairs in these platforms. There are more than 1.52 billion users active daily on Facebook and 321 million users active monthly on Twitter. The logic of social platforms can be represented as online social network (OSNs), which is a directed graph, including individuals and their relationship. The theory of Viral marketing was formulated by Domingos and Richardson (Domingos and Richardson, 2001) (Richardson and Domingos, 2002). By giving the most influential users free or coupon samples in social networks, it aims to make the follow-up adoptions maximized. Inspired by that, the concept of Influence Maximization (IM) was used as the spread of trust, advertisements or innovations (Kempe et al., 2003) (Chen et al., 2011) (Zhang et al., 2014). It was stated formulated formally by Kempe et al. (Kempe et al., 2003) as a discrete optimization problem: selects a subset of nodes with size constraint to make the expected number of follow-up adoptions (influence) maximized. Then, they proposed two classical diffusion models called IC model and LT model, and proved the IM problem is NP-hard and monotone submodular under both IC model and LT model. Since this seminal work, a large number of follow-up researches have been done on IM, which mainly concentrate on improve the running time of the greedy algorithm in IC/LT model and their variant model, such as (Chen et al., 2009) (Chen et al., 2010a) (Guo and Wu, 2019) (Guo et al., 2019) (Borgs et al., 2014) (Tang et al., 2014) (Tang et al., 2015). Besides, another branch of follow-up researches focuses on the variant problem of IM. Profit maximization (PM) (Lu and Lakshmanan, 2012) (Tang et al., 2016) is a kind of variant problem. Here, we need to consider some more complex factors, such as product price, cost, discount, coupon and their impact on diffusion.

Obviously, maximized influence does not mean the highest profit in a real scene when there are some marketing tools being adopted, such as coupons, rewards, free samples and so on. They are effective promotional tools to stimulate consumption and change the customers' purchase behavior. For example, in online social platforms, eBay and Amazon, those

sellers can prompt some influential customers to spread their products by giving coupons. In this paper, we think over such a marketing strategy that the initial clients are provided with coupons. We propose the problem of Budget Profit Maximization with Coupon Advertisement (BPMCA), which is based on three parameters: market price, cost and the value of coupon. From the perspective of merchant, coupon is not free since it lowers the price of product actually, thereby reducing the profit. Therefore, it does not bring higher profit by giving more initial clients coupons, which leads to the non-monotonicity of the objective function of BPMCA problem.

Most of existing researches about PM mainly consider the unconstrained situations. In other words, the number of initial adopters is unlimited (Lu and Lakshmanan, 2012) (Tang et al., 2016). It can be classified as the problem: Unconstrained Submodular Maximization (USM), where the best approximation ratio we can obtain is 1/2 by randomized double greedy algorithm proposed by Buchbinder et al. (Buchbinder et al., 2015). However, this is not completely in line with the real scene by two following reasons: (1) It is impossible to give every potential user a coupon, because for seller, they may not be able to get complete network topology information and the interconnections between users, who will be a potential client; (2) For advertisement, giving too many coupons is sometimes counterproductive, because this will make your company's products very cheap and lack competitiveness. In order to overcome above defect, we add the budget constraint to our problem, where the number of coupons is limited. Then, it can be classified as the problem: Submodular Maximization with Cardinality Constraint (SMCC), whose best approximation is an open question. With the latest research progress, the state-of-the-art algorithm to solve SMCC is Random Greedy (RG) and Continuous Double Greedy (CDG) (Buchbinder et al., 2014) Algorithm. But even with such algorithms, BPMCA problem is still not easy to be solved because the objective function of CDG is continuous and there are some details that are difficult to handle. In this paper, we give the discretization process and its implementation details of CDG. We

propose BPMCA-Framework combining the idea of RG and CGD. Besides, the ground-truth diffusion probabilities on edges cannot be determined accurately, even though there are some learning methods (Saito et al., 2008) (Goyal et al., 2010) existing to solve it. Because of the uncertainty, He et al. (He and Kempe, 2015) assumed for each edge, there exists an interval that the ground-truth probability lies in. Hence, we propose Robust Budget Profit Maximization with Coupon Advertisement (Robust-BPMCA) problem, which aims to maximize the worst ratio between the profit of any feasible seed set and the optimal seed set, and design LU-B-Framework to solve it. It first solves BPMCA problem on the maximum and minimum parameter vectors respectively, and selects the better one on the minimum parameter vectors as the result. Besides, the robustness of our algorithm can be improved further, we use uniform sampling to sample the edge probability. It shortens the parameter space, meaning that the uncertainty is reduced, and the robustness is improved. Our contribution in this paper are summarized as follows: (1) This is the first time to study PM under coupon and limited budget. We propose BPMCA problem and prove the objective function is negative, non-monotone and submodular; (2) We design the BPMCA-Framework, give the discretization process and implementation details, and achieve a dependent $h(G, k)$-approximation. (3) For Robust-BPMCA problem, we propose LU-B-Framework using BPMCA-Framework as subroutine, and prove get a dependent approximation ratio $h(G, k) \cdot \alpha(\Theta)$. To improve the robust ratio of BPMCA, we combine uniform sampling into LU-B-Framework and get a dependent $h(G, k) \cdot (1 - \epsilon)$ theoretical bound; (4) Our proposed algorithms are evaluated on real-world social networks, which verify the effectiveness and correctness of them.

4.1 Related Work

Domingos and Richardson (Domingos and Richardson, 2001) (Richardson and Domingos, 2002) was the first to study viral marketing and the value of customers in social networks. Kempe et al. (Kempe et al., 2003) studied IM as a discrete optimization problem and

generalized IC model and LT model to triggering model, who provided us with a greedy algorithm implemented by Monte Carlo simulation. A series of follow-up researches about IM mainly aimed to improve the efficiency of greedy algorithm (Chen et al., 2011) (Zhang et al., 2014) (Leskovec et al., 2007) (Goyal et al., 2011a), especially based on reverse influence sampling (Borgs et al., 2014) (Tang et al., 2014) (Tang et al., 2015). PM is an important variant problem of IM, whose related researches can be divided into two categories roughly. One branch focused on pricing strategies of the product. Arthur et al. (Arthur et al., 2009), Zhou et al. (Zhou et al., 2015) and Lu et al. (Lu et al., 2016) stated price setting from the perspective of game theory. Yang et al. (Yang et al., 2016) considered the optimal discount setting such that the subsequent adoptions can be maximized. Another branch, more related to us, is the problem of selecting high quality seed users (Lu and Lakshmanan, 2012) (Tang et al., 2016) such that maximizing profit. Lu et al. (Lu and Lakshmanan, 2012) extended the LT model to include prices and valuation, who used a heuristic unbudgeted greedy framework to solve this problem. Tang et al. (Tang et al., 2016) provided a strong approximation guarantee by use of the techniques of USM.

The robustness problem of IM has attracted researchers' attention recently. He et al. (He and Kempe, 2015) was the first one trying to consider that the uncertainty of diffusion probability affects the objective value of IM. Influence difference maximization was formulated, whose purpose was to find a seed set maximizing the difference of two influence under the different parameter value and see how large the difference is because of the uncertainty of parameter space. Jung et al. (Jung et al., 2012) proposed the IRIE algorithm that combines influence ranking and estimation for IM under the IC model, and showed it is more robust than others. Later, the robust IM problem was constructed by Chen et al. (Chen et al., 2016), which aims to maximize the worst ratio between influence function of given seeds and optimal solution. The main idea is similar to our robust analysis, but our objective function is total profit, and it is not monotone non-decreasing.

4.2 Problem Formulation

In this section, we formulate the BPMCA problem. From the aforementioned insights, we consider the simplest coupon scenario as follows: There are three parameters for each product the merchant wants to advertise, market price p, product cost c and coupon value b. Here, we need to require $0 \leq b \leq p - c$, which ensures the merchant does not lose money. Given a seed set S, we define $f(S|p, c, b)$ as the expected profit when diffusion terminates under the IC/LT model according to parameters p, c and b shown as above. When the context is clear, we would use $f(S)$ instead of $f(S|p, c, b)$. As we know, profit is equal to market price minus product cost. Thus, $f(S)$ can be expressed as

$$f(S) = (p - c) \cdot \sigma(S) - b \cdot |S| \tag{4.1}$$

$$= (p - c) \cdot (\sigma(S) - |S|) + (p - c - b) \cdot |S|. \tag{4.2}$$

After obtaining profit function, the problem of Budget Profit Maximization with Coupon Advertisement (BPMCA) is formulated as follows:

Problem 2 (BPMCA)**.** *Given a social network $G = (V, E)$, product parameter p, c and b, we aim to find a seed set $S \subseteq V$ and $|S| \leq k$ such that the expected profit $f(S|p, c, b)$ can be maximized under the IC/LT model.*

From Problem 1, BPMCA problem aims to find a optimal solution S^* such that

$$S^* = \arg \max_{S \subseteq V, |S| \leq k} f(S). \tag{4.3}$$

When setting $p = c + 1$ and $b = 0$, we have $f(S) = \sigma(S)$, therefore, BPMCA problem is NP-hard.

Theorem 4.2.1. *The objective function $f(\cdot)$ is non-negative submodular, but not monotone non-decreasing.*

Proof. Under the IC/LT model, $\sigma(S)$ is submodular from Theorem 1. According to Equation (4), the first term $(p-c)\cdot\sigma(S)$ is submodular and second term $b\cdot|S|$ is modular with respect to S, thus, $f(S)$ is submodular. Then, let us look at Equation (5), $(p-c)\cdot(\sigma(S)-|S|)\geq 0$ because of $\sigma(S)\geq|S|$ and $b\cdot|S|\geq 0$ because we require $0\leq b\leq p-c$. Therefore, we have $f(S)\geq 0$ for $S\subseteq V$.

To show monotonicity, let us consider a simple example under IC model. Given $G=(V,E)$, we suppose $V=\{u,v\}$, $E=(u,v)$ and $p_{uv}=1$. Here, $f(\{u\})=2\cdot(p-c)-b$ and $f(\{u,v\})=2\cdot(p-c)-2b$, so $f(\{u\})\geq f(\{u,v\})$ and $f(\cdot)$ is not monotone non-decreasing. □

4.3 Algorithm for BPMCA

From the last section, BPMCA problem is stated, and in this section, we try to give some effective solutions to it.

4.3.1 Preliminaries

We have known that the objective function of BPMCA is non-negative submodular, but not monotone non-decreasing. It can be classified to a classical problem: Submodular Maximization with Cardinality Constraint (SMCC). Given a non-negative submodular function $f(\cdot)$, SMCC aims to find a subset $|S|\leq k$ such that $f(\cdot)$ is maximized. With the latest works, there are many effective methods to solve SMCC problem, and the famous one, proposed by (Buchbinder et al., 2014), is Random Greedy (RG) Algorithm. This simple algorithm is a natural replacement for the classical hill-climbing algorithm proposed by Nemhauser et al. (Nemhauser et al., 1978), because it can obtain the same tight $(1-1/e)$-approximation for monotone non-decreasing objective function and same time complexity of $O(nk)$, provided that f is value oracle, but giving $(1/e)$-approximation for general submodular function (not necessarily monotone). However, given a seed set S, computing the exact value of f is #P-hard under the IC model and LT model (Chen et al., 2010a) (Chen et al., 2010b). To

Algorithm 8: RandomGreedy (G, f, k)

```
1 Initialize: S_0 ← ∅;
2 for i = 1 to k do
3 |   Let M_i ⊂ V\S_{i-1} be a subset of size k maximizing Σ_{u∈M_i} f(u|S_{i-1});
4 |   Let u_i be a uniformly random element from M_i;
5 |   if f(u_i|S_{i-1}) ≥ 0 then
6 |   |   S_i ← S_{i-1} ∪ {u_i};
7 return S_k;
```

estimate the value of f by Monte Carlo simulations, the time complexity of RG is $O(kmnr)$ under the IC model and LT model, where r is the number of Monte Carlo simulations.

The RG Algorithm is shown as algorithm 8, where the marginal gain is $f(u|S) = f(S \cup \{u\}) - f(S)$ above. It is a little different from the statement in (Buchbinder et al., 2014), here, we use the judgment statement in line 4 of Algorithm 8 to take the place of dummy assumption in (Buchbinder et al., 2014), dummy element whose marginal gain to any set is 0. In (Buchbinder et al., 2014), they assume there are $2k \leq n$ dummy elements in the ground set.

Lemma 4.3.1. *Given non-negative submodular function $f(\cdot)$ and cardinality k, the set produced by RandomGreedy algorithm is a $(1/e)$-approximate solution with $O(kmnr)$ time complexity.*

Even though RandomGreedy algorithm is simple and easy to be implemented, there is a fatal flaw because in extremely bad cases, it can only reach $1/e$ of the optimal value. Buchbinder et al. (Buchbinder et al., 2014) extended the double greedy method of (Buchbinder et al., 2012) to continuous setting, which can obtain a solution-dependent approximation ratio. It is better than RandomGreedy algorithm in the most cases. Let $\vec{x} = (x_1, x_2, ..., x_n)$ and $\vec{y} = (y_1, y_2, ..., y_n)$ be two n-dimensional vector and $\vec{x}, \vec{y} \in [0, 1]^V$. In our BPMCA problem, for each variable $x_u, y_u \in [0, 1]$, it means that we choose node u as the seed node with the

Algorithm 9: Continuous-DGreedy (G, F, k)

Initialize: $\vec{x}^0 \leftarrow \vec{0}$, $\vec{y}^0 \leftarrow \vec{1}$;
for *any time* $t \in [0,1]$ **do**
  for *each* $u \in V$ **do**
    $a_u \leftarrow \frac{\partial F(\vec{x}^t)}{\partial x_u}$, $b_u \leftarrow -\frac{\partial F(\vec{y}^t)}{\partial y_u}$;
    $a'_u(\ell) \leftarrow \max\{a_u - \ell, 0\}$, $b'_u(\ell) \leftarrow \max\{b_u + \ell, 0\}$;
    $\frac{\partial x_u}{\partial t}(\ell) \leftarrow \frac{a'_u}{a'_u + b'_u}$, $\frac{\partial y_u}{\partial t}(\ell) \leftarrow -\frac{b'_u}{a'_u + b'_u}$;
  Let ℓ' be a value such that $\sum_{u \in V} \frac{\partial x_u}{\partial t}(\ell) = k$;
  Let $\ell^* \leftarrow \max\{\ell', 0\}$;
  for *each* $u \in V$ **do**
    $\frac{\partial x_u}{\partial t} \leftarrow \frac{\partial x_u}{\partial t}(\ell^*)$, $\frac{\partial y_u}{\partial t} \leftarrow \frac{\partial y_u}{\partial t}(\ell^*)$;
return $\vec{x}^1 = \vec{y}^1$;

probability x_u, y_u. The Continuous Double Greedy (CDG) Algorithm is shown as Algorithm 9.

Definition 4.3.2 (Multilinear Extension)**.** *For $\vec{x} \in [0,1]^V$, we can get a random vector $\hat{\vec{x}} \in \{0,1\}^V$ by rounding each component of $\vec{x}$ to 1 with probability x_i or 0 otherwise. Let $S \subseteq V$ corresponding to the indicator vector $\hat{x}$, we have*

$$F(\vec{x}) = \mathbb{E}[f(\hat{\vec{x}})] = \sum_{S \subseteq V} f(S) \prod_{i \in S} x_i \prod_{j \in V \setminus S} (1 - x_j). \tag{4.4}$$

In Algorithm 9, we consider an oracle access to the multilinear extension F. The Continuous-DGreedy algorithm begins to run from time $t = 0$ to time $t = 1$ as a continuous manner. For any time $t \in [0,1]$, the two vector satisfies $\vec{x}^t \leq \vec{y}^t \in [0,1]^V$, besides, $\vec{x}^t$ increases and $\vec{y}^t$ decreases gradually. The algorithm meets the following invariants (Buchbinder et al., 2014) for each u at any time t:

1. $\sum_{u \in V} \frac{\partial x_u}{\partial t}(\ell') = k$
2. $\frac{\partial x_u}{\partial t}, -\frac{\partial y_u}{\partial t} > 0$ and $\frac{\partial x_u}{\partial t} - \frac{\partial y_u}{\partial t} = 1$

Thus, at the end of the execution, at time $t = 1$, we have $\vec{x}^1 = \vec{y}^1$ as the output of Algorithm 9.

4.3.2 Discretization of CDG Algorithm

Although we have obtained CDG Algorithm, it is not enough to solve our BPMCA problem. An implementation of the algorithm should be done further by discretization. Here, we need to discretize the time scale and balance the granularity of discretization and the error incurred. At the time t, we denote current vector $\vec{x}$ and $\vec{y}$ as $\vec{x}(t)$ and $\vec{y}(t)$. Thus, to compute $\nabla F(\vec{x}(t))$, we have

$$\frac{\partial F}{\partial x_i} = \mathbb{E}[f(\vec{\hat{x}})|\hat{x}_i = 1] - \mathbb{E}[f(\vec{\hat{x}})|\hat{x}_i = 0]. \tag{4.5}$$

Let $X(t), Y(t)$ be the set that contains each element i independently with probability $a_i(t)$ and $y_i(t)$, respectively. According to Equation (8), each element i from $\nabla F(\vec{x}(t))$ can be estimated as $a_i(t) = \mathbb{E}[f(X(t) \cup \{i\}) - f(X(t) \backslash \{i\})]$ and $\vec{a}(t) = \{a_1(t), a_2(t), ..., a_n(t)\}$. So far, the discretized process is formulated, called Discretized-CDG. Given time step Δt and sampling number λ, we have

1. Start with $t = 0$, $\vec{x}(0) = \vec{0}$, $\vec{y}(0) = \vec{1}$.

2. Let $X(t)$ contains each element i independently with probability $x_i(t)$, $Y(t)$ contains each element i independently with probability $y_i(t)$. For each element $i \in V$, we define $a_i(t) = \mathbb{E}[f(X(t) \cup \{i\}) - f(X(t) \backslash \{i\})]$ and $b_i(t) = \mathbb{E}[f(Y(t) \backslash \{i\}) - f(Y(t) \cup \{i\})]$, which can be obtained by taking the average of λ independent sampling of $X(t)$ and $Y(t)$.

3. For each element $i \in V$, we set $a'_i(t, \ell) = \max\{w_i(t) - \ell, 0\}$ and $b'_i(t, \ell) = \max\{z_i(t) + \ell, 0\}$.

4. For each element $i \in V$, we set $w_i(t,\ell) = a_i'(t,\ell)/(a_i'(t,\ell) + b_i'(t,\ell))$ and $z_i(t,\ell) = -b_i'(t,\ell)/(a_i'(t,\ell) + b_i'(t,\ell))$. Actually, we can compute $w_i(t,\ell)$ and $z_i(t,\ell)$ directly as

$$w_i(t,\ell) = \begin{cases} 1 & \ell < -b_i(t) \\ \frac{a_i(t)-\ell}{a_i(t)+b_i(t)} & -b_i(t) \le \ell \le a_i(t) \\ 0 & \ell > a_i(t) \end{cases}$$

and

$$z_i(t,\ell) = \begin{cases} 0 & \ell < -b_i(t) \\ -\frac{b_i(t)+\ell}{a_i(t)+b_i(t)} & -b_i(t) \le \ell \le a_i(t) \\ -1 & \ell > a_i(t) \end{cases}$$

5. Let ℓ' be a value such that $\sum_{i\in V} w_i(t,\ell) = k$.

6. Let $\ell^* \leftarrow \max\{\ell', 0\}$.

7. Update vector $\vec{x}$ and $\vec{y}$, let $\vec{x}(t+\Delta t) = \vec{x}(t) + \vec{w}(t,\ell^*) \cdot \Delta t$ and $\vec{y}(t+\Delta t) = \vec{y}(t) + \vec{z}(t,\ell^*) \cdot \Delta t$.

8. Increment $t = t + \Delta t$; If $t < 1$, go back to step (2) Otherwise, return $\vec{x}(1)$ or $\vec{y}(1)$.

At the time t, let $g(\ell) = \sum_{i\in V} w_i(t,\ell)$, in step (5), the solution of $g(\ell) = k$ cannot be computed directly. Fortunately, we observe that $\min g(\ell) = \sum_{i\in V} 0 = 0$ and $\max g(\ell) = \sum_{i\in V} 1 = n$, thus, for any $0 \le k \le n$, there exists a value ℓ' such that $g(\ell') = k$ if $g(\ell)$ is continuous, in other words, $a_i(t) + b_i(t) \neq 0$ for every $i \in V$. Then, we denote $\ell_{min} = \min_{i\in V}\{-b_i(t)\}$ and $\ell_{max} = \max_{i\in V}\{a_i(t)\}$, and the solution ℓ' satisfies $\ell' \in [\ell_{min}, \ell_{max}]$ definitely. Obviously, the $g(\ell)$ is monotone non-increasing in the interval $[\ell_{min}, \ell_{max}]$. Given a predefined threshold (precision) β, we can use binary search to find a approximately accurate ℓ' in the interval $[\ell_{min}, \ell_{max}]$ as follows:

1. Initialize: $low = \ell_{min}$ and $high = \ell_{max}$ first.

2. Compute $mid = (low + high)/2$.

3. Check whether the solution mid satisfies the predefined precision β. If we have $|g(mid) - k| \leq \beta$, then terminate this searching and output $\ell' = mid$.

4. If $g(mid) > k$, we set $low = mid$; Otherwise, we set $high = mid$.

5. Go back to step (2)

If there is no such ℓ' that $g(\ell') = k$, it means that the $g(\ell)$ is not a continuous function, in other words, there exists element $i \in V$ such that $a_i(t) + b_i(t) = 0$. If there is a non-continuous point ℓ' following this condition: Given any $\varepsilon > 0$, $g(\ell' - \varepsilon) > k$ and $g(\ell' - \varepsilon) < k$, we define such ℓ' that $g(\ell') = k$. Therefore, we can find ℓ' satisfying $g(\ell') = k$ at any time and any cases.

Then, let us consider step (6), $\ell^* = \max\{\ell', 0\}$, which indicates that $\ell^* \geq \ell'$. We have $g(\ell^*) \leq g(\ell')$ due to the fact that $g(\ell)$ is monotone non-increasing. Thus, $g(\ell^*) \leq k$, it results in $\sum_{i \in V} x_i(1) \leq k$ eventually. In order to transform the resulting fractional vector to an integral solution without losing budget, we may get help from some rounding techniques, such as pipage rounding (Calinescu et al., 2011) (Vondrák, 2013), or select the nodes with the highest component in vector $\vec{x}(1)$ heuristicly. When $g(\ell^*) = k$, we select top k nodes with the highest component as our result. However, when $g(\ell^*) < k$, we compare the objective value of top $\lceil \sum_{i \in V} x_i(1) \rceil$ and $\lfloor \sum_{i \in V} x_i(1) \rfloor$ nodes with the highest component, then choose the better one as our result.

4.3.3 Time Complexity

Considering the step (2), we need to compute $a_i(t)$ and $b_i(t)$ by taking the average of λ independent sampling. For each element $i \in V$, the running time is bounded by $O(4\lambda mr)$ where r is the number of Monte Carlo simulations. Thus, the total running time of step

(2) is $O(4\lambda mnr)$. The running time of Discretized-CDG is determined by its step (2), so we have its time complexity $O(4\lambda mnr/\Delta t)$. We have mentioned that we need to balance the granularity of discretization and the error incurred. Here, the granularity refers to the value of time step Δt. The smaller the granularity, the smaller the incurred error, so the result $\vec{x}$ returned by Discretized-CDG can show the effect of CDG algorithm more accurately. Generally, this result will also be more satisfying. However, the smaller granularity will lead to longer running time, thus, if the accuracy requirements are not so strict, we can choose a larger granularity to improve the efficiency of this algorithm.

In addition, another constraint on the efficiency of Discretized-CDG is to estimate the value of $f(\cdot)$ by Monte Carlo simulations. Here, we give some idea to address the intractability in computing influence spread. A direct technique is based on the reverse influence sampling (Borgs et al., 2014) for IM problem. There is an important concept, called Random RR-Set (Reverse Reachable Set). Let v is a node sampled randomly from V and g is a realization sampled from the probability distribution of IC/LT model, Random RR-Set is the set of nodes in g that can reach v. Give a Random RR-Set R, we have $\sigma(S) = n \cdot \Pr[S \text{ covers } R]$. Given a seed set S, we can obtain a (μ, η)-Estimation of $f(S)$, $\Pr\left|\hat{f}(S) - f(S)\right| \geq \mu f(S)] \leq \eta$, by sampling Random RR-Set π times, and get a collection of Random RR-Set, denoted by $\mathcal{R}$, where $\mathcal{R} = \{R_1, R_2, \cdots, R_\pi\}$. We denote this estimated value by $\hat{f}(S)$,

$$\hat{f}(S) = \frac{(p-c)n}{\pi} \cdot \sum_{i=1}^{\pi} z(S, R_i) - b \cdot |S|, \tag{4.6}$$

where $z(S, R_i) = 1$ if $S \cap R_i \neq \emptyset$, otherwise $z(S, R_i) = 0$. Then, we replace the $f(\cdot)$ in step (2) with $\hat{f}(\cdot)$. The error can be controlled by the two adjustable parameters μ and η. Since the sampling of the collection of Random RR-Set can be done at once, the running time of step (2) is reduced greatly, and the approximation performance is hardly affected. Besides, there are some existing heuristic algorithms can be used to estimate the influence spread. For example, PMIA (Wang et al., 2012) is a classical algorithm for IC model, whose

Algorithm 10: B-Framework (G, f, k)

$S_1 \leftarrow$ RandomGreedy (G, f, k);
$\vec{x} \leftarrow$ Discretized-CDG (G, f, k);
$S_2 \leftarrow$ Pipage rounding or heuristic rounding of fractional vector $\vec{x}$;
$S^* \leftarrow \arg\max\{f(S_1), f(S_2)\}$;
return S^*;

main idea is to transform the diffusion of the influence of a node on a general graph into a representative maximum influence propagation subtree in the area near the node. The IRIE algorithm (Jung et al., 2012) uses the overall iterative method on the graph to improve the algorithm speed and save memory usage at the same time. There are also LDAG algorithm (Chen et al., 2010b) and SIMPATH algorithm (Goyal et al., 2011b) for the LT model. The advantage of these heuristic algorithms is that they are fast and usually have good results, but they lack theoretical guarantees unfortunately.

4.3.4 Solution for BPMCA

The discretization of CDG algorithm reduces its approximation ratio by a low order term. Time step Δt and sampling number λ are adjustable parameters, which balances the running time and performance. Thus, combining with the conclusion from (Buchbinder et al., 2014), we have

Lemma 4.3.3. *Given non-negative submodular function $f(\cdot)$ and cardinality k, the approximation ratio of set produced by Discretized-CDG can be considered the same as CDG roughly when Δt is small enough and λ is large enough. CDG achieves an approximation (Buchbinder et al., 2014) of $(1 + (n/(2\sqrt{(n-k)k}))^{-1})$.*

From Fig. 4.1, we can see that the theoretical approximation performance of CDG is better than RG when the ratio of k/n is greater than 10. Therefore, the algorithm to solve BPMCA problem is formulated by combining RG and Discretized-CDG algorithm, called

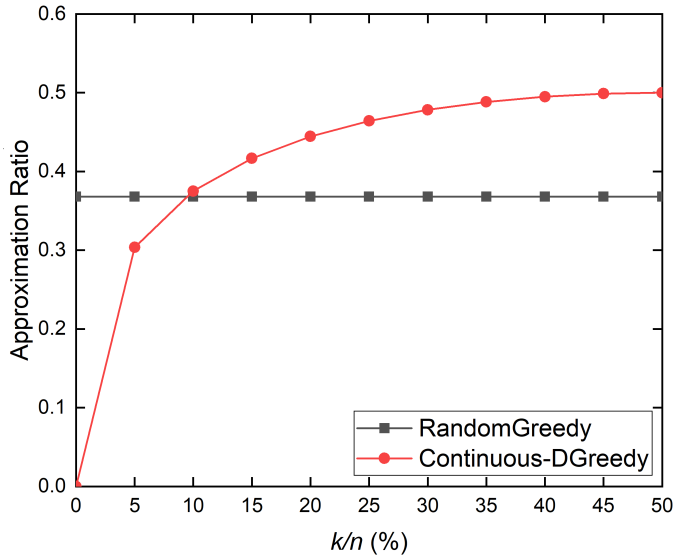


Figure 4.1. The approximation performance of RandomGreedy and Continuous-DGreedy algorithm with the different k/n.

B-Framework. It is shown as Algorithm 10. Let S^* be the final solution from B-Framework, its theoretical bound can be guaranteed:

Theorem 4.3.4. *Given non-negative submodular function $f(\cdot)$ and cardinality k, we can achieve an approximate solution S^* such that $|S^*| \leq k$ and*

$$f(S^*) \geq \max\left\{\frac{1}{e}, \left(1 + \frac{n}{2\sqrt{(n-k)k}}\right)^{-1}\right\} \cdot f(S_{opt}), \tag{4.7}$$

where S^* is the solution returned by B-Framework and S_{opt} is the optimal solution.

Proof. Combining Lemma 1 and Lemma 2, the approximation ratio of B-Framework holds naturally. □

4.4 Robust Analysis

However, in a real business scenario, it is not an easy task to get the strength of the association (diffusion probability) among users. In general, this is based on an estimate of the user

log (Netrapalli and Sanghavi, 2012) (Tang et al., 2009), which is usually not accurate. The probability for each edge we learn from user log exists a certain range of errors compared to its ground-truth value. Supposed there is an confidence interval $[l_e, r_e]$ $(0 \leq l_e \leq r_e \leq 1)$ corresponding to each edge $e \in E$, the true probability p_e of the edge e belongs to this interval definitely. It can be applied to the IC model naturally. Similarly, in LT model, the weight b_{uv} for each edge (u, v) lies in interval $[l_{uv}, r_{uv}]$, but we need to make sure $\Sigma_{u \in N^-(v)} r_{uv} \leq 1$ for each node v. This idea can be extended to any other model easily as long as the influence is spread in probability.

Given $G = (V, E)$, we denote $\Theta = \times_{e \in E}[l_e, r_e]$ as the parameter space of G, and $\theta = (p_e)_{e \in E} \in \Theta$ as a parameter vector. In this section, the objective function of BPMCA and influence function in Equation (4) and (5) can be represented by $f_\theta(\cdot)$ and $\sigma_\theta(\cdot)$ instead of $f(\cdot)$ and $\sigma(\cdot)$, which means that the expected profit and expected influence under the parameter vector θ given IC/LT model. Thus, given the seed set S, we have

$$f_\theta(S) = (p - c) \cdot \sigma_\theta(S) - b \cdot |S| \tag{4.8}$$

$$= (p - c) \cdot (\sigma_\theta(S) - |S|) + (p - c - b) \cdot |S|. \tag{4.9}$$

Particularly, we denote $\theta^+(\Theta) = (r_e)_{e \in E}$ and $\theta^-(\Theta) = (l_e)_{e \in E}$ as the maximum and minimum parameter vectors, respectively, and we would only use θ^+ and θ^- if the context is clear. For different parameter settings, $\theta \in \Theta$, the optimal solution and function value of any seed set is different. Given $S \in V$, the robust ratio under parameter space Θ can be defined as

$$g(\Theta, S) = \min_{\theta \in \Theta} \frac{f_\theta(S)}{f_\theta(S^*_\theta)}, \tag{4.10}$$

where $S^*_\theta \in V$ is the optimal solution when the parameter vector is given by θ. Given Θ and solution S, the robust ratio $g(\Theta, S)$ shows us with the worst ratio of profit under θ and its corresponding optimal solution S^*_θ, when the true θ is not sure but knowing that $\theta \in \Theta$. Then the Robust Budget Profit Maximization with Coupon Advertisement (Robust-BPMCA) problem can be defined as follows:

Problem 3 (Robust-BPMCA)**.** *Given a social network $G = (V, E)$, product parameter p, c and b and parameter space Θ, we aim to find a seed set $S \subseteq V$ and $|S| \leq k$ such that the robust ratio $g(\Theta, S)$ can be maximized under the IC/LT model.*

From Problem 2, Robust-BPMCA problem aims to find a optimal solution S^*_Θ with the largest robust ratio under parameter space Θ. Thus, we have

$$S^*_\Theta = \arg\max_{|S|\leq k} g(\Theta, S) = \arg\max_{|S|\leq k} \left(\min_{\theta\in\Theta} \frac{f_\theta(S)}{f_\theta(S^*_\theta)} \right). \tag{4.11}$$

Here, the ground-truth parameter vector θ we do not know in advanced, and S^*_Θ should make the worst ratio of its profit and the largest profit maximized. The Robust-BPMCA can be reduced back to BPMCA problem when there is no uncertainty for edges, at this time, the parameter space Θ can be down to a vector θ.

4.4.1 Solution for Robust-BPMCA

Given parameter space Θ, we design a Lower-Upper B-Framework to solve Robust-BPMCA problem. Its main thought is to use B-Framework to solve BPMCA problem under the maximum and minimum parameter vector $\theta^+(\Theta)$ and $\theta^-(\Theta)$, respectively. It is shown as Algorithm 11, called LU-B-Framework. In Algorithm 11, B-Framework can be acted as a subroutine to solve BPMCA problem under the setting of parameter vector θ^+ and θ^-, then return the better one under the minimum parameter vector θ^-. Thus, we have

$$S^{LU}_\Theta = \arg\max_{S\in\{S^B_{\theta^+}, S^B_{\theta^-}\}} f_{\theta^-}(S). \tag{4.12}$$

As we said before, the parameter space Θ is down to vector θ when there is no uncertainty, in this way, Robust-BPMCA is reduced to BPMCA problem naturally. Robust-BPMCA is NP-hard because of the NP-hardness of BPMCA.

Algorithm 11: LU-B-Framework (G, f, k, Θ)

$\theta^+ \leftarrow \theta^+(\Theta)$, $\theta^- \leftarrow \theta^-(\Theta)$;
$S^{BF}_{\theta^+} \leftarrow$ B-Framework (G, f_{θ^+}, k);
$S^{BF}_{\theta^-} \leftarrow$ B-Framework (G, f_{θ^-}, k);
$S^{LU}_{\Theta} \leftarrow \arg\max_{S \in \{S^B_{\theta^+}, S^B_{\theta^-}\}} f_{\theta^-}(S)$;
return S^{LU}_{Θ};

Obviously, the theoretical guarantee of LU-B-Framework is impossible to be better than the approximation of B-Framework, shown as Theorem 3. Even though that, for Robust-BPMCA problem, we can achieve a dependent bound by use of B-Framework. The dependence comes from two parts: (1) The number of nodes in graph G and budget k; (2) The solution S^{LU}_{Θ} and $S^{BF}_{\theta^+}$ from B-Framework. In order to assess the approximate performance of solution S^{LU}_{Θ}, we introduce the concept of gap ratio $\alpha(\Theta) \in [0, 1]$ as

$$\alpha(\Theta) = \frac{f_{\theta^-}(S^{LU}_{\Theta})}{f_{\theta^+}(S^{BF}_{\theta^+})} \tag{4.13}$$

for any input parameter space Θ. For LU-B-Framework, we have following theorem:

Theorem 4.4.1. *Given a social network $G = (V, E)$, budget k and parameter space Θ, the seed seed S^{LU}_{Θ} returned by LU-B-Framework satisfies*

$$g(\Theta, S^{LU}_{\Theta}) \geq h(G, k) \cdot \alpha(\Theta), \tag{4.14}$$

where

$$h(G, k) = \max\left\{\frac{1}{e}, \left(1 + \frac{|V(G)|}{2\sqrt{(|V(G)| - k)k}}\right)^{-1}\right\}. \tag{4.15}$$

Proof. First, we need to show that fixing seed set S, $f_\theta(S)$ is monotone non-decreasing with respect to θ. In the IC/LT model, $\sigma_\theta(S)$ is monotone non-decreasing with the increase of θ under fixed S. From Equation (10), $(p - c) \cdot \sigma_\theta(S)$ is monotone non-decreasing on θ as well, and $b \cdot \cdot |S|$ is a constant. Thus, $f_\theta(S)$ is monotone non-decreasing with respect to θ. Then,

we have $f_\theta(S^*_\theta) \leq f_{\theta^+}(S^*_\theta) \leq f_{\theta^+}(S^*_{\theta^+})$ according to the monotonicity of f about θ. On the report of Theorem 3, we have

$$f_{\theta^+}(S^{BF}_{\theta^+}) \geq h(G,k) \cdot f_{\theta^+}(S^*_{\theta^+}). \tag{4.16}$$

Moreover, it can be implied that

$$\begin{aligned} g(\Theta, S) &= \min_{\theta \in \Theta} \frac{f_\theta(S)}{f_\theta(S^*_\theta)} \\ &\geq \min_{\theta \in \Theta} \frac{f_\theta(S)}{f_{\theta^+}(S^*_{\theta^+})} \geq \min_{\theta \in \Theta} \frac{f_\theta(S)}{f_{\theta^+}(S^{BF}_{\theta^+})} \cdot h(G,k) \\ &\geq \frac{f_{\theta^-}(S)}{f_{\theta^+}(S^{BF}_{\theta^+})} \cdot h(G,k). \end{aligned}$$

Substituting S^{LU}_Θ from LU-B-Framework into the above inequality, we have

$$\begin{aligned} g(\Theta, S^{LU}_\Theta) &\geq \frac{f_{\theta^-}(S^{LU}_\Theta)}{f_{\theta^+}(S^{BF}_{\theta^+})} \cdot h(G,k) \\ &= \alpha(\Theta) \cdot h(G,k). \end{aligned}$$

Therefore, the theorem is proved. □

We notice that the value of $\alpha(\Theta) \cdot h(G,k)$ not only depends on input variables G and k, but also on the solution S^{LU}_Θ and $S^{BF}_{\theta^+}$ of LU-B-Framework. Thus, this approximation value is dependent.

4.4.2 Solution with Uniform Sampling

Before, we propose the solution for Robust-BPMCA problem, B-Framework, which has a dependent theoretical bound shown as Theorem 4, and indicate the reason why it is dependent is that approximation performance relies on the value of $\alpha(\Theta)$. The width of confidence interval for each edge will impact the value of $\alpha(\Theta)$ because uncertainty will be exacerbated if and only if the confidence interval becomes wider. Thus, the wider confidence interval is,

Algorithm 12: B-UniSampling $(G, f, k, \Theta, (\epsilon, \gamma))$

1 $t \leftarrow 2\left(\frac{mn(p-c)}{\epsilon h(G,k)(p-c-b)k}\right)^2 \cdot \ln\left(\frac{2m}{\gamma}\right)$;
2 **for** *each* $e \in E$ **do**
3 Sample e for t times, and observe $x_e^1, x_e^2, ..., x_e^t$;
4 $p_e \leftarrow \frac{1}{t}\sum_{i=1}^{t} x_e^i$;
5 $\delta \leftarrow \frac{\epsilon h(G,k)(p-c-b)k}{mn(p-c)}$;
6 $r_e \leftarrow \min\{1, p_e + \delta/2\}$;
7 $l_e \leftarrow \max\{0, p_e - \delta/2\}$;
8 $\Theta_{out} \leftarrow \times_{e\in E}[l_e, r_e]$;
9 $S_{out} \leftarrow$ LU-B-Framework (G, f, k, Θ_{out});
10 **return** (Θ_{out}, S_{out});

the worse the corresponding robust ratio we have. At this time, the satisfactory approximation ratio cannot be obtained. A natural question is how can we improve our robust ratio further? Sampling method works here. We sample the diffusion probability of each edge as much as possible, which enhances the accuracy of estimation by use of shortening the width of the confidence interval. It can be shortened from Θ to Θ'. In light of Chernoff's bound, the more samples we collect, the shorter the confidence interval becomes where ground-truth diffusion probability lies in.

The next question is how to correspond our profit function by influence spread with the width of confidence interval. Chen et al. (Chen et al., 2016) proposed two properties, additive and multiplicative confidence interval, to solve this question. Here, we use additive property, combining it with LU-B-Framework and getting a theoretical bound. Relied on this idea, we design a sampling-based LU-B-Framework by incorporating into uniform sampling. It is shown as Algorithm 12, called B-UniSampling. Here, we aim to generate a narrower parameter space Θ_{out} instead of Θ, which can obtain a new seed set S_{out} such that $g(\Theta_{out}, S_{out})$ is larger than before with high probability. First, in Algorithm 12, we sample the diffusion probability t times for each edge in line 3, and then, new parameter space Θ_{out} can be generated according to this results under the adjustable parameter ϵ and γ. Then,

we take Θ_{out} as the input of LU-B-Framework to get S_{out}. Based on this idea, we need to establish the connection between the width of confidence interval and the profit of BPMCA problem in the additive form. Let us look at the following Lemma:

Lemma 4.4.2. *Given a social network $G = (V, E)$, parameter space Θ and $B = (p-c)\cdot n$, let S be any seed set such that $S \subseteq V$ and $||\theta^+ - \theta^-||_\infty = \max_{e\in E} |r_e - l_e|$. Then, the difference of $f_{\theta^+}(S)$ and $f_{\theta^-}(S)$ can be bounded as*

$$f_{\theta^+}(S) - f_{\theta^-}(S) \leq mB \cdot ||\theta^+ - \theta^-||_\infty. \tag{4.17}$$

Proof. The proof can be extended from Lemma 4 in (Chen et al., 2016). □

The bound of the difference between $f_{\theta^+}(\cdot)$ and $f_{\theta^-}(\cdot)$ for any seed set $S \subseteq V$ can be determined by Equation (19) theoretically. It shows that this difference will be reduced certainly if the value of $||\theta^+ - \theta^-||_\infty$ is smaller. This means the narrower the parameter space Θ_{out} is, the better the robust ratio becomes. Then, the performance of B-UniSampling algorithm can be concluded as:

Theorem 4.4.3. *Given a social network $G = (V, E)$, budget k, parameter space Θ and adjustable parameters (ϵ, γ), let $a = h(G, k) \cdot (p - c - b) \cdot k$, $\delta = \epsilon a/mB$ and $t = 2m^2B^2 \cdot \ln(2m/\gamma)/(\epsilon^2 a^2)$. The solution pair (Θ_{out}, S_{out}) returned by B-UniSampling algorithm satisfies*

$$g(\Theta_{out}, S_{out}) \geq h(G, k) \cdot (1 - \epsilon), \tag{4.18}$$

with probability $\Pr[\theta \in \Theta_{out}] \geq 1 - \gamma$.

Proof. From Algorithm 12, we get a pair (Θ_{out}, S_{out}) and according to Theorem 4, we have

$$\begin{aligned}
&g(\Theta_{out}, S_{out}) \geq h(G,k) \cdot \alpha(\Theta) \\
&= h(G,k) \cdot \frac{f_{\theta^-_{out}}(S^{LU}_{\Theta_{out}})}{f_{\theta^+_{out}}(S^{BF}_{\theta^+_{out}})} \\
&\geq h(G,k) \cdot \frac{f_{\theta^-_{out}}(S^{BF}_{\theta^+_{out}})}{f_{\theta^+_{out}}(S^{BF}_{\theta^+_{out}})} \\
&= h(G,k) \cdot \left(1 - \frac{f_{\theta^+_{out}}(S^{BF}_{\theta^+_{out}}) - f_{\theta^-_{out}}(S^{BF}_{\theta^+_{out}})}{f_{\theta^+_{out}}(S^{BF}_{\theta^+_{out}})}\right).
\end{aligned}$$

The second inequality holds definitely since $f_{\theta^-_{out}}(S^{LU}_{\Theta_{out}}) \geq f_{\theta^-_{out}}(S^{BF}_{\theta^+_{out}})$ according to the line 4 of Algorithm 11. Based on the definition of parameter space, $\Theta_{out} = \times_{e \in E}[l_e, r_e]$, we have, for any $e \in E$ and $\delta > 0$, the lower bound and upper bound of ground-truth probability p_e can be denoted as $l_e = \frac{1}{t}\sum_{i=1}^{t} x_e^i - \delta/2$ and $r_e = \frac{1}{t}\sum_{i=1}^{t} x_e^i + \delta/2$. It is shown as line 6 and 7 of Algorithm 12. Then, let $||\theta^+_{out} - \theta^-_{out}||_\infty = \delta$. Because of Lemma 3, we have

$$f_{\theta^+_{out}}(S^{BF}_{\theta^+_{out}}) - f_{\theta^-_{out}}(S^{BF}_{\theta^+_{out}}) \leq \delta m B.$$

Keeping up with the above step, we have

$$\begin{aligned}
&g(\Theta_{out}, S_{out}) \\
&\geq h(G,k) \cdot \left(1 - \frac{f_{\theta^+_{out}}(S^{BF}_{\theta^+_{out}}) - f_{\theta^-_{out}}(S^{BF}_{\theta^+_{out}})}{f_{\theta^+_{out}}(S^{BF}_{\theta^+_{out}})}\right) \\
&\geq h(G,k) \cdot \left(1 - \frac{\delta m B}{f_{\theta^+_{out}}(S^{BF}_{\theta^+_{out}})}\right) \\
&\geq h(G,k) \cdot \left(1 - \frac{\delta m B}{a}\right) \qquad (4.19) \\
&= h(G,k) \cdot \left(1 - \frac{\epsilon a}{mB} \cdot \frac{mB}{a}\right) \qquad (4.20) \\
&= h(G,k) \cdot (1 - \epsilon),
\end{aligned}$$

where the Inequality (22) and Equation (23) can be established by setting $a = h(G,k) \cdot (p - c - b) \cdot k$ and $\delta = \epsilon a / mB$. Here, we can prove $f_{\theta^+_{out}}(S^{BF}_{\theta^+_{out}}) \geq a$. According to Theorem 2,

we understand that $f_{\theta^+_{out}}(\cdot)$ is non-negative and let $|S^*_{\theta^+_{out}}| \leq k$ be the optimal solution under the parameter vector θ^+_{out}, we have $f_{\theta^+_{out}}(S^*_{\theta^+_{out}}) \geq (p-c-b)\cdot k$. The profit is impossible to be less than $(p-c-b)\cdot k$ if we select k seeds, since $\sigma_\theta(S) - |S| \geq 0$. Based on Theorem 3, we have

$$\begin{aligned} f_{\theta^+_{out}}(S^{BF}_{\theta^+_{out}}) &\geq h(G,k)\cdot f_{\theta^+_{out}}(S^*_{\theta^+_{out}}) \\ &\geq a = h(G,k)\cdot(p-c-b)\cdot k. \end{aligned}$$

Therefore, a is a lower bound of that. For each edge $e \in E$, we need to generate t random variables $\{x^1_e, x^2_e, ..., x^t_e\}$, where $x^i_e \in [l_e, r_e]$ and $i \in \{1, 2, ..., t\}$. How can we determine t such that $g(\Theta_{out}, S_{out}) \geq h(G,k)\cdot(1-\epsilon)$ holds with probability $\Pr[\theta \in \Theta_{out}] \geq 1-\gamma$. Let us denote the estimation $\hat{p}_e = \frac{1}{t}\sum_{i=1}^{t} x^i_e$ and by the additive form of Chernoff-Hoeffding Inequality, we have

$$\begin{aligned} \Pr\left[|\hat{p}_e - p_e| > \frac{\delta}{2}\right] &\leq 2\exp\left(\frac{-2(\delta t/2)^2}{\sum_{i=1}^{t}(r_e - l_e)^2}\right) \\ &\leq 2\exp\left(-\frac{\delta^2 t}{2}\right) \\ &= 2\exp\left(-\frac{\delta^2}{2}\cdot\frac{2\ln(2m/\gamma)}{\delta^2}\right) \\ &= \frac{\gamma}{m}, \end{aligned} \tag{4.21}$$

where we set $t = 2\ln(2m/\gamma)/\delta^2$. Since $\delta = \epsilon a/mB$, we have $t = 2m^2B^2\cdot\ln(2m/\gamma)/(\epsilon^2 a^2)$ and the Equation (24) holds becasue of $r_e - l_e \leq 1$. Therefore, $\Pr[\theta \in \Theta_{out}] = \Pr[\forall e : |\hat{p}_e - p_e| \leq \delta^2/2] \geq 1 - \sum_{i=1}^{m}\Pr[|\hat{p}_e - p_e| > \delta^2/2] = 1 - m\cdot\Pr[|\hat{p}_e - p_e| > \delta^2/2] \geq 1-\gamma$. The theorem is proved. □

Finally, the value of δ will be smaller and smaller with the increase of samples. At this time, we can obtain better and better approximation ratio of B-UniSampling until approaching to $h(G,k)$, whose parameter space degenerates to parameter vector with no uncertainty. It can be solved by B-Framework.

4.5 Experiment

In this section, we show the effectiveness and efficiency of our proposed algorithms on several real social networks. Our experiments are based on the dataset from networkrepository.com (Rossi and Ahmed, 2015), an online network repository. There are two datasets in our experiment. The dataset-1 is a co-authorship network, where each edge is a co-authorship among scientists in network theory and experiments. The dataset-2 is a Wiki network, which is a who-votes-on-whom collected from Wikipedia. The number of nodes of dataset-1 and dataset-2 are 0.4K and 1K, respectively.

4.5.1 Experimental Settings

As mentioned earlier, our proposed algorithms are based on IC model and LT model for experiments. Under the IC model, the diffusion probability for each edge is set as $p_e = 1/|N^-(v)|$, and for the LT model, the weight for each edge $e = (u, v)$ is set as $b_{uv} = 1/|N^-(v)|$. These settings are widely used by previous researchers. There are two experiments we perform for each dataset: Algorithm performance and Robust analysis.

Algorithm performance: In this experiment, we test our B-Framework (Algorithm 10) with some common heuristic algorithms and compare the result of RG and discretized-CDG algorithm. It aims to evaluate the effectiveness and efficiency of our discretization process and heuristic rounding technique proposed before. The profit function can be defined as $f(\cdot|15, 10, 5)$ in this experiment, where $p = 15$, $c = 10$ and $b = 5$. The time step $\Delta t = 0.1$, sampling number $\lambda = 10$ and the precision $\beta = 0.01$. The budget k is from 0 to $n/2$ and the number of Monte Carlo simulation is set to $r = 500$. Our proposed algorithms are compared with some baseline algorithms:

1. B-Framework: Shown as Algorithm 10, select the better seed set from the result of RG and Discretized-CDG algorithms.

2. B-Framework (RIS): Shown as Algorithm 10, but we use RIS technique, Equation (9), to estimate objective function $f(\cdot)$ in step (2) of Discretized-CDG. Thu number of Random RR-Set we sample is 20,000.

3. Random: Select the node randomly from V under the budget k.

4. MaxDegree: Select the node with the highest outdegree under the budget k.

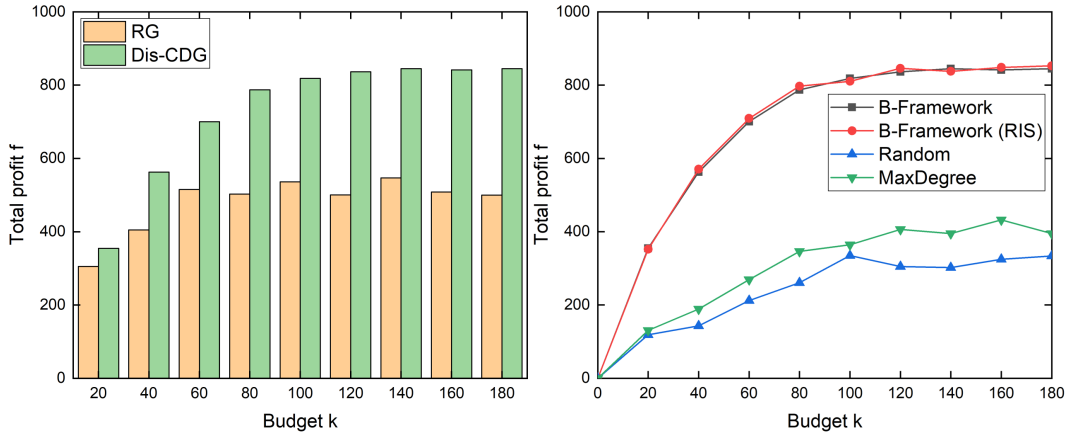
Robust analysis: In this experiment, we assess our LU-B-Framework (Algorithm 11) under any given parameter space and verify the sampling-based B-UniSampling (Algorithm 12). Here, the objective function is $g(\Theta, S)$, shown as Equation (13). We cannot obtain the exact value of $g(\Theta, S)$ due to the unknown optimal solution S^*_θ. However, we can know its the upper bound instead of $g(\Theta, S)$. The profit function can be defined as $f(\cdot|15, 10, 3)$ in this experiment, where $p = 15$, $c = 10$ and $b = 3$. According to Equation (17), Theorem 4, we have the lower bound as

$$g_{LB}(\Theta, S^{LU}_\Theta) = h(G, k) \cdot \alpha(\Theta). \tag{4.22}$$

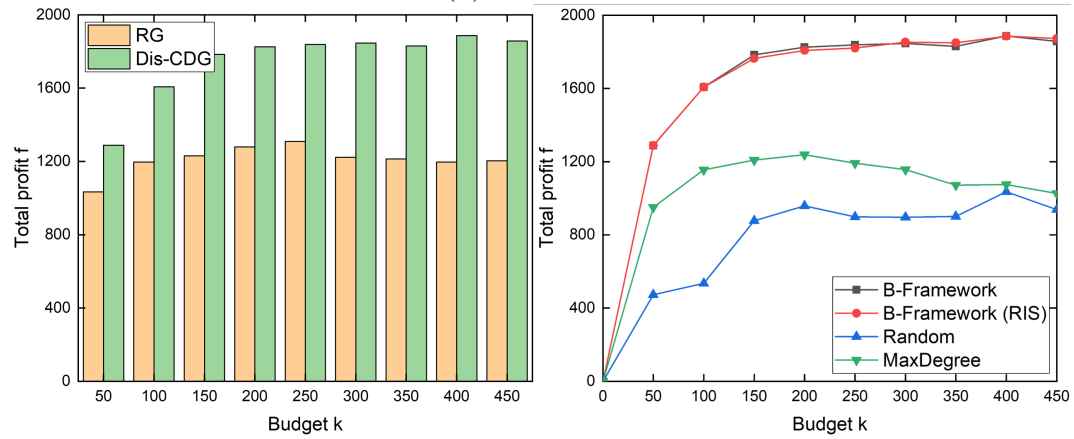
Under the IC model, for Algorithm 11, we can set $l_e = \max\{p_e - w/2, 0\}$ and $r_e = \min\{p_e + w/2, 1\}$ for any $e \in E$ given the width of interval w. Then, the lower bound of $g(\Theta, S^{LU}_\Theta)$ can be computed. For Algorithm 12, we assume there is a probability sample subject to a normal distribution whose expected value is equal to ground-truth diffusion probability defined above for each $e \in E$. We set $\gamma = 0.1$, which means that $\Pr[\theta \in \Theta_{out}] \geq 0.9$, and t as a variable.

4.5.2 Experimental Results

Fig. 4.2 and Fig. 4.3 draw the performance achieved by B-Framework under the IC/LT model and the comparison between the RG and Discretized-CDG algorithm. Theoretically, the approximate performance will be better and better with the increase of budget k. From



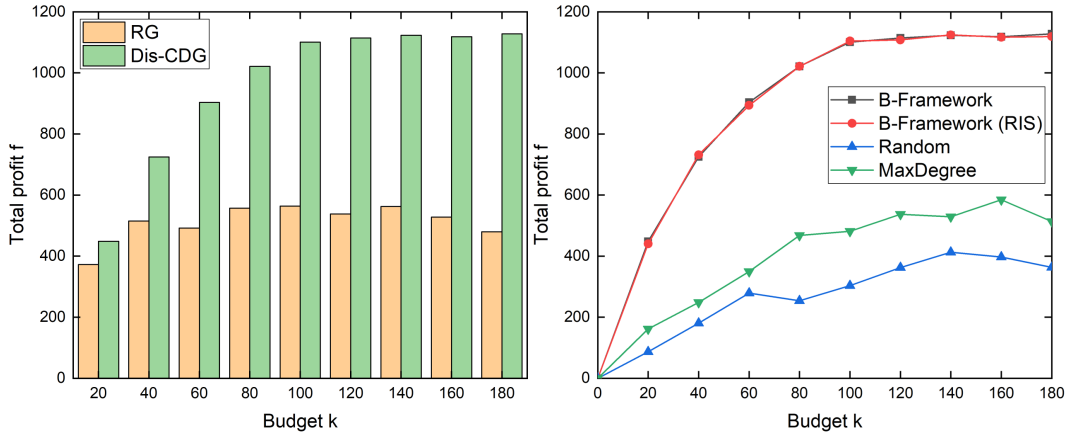
(a) Dataset-1



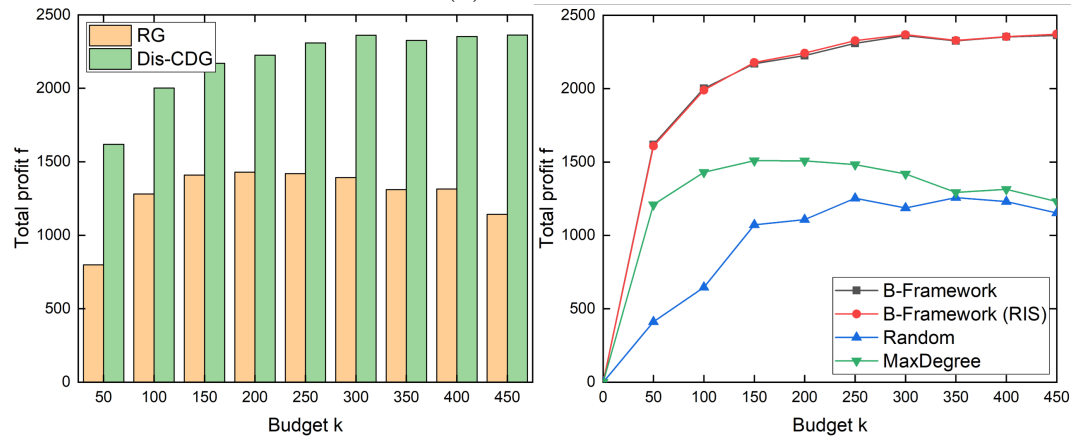
(b) Dataset-2

Figure 4.2. The performance of B-Framework changes over budget k under the IC model. Left column is the comparison between RG and Discretized-CDG; Right column is achieved by different algorithms.

the left column of Fig. 4.2 and Fig. 4.3, the total profit obtained by Discretized-CDG is larger than that obtained by RG, and with the increase of budget k, the superiority of Discretized-CDG is more manifest, which is in line with our expectations. From the right column of Fig. 4.2 and Fig. 4.3, the total profit returned by B-Framework is the largest among these three algorithms, so its performance is the best. With the help of Discretized-CDG, we can overcome the bad approximation ratio of SMCC before. Maybe, it can be improved further by setting a smaller time step Δt at the expense of time, and we are not to expand here due to space limitation. Besides, the performance of B-Framework and B-



(a) Dataset-1



(b) Dataset-2

Figure 4.3. The performance of B-Framework changes over budget k under the LT model. Left column is the comparison between RG and Discretized-CDG; Right column is achieved by different algorithms.

Framework (RIS) is very close, but the running time is reduced significantly by the technique of RIS sampling. Thus, it is an efficient estimation to our objective function.

Fig. 4.4 draws the results of robust analysis with different budget k under the IC model and dataset-1. From the left one of Fig. 4.4, given a parameter space, the width of parameter space Θ can be determined by w, shown as Section 6.1, and with the increase of the value of w, the gap ratio becomes smaller and smaller. Robustness will worsen with increasing in uncertainty, which is consistent with the definition of robust ratio. The downward trend of these curves is more apparent when w is small. Besides, for a given budget k, $k \in$

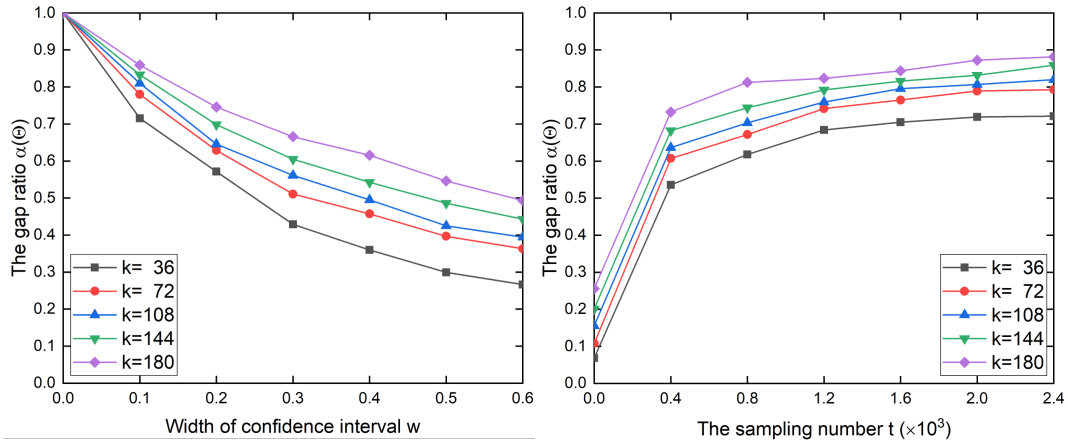


Figure 4.4. The results of robust analysis with different budget k under the IC model and dataset-1. Left column is the gap ratio obtained by given parameter space (LU-B-Framework); Right column is the gap ratio obtained by uniform sampling (B-UniSamling).

$\{0, 1, 2, ..., n/2\}$, the larger the k is, the better gap ratio we can get. According to Theorem 4, robust ratio is larger than $h(G, k) \cdot \alpha(\Theta)$, and $h(G, k)$ increases along with budget k. Thus, comparing to gap ratio, robust ratio will be better and more obvious along with increasing k. From the right one of Fig. 4.4, the gap ratio becomes larger and larger with the increase of sampling number t. According to Theorem 5, the value of ϵ decreases when sampling number t increasing, and it results in a smaller confidence interval δ. Thus, our observation is reasonable. The upward trend of there curves is more notable when t is small, then this trend weakens gradually. Besides, same as before, the larger the k is, the better gap ratio we can get. The results indicate that uniform sampling is an effective way to reduce the uncertainty and achieve better robustness.

CHAPTER 5

ADAPTIVE MULTI-FEATURE BUDGETED PROFIT MAXIMIZATION [1]

Authors – Tiantian Chen, Jianxiong Guo, and Weili Wu

The Computer Science Department, EC 31

The University of Texas at Dallas

800 West Campbell Road

Richardson, Texas 75080-3021

[1]Reproduced with permission from Springer Nature: Tiantian Chen, Jianxiong Guo, and Weili Wu, "Adaptive multi-feature budgeted profit maximization in social networks", Social Network Analysis and Mining, 12, 164, 2022, Springer Nature. DOI: 10.1007/s13278-022-00989-3

5.1 Introduction

Online social network, like Facebook, Twitter, LinkedIn, etc., has been one of the most important platforms for marketing and communication. Many companies have taken social network as main method to promote products by word-of-mouth effects. To maximize the product influence and obtained profit, companies may apply many methods, such as distributing coupons, free samples or offering some discounts when purchasing. Many researches have been focused on the diffusion phenomenon on social networks, including diffusion of ideas, news, adoptions of new products, etc. One topic extensively studied is the Influence Maximization (IM) problem (Borodin et al., 2010; Chen et al., 2009, 2010a; Kempe et al., 2003), which asks for k seeds to maximize the expected number of influenced users under some diffusion model. There are two classical diffusion models: Independent Cascade (IC) model and Linear Threshold (LT) model.

However, it has been proved IM problem is NP-hard and computing the expected spread from a node set is #P-hard in general under IC model (Chen et al., 2010a) and LT model (Chen et al., 2010b). (Kempe et al., 2003) presented a $(1 - 1/e)$-approximation scheme, classical greedy algorithm, for IM problem and they used Monte Carlo method to estimate expected spread of a seed set, but it is time-consuming. Many recent works (Tang et al., 2014, 2015; Nguyen et al., 2016b; Huang et al., 2017, 2020; Han et al., 2018; Tang et al., 2018) have been focused on solving this problem, which not only could obtain a $(1-1/e-\epsilon)$-approximation solution with high probability but are efficient even for large-scale datasets.

Most of existing papers related to IM problem only consider a single diffusion. That is, only one piece of information about the product is spread on social networks. Some papers (Chen et al., 2020a; Guo and Wu, 2019; Zhang et al., 2016, 2019) indeed consider multiple diffusions of products. But the diffusions are for multiple products and each diffusion is for one product. However, in reality, one product may have multiple features and the information about all these features can spread on the social network. For instance, when a customer

wants to buy a phone, he may consider many features, such as price, brand, camera, display, speed, etc. He has his own preference for each feature, which can be regarded as weight for the feature, and has a threshold to purchase the phone. He heard the information about features of the phone on networks, and he will purchase the phone only when the sum of weights of features satisfying his requests is larger than or equal to the threshold.

(Guo et al., 2020c) first proposed a multi-feature diffusion model (MF-model) to describe multiple features about one product spreading on the social network, where different feature information spreads independently according to different successful probabilities and whether to accept a product is determined by overall evaluation on all these features. They considered the rumor blocking problem under this model, but they assume the weights of each feature for each node are equal when solving the problem.

Based on the MF-model (Guo et al., 2020c), we propose a novel budgeted profit maximization problem, MBPM problem. Given a social network, MBPM problem assumes that multiple information about multiple features of a product are spread on it. Each feature has its own propagation probability when spreading from one user to another, and each user has its own weights for each feature. A user will purchase the product only when the sum of weights of features he accepts is larger than or equal to his threshold. Each node has an activation cost and a profit. MBPM problem seeks for a seed set with expected cost no more than the budget to make the obtained profit as large as possible. We consider MBPM problem under the adaptive setting, in which the next seed is selected based on the diffusion result of current seeds. That is, we first select a seed and then observe which nodes would be activated by the seed. According to diffusion results, we would select the next seed to maximize the profit as much as possible.

Main contributions of this work are as follows: (1) We propose a novel practical problem, MBPM problem, consider it under both the non-adaptive and adaptive settings and propose efficient strategies to solve them, respectively; (2) For the non-adaptive MBPM problem,

we show its objective is monotone submodular and give a randomized algorithm which could achieve $(1 - 1/e)$ expected approximation guarantee; (3) For the adaptive MBPM problem, we prove its objective function is adaptive monotone and adaptive submodular. We consider adaptive MBPM problem under two models, oracle model and noise model. A policy with $(1 - 1/e)$ expected approximation ratio is given in oracle model. Under noise model, we estimate the conditional expected marginal profit of any node by modifying the EPIC algorithm and propose a sampled adaptive greedy policy which could achieve a $(1 - e^{-(1-\epsilon)})$ expected approximation guarantee, where $0 < \epsilon < 1$; (4) Experimental results on realistic datasets confirm effectiveness and superiority of our algorithm.

5.2 Related Works

(Kempe et al., 2003) first formulated IM problem as a combinatorial optimization problem, which aims to choose k seeds to make the expected influence as large as possible. They presented a $(1 - 1/e)$-approximation algorithm, classical greedy scheme, to solve IM. Later, many variants of IM problem appeared, such as coupon based profit maximization (Liu et al., 2020; Tong et al., 2018; Guo et al., 2020a), multiple products profit maximization (Chen et al., 2020a; Zhang et al., 2016, 2019), etc. The one related to our work is cost-aware targeted viral marketing (CTVM) problem (Nguyen et al., 2017), which maximizes the expected total benefit by choosing a seed set under the budget. The difference between CVTM and our MBPM problem is CVTM only considered one information diffusion on networks under the classical IC and LT model. And they studied CVTM problem under non-adaptive setting and designed a $(1 - 1/\sqrt{e} - \epsilon)$-approximation algorithm. (Banerjee et al., 2020) considered targeted CVTM problem where only nodes in target set have an activation profit, and they proposed a $(1 - 1/\sqrt{e})$-approximation algorithm. However, we consider multiple diffusions of a product's features and studied the MBPM problem under the adaptive setting. (Shan et al., 2019) considered a diffusion model of multiple diffusions

of a product's features, which is similar to ours, but the activation threshold for each node in their model is fixed. They measured the amount of information that a user received as the probability that the user is activated in an information cascade. However, in our MF-model, each node has a weight for each feature which measures how the user cares about the feature, and the activation threshold for each node is distributed uniformly in [0, 1].

For adaptive problem related research, (Golovin and Krause, 2011) proved the objective of adaptive IM problem is adaptive monotone and submodular under full-adoption model and IC model. They proposed an adaptive greedy scheme, which is a $(1-1/e)$-approximation scheme for adaptive IM problem. They also proved this algorithm can be used when a set function with adaptive monotonicity and submodularity subjects to the knapsack constraint. (Han et al., 2018) considered an variant of the adaptive IM problem, where k seeds are selected in batches of equal size b. They designed an AdaptGreedy framework instantiated by scalable IM algorithms, which could achieve a $(1 - e^{-(1-1/e)+\epsilon})$ approximation guarantee with high probability. (Sun et al., 2018) studied a Multi-Round Influence Maximization problem under the adaptive setting where information spreads in multiple rounds independently from probably different seed sets. They proposed an adaptive algorithm instantiated by the IMM (Tang et al., 2015), which could guarantee $(1 - e^{-(1-1/e)} - \epsilon)$ approximation. Recently, (Huang et al., 2020) pointed out there are some gaps in analysis of approximation guarantee for adaptive policies in (Han et al., 2018) and (Sun et al., 2018). They fixed the previous AdaptGreedy framework in (Han et al., 2018) by instantiating with their improved EPIC algorithm and showed it could provide a $(1 - e^{\rho_b(\epsilon-1)})$ expected approximation guarantee. (Guo and Wu, 2020a) considered the adaptive influence maximization with multiple activations problem, where a selected node in each iteration can be unwilling to be the seed and a node not being the seed in previous iteration can be activated again later but with higher activation cost. The goal is to find a randomized policy subject to expected knapsack constraint to maximize the expected influence spread. They designed an adaptive greedy

policies by modifying EPIC algorithm in (Huang et al., 2020). (Peng and Chen, 2019) showed that the adaptivity gap of the IM problem under the IC model with myopic feedback is at least $e/(e-1)$ and at most 4, and that both the non-adaptive and adaptive greedy algorithms achieve a $\frac{1}{4}(1-1/e)$-approximation to the adaptive optimum. (Chen and Peng, 2019) showed the adaptivity gap of the IM problem under the IC model with the full-adoption feedback on several families of well-studied influence graphs. (Chen et al., 2020b) proposed the concept of greedy adaptivity gap, comparing the performance of adaptive greedy algorithm to its non-adaptive counterpart.

5.3 Influence Maximization Problem under the MF-model

A social network is generally denoted by a directed graph $G = (V, E)$, where $|V| = n$ and $|E| = m$. For each $(v, w) \in E$, v is named the in-neighbor of w and w is called the out-neighbor of v. Here, each node $u \in V$ represents a user (customer) in this paper.

5.3.1 Multi-Feature Diffusion Model

Consider a product with multiple features and the information about each feature may spread from one customer to another. To characterize it, we consider the multi-feature diffusion model (MF-model) (Guo et al., 2020c) in this paper.

1. Given a social network $G = (V, E)$, q pieces of information about q features of a product are spread on it, respectively. For each $(u, v) \in E$, there is a q-dimensional propagation probability vector $\bar{p}_{u,v} = (p^1_{u,v}, \ldots, p^q_{u,v})$ associated with it, where $p^i_{u,v} \in (0, 1]$ is the successful probability when u tries to motivate v to accept the information about feature i of the product.

2. Each $u \in V$ has a threshold θ_u distributed in [0,1] uniformly and a weight vector $\bar{w}_u = (w^1_u, \ldots, w^q_u)$, where w^i_u denotes the weight of feature i for user u and $\sum_{i=1}^{q} w^i_u = 1$.

3. When user u accepts feature i at timestamp t (called i-accepted, otherwise called i-unaccepted), it will attempt to motivate its i-unaccepted out-neighbor v with successful probability $p^i_{u,v}$ at timestamp $t+1$. The information about different features is diffused independently on the social network, and user v will purchase the product if and only if the sum of corresponding weights of features that have already been accepted by v is no less than θ_v (called purchase condition).

4. Initially, a set of seeds is activated to spread all of the q features. At every step, each node that hasn't purchased the product would check whether the purchase condition is satisfied. The diffusion process will continue until there is no more node activated.

To further illustrate the model by the phone example, say, the five features of a phone corresponding to price, brand, camera, display, and speed are 1, 2, 3, 4, and 5, respectively. Each node can either accept or not accept each feature. For example, a potential customer may either be convinced that the display is good or not. Each node $v \in V$ also has a weight for each of the five features: $w^1_v, w^2_v, w^3_v, w^4_v$, and w^5_v, which measures how much he cares about each feature. Before the diffusion, each node also has to sample a threshold $\theta_v \in [0,1]$. Initially, all features for each seed are accepted. Then, we have five different cascade processes corresponding to the five features. Each of them follows the IC model, and the five cascade processes are independent. Now, at the end of the cascade process, each node is infected by some of the five features. A node will eventually buy the product if the sum of the weights of the accepted features exceeds its threshold. For example, if v accepts features 2 and 4 but not 1, 3, 5, then v will be considered activated if the weighted sum $w^2_v + w^4_v$ exceeds his threshold θ_v.

Even though (Guo et al., 2020c) first proposed this MF-model, they assume the weight of each feature for each node is the same, namely $w^i_u = w^i_v$ for any user $u, v \in V$, in their submodularity proof and algorithm analysis. This is only a special case and not that realistic. In this paper, we consider the general case where the weight vector $\bar{w}_u = (w^1_u, \ldots, w^q_u)$ for each

node u is arbitrary. Denote by $\sigma(S)$ the expected number of nodes (users) purchasing the product when S is the seed. Actually, we could prove that $\sigma(S)$ is monotone non-decreasing and submodular with respect to S under the general MF-model. Our following analysis is based on the general MF-model, which is an important improvement and extension. Before showing properties of $\sigma(S)$, let's first see the equivalent diffusion process of the MF-model.

Remark 4. *For convenience, we still use $G = (V, E)$ to represent the social network $G = (V, E)$ with propagation probability $p : E \to (0, 1]^q$, threshold $\theta : V \to [0, 1]$, and weight $w : V \to [0, 1]^q$.*

5.3.2 Equivalent Diffusion Process

Since information of different features is spread independently on the social network in MF-model, that is, the diffusion of one piece of information about one feature has no interference on information of other features, we can view the propagation process of MF-model as follows.

Definition 5.3.1 (Multi-level Graph)**.** *Given a social network $G = (V, E)$, define its multi-level graph as $\widehat{G} = (\widehat{V}, \widehat{E}) = G^1 \cup G^2 \cup \cdots G^q$, where $G^i = (V^i, E^i)$ and each node $v_i \in V^i$ is a copy node of $v \in V$, called feature node of v. For each $(u, v) \in E$, there is a corresponding edge $(u_i, v_i) \in E^i$, $i = 1, \cdots, q$, and the propagation probability on (u_i, v_i) is $p^i_{u,v}$, that is, the successful probability when u attempts to motivate v to accept feature i.*

An example of the multi-level graph can be seen in Fig. 5.1. For each node set $S \subseteq V$, denote by $\widehat{S} = S^1 \cup \ldots \cup S^q$ the corresponding feature node set in $\widehat{G}$ of nodes in S, where $S^i = \{u_i \in V^i : u_i \text{ is the corresponding feature node of } u \in S \text{ for feature } i\}$. For each $u \in V$, denote the corresponding feature node set of u as $\widehat{u} = \{u_1, \ldots, u_q\}$. Then we could give the equivalent diffusion process of the general MF-model.

1. Given a social network $G = (V, E)$ and its multi-level graph $\widehat{G} = G^1 \cup \ldots \cup G^q$, q pieces of information about q different features are spread on $\widehat{G}$, but the information about feature

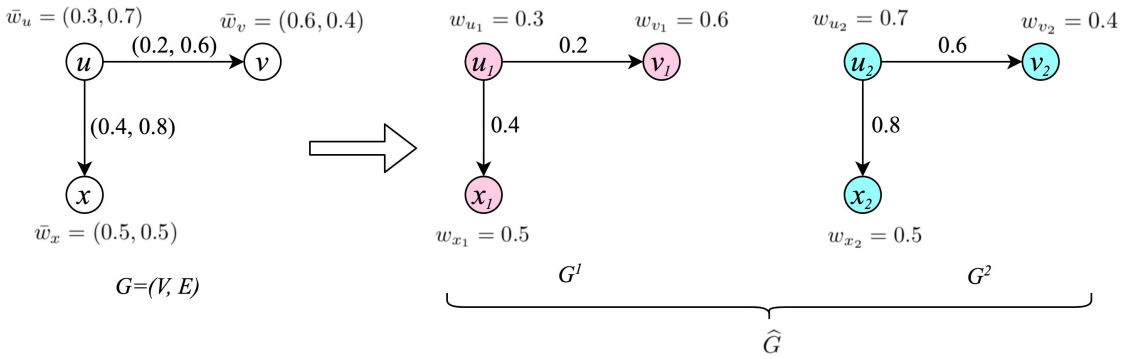


Figure 5.1. An example of $G = (V, E)$ with its multi-level graph $\widehat{G}$.

i is only spread on G^i. Each node $v \in V$ samples a threshold θ_v independently uniformly at random from [0,1].

2. Initially, we choose the seed set $S \subseteq V$ for the product. Then nodes in $\widehat{S}$ are seeds of the corresponding features.

3. The information about different features is diffused independently from their own seeds according to the classic IC model. A node in G^i can only have two states: i-accepted or i-unaccepted. A node in G^i accepting the information of feature i is called i-accepted. Otherwise, it is called i-unaccepted.

4. After the propagation process of all features terminates, we could determine whether each node $v \in V$ would purchase the product. That is, we would check whether the sum of weights of i-accepted nodes $v_i, i = 1, \cdots, q$, is larger than or equal to θ_v. If it satisfies the purchase condition, then node v would purchase the product and we call v active. Otherwise, v is called inactive.

Remark 5. *To avoid confusion, we will use "infect" when we say a feature node u_i tries to activate its out-neighbor v_i to accept feature i, and use "activate" for user node.*

5.3.3 Property of $\sigma(S)$

Definition 5.3.2 (Realization)**.** *Given a social network $G = (V, E)$ with probability $p : E \to (0, 1]$, a (full) realization ϕ of G is defined as $\phi : E \to \{0, 1\}$. For $e \in E$, $\phi(e) = 0$ (resp. 1) means edge e is blocked (resp. live) under ϕ.*

Let Φ be a random variable denoting a random realization. Then we have

$$\Pr[\phi] := \Pr[\Phi = \phi] = \prod_{\substack{e \in E: \\ \phi(e)=1}} p_e \prod_{\substack{e \in E: \\ \phi(e)=0}} (1 - p_e). \tag{5.1}$$

Denote by Ω the set of all possible realizations of multi-level graph $\widehat{G}$. Let S be the seed set of the product and $\widehat{S} = S^1 \cup \ldots \cup S^q$ be its corresponding feature node set. Then S^i is the seed set of feature i in G^i. Given a realization $\phi \in \Omega$, for each node u_i in $\widehat{G}$, define

$$x_\phi(S^i, u_i) = \begin{cases} 1, & u_i \text{ accepts feature } i \text{ in } \phi \text{ under } S^i \\ 0, & \text{otherwise} \end{cases}. \tag{5.2}$$

Therefore, node $u \in V$ will purchase the product under ϕ if and only if $\sum_{i=1}^{q} x_\phi(S^i, u_i) \cdot w_u^i \geq \theta_u$. Denote $I_\phi(S^i)$ as the node set in G^i containing the i-accepted nodes in ϕ under S^i.That is, $I_\phi(S^i) = \{u_i \in V^i | x_\phi(S^i, u_i) = 1\}$. Let $I_\phi(S)$ be the set of active nodes in V when diffusion process of nodes in $\widehat{S}$ on ϕ terminates. Then for any $u \in V$, we have

$$\Pr[u \in I_\phi(S)] = \sum_{\substack{1 \leq i \leq q: \\ u_i \in I_\phi(S^i)}} w_u^i. \tag{5.3}$$

Remark 6. *$I_\phi(S^i), i = 1, \ldots, q$ are deterministic sets while $I_\phi(S)$ is a random set.*

Theorem 5.3.3. *$\sigma(S)$ is monotone non-decreasing and submodular with respect to S.*

Proof. For any node set $S \subseteq T \subseteq V$, denote by $\hat{S} = S^1 \cup \ldots \cup S^q$ and $\hat{T} = T^1 \cup \ldots \cup T^q$ their corresponding copy node set in $\hat{G}$, respectively. Then $\sigma(S)$ under the MF-model can

be represented as:

$$\sigma(S) = \sum_{\phi \in \Omega} \Pr[\phi] \cdot \sum_{v \in V} \Pr[v \in I_\phi(S)] = \sum_{\phi \in \Omega} \Pr[\phi] \cdot \sum_{v \in V} \sum_{\substack{1 \le i \le q: \\ v_i \in I_\phi(S^i)}} w_v^i.$$

Since $S \subseteq T$, then $S^i \subseteq T^i, i = 1, \ldots, q$. Clearly, $I_\phi(S^i) \subseteq I_\phi(T^i)$ since any node in $I_\phi(S^i)$ can also be i-accepted under T^i in ϕ. Therefore, $\sum_{v_i \in I_\phi(S^i)} w_v^i \le \sum_{v_i \in I_\phi(T^i)} w_v^i$ and $\sigma(S)$ is monotone with respect to S. For any $u \in V \setminus T$, we have

$$\sigma(S \cup \{u\}) - \sigma(S) = \sum_{\phi \in \Omega} \Pr[\phi] \cdot \sum_{v \in V} \sum_{\substack{1 \le i \le q: \\ v_i \in (I_\phi(S^i \cup \{u_i\}) \setminus I_\phi(S^i))}} w_v^i,$$

and $I_\phi(S^i \cup \{u_i\}) \setminus I_\phi(S^i)$ contains the nodes in G^i that can only be infected by u_i but cannot by S^i under ϕ. Clearly, $(I_\phi(S^i \cup \{u_i\}) \setminus I_\phi(S^i)) \supseteq (I_\phi(T^i \cup \{u_i\}) \setminus I_\phi(T^i))$, since u_i could infect more feature nodes when adding to S^i than T^i under ϕ. Therefore, $\sigma(S \cup \{u\}) - \sigma(S) \ge \sigma(T \cup \{u\}) - \sigma(T)$ and the proof of Theorem 5.3.3 is completed. □

5.4 Multi-feature Budgeted Profit Maximization Problem

A company wants to promote a new product by distributing coupons on social networks to maximize its profit as much as possible. However, the advertisement budget is usually limited. Thus, it is important to wisely select customers to allocate coupons. The product has multiple features and the information about each feature spreads from one customer to another. Given the social network $G = (V, E)$, for each $u \in V$, assume the cost of picking u as the seed of product and profit obtained when u purchases the product are $c(u)$ and $b(u)$, respectively. For any $S \subseteq V$, the activation cost and profit of S are defined as $c(S) = \sum_{u \in S} c(u)$ and $\sum_{u \in S} b(u)$, respectively. Since we will consider randomized algorithm in the adaptive case later, in this section we will also consider the randomized algorithm for comparison. Given a budget B, we want to find a seed set S with expected cost at most B to maximize the obtained profit.

5.4.1 Problem Definition

Definition 5.4.1 (Multi-feature Budgeted Profit Maximization (MBPM) Problem)**.** *Given $G = (V, E)$, q pieces of information about q features of a product are spread on the social network according to the MF-model. The MBPM problem seeks for a seed set $S \subseteq V$ with expected activation cost at most B, i.e., $\mathbb{E}[c(S)] \leq B$, to maximize the total expected profit $P(S)$.*

Given the equivalent diffusion process of the MF-model, we could solve the MBPM problem by solving the profit maximization problem on the multi-level graph.

Then the MBPM problem can be formulated as:

$$\begin{aligned} \max \quad & P(S) = \sum_{\phi \in \Omega} \Pr[\phi] \cdot \sum_{u \in V} \Pr[u \in I_\phi(S)] \cdot b(u) \\ s.t. \quad & \mathbb{E}[c(S)] \leq B \end{aligned} \tag{5.4}$$

Based on the proof of Theorem 5.3.3, we have the following result.

Theorem 5.4.2. *The objective function of MBPM Problem is monotone submodular with respect to the seed set of the product.*

5.4.2 Algorithm

Before presenting the algorithm of MBPM problem, we first introduce another problem, maximization of a monotone submodular function under the cardinality constraint. Let $g : 2^V \to R_{\geq 0}$ be a monotone submodular function. For $u \in V$ and $S \subseteq V$, the marginal gain by adding v to S is denoted as $g_v(S) = g(S \cup \{v\}) - g(S)$. For the problem $\max_{S \subseteq V, |S| \leq k} g(S)$, classical greedy scheme could return $(1 - 1/e)$-approximation solutions ((Nemhauser et al., 1978)). The algorithm always selects the element with largest marginal gain to current selected set until k nodes are chosen. That is, for current selected set S_0, the algorithm will select $v^* = \arg\max_{v \in V \setminus S_0} g_v(S_0)$ and add it into S_0. Under cardinality constraint, each

Algorithm 13: Modified Greedy Algorithm

$S \leftarrow \emptyset$;
while $c(S) < B$ **do**
  $v^* = \arg\max_{v \in V \setminus S} \frac{P(S \cup \{v\}) - P(S)}{c(v)}$;
  if $c(S) + c(v^*) > B$ **then**
    break with $1 - \frac{B - c(S)}{c(v^*)}$ probability;
  $S \leftarrow S \cup \{v^*\}$;
return S;

node actually has a cost of 1 and greedy scheme always selects the element with the largest marginal gain per unit cost.

Since the objective of MBPM problem is monotone submodular, inspired by ideas of classical greedy scheme, we could utilize Algorithm 13 to solve it. Assume current selected set is S. Alg. 13 always selects the node v^* with largest ratio of marginal gain to S to cost among remaining nodes. If $c(S) + c(v^*) \leq B$, v^* will be added into S. Otherwise, add v^* to S with $\frac{B-c(S)}{c(v^*)}$ probability or break with probability $1 - \frac{B-c(S)}{c(v^*)}$. It could guarantee that the output S satisfies $\mathbb{E}[c(S)] \leq B$. For the node found in the last iteration of our Algorithm, it will be selected with a very low probability if it is far more than the remaining budget.

Let $v_1, \ldots, v_n$ be the result sorted by increasing order of activation cost of nodes in V. That is, $c(v_1) \leq c(v_2) \leq \ldots \leq c(v_n)$. Denote p as the minimum number satisfying $\sum_{i=1}^{p} c(v_i) \geq B$. Assume $p < n$. Otherwise, we could select all nodes as the seeds. Then we know Algorithm 13 would execute at most p iterations.

Theorem 5.4.3. *Algorithm 13 could achieve a* $(1 - 1/e)$ *expected approximation guarantee of MBPM problem. The algorithm requires* $O(n^2)$ *function value computation.*

Proof. Since the expected knapsack constraint is somewhat different from the classical knapsack constraint, we think it's necessary to provide the proof for this theorem here. Our proof is inspired by (Nemhauser et al., 1978). Assume $S^* = \{u_1, \ldots, u_t\}$ is an optimal solution to

MBPM problem. Let $S_r = \{v_1, \ldots, v_r\}$ be the node set obtained by Algorithm 13 after r iterations and $S_0 = \emptyset$. Assume $v_{r+1} = \arg\max_{v \in (V \setminus S_r)} \{\frac{P(S_r \cup \{v\}) - P(S_r)}{c(v)}\}$. Assume $c(S_r) \leq B$ and $c(S_r) + c(v_{r+1}) > B$. Let S_G be the node set returned by Algorithm 13. Denote $P_v(S_i) = P(S_i \cup \{v\}) - P(S_i)$. Since $P(\cdot)$ is monotone submodular, for $1 \leq i \leq r$, we have

$$\begin{aligned}
&P(S^*) = P(S_i) + P_{u_1}(S_i) + P_{u_2}(S_i \cup \{u_1\}) + \ldots + P_{u_t}(S_i \cup \{u_1, \ldots, u_{t-1}\}) \\
&\leq P(S_i) + c(u_1) \cdot \frac{P_{v_{i+1}}(S_i)}{c(v_{i+1})} + \ldots + c(u_t) \cdot \frac{P_{v_{i+1}}(S_i)}{c(v_{i+1})} \\
&\leq P(S_i) + B \cdot \frac{P(S_{i+1}) - P(S_i)}{c(v_{i+1})}.
\end{aligned}$$

Denote $a_i = P(S^*) - P(S_i)$. Then we know $a_i \leq \frac{B}{c(v_{i+1})}(a_i - a_{i+1})$. Thus, $a_{i+1} \leq (1 - \frac{c(v_{i+1})}{B})a_i \leq \Pi_{j=1}^{i+1}\left(1 - \frac{c(v_j)}{B}\right) a_0$. Therefore,

$$P(S_{i+1}) \geq \left(1 - \Pi_{j=1}^{i+1}\left(1 - \frac{c(v_j)}{B}\right)\right) P(S^*).$$

Then we have

$$\begin{aligned}
&\mathbb{E}[P(S_G)] = \frac{B - c(S_r)}{c(v_{r+1})} P(S_{r+1}) + \left(1 - \frac{B - c(S_r)}{c(v_{r+1})}\right) P(S_r) \\
&\geq P(S_r) + (B - c(S_r)) \cdot \frac{P(S^*) - P(S_r)}{B} = \left(1 - \frac{c(S_r)}{B}\right) P(S^*) + \frac{c(S_r)}{B} P(S_r) \\
&\geq \left(1 - \frac{c(S_r)}{B}\right) P(S^*) + \frac{c(S_r)}{B}\left(1 - \prod_{j=1}^{r}\left(1 - \frac{c(v_j)}{B}\right)\right) P(S^*) \\
&= \left(1 - \prod_{j=1}^{r}\left(1 - \frac{c(v_j)}{B}\right) \cdot \frac{c(S_r)}{B}\right) P(S^*) \\
&= \left(1 - \prod_{j=1}^{r}\left(1 - \frac{c(v_j)}{B}\right)\left(1 - \left(1 - \frac{c(S_r)}{B}\right)\right)\right) P(S^*) \\
&\geq 1 - \exp\left(-\sum_{j=1}^{r} \frac{c(v_j)}{B}\right) \cdot \exp\left(-\left(1 - \frac{c(S_r)}{B}\right)\right) P(S^*) \\
&= (1 - 1/e) P(S^*).
\end{aligned}$$

□

5.5 Adaptive Multi-feature Budgeted Profit Maximization Problem

In practice, the decision maker may select one seed at a time and then observe the propagation result. He could make a choice to select the next seed based on currently observed results. And this strategy is usually called adaptive seed selection strategy. This strategy may bring more advantages and profits since the decision maker could adaptively revise the strategy according to the current situation rather than select all seeds once before the actual propagation process starts. Therefore, it is worth considering whether adaptive selection strategy helps a lot or not. In this section, we will introduce the adaptive MBPM problem and some related definitions.

5.5.1 Problem Definition

In the adaptive MBPM problem, we also choose seeds S from $G = (V, E)$ and observe the propagation process of corresponding seeds $\widehat{S}$ in its multi-level graph $\widehat{G}$ like in the non-adaptive MBPM problem. But under the adaptive setting, seeds are selected one by one and we need observe the diffusion result once a seed u is chosen. Specifically, thresholds for each node $v \in V$ are sampled independently uniformly at random from $[0, 1]$ at first. Then we select one seed at each step. When node u is selected as the next seed, equivalently we infect all of its feature nodes $u_1, \ldots, u_q$. Then we need to observe states of edges in $\widehat{G}$: observe the propagation result of u_i on G^i (related edges are live or blocked), $i = 1, \ldots, q$. After all q diffusions on $\widehat{G}$ stop, we could determine whether nodes in G not buying the product before selecting u would purchase the product or not now, according to the current propagation results on $\widehat{G}$. Then we select next seed and repeat this process until there is no seed budget.

In this adaptive seeding process, after selecting a node $u \in V$ and all feature nodes $u_1, \ldots, u_q$ of u are infected, we could observe all edges exiting $v_i, i = 1, \ldots, q$, which can be reached from u_i by currently live edges in G^i. That is, the full-adoption feedback model (Golovin and Krause, 2011) is considered in this paper. Our observation so far could be

described by the partial realization φ, a function mapping from currently observed items to their states. For $(u_i, v_i) \in \widehat{E}$, $\varphi((u_i, v_i)) \in \{0, 1, ?\}$ and $\varphi((u_i, v_i)) = 1$ (resp. 0) if edge (u_i, v_i) has been observed live (resp. blocked). $\varphi((u_i, v_i)) =?$ if the status of (u_i, v_i) is not known yet.

For any partial realization φ, define the domain $\mathrm{dom}(\varphi)$ of φ as the seed set for the product that has already been picked from V. Denote $dx(\varphi)$ as the set of edges in $\widehat{E}$ whose states have been known under φ. Let $\phi : \widehat{E} \to \{0, 1\}$ be a full realization of $\widehat{G}$. We say a partial realization φ is consistent with ϕ if they are equal everywhere in the domain of φ, denoted by $\phi \sim \varphi$. If φ and φ' are both consistent with some full realization ϕ, satisfying $\mathrm{dom}(\varphi) \subseteq \mathrm{dom}(\varphi')$, we say φ is a subrealization of φ', denoted as $\varphi \subseteq \varphi'$.

Let $\pi(\tau)$ be a randomized policy where τ represents all random sources of the randomized policy. Specifically, $\pi(\tau)$ is a function mapping from an already chosen seed set $S \subseteq V$ and a set of partial realizations to V, specifying which node to select as the next seed of the product within the budget. Let $S(\pi(\tau), \phi)$ be the set of nodes in V chosen by $\pi(\tau)$ under realization ϕ. Let $I^i_\phi(S(\pi(\tau), \phi))$ be the set of nodes in V^i accepting feature i when diffusion process of feature nodes of $S(\pi(\tau), \phi)$ on ϕ terminates. The profit obtained by policy $\pi(\tau)$ under realization ϕ is defined as:

$$f\left(S(\pi(\tau), \phi), \phi\right) = \sum_{u \in V} b(u) \cdot [\sum_{\substack{1 \le i \le q: \\ u_i \in I^i_\phi(S(\pi(\tau), \phi))}} w^i_u]. \tag{5.5}$$

Thus, the expected profit obtained by policy $\pi(\tau)$ can be formulated as:

$$f_{\mathrm{avg}}(\pi(\tau)) = \mathbb{E}_\Phi \left[f\left(S(\pi(\tau), \Phi), \Phi\right)\right]. \tag{5.6}$$

Definition 5.5.1 (Adaptive Multi-feature Budgeted Profit Maximization (AMBPM) Problem)**.** *Given $G = (V, E)$, assume q pieces of information about q features of a product are spread on G according to the MF-model. The AMBPM problem seeks for a randomized policy*

to maximize the total expected profit obtained:

$$\max_{\pi} \mathbb{E}_{\tau}[f_{avg}(\pi(\tau))]$$

$$s.t.\ \mathbb{E}_{\tau}[c(S(\pi(\tau), \phi))] \leq B,\ \textit{for any realization } \phi$$

Definition 5.5.2 (Conditional Expected Marginal Profit)**.** *Given a partial realization φ and a node u, the conditional expected marginal profit of u conditioned on having observed φ is defined as:*

$$\Delta(u|\varphi) = \mathbb{E}\left[f(dom(\varphi) \cup \{u\}, \Phi) - f(dom(\varphi), \Phi)|\Phi \sim \varphi\right], \tag{5.7}$$

where the expectation is taken over $p(\phi|\varphi) = \mathbb{P}(\Phi = \phi|\Phi \sim \varphi)$.

Definition 5.5.3 (Adaptive Monotonicity)**.** *A function $f(\cdot, \phi)$ is adaptive monotone with respect to distribution $p(\phi)$, if for all partial realization φ with $\Pr[\Phi \sim \varphi] > 0$ and all $u \notin dom(\varphi)$, we have $\Delta(u|\varphi) \geq 0$.*

Definition 5.5.4 (Adaptive Submodularity)**.** *A function $f(\cdot, \phi)$ is adaptive submodular with respect to distribution $p(\phi)$, if for all partial realizations φ and φ' satisfying $\varphi \subseteq \varphi'$ and for all $u \notin dom(\varphi')$, we have $\Delta(u|\varphi) \geq \Delta(u|\varphi')$.*

Theorem 5.5.5. *The objective function $f(\cdot, \phi)$ is adaptive monotone and adaptive submodular.*

Proof. We first show adaptive monotonicity of f. Consider a fixed partial realization φ. For a node $u \notin \mathrm{dom}(\varphi)$, when selecting u as the seed under φ, if all feature nodes $u_1, \ldots, u_q$ of u have been infected before u is selected under φ, then for any realization $\phi \sim \varphi$, we have $f(dom(\varphi) \cup \{u\}, \phi) = f(dom(\varphi), \phi)$. Otherwise, there exists at least one of $u_1, \ldots, u_q$ not infected before u is selected, and assume u_1 is one of the feature node satisfying the condition. Then for any realization $\phi \sim \varphi$, we have $f(dom(\varphi) \cup \{u\}, \phi) - f(dom(\varphi), \phi) \geq b(u) \cdot w_u^1 \geq 0$. Thus, no matter which case happens, for any realization $\phi \sim \varphi$, $f(dom(\varphi) \cup \{u\}, \phi) \geq$

$f(dom(\varphi), \phi)$ always holds. Since $\Delta(u|\varphi)$ is a linear combination of each realization $\phi \sim \varphi$, we know that $\Delta(u|\varphi) \geq 0$.

Next we prove the adaptive submodularity of f. For any pairs of partial realizations φ, φ' satisfying $\varphi \subseteq \varphi'$ and any $u \notin \text{dom}(\varphi')$, we have to show $\Delta(u|\varphi) \geq \Delta(u|\varphi')$. Our proof is inspired by the proof technique in (Golovin and Krause, 2011) and (Guo and Wu, 2020b).

Consider two fixed partial realizations φ, φ' satisfying $\varphi \subseteq \varphi'$. Assume there are two realizations ϕ and ϕ' with $\phi \sim \varphi, \phi' \sim \varphi'$, satisfying $\phi((u_i, v_i)) = \phi'((u_i, v_i))$ for all $(u_i, v_i) \notin dx(\varphi')$. Thus, ϕ and ϕ' have the same area $\beta = \varphi \cup (\phi' \setminus \varphi')$.

Let $\sigma(\text{dom}(\varphi) \cup \{u\}, \phi) = \cup_{i=1}^{q} I_{\phi}^{i}(\text{dom}(\varphi) \cup \{u\}, \phi))$ be the set of infected feature nodes in $\widehat{G}$ when feature nodes of $\text{dom}(\varphi) \cup \{u\}$ are seeds under the realization ϕ. Denote $T = \sigma(\text{dom}(\varphi) \cup \{u\}, \phi)$ and $M = \sigma(\text{dom}(\varphi), \phi)$. Let $N = T \setminus M$. Similarly, denote $T' = \sigma(\text{dom}(\varphi') \cup \{u\}, \phi')$ and $M' = \sigma(\text{dom}(\varphi'), \phi')$, and let $N' = T' \setminus M'$.

We first show that $M \subseteq M'$. Fix $w_i \in M$. Then there must exist a path P_i from some feature node v_i of $v \in \text{dom}(\varphi)$ to w_i. Therefore, edges on path P_i are observed to be live by φ. Since $\phi \sim \varphi, \phi' \sim \varphi'$ and $\varphi \subseteq \varphi'$, edges observed by φ have same states in ϕ and ϕ'. That is, each edge on P_i is also live under ϕ'. Since $\varphi \subseteq \varphi'$, it is clear that $v \in \text{dom}(\varphi')$. Therefore, w_i will be i-accepted when feature nodes of $\text{dom}(\varphi')$ are seeds in $\widehat{G}$ under realization ϕ', i.e., $w_i \in M'$.

We next show $N' \subseteq N$. We prove this by contradiction. Fix $v_j \in N'$. Assume $v_j \notin N$. Since $v_j \in N'$ and $M' \cap N' = \emptyset$, we have that $v_j \notin M'$. Since we have proven $M \subseteq M'$, it is obvious that $v_j \notin M$. As $v_j \in N'$, there must exist some path P_j from u_j to v_j in ϕ' but at least one edge on path P_j is blocked in ϕ. Assume one such edge is (s_j, t_j). Since the status of edge (s_j, t_j) is different in realization ϕ and ϕ', and ϕ and ϕ' have the same area β, thus (s_j, t_j) must be observed by φ' but not by φ. Since (s_j, t_j) is observed by φ', s_j must be infected after selecting $\text{dom}(\varphi')$ according to the full-adoption feedback model. That is, s_j and the nodes that can be reachable from s_j must be infected after we select $\text{dom}(\varphi')$.

Therefore, s_j and the nodes that can be reachable from s_j, including v_j, will belong to M', a contradiction.

Define

$$\begin{aligned}
&\delta(u|\phi, \phi \sim \varphi) = f(\mathrm{dom}(\varphi) \cup \{u\}, \phi) - f(\mathrm{dom}(\varphi), \phi) \\
=&\sum_{v \in V} b(v) \sum_{\substack{1 \le i \le q: \\ v_i \in T}} w_v^i - \sum_{v \in V} b(v) \sum_{\substack{1 \le i \le q: \\ v_i \in M}} w_v^i \\
=&\sum_{v \in V} b(v) \sum_{\substack{1 \le i \le q: \\ v_i \in (T \setminus M)}} w_v^i = \sum_{v \in V} b(v) \sum_{\substack{1 \le i \le q: \\ v_i \in N}} w_v^i.
\end{aligned}$$

Since we have shown that $N' \subseteq N$, we could obtain that $\delta(u|\phi, \phi \sim \varphi) \geq \delta(u|\phi', \phi' \sim \varphi')$. Since $\sum_{\phi \sim \beta} \Pr[\phi|\phi \sim \beta] = 1$, we know

$$\begin{aligned}
\Delta(u|\varphi) &= \sum_{\phi \sim \varphi} \Pr[\phi|\phi \sim \varphi] \cdot \delta(u|\phi, \phi \sim \varphi) \\
&= \sum_{\phi' \sim \varphi'} \Pr[\phi'|\phi' \sim \varphi'] \sum_{\phi \sim \beta} \Pr[\phi|\phi \sim \beta] \cdot \delta(u|\phi, \phi \sim \varphi) \\
&\geq \sum_{\phi' \sim \varphi'} \Pr[\phi'|\phi' \sim \varphi'] \sum_{\phi \sim \beta} \Pr[\phi|\phi \sim \beta] \cdot \delta(u|\phi', \phi' \sim \varphi') \\
&= \sum_{\phi' \sim \varphi'} \Pr[\phi'|\phi' \sim \varphi'] \cdot \delta(u|\phi', \phi' \sim \varphi') = \Delta(u|\varphi'),
\end{aligned}$$

which completes the proof. □

5.6 Algorithm and Theoretical Analysis

Since the objective $f(\cdot, \phi)$ of AMBPM problem is adaptive monotone and adaptive submodular, we could utilize adaptive greedy policy proposed in (Golovin and Krause, 2011) to solve it. The seed selection rule of adaptive greedy policy is straightforward, i.e., always selecting the node with the largest ratio of conditional expected marginal profit to cost. However, given a partial realization φ and a node $u \notin \mathrm{dom}(\varphi)$, it is difficult to compute the conditional expected marginal profit $\Delta(u|\varphi)$ since there are almost exponential possible realizations ϕ

Algorithm 14: Adaptive-Greedy

$S \leftarrow \emptyset$;
$\varphi = \{?\}^{\widehat{E}}$;
while $c(S) < B$ **do**
  $v_{max} = \arg\max_{v \in V \setminus S} \Delta(v|\varphi)/c(v)$;
  if $c(S) + c(v_{max}) > B$ **then**
    break with $1 - \frac{B-c(S)}{c(v_{max})}$ probability;
  $S \leftarrow S \cup \{v_{max}\}$;
  Observe the node set $A(v_{max})$ infected by feature nodes of v_{max}, $1 \leq i \leq q$;
  Update φ by updating states of edges related to nodes in $A(v_{max}) \cup \hat{v}_{max}$;
return $S, f(S, \varphi)$;

with $\phi \sim \varphi$. This section would consider algorithms of AMBPM problem under both the oracle model and noise model.

5.6.1 Adaptive Greedy Algorithm under the Oracle Model

Under the oracle model, assume conditional expected marginal profit of any node under any partial realization can be obtained in constant time. Define a randomized adaptive greedy policy $\pi_{ag}(\tau)$. The main idea of adaptive greedy policy to solve this problem can be seen in Algorithm 14, which is based on the adaptive greedy policy proposed in (Golovin and Krause, 2011). Under the current partial realization φ and seed set S, the $\pi_{ag}(\tau)$ would select a node v_{max} satisfying $v_{max} := \arg\max_{v \in V \setminus S}\{\frac{\Delta(v|\varphi)}{c(v)}\}$. If $c(S) + c(v_{max}) \leq B$, then v_{max} is the next seed. Otherwise, $\pi_{ag}(\tau)$ would select v_{max} as the next seed with probability $\frac{B-c(S)}{c(v_{max})}$. After selecting v_{max}, we observe the nodes infected by feature nodes of v_{max}, denoted by $A(v_{max})$ and update the partial realization φ by changing states of edges related to nodes in $A(v_{max}) \cup \hat{v}_{max}$ from ? to 0 or 1. The algorithm repeats the above process, and terminates until $c(S) \geq B$, or terminates with a probability. In this way, we could guarantee $\mathbb{E}[c(S)] \leq B$. The random source τ in this adaptive greedy policy indicates whether to contain the node found in the last iteration.

Since the objective $f(S, \phi)$ of AMBPM problem is adaptive monotone and adaptive submodular, according to the result in (Golovin and Krause, 2011), we have the following conclusion.

Theorem 5.6.1. *The adaptive greedy policy shown in Algorithm 14 could obtain a* $(1-1/e)$ *expected approximation solution of the AMBPM problem. It requires* $O(n^2)$ *function value computations.*

5.6.2 Adaptive Greedy Algorithm under the Noise Model

This section will present algorithms of AMBPM problem under the noise model. The basic seed selection strategy is similar to that in oracle model, but the difference is we will estimate the conditional expected marginal profit of any node under a fixed partial realization, $\Delta(u|\varphi)$, by the reverse influence sampling technique. However, maximizing the estimation of $\Delta(u|\varphi)$ by sampling technique is likely to obtain an extremely worse node with some probability, although the probability is very small. That is, the node u^* maximizing the estimation may not be the optimal solution to $\max_{v \in V \setminus S} \Delta(u|\varphi)/c(u)$. In this case, the expected approximation ratio in Theorem 5.6.1 is not guaranteed.

5.6.2.1 Technique

Definition 5.6.2 (Reverse Reachable (RR) set (Tang et al., 2014))**.** *For any graph realization* $\phi \in \Omega$ *and* $v_i \in \widehat{V}$*, the RR set for* v_i *is denoted by* $R_\phi(v_i)$*, which contains all nodes that could reach* v_i *in* ϕ*.* v_i *is called the target node of* $R_\phi(v_i)$*.*

Intuitively, RR set $R_\phi(v_i)$ of v_i contains feature nodes that are likely to infect v_i during the propagation. A random RR set is an RR set whose target node v_i is selected randomly from $\widehat{V}$. Given a random RR set $R_\phi(v_i)$ and $\widehat{S} \subseteq \widehat{V}$, we say $\widehat{S}$ covers $R_\phi(v_i)$ if $\widehat{S} \cap R_\phi(v_i) \neq \emptyset$. A set with larger expected influence has a higher probability to cover a random RR set.

Specifically, given a graph $G = (V, E)$ and a random RR set R, the expected influence $\mathbb{E}[I(S)]$ of a set S in G is $\mathbb{E}[I(S)] = |V| \cdot \Pr[S \cap R \neq \emptyset]$ (Tang et al., 2014). Therefore, if we could generate a large number of random RR sets, a set with large expected influence would cover a large amount of the generated random RR sets. We will use this idea in our estimation of the conditional expected marginal profit of a node.

Given a partial realization φ, let $G_\varphi = (V_\varphi, E_\varphi)$ be the subgraph induced by the i-unaccepted nodes under φ, $i = 1, \ldots, q$. That is, G_φ is obtained by deleting all of the i-accepted feature nodes and their related edges in $\widehat{G}$, $i = 1, \ldots, q$. Let Ω_φ be the set containing all realizations of G_φ. Denote $W = \sum_{v_i \in V_\varphi} b(v) \cdot w_v^i$. Assume each node $v_i \in V_\varphi$ is selected randomly from V_φ with probability $\frac{b(v) \cdot w_v^i}{W}$ as the target node of an RR set.

Given a partial realization φ and $u \in V$, let R_φ be a random RR set generated from a realization $\phi \in \Omega_\varphi$. Define

$$h(u, R_\varphi) = \begin{cases} 1, & \text{if } \widehat{u} \cap R_\varphi \neq \emptyset \\ 0, & \text{otherwise} \end{cases}. \tag{5.8}$$

By the reverse Breadth First Search algorithm (Moore, 1959), we could produce a large number of random RR sets $\mathcal{R}(\varphi) = \{R_1, R_2, \ldots, R_\alpha\}$ of G_φ. Define $F_{\mathcal{R}(\varphi)}(u) = \frac{1}{\alpha} \sum_{j=1}^{\alpha} h(u, R_j)$. Denote $\rho(u|\varphi) = W \cdot F_{\mathcal{R}(\varphi)}(u) = W \cdot \frac{1}{\alpha} \sum_{j=1}^{\alpha} h(u, R_j)$. Then the following result holds.

Theorem 5.6.3. *Given a node $u \in V$ and a partial realization φ, we have $\mathbb{E}[\rho(u|\varphi)] = \Delta(u|\varphi)$.*

Proof. Given a realization $\phi \in \Omega_\varphi$ and a user node $u \in V$, let $I^i_\phi(u)$ be the feature nodes infected by feature node u_i under ϕ. Then we have

$$\mathbb{E}\left[\rho(u|\varphi)\right] = W \cdot \mathbb{E}[\frac{1}{\alpha}\sum_{j=1}^{\alpha} h(u, R_j)] = W \cdot \sum_{\phi\sim\Omega_\varphi} \Pr[\phi] \sum_{v_i \in V_\varphi} \Pr[v_i] \cdot h(u, R_\varphi(v_i))$$

$$= \sum_{\phi\sim\Omega_\varphi} \Pr[\phi] \sum_{v_i \in V_\varphi} b(v) \cdot w^i_v \cdot h(u, R_\varphi(v_i))$$

$$= \sum_{\phi\sim\Omega_\varphi} \Pr[\phi] \sum_{v\in V} b(v) \cdot \sum_{\substack{1\le i\le q:\\ v_i \in I^i_\phi(u)}} w^i_v$$

$$= \sum_{\phi\sim\Omega} \Pr[\phi|\phi\sim\varphi] \cdot \sum_{v\in V} b(v) \sum_{\substack{1\le i\le q:\\ v_i \in \left(I^i_\phi(\mathrm{dom}(\varphi)\cup\{u\})\setminus I^i_\phi(\mathrm{dom}(\varphi)\right)}} w^i_v$$

$$= \sum_{\phi\sim\Omega} \Pr[\phi|\phi\sim\varphi] \cdot [f(\mathrm{dom}(\varphi)\cup\{u\}, \phi) - f(\mathrm{dom}(\varphi), \phi)] = \Delta(v|\varphi).$$

□

Given a partial realization φ and a set of random RR sets $R(\varphi)$ generated from subgraph G_φ, define $Q_{R(\varphi)}(u) = F_{\mathcal{R}(\varphi)}(u)/c(u)$. According to Theorem 5.6.3, we know $\mathbb{E}[W \cdot Q_{R(\varphi)}(u)] = W \cdot \mathbb{E}[Q_{R(\varphi)}(u)] = \Delta(u|\varphi)/c(u)$. Thus, $W \cdot Q_{R(\varphi)}(u)$ is an unbiased estimation of $\Delta(u|\varphi)/c(u)$. When $|R(\varphi)|$ is sufficiently large, $W \cdot Q_{R(\varphi)}(u)$ could be convergent to $\Delta(u|\varphi)/c(u)$. Thus, we could use $W \cdot Q_{R(\varphi)}(u)$ as an estimation for $\Delta(u|\varphi)/c(u)$.

Algorithm 15 show the adaptive greedy policy with the above sampling technique, named Sampled-AdapGreedy. It is denoted by $\pi_{sag}(\tau, \omega)$ where ω usually represents the random source of sampling. At each iteration, instead of finding a node maximizing $\Delta(u|\varphi)/c(u)$ from currently unselected user nodes, we select a node v^* which could maximize $Q_{R(\varphi)}(u)$, which is obtained by Algorithm 16 (Huang et al., 2020). If $c(v^*)$ is larger than the current remaining budget, then we add v^* into the current seed set with $(B - c(v^*))/c(v^*)$ probability. Otherwise, we add v^* to the current seed set, observe the corresponding propagation result on G_φ, and update partial realization φ and subgraph G_φ.

Algorithm 15: Sampled-AdapGreedy (SAG)

1 $S = \emptyset$;
2 $\varphi = \{?\}^{\widehat{E}}$;
3 $G_\varphi = G$;
4 $W = \sum_{u \in V} b(u)$;
5 $W^* = \min_{v_i \in \widehat{V}} b(v) \cdot w_v^i$;
6 **while** $c(S) < B$ **do**
7 $\quad T = V \setminus S$;
8 $\quad n_\varphi = |T|$;
9 $\quad v^* \leftarrow$ Modified-EPIC$(G_\varphi, T, W, W^*, n_\varphi, \epsilon)$;
10 $\quad$ **if** $c(S) + c(v^*) > B$ **then**
11 $\quad\quad$ break with $1 - \frac{B - c(S)}{c(v^*)}$ probability;
12 $\quad S \leftarrow S \cup \{v^*\}$;
13 $\quad$ Observe the node set $A(v^*)$ infected by the feature nodes of v^*, $1 \le i \le q$;
14 $\quad$ Update φ by updating states of edges related to nodes in $A(v^*) \cup \widehat{v^*}$;
15 $\quad W = W - \sum_{u_i \in A(v^*)} b(u) w_u^i - \sum_{v_i \in \widehat{v^*} \cap G_\varphi} b(v^*) w_v^i$;
16 $\quad$ Update G_φ by removing nodes in $A(v^*) \cup \widehat{v^*}$ and their corresponding edges;
17 **return** S and $f(S, \varphi)$;

5.6.2.2 Theoretical Analysis

At each iteration of Algorithm 15, it needs use Algorithm 16 (line 9 of Alg. 15) to obtain a node which could achieve the maximum of function $Q_{R(\varphi)}(u)$. Alg. 16 is obtained by modifying the EPIC algorithm proposed in (Huang et al., 2020). However, there are some difference between EPIC and Modified-EPIC (MEPIC) algorithm: (1) The seed selected at each iteration is one in MEPIC. (2) The target estimation function in MEPIC is $Q_{R(\varphi)}(u)$ instead of $F_{R(\varphi)}(u)$.

At each iteration of Algorithm 15, denote the current partial realization as φ. We could obtain its corresponding induced subgraph G_φ. Alg. 16 first initializes some parameters and then generates two same size sets of random RR sets of G_φ, $\mathcal{R}_1(\varphi)$ and $\mathcal{R}_2(\varphi)$. At each iteration, it chooses a node v^* maximizing $Q_{R_1(\varphi)}(\cdot)$, which can be achieved in polynomial time. Assume $v_{max} = \arg\max_{v \in T} \Delta(v|\varphi)/c(v)$.Then $Q^u(v_{max}) = Q_{R_1(\varphi)}(v^*) \ge Q_{R_1(\varphi)}(v_{max})$. That is, $Q^u(v_{max})$ is an upper bound of $Q_{R_1(\varphi)}(v_{max})$. And $W \cdot Q^l(v^*)$ is an accurate lower

Algorithm 16: Modified-EPIC (MEPIC) ((Huang et al., 2020))

1 $\delta = 0.01 \cdot \epsilon / W$;
2 $\epsilon' = (\epsilon - \delta \cdot W)/(1 - \delta \cdot W)$;
3 $\bar{\epsilon} = \epsilon'/(1 - \epsilon')$;
4 $i_{max} = \lceil \log_2 \frac{(2+2\bar{\epsilon}/3)\cdot W}{\bar{\epsilon}^2} \rceil + 1$ and $a = \ln(\frac{2 \cdot i_{max}}{\delta})$;
5 $\theta_0 = \frac{1}{W^*}\left(\ln \frac{2}{\delta} + \ln\left(n_\varphi\right)\right)$;
6 Generate two sets of random RR sets $\mathcal{R}_1(\varphi)$ and $\mathcal{R}_2(\varphi)$ of G_φ with $|\mathcal{R}_1(\varphi)| = |\mathcal{R}_2(\varphi)| = \theta_0$;
7 **for** $i = 1$ *to* i_{max} **do**
8 $\quad v^* = \arg\max_{v \in T} Q_{R_1(\varphi)}(v)$;
9 $\quad Q^u(v_{max}) \leftarrow Q_{R_1(\varphi)}(v^*)$;
10 $\quad Q^l(v^*) \leftarrow \left(\sqrt{Q_{R_2(\varphi)}(v^*) + \frac{2a}{9|R_2(\varphi)|}} - \sqrt{\frac{a}{2|R_2(\varphi)|}}\right)^2 - \frac{a}{18 \cdot |R_2(\varphi)|}$;
11 $\quad$ **if** $\frac{Q^l(v^*)}{Q^u(v_{max})} \geq 1 - \epsilon'$ *or* $i = i_{max}$ **then**
12 $\quad\quad$ **return** v^*;
13 $\quad$ Double the sizes of $R_1(\varphi)$ and $R_2(\varphi)$ with new random RR sets;

bound of $\Delta(v^*|\varphi)/c(v^*)$ with high probability. Then MEPIC checks whether the stopping condition (line 11) is satisfied. If satisfied, it returns v^* as output. Otherwise, it doubles the size of $\mathcal{R}_1(\varphi)$ and $\mathcal{R}_2(\varphi)$, and repeats the above process.

Lemma 5.6.4. *Given a partial realization φ and its corresponding induced subgraph $G_\varphi = (V_\varphi, E_\varphi)$, denote by T the set of current unselected nodes in V. Then MEPIC algorithm could return a user node v^* satisfying that*

$$\mathbb{E}_\tau\left[\frac{\Delta(v^*|\varphi)}{c(v^*)}\right] \geq (1-\epsilon) \cdot \max_{v \in T}\left\{\frac{\Delta(v|\varphi)}{c(v)}\right\}, \tag{5.9}$$

within $O((|V_\varphi| + |E_\varphi|)(\log(|T|) + \log\frac{1}{\epsilon})/\epsilon^2)$ expected time.

Proof. Given a partial realization φ and its corresponding induced subgraph $G_\varphi = (V_\varphi, E_\varphi)$, the target function $Q_{R_1(\varphi)}(v) = F_{R_1(\varphi)}(u)/c(u)$ is a weighted coverage function on $R_1(\varphi)$, where the weight for each $u \in (V \setminus \mathrm{dom}(\varphi))$ is $1/c(u)$. Since $Q_{R_1(\varphi)}(v)$ is a monotone submodular function and maximizing a monotone submodular weighted coverage function

can be solved in polynomial time, thus the node $v^* = \arg\max_{v \in T} Q_{R_1(\varphi)}(v)$ can be obtained in polynomial time. Also, the expected approximation guarantee can be obtained accordingly from results of EPIC algorithm in (Huang et al., 2020). □

Recall that p is the minimum number satisfying $\sum_{i=1}^{p} c(v_i) \geq B$. Then we know Algorithm 15 would execute at most p iterations. Now, we could give the approximation guarantee of our AG algorithm.

Theorem 5.6.5. *Given $\epsilon \in (0, 1)$, the sampled adaptive greedy policy $\pi_{sag}(\tau, \omega)$ (Algorithm 15) could achieve a $(1 - e^{-(1-\epsilon)})$ expected approximation ratio within $O(pq \cdot (n + m)(\log(n + \log \frac{1}{\epsilon})/\epsilon^2)$ expected time. That is, for any realization ϕ and any policy $\pi(\tau)$ satisfying $\mathbb{E}_\tau[c(S(\pi(\tau), \phi))] \leq B$, we have*

$$\mathbb{E}_\tau[\mathbb{E}_\omega[f_{avg}(\pi_{sag}(\tau, \omega))]] \geq (1 - e^{-(1-\epsilon)}) \cdot \mathbb{E}_\tau[f_{avg}(\pi(\tau))]. \tag{5.10}$$

Proof. According to Lemma 5.6.4, the node selected in each iteration of Algorithm 15 satisfies $(1-\epsilon)$ expected approximation. Since Algorithm 15 would execute no more than p iterations, thus the total expected error of all iterations is $(1/p) \cdot \sum_{i=1}^{p} \epsilon = \epsilon$. According to the Theorem 5.4.3 and Lemma 5.6.4, Theorem 5.6.5 holds by inferring from Theorem 6 in (Huang et al., 2020). □ □

5.7 Experiments

We verify efficiencies of our proposed policy by comparing the running time and its obtained profit with other algorithms. We run experiments on a Linux machine with an Intel Xeon 3.5GHz CPU and 32GB RAM. For each dataset, 30 possible realizations are produced randomly and the average performance of each algorithm is reported.

5.7.1 Experimental Setup

Datasets. Five real-world social network datasets are used in this paper and detailed statistics are shown in Table 5.1. Epinions dataset could be found in (Leskovec and Krevl, 2014) and all other datasets are from (Rossi and Ahmed, 2015). According to the structure of MF-model, the number of nodes in multi-level graph is different from these basic information, which is also determined by the number of features. For the undirected graph, we replace each edge with two reversed directed edges.

Table 5.1. Dataset characteristics

Dataset	n	m	**Type**	**Average degree**
Twitter	0.8k	1k	directed	2
Wiki	0.9k	3k	directed	6
Hamsterster	2.4k	16.6k	undirected	13
DBLP	12.6k	49.7k	undirected	7.9
HepPh	12k	118k	undirected	19
Epinions	75.9k	508.8k	directed	13

Propagation Model and Parameters. We use the MF-model as diffusion model in experiments and for each edge $e = (u, v) \in E$, set $p^i_{u,v} = 1/|N^{in}(v)|, i = 1, \ldots, q$, where $N^{in}(v)$ is the set of in-neighbors of v. This setting is widely used in prior works (Tang et al., 2014; Goyal et al., 2011a; Jung et al., 2011). For $u \in V$, the weight vector of u is generated randomly from $(0, 1]^q$ such that the sum of weights of all features for u is 1. Also, we generate random numbers from $(0, 1]$ as the cost and profit of each node. For each dataset, we vary budget B such that $B \in \{0, 10, 20, 30, 40, 50\}$.

We conduct two groups of experiments to test the time efficiency and performance of our proposed policy, respectively. The first group of experiments is performed to verify the time efficiency of adaptive greedy policy (Algorithm 14) and sampled adaptive greedy policy (Algorithm 15). We compare the running time and obtained profit of adaptive greedy policy

and sampled adaptive greedy policy with their non-adaptive corresponding algorithms, with different implementations.

(1) Modified greedy algorithm sampled by Monte Carlo (MGMC): Shown as Algorithm 13 and the profit $P(S)$ of any node set S is estimated by Monte Carlo method.

(2) Modified greedy algorithm sampled by reverse influence sampling (MGRIS): Shown as Algorithm 13 and the profit $P(S)$ of any node set S is estimated by reverse influence sampling method. Let $Q = \sum_{u \in V} b(u)$. Each feature node v_i in multi-level graph $\widehat{G}$ is selected as a target node of a RR set with $b(v) \cdot w_v^i / Q$ probability. Let $\mathcal{R} = \{R_1, \ldots, R_\lambda\}$ be a set of random RR sets generated from $\widehat{G}$. Then it can be proved $K_{\mathcal{R}}(S) = Q \cdot F_{\mathcal{R}}(S)$ is an unbiased estimation of $P(S)$. According to Chernoff Bounds (Motwani and Raghavan, 1995), if $\lambda \geq \frac{(2+\eta)Q}{\eta^2 P(S)} \cdot \ln \frac{1}{\delta'}$, then for any node set S with $c(S) \leq B$, we have $\Pr[|P(S) - K_{\mathcal{R}}(S)| > \eta \cdot P(S)] < \delta'$. Let p^* be the minimum number such that $\sum_{j=p^*}^{n} c(v_j) \leq B$ and $Q^* = \sum_{j=p^*}^{n} b(v_j)$. By setting $\lambda = \frac{(2+\eta)Q}{\eta^2 Q^*} \cdot \ln \frac{1}{\delta'}$, we could guarantee $\lambda \geq \frac{(2+\eta)Q}{\eta^2 P(S)} \cdot \ln \frac{1}{\delta'}$. Here we set $\eta = \delta' = 0.1$.

(3) Adaptive greedy policy (AG): Shown as Algorithm 14 and the conditional expected marginal profit of a node u under any partial realization, $\Delta(u|\varphi)$, is estimated by Monte Carlo method.

(4) Sampled adaptive greedy policy (ASG): Shown as Algorithm 15 and set the error parameter $\epsilon = 0.5$.

The second group of experiments is to compare the performance of our SAG policy with three heuristic adaptive policies: Adaptive Random (AR), Adaptive Max-degree (AMD) and Adaptive Max-profit (AMP).

(1) AR is the adaptive version of the simple random algorithm. It uniformly selects currently unselected nodes in V as the next seed.

(2) AMD picks the node with maximum out-degree from currently unselected nodes in V as the next seed.

(3) Given the partial realization φ and currently selected seed set S, AMP selects the node u^* satisfying $u^* \in \arg\max_{u \in (V \setminus S)} \Delta(u|\varphi)$ and estimates $\Delta(u|\varphi)$ by reverse influence sampling technique. According to Theorem 5.6.3, we know $\rho(u|\varphi)$ is an unbiased estimation of $\Delta(u|\varphi)$. According to Chernoff Bounds (Motwani and Raghavan, 1995), if $\alpha \geq \frac{(2+\hat{\epsilon})W}{\hat{\epsilon}^2 \Delta(u|\varphi)} \cdot \ln \frac{1}{\delta'}$, then we have $\Pr[|\rho(u|\varphi) - \Delta(u|\varphi)| > \hat{\epsilon} \cdot \Delta(u|\varphi)] < \delta'$. Let $W^* = \min_{v_i \in \widehat{V}} b(v) \cdot w_v^i$. By setting $\alpha = \frac{(2+\hat{\epsilon})W}{\hat{\epsilon}^2 W^*} \cdot \ln \frac{1}{\delta'}$, we could guarantee $\alpha \geq \frac{(2+\hat{\epsilon})W}{\hat{\epsilon}^2 \Delta(u|\varphi)} \cdot \ln \frac{1}{\delta'}$. Here we set $\hat{\epsilon} = \delta' = 0.1$.

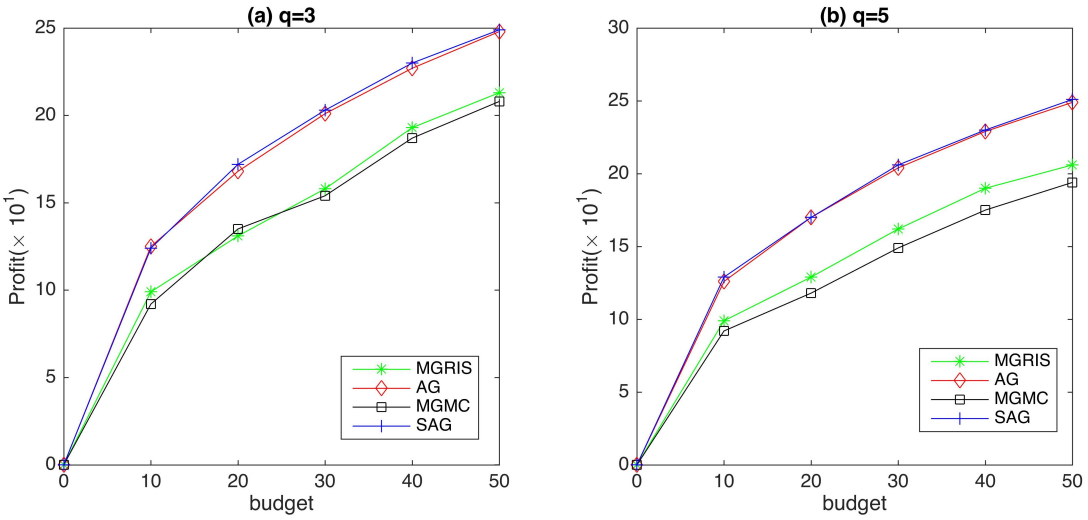


Figure 5.2. Profit VS budget on Twitter.

5.7.2 Experimental Results

5.7.2.1 Results of first group of experiment

Fig. 5.2 and Fig. 5.3 present results of the first group of experiments on Twitter and Wiki datasets. Fig. 5.2 and Fig. 5.3 present profits obtained by our proposed adaptive greedy policy (AG and SAG) with their non-adaptive versions (MGMC and MGRIS). We implement the experiments under two values of q, 3 and 5. Since AG policy and MGMC algorithm are implemented by Monte Carlo method and they are time-consuming, thus we only conduct

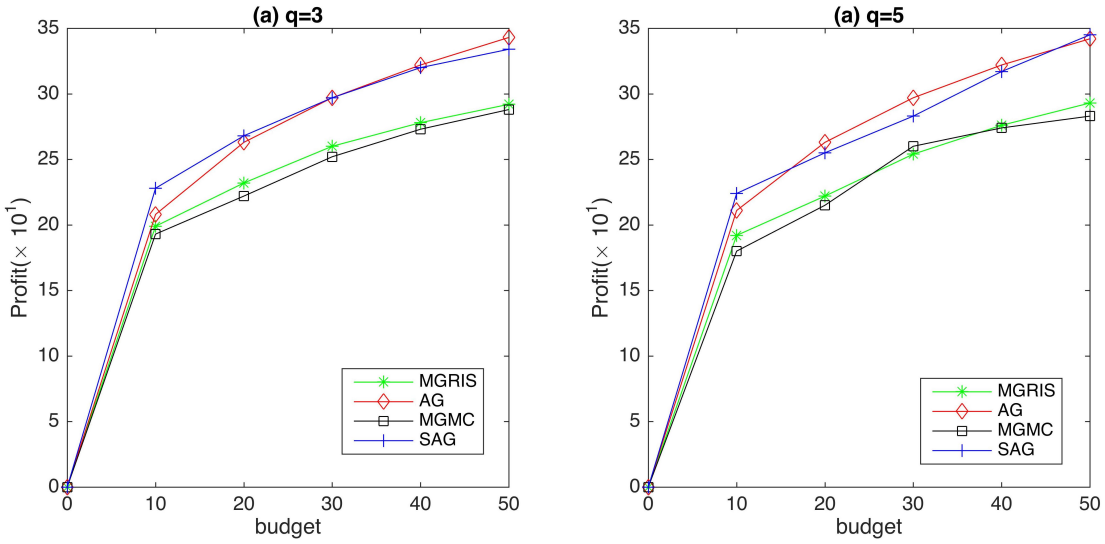


Figure 5.3. Profit VS budget on Wiki.

the first group of experiments on two small datasets. Here the number of Monte Carlo simulations is set to 500. The results show that AG and SAG policies are always evidently superior to MGMC and MGRIS algorithms with respect to the obtained profit, which shows the benefits of adaptive policies. The profits obtained by AG and SAG policies are very close, which indicates effectiveness of our sampling technique. The profits obtained by MGRIS and MGMC algorithms are close at most times, but in some cases, profits of MGMC are smaller than those of MGRIS. This may be because the number of Monte Carlo simulations is not enough.

Table 5.2 presents the running time of our proposed AG and SAG policies with MGMC and MGRIS on Twitter and Wiki datasets under budget 10, 30 and 50, respectively. To compare running time of different strategies fairly, parallel computing is not used here. We can see that MGRIS is fastest among all of the four algorithms since it only needs to generate a set of random RR sets once and choose seeds once. Our SAG policy is the second fastest strategy and faster than AG and MGMC. AG is much faster than MGMC and this may be because the induced subgraph of partial realization becomes smaller and smaller.

Table 5.2. Running time VS budget on Twitter and Wiki

Twitter						
	$q = 3$			$q = 5$		
Algorithm	**10**	**30**	**50**	**10**	**30**	**50**
MGRIS(s)	5.36	20.64	39.57	9.1	31.54	61.3
MGMC(h)	0.7	3.01	5.46	0.97	4.35	7.57
AG(s)	175.23	419.59	667.5	271.48	677.69	1012.19
SAG(s)	42.64	137.58	231.2	59.84	186.75	313.62
Wiki						
	$q = 3$			$q = 5$		
Algorithm	**10**	**30**	**50**	**10**	**30**	**50**
MGRIS(s)	7.31	25.82	46.75	10.37	39.03	68.52
MGMC(h)	3.56	11.41	19.01	6.67	22.2	36.92
AG(s)	186.81	445.79	690.42	268.85	639.21	970.45
SAG(s)	39.29	143.84	239.65	67.55	190.42	323.28

5.7.2.2 Results of second group of experiment

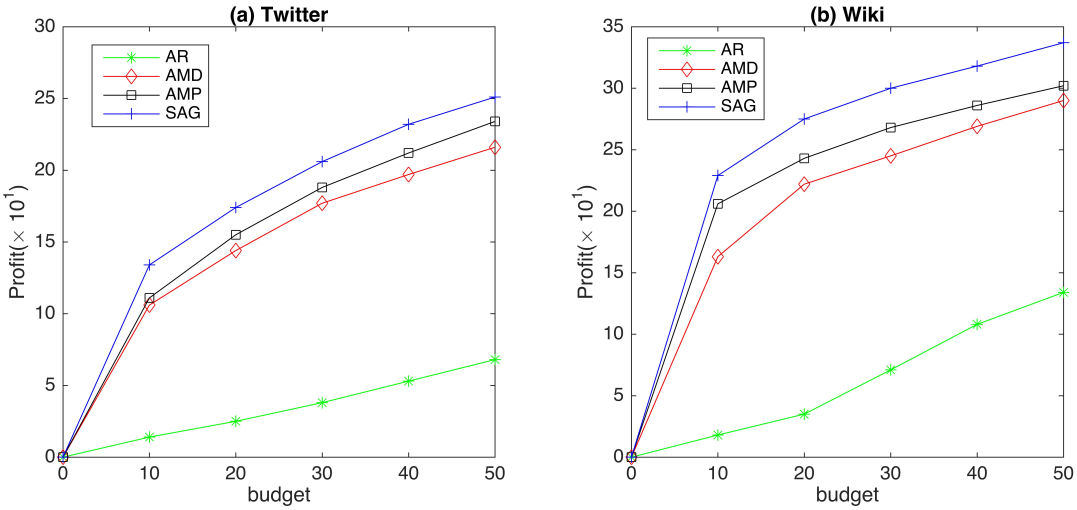


Figure 5.4. Profit VS budget on Twitter and Wiki.

Fig. 5.4 to Fig. 5.6 present the performance of our SAG policy and other three heuristic adaptive policies on all of the six listed datasets. In this group of experiments, the value of q is set to 3. We can see that the profits obtained by any policies increase with the value of the budget. And profits obtained by our SAG policy are always higher than those obtained

by other three heuristic adaptive policies no matter on which one of the six datasets. Among the three heuristic adaptive policies, adaptive max-profit (AMP) policy performs better than AR and AMD policies at most times. This is intuitive since AMP considers the profit not just the degree and a node with a large degree may not bring many profits. And our SAG policy usually can obtain about 10% higher profits than AMP policy, which also indicates the effectiveness of our SAG policy. But the results are not so stable and this may be due to the different graph structures and other features of different datasets.

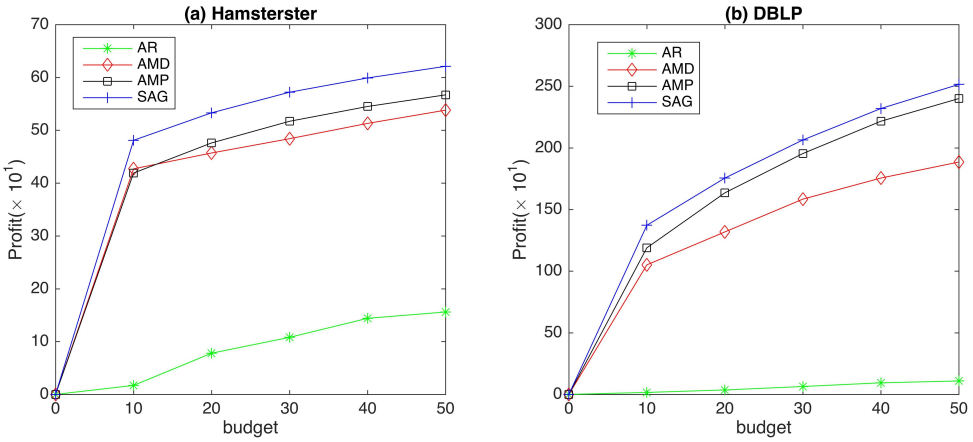


Figure 5.5. Profit VS budget on Hamsterster and DBLP.

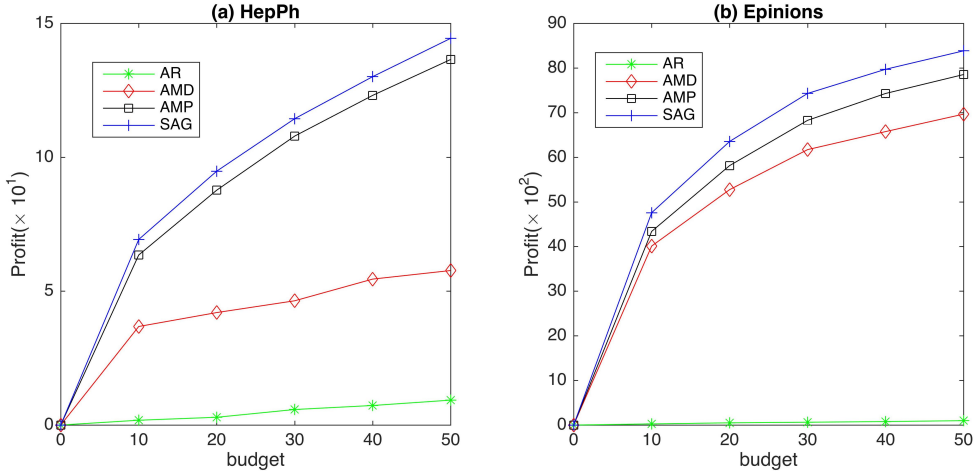


Figure 5.6. Profit VS budget on HepPh and Epinions.

CHAPTER 6

LEARNING-BASED INFLUENCE MAXIMIZATION

Authors – Tiantian Chen, Jianxiong Guo, Siwen Yan, and Weili Wu

The Computer Science Department, EC 31

The University of Texas at Dallas

800 West Campbell Road

Richardson, Texas 75080-3021

6.1 Introduction

Online social platforms, such as Twitter, LinkedIn and WeChat, have shown to be one of the most effective ways for people to communicate and share information with each other. Many companies have turned to the social network as a primary way of promoting products, and use "word of mouth" effects to maximize the product influence. To maximize earned profits, companies may apply a variety of methods, such as distributing free samples or coupons. Many works have focused on the diffusion phenomenon on social networks. Kempe *et al.* (Kempe et al., 2003) first formally defined Influence Maximization (IM) problem as a combinatorial optimization (CO) problem, and presented Independent Cascade (IC) model and Linear Threshold (LT) model to depict the information diffusion process.

It has been proved IM problem is NP-hard, and the objective (influence spread) is monotone and submodular under IC and LT models (Kempe et al., 2003). Kempe *et al.* (Kempe et al., 2003) used Greedy algorithm, which selects the node with maximum marginal gain of influence spread, to solve IM. Greedy scheme can achieve $(1 - 1/e)$-approximation ratio for the maximization of monotone submodular function under budget constraint. However, it is #P-hard to compute the influence spread of a seed set under both IC (Chen et al., 2010a) and LT models (Chen et al., 2010b). The hardness of estimating the influence spread lies in the randomness of the probabilistic diffusion models, i.e., random choices and diffusion paths. The key to approximate the influence spread is to effectively and efficiently sample diffusion paths. Kempe *et al.* (Kempe et al., 2003) used Monte Carlo method to simulate diffusion paths, which can obtain good estimations when simulation times are large enough. But it is too time-consuming. Borgs *et al.* (Borgs et al., 2014) first proposed a novel Reverse Influence Sampling (RIS) technique to reduce the running time. However, RIS still incurs significant computational overheads in practice in order to obtain a good solution. Subsequently, a series of algorithms based on RIS were proposed, such as TIM/TIM+ (Tang et al., 2014), IMM (Tang et al., 2015), SSA/D-SSA (Nguyen et al., 2016b) and OPIM-C

(Tang et al., 2018), which can achieve $(1 - 1/e - \epsilon)$-approximation solution with high probability when the number of generated random reachable reverse (RR) sets are large enough, and were recognized as the state-of-the-art methods to solve IM. However, these algorithms, such as IMM, still have scalability issues in large influence networks.

On the other hand, the development of deep learning and reinforcement learning (RL) has blossomed in the last few years, resulting in an increasing number of works addressing CO problem by learning-based methods. A natural question is: can we estimate the influence spread by learnable parametric function and avoid costly sampling random RR sets? The answer is Yes. Khalil *et al.* (Khalil et al., 2017) first designed an end-to-end deep reinforcement learning (DRL) framework, S2V-DQN, to solve the common CO problem. Then, Manchanda *et al.* (Manchanda et al., 2020) proposed a supervised deep learning based model for the CO problem, called GCOMB, where IM was used as an example to test the performance. However, the exact value of influence spread is not available, and therefore, no accurate target value can be used for supervised learning. On the contrary, Li *et al.* (Li et al., 2022) presented an end-to-end DRL model, called PIANO, which is revised from S2V-DQN (Khalil et al., 2017). PIANO is trained on subgraphs of the entire network and then tested on the entire network, which makes it not able to generalize on non-homogeneous networks with different topological characteristics. To address the above drawbacks, we integrate the latest strategies and design a new solution framework for the IM problem.

In this paper, we model the IM problem as a RL problem, which aims to find the optimal policy of selecting b seeds (b action sequences) to maximize the influence spread (cumulative rewards) of these b seeds. However, the exact Q-value in this RL is not available, and therefore deep Q-network (Mnih et al., 2015) (DQN) is a natural solution to solve this issue. Instead of using DQN, we use its improvement double DQN (DDQN) (Van Hasselt et al., 2016), which can avoid the over-optimistic issue of a simple DQN and achieve better performance. On the other hand, except for the network topology structure, the function

approximator in DDQN also needs to well capture the crucial influence cascading effects in IM, which makes it more challenging. The cascading effect represents that the activation of a node will trigger its neighbors in a successive manner, forming a diffusion cascade on social networks. This is consistent with the message passing effect in graph neural networks (GNNs) (Zhou et al., 2020). Therefore, based on these two techniques, in this paper, we propose a novel end-to-end DRL framework, called ToupleGDD (**T**hree C**ouple**d **G**raph Neural Networks with **D**ouble **D**eep Q-networks), to solve the IM problem, which incorporates three coupled GNNs for network embedding and DDQN technique for parameter learning. The main contributions can be summarized as follows: (1) To the best of our knowledge, we are the first to present such an end-to-end framework, ToupleGDD, which combines coupled GNNs and DRL method to effectively solve the IM problem; (2) We propose a personalized DeepWalk method to learn initial node embedding as input features for the following customized GNN layer, which considers both local and global influence contexts of nodes; (3) To capture the crucial cascading effects of information diffusion and network topology, we design three coupled GNNs to learn node embeddings; (4) We show that ToupleGDD can be applied on large-scale networks without compromising on solution quality. Extensive experiments are conducted on synthetic graphs and real-world datasets. Empirical results show that our model can achieve performance very close to IMM, and even outperform OPIM-C on several datasets, which demonstrate the superiority and effectiveness of our proposed model.

6.2 Related Works

IM. Kempe *et al.* (Kempe et al., 2003) first formulated IM as a CO problem, and proposed two classical diffusion models: IC and LT model. They proved that IM problem is NP-hard, and presented a $(1 - 1/e - \epsilon)$-approximation algorithm, Greedy, by applying Monte Carlo method to estimate expected spread of a seed set. But it is too time-consuming. Borgs *et*

al. (Borgs et al., 2014) made a breakthrough for this issue with the RIS technique, which guaranteed $(1 - 1/e - \epsilon)$-approximation solutions and significantly reduced the expected running time. Subsequently, a series of more efficient randomized approximation algorithms were proposed, such as TIM/TIM+ (Tang et al., 2014), IMM (Tang et al., 2015), SSA/D-SSA (Nguyen et al., 2016b), OPIM-C (Tang et al., 2018), and HIST (Guo et al., 2020d). They can not only provide $(1 - 1/e - \epsilon)$-approximation solution but also very efficient even on networks with millions of nodes, which were state-of-the-art approximation algorithms for IM.

ML/RL for CO. Recent advancements of deep learning and RL has resulted in an increasing number of works addressing IM by learning-based methods. Since IM can be formulated as a CO problem, many works aiming for CO problems have used IM as an example to test the performance of their models. Khalil *et al.* (Khalil et al., 2017) first proposed a DRL model for CO problems, called S2V-DQN, which utilized the graph embedding method, structure2vec (Dai et al., 2016), to encode nodes states to formulate the value approximator, and the fitted Q-learning to select the node to add to the current seed set. Li *et al.* (Li et al., 2018) approximated the solution quality by graph convolutional networks, and applied a learning framework based on guided tree search. Manchanda *et al.* (Manchanda et al., 2020) proposed a supervised deep learning based model, GCOMB, for CO problems over large graphs. The key contribution of GCOMB is its hybrid learning model, i.e., combining supervised learning and RL. By introducing a supervised learning step into Q-learning framework, GCOMB can predict the quality of nodes and filter out "bad nodes" at an early step. Instead of solving CO problems on the entire graph, (Kamarthi et al., 2020) and (Ireland and Montana, 2022) are focused on how to prune graph and discover a subgraph which can act as a surrogate to the entire graph. Ireland *et al.* (Ireland and Montana, 2022) introduced a novel graph pruning algorithm, LeNSE, based on supervised learning and RL. LeNSE learns how to identify a subgraph by removing vertices and edges

to significantly reduce the size of the problem, so that heuristics can find a nearly optimal solution of a CO problem with a high likelihood. The first phase of GCOMB can be viewed as the graph pruning, which filters out the "bad nodes" and only considers the "good" nodes as the candidate. For readers interested in more works of CO, please refer to (Bengio et al., 2021) (Mazyavkina et al., 2021) (Yang and Whinston, 2020) for detailed reviews.

ML/RL for IM. Fan *et al.* (Fan et al., 2020) proposed the DRL model for network dismantling problem, FINDER, which aimed to find key players in complex networks, and applied GraphSAGE as the function approximator for DQN. Kamarthi *et al.* (Kamarthi et al., 2020) utilized deep Q-learning for discovering subgraph and solved the IM problem on the subgrah and utilized the selected influential node set as the seeds on the complete graph. There were some researches (Lin et al., 2015) (Ali et al., 2018) (Ali et al., 2020) focusing on using DRL to solve the competitive IM problem, which aims to find an optimal strategy against competitor to maximize the commutative reward under the competition against other agents. Besides, (Yadav et al., 2018) (Chen et al., 2021) considered the contingency-aware IM problem, where there is a probability of a node willing to be seed when selected as seed node. Tian *et al.* (Tian et al., 2019) proposed DIEM model for the topic-aware IM problem, which aims to maximize the activated number of nodes under the specific query topics. DIEM modified the structure2vec method (Dai et al., 2016) for network embeddings, and utilized DDQN with prioritized experience replay to learn parameters. The work most related to ours is (Li et al., 2022), which proposed a DRL model, called PIANO, for the IM problem, and presented with small modification from S2V-DQN (Khalil et al., 2017).

Comparisons of related models to our model. FINDER model (Fan et al., 2020) was proposed for network dismantling problem, and cannot work on directed graphs and weighted graphs. However, our model can work on undirected graphs and different edge weight settings. GCOMB framework (Manchanda et al., 2020) was based on supervised learning which introduced large extra computational overhead and efforts of hand-crafting

the learning pipeline, while our model can learn parameters end-to-end. PIANO method (Li et al., 2022) applied structure2vec to learn node embeddings, while we designed three coupled GNNs to learn the network representation. Additionally, both GCOMB and PIANO are trained on subgraphs of the entire graph, and tested on the rest or the entire network, which makes them graph-specific. However, our ToupleGDD model does not have this limitation and performs well on different training and testing datasets, which shows more generalization ability.

6.3 Preliminaries and Framework

6.3.1 Background

Social network is usually represented by a directed graph $G = (V, E)$, where V denotes the node (user) set and E is a set of relationships between nodes. For an edge $(u, v) \in E$, u is called the in-neighbor of v, and v is called the out-neighbor of u. For a node v, denote by $N_{in}(v)$ and $N_{out}(v)$ the in-neighbor set and out-neighbor set of v, respectively.

Denote by $\sigma(v; S) = \sigma(S \cup \{v\}) - \sigma(S)$ the marginal gain obtained by adding v into a seed set S. Let S_t be the currently selected seed set. The greedy algorithm will select the node which can achieve the maximum of $\sigma(v; S_t)$ as the next seed. However, computing the influence spread of a seed set is #P-hard under the IC (Chen et al., 2010a) and LT model (Chen et al., 2010b), which results in the difficulty of calculating the marginal gain. Instead of generating a large number of RR sets like in the state-of-art approximation algorithms, in this paper, we regard IM as an RL problem, which aims to find an optimal policy to select k nodes or k action sequence with the maximum influence spread. In this case, the marginal gain can be considered as the value function in RL, whose value is difficult to be obtained in our problem. To address this issue, we approximate the value function (marginal gain) by a parameterized function through DRL method.

Definition 6.3.1 (Learning-based IM Problem)**.** *It can be divided into two phases:* ***(1) Learning Phase:*** *Given a set of training graphs $\mathcal{G} = \{G_1, G_2, \cdots, G_c\}$, diffusion model ψ and influence spread function $\sigma : S \to \mathbb{R}^+$, train a group of parameters Θ such that $\hat{\sigma}(v, S; \Theta)$ could approximate $\sigma(v; S)$ as accurately as possible.* ***(2) Testing Phase:*** *Given a target social network G, the learned parameters Θ and an integer b, solve the IM problem with respect to budget b under some diffusion model ψ.*

As a special type of RL, DRL applies deep neural networks for state representation and function approximation for value function, policy, transition model, or reward function. In this paper, we use GNNs to obtain node embeddings and formulate the parameterized function using node embeddings, where all parameters are learned by DDQN.

6.3.2 General Framework of GNN

As an effective framework of nodes embedding learning, GNN usually follows a neighbor-aggregation strategy, where the embedding of a node is updated by recursively aggregating embedding from its neighborhood. Formally, u's embedding at $k+1$-th layer $F_u^{(k+1)}$ is updated by:

$$m_{\mathcal{N}(u)}^{(k)} = \text{AGGREGATE}^{(k)}(F_v^{(k)} : v \in \mathcal{N}(u)), \tag{6.1}$$

$$F_u^{(k+1)} = \text{UPDATE}(F_u^{(k)}, m_{\mathcal{N}(u)}^{(k)}), \tag{6.2}$$

where AGGREGATE and UPDATE are neural networks and $\mathcal{N}(u)$ is u's neighborhood.

6.3.3 Framework of ToupleGDD

In this subsection, we present the proposed framework ToupleGDD, which solves the IM problem by incorporating three coupled GNNs and DDQN. The framework of ToupleGDD is illustrated in Fig. 6.1. Given a set of training graphs $\mathcal{G} = \{G_1, G_2, \cdots, G_c\}$, we first apply the personalized DeepWalk (PDW) method to get the initial node embedding, since

it has been found that DeepWalk embedding rather than randomly initialized embedding is vital for stable training of Geometric-DQN, which also works well in our model and will be shown in experiments. Then GNN and attention mechanism are combined to learn node embeddings. Specifically, three coupled GNN (ToupleGNN) are designed to capture the cascading effect of information diffusion. After K iterations of ToupleGNN, we use the obtained node embedding to construct the parameterized function $\hat{Q}(v, S; \Theta)$ and use RL technique to learn the parameters. Instead of using DQN, we apply the DDQN to learn the parameters Θ for $\hat{Q}(v, S; \Theta)$ to approximate the marginal gain $\sigma(v; S)$, and adopt ε-greedy policy to select the next seed. The reason why we use DDQN will also be explained through experiments.

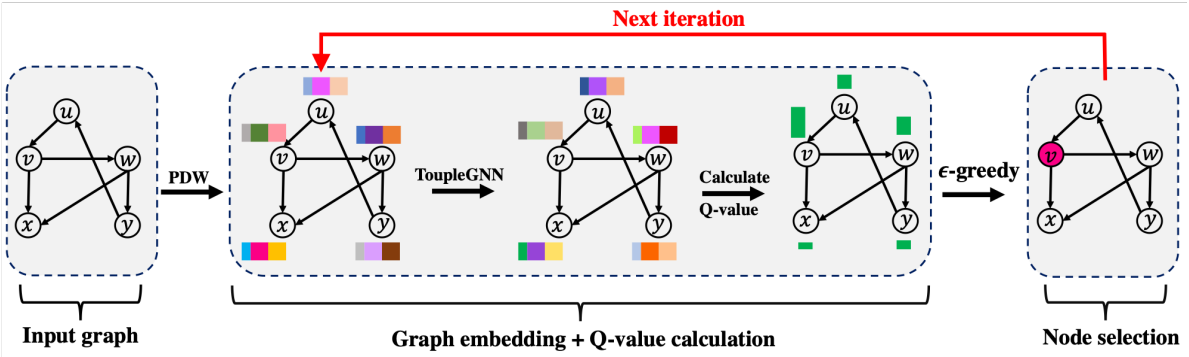


Figure 6.1. The framework of ToupleGDD: (a) Apply PDW to obtain initial embedding; (b) Utilize ToupleGNN to capture network topology structures and influence cascading effects to get node embedding; (c) Construct the parameterized function $\hat{Q}(v, S; \Theta)$ based on node embedding input from ToupleGNN; (d) Use ε-greedy to select the next seed and DDQN to learn the parameters.

6.4 Representation: Node Embedding

As a way of representing the node as a vector, node embeddings can capture the network topology. For our IM problem, more importantly, node embeddings need to capture the influence cascading effects, which represents that the activation of a node will trigger its out-neighbors in a successive manner, forming a diffusion cascade on networks. For a target

node, whether it will be activated is intrinsically governed by three components: the states of in-neighbors, the influence capacity of in-neighbors and its tendency to be influenced by in-neighbors. In this sense, the cascading effect is intrinsically the iterative interplay between node states, nodes' influence capacity and nodes' tendency to be influenced by others. Therefore, for each node u, we include three parts in u's embedding: X_u, S_u and T_u, where $X_u \in \mathbb{R}$ indicates the activation state of node u, $S_u \in \mathbb{R}^l$ is the capacity of u to influence other users and $T_u \in \mathbb{R}^l$ is the tendency of being activated by other users.

6.4.1 Initial Embedding Learning

Instead of randomly generating initial embeddings, we proposed the personalized DeepWalk (PDW) method to learn embeddings as input features for the following GNN layer. The main part of PDW is to generate node contexts, and then utilize skip-gram technique to predict contexts for a given node. Inspired by Inf2vec model (Feng et al., 2018), for node $u \in V$, our method includes two parts as u's influence context C_u: local and global influence context, where local context is a sampled set of nodes that can be activated by u and global contexts are sampled from the r-hop out-neighbors of u. To limit the size of node contexts, assume length threshold of the node context is L and $\alpha \in [0, 1]$. For a node u, we use random walk with restart (RWR) strategy (restart probability is set as 0.15 in this paper) to obtain the local influence context L_u of node u, and the walk will stop when threshold $\alpha \cdot L$ is reached. After generating local contexts, we randomly sample $(1 - \alpha) \cdot L$ nodes from the r-hop out-neighbor set N^r_{out} of u as global influence context G_u.

Given a user u, the probability of user v being influenced by user u, is formulated as a softmax function by their node embeddings: $\Pr(v|u) = e^{X_u \cdot S_u \cdot T_v + X_v}/Z(u)$, where $Z(u) = \sum_{w \in V} e^{X_u \cdot S_u \cdot T_w + X_w}$ is the normalization term. Assume users in C_u are independent with each other, then the probability of observing context C_u conditioned on u's embedding is $\Pr(C_u|u) = \Pi_{v \in C_u} \Pr(v|u)$. We will sample a set of influence contexts, $\mathcal{D} =$

$\{(u_1, C_{u_1}), \ldots, (u_q, C_{u_q})\}$ from social network G. We consider all the observed influence contexts, and attempt to maximize the log probability of them:

$$\max \sum\nolimits_{(u,C_u)\in\mathcal{D}} \sum\nolimits_{v\in C_u} \log \Pr(v|u). \tag{6.3}$$

However, it is time-consuming to compute $Z(u)$ directly since we need to enumerate each $w \in V$. In this paper, we utilize the negative sampling technique, which is popularly used to compute softmax functions. Instead of enumerating all nodes, negative sampling method only considers a small set of sampled nodes. For each node $u \in V$, we randomly generate a small set of nodes N as negative instances to approximate the softmax function:

$$\log \Pr(v|u) \approx \log \sigma(z_v) + \sum\nolimits_{w\in N} \log \sigma(-z_w), \tag{6.4}$$

where $z_v = X_u \cdot S_u \cdot T_v + X_v, z_w = X_u \cdot S_u \cdot T_w + X_w$ and $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid function.

Stochastic Gradient Descent (SGD) method is applied to learn all the parameters. In each step, we update the parameters Φ by calculating the gradient:

$$\Phi \leftarrow \Phi + \eta \frac{\partial}{\partial \Phi} (\log \Pr(v|u)), \tag{6.5}$$

where η is the learning rate and $\frac{\partial}{\partial \Phi}$ represents the gradient of parameters Φ. Based on (6.4), the gradient for corresponding parameters can be computed as follows:

$$\begin{aligned}
&\frac{\partial}{\partial S_u} = (1 - \sigma(z_v)) \cdot X_u \cdot T_v + \sum\nolimits_{w\in N} (-\sigma(z_w)) \cdot X_u \cdot T_w \\
&\frac{\partial}{\partial T_v} = (1 - \sigma(z_v)) \cdot X_u \cdot S_u, \ \frac{\partial}{\partial T_w} = (-\sigma(z_w)) \cdot X_u \cdot S_u \\
&\frac{\partial}{\partial X_u} = (1 - \sigma(z_v)) \cdot S_u \cdot T_v + \sum\nolimits_{w\in N} (-\sigma(z_w)) \cdot S_u \cdot T_w \\
&\frac{\partial}{\partial X_v} = 1 - \sigma(z_v), \ \frac{\partial}{\partial X_w} = -\sigma(z_w)
\end{aligned} \tag{6.6}$$

The proposed PDW method is summarized in Algorithm 17. It contains two parts: influence context generation (lines 3-8) and parameters learning (lines 9-14), which have

Algorithm 17: PDW

Initialize X_u, S_u, T_u by Gaussian distribution $\mathcal{N}(0, 0.01)$;
Initialize $W \leftarrow \emptyset$;
foreach $u \in V$ **do**
    $L_u \leftarrow \emptyset, G_u \leftarrow \emptyset, C_u \leftarrow \emptyset$;
    $L_u \leftarrow$ Sample αL nodes by RWR starting from u;
    $G_u \leftarrow$ Uniformly sample $(1-\alpha)L$ nodes from $N^r_{out}(u)$;
    $C_u \leftarrow L_u \cup G_u$;
    Insert (u, C_u) into W;
foreach $(u, C_u) \in W$ **do**
    foreach $v \in C_u$ **do**
        Update X_u, S_u, X_v, T_v;
        Sample a set of negative samples N;
        foreach $w \in N$ **do**
            Update X_u, S_u, X_w, T_w;
return X_u, S_u, T_u for each node u;

been illustrated in the above. In the influence context generation part, for each node u, local influence context is sampled by RWR strategy, and we use breath first search method to obtain u's r-hop out-neighbor set $N^r_{out}(u)$ for generating global influence context (upper bounder by $|E|$). Therefore, the time complexity of influence context generation part is $O(|V|(\alpha \cdot L + |E|)) = O(|V||E|)$. For the parameters learning part, for each tuple $(u, C_u) \in W$ (where $|W| = |V|$), L iterations are performed for nodes in C_u. At each iteration, we first update node embeddings of u and v, and then update node embeddings for each node in the negative samples set N. Therefore, the running time of the parameters learning part is $O(|V| \cdot L \cdot |N|) = O(|V|)$. Here we consider L and $|N|$ are fixed constants. Thus, the total time complexity of Algorithm 17 is $O(|V| + |V||E|) = O(|V||E|)$.

6.4.2 ToupleGNN

Inspired by (Cao et al., 2020), we design three coupled GNNs (ToupleGNN) to naturally capture the iterative interplay between node states, nodes' influence capacity and nodes'

tendency to be influenced by others. Taking initial node embeddings as input, ToupleGNN includes three coupled GNNs: (1) state GNN: model the activation states of nodes; (2) source GNN: model the influence capacity of nodes; (3) target GNN: model the tendency of nodes to be influenced by others. The framework of these three GNNs is illustrated in the middle part of Fig. 6.2, and we will introduce the detailed structures of them in the following part. Given the currently selected seed set S_t, we need update node representations accordingly by ToupleGNN.

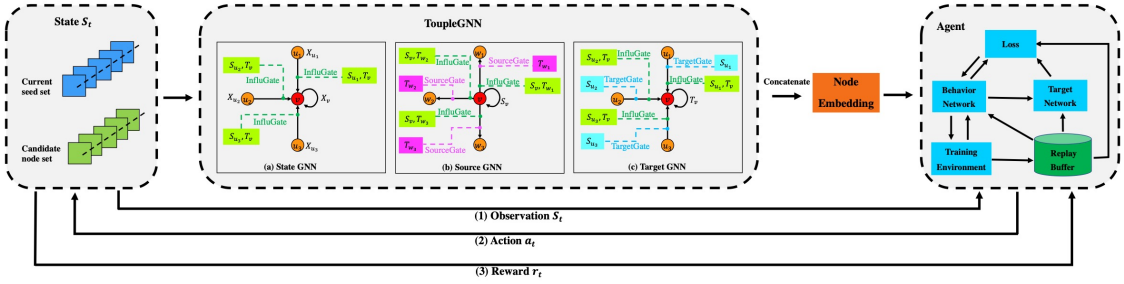


Figure 6.2. Mechanism of DDQN incorporated ToupleGNN as function approximator.

6.4.2.1 State GNN

The state GNN is used to model the activation state of each node during the cascading effect. For a target user v, it will be activated by its active in-neighbors. Therefore, its activation state X_v is determined by the activation states of its in-neighbors and the influence weight/probability of these in-neighbors to it. Since the interaction strength between users will change with nodes' states, only using the given static edge weight is not enough to capture the importance and influence weight between users. Therefore, except for the given edge weights, we also consider applying v's in-neighbors' capacity embedding and v's tendency embedding by an influence attention mechanism to dynamically capture the diffusion weight between them. Specifically, define $e_{uv}^{(k)} = \eta^{(k)}[W^{(k)}S_u^{(k)}, W^{(k)}T_v^{(k)}]$ to measure the dynamic importance of node u to v, where $\eta^{(k)} \in \mathbb{R}^{2h^{(k+1)}}$ is a weight vector, $W^{(k)} \in \mathbb{R}^{h^{(k+1)} \times h^{(k)}}$

is a weight matrix to transform the source and target representation from dimension $h^{(k)}$ to $h^{(k+1)}$, and $[\cdot,\cdot]$ denotes the concatenation of vectors. To make coefficients comparable among nodes, a softmax function incorporated with the LeakyReLU (Maas et al., 2013) is adopted to normalize the attention coefficients:

$$\text{InfluGate}(S_u^{(k)}, T_v^{(k)}) = \frac{\exp(\text{LeakyReLU}(e_{uv}^{(k)}))}{\sum_{u \in N_{in}(v)} \exp(\text{LeakyReLU}(e_{uv}^{(k)}))}, \tag{6.7}$$

where LeakyReLU has negative slope 0.2.

The expected influence that node v aggregates from its in-neighbors is:

$$a_v^{(k)} = \sum_{u \in N_{in}(v)} (\delta_1^{(k)} p_{uv} + \delta_2^{(k)} \text{InfluGate}(S_u^{(k)}, T_v^{(k)})) \cdot X_u^{(k)}. \tag{6.8}$$

Since we expect that the activation state should indicate the possibility of a node being activated, the activation state of node v is set to 1 when it is selected into the current seed set S_t. Otherwise, v's activation state is updated by aggregating influence from its in-neighbors. That is, node v's activation state at $(k+1)$-th layer is updated by:

$$X_v^{(k+1)} = \begin{cases} 1, & \text{if } v \in S_t \\ \sigma(\xi_X^{(k)} X_v^{(k)} + \xi_a^{(k)} a_v^{(k)}), & \text{otherwise} \end{cases} \tag{6.9}$$

where $\xi_X^{(k)}, \xi_a^{(k)} \in \mathbb{R}$ are weight parameters and $\sigma(\cdot)$ is the sigmoid function.

6.4.2.2 Source GNN

The source GNN is used to model the capacity of nodes to influence others. Intuitively, the capacity of a node v to activate others can be measured by both its activation state and how much influence its out-neighbors can get when the information is spread from v to out-neighbors, which can be modeled by v's out-neighbors' tendency to be activated. Similar to the dynamic influence weight defined in state GNN, for edge $(v, w) \in E$, we also

define the dynamic attention weight $f_{vw}^{(k)} = \beta^{(k)}[W^{(k)}S_v^{(k)}, W^{(k)}T_w^{(k)}]$ and its corresponding normalization for the weighted aggregation:

$$\alpha_{vw}^{(k)} = \frac{\exp(\text{LeakyReLU}(f_{vw}^{(k)}))}{\sum_{w \in N_{out}(v)} \exp(\text{LeakyReLU}(f_{vw}^{(k)}))}, \tag{6.10}$$

where $\beta^{(k)} \in \mathbb{R}^{2h^{(k+1)}}$ is a weight vector. Then the neighborhood aggregation is defined as:

$$b_v^{(k)} = \sum_{w \in N_{out}(v)} (\lambda_1^{(k)} p_{vw} + \lambda_2^{(k)} \alpha_{vw}^{(k)}) \cdot \text{SourceGate}(T_w^{(k)}), \tag{6.11}$$

where SourceGate($\star$) is the source gating mechanism implemented by a 3-layer MLP in this paper to reflect the nonlinear effect of out-neighbors' target tendency.

The source representation of node v at $(k+1)$-th layer is updated by incorporating its k-th layer source representation, neighborhood aggregation and its activation state:

$$S_v^{(k+1)} = \sigma(\gamma_S^{(k)} S_v^{(k)} + \gamma_b^{(k)} b_v^{(k)} + \gamma_X^{(k)} X_v^{(k)}), \tag{6.12}$$

where $\gamma_S^{(k)}, \gamma_b^{(k)}, \gamma_X^{(k)} \in \mathbb{R}$ are weight parameters.

6.4.2.3 Target GNN

The target GNN is used to model the nodes' tendency to be influenced by others. Generally, the tendency of a node to be activated is determined by its current activation state and the influence diffusion from its in-neighbors to it. Similarly, for edge $(u, v) \in E$, define $d_{uv}^{(k)} = \tau^{(k)}[W^{(k)}S_u^{(k)}, W^{(k)}T_v^{(k)}]$ and

$$\phi_{uv}^{(k)} = \frac{\exp(\text{LeakyReLU}(d_{uv}^{(k)}))}{\sum_{u \in N_{in}(v)} \exp(\text{LeakyReLU}(d_{uv}^{(k)}))}, \tag{6.13}$$

where $\tau^{(k)} \in \mathbb{R}^{2h^{(k+1)}}$ is a weight vector. Then the neighborhood aggregation is defined as:

$$c_v^{(k)} = \sum_{u \in N_{in}(v)} (\rho_1^{(k)} p_{uv} + \rho_2^{(k)} \phi_{uv}^{(k)}) \cdot \text{TargetGate}(S_u^{(k)}), \tag{6.14}$$

where TargetGate($\star$) is the target gating mechanism implemented by a 3-layer MLP in this paper to reflect the nonlinear effect of in-neighbors' source ability.

The target representation of node v at $(k+1)$-th layer is updated by incorporating its k-th layer target representation, neighborhood aggregation and its activation state:

$$T_v^{(k+1)} = \sigma(\mu_S^{(k)} T_v^{(k)} + \mu_c^{(k)} c_v^{(k)} + \mu_X^{(k)} X_v^{(k)}), \tag{6.15}$$

where $\mu_S^{(k)}, \mu_c^{(k)}, \mu_X^{(k)} \in \mathbb{R}$ are weight parameters.

6.4.3 Putting It Together

At each iteration of ToupleGNN, information diffusion and network structure features can be passed across nodes. After K iterations, nodes embedding can aggregate information from its K-hop neighbors. For node u, denote by $X_u^{(K)}, S_u^{(K)}, T_u^{(K)}$ the three components of u's node embedding after K iterations. Then u's node embedding can be obtained by concatenating these three parts: $[X_u^{(K)}, S_u^{(K)}, T_u^{(K)}]$. For the k-th layer of state GNN, the time complexity is $O(|V| + |E|)$, which is same for source GNN and target GNN. Therefore, the overall time complexity of ToupleGNN is $O(K(|V| + |E|))$.

Based on the obtained node embeddings, the score function to measure the marginal gain of a node $u \in \bar{S}_t = V \setminus S_t$ with respect to the current seed set S_t is defined as $\hat{Q}(u, S_t; \Theta) =$

$$\theta_1^\top \mathrm{ReLU}\Big(\Big[\theta_2 S_u^{(K)}, \theta_3 \sum_{v \in S_t} S_v^{(K)}, \theta_4 \sum_{w \in V \setminus (S_t \cup \{u\})} T_w^{(K)}\Big]\Big), \tag{6.16}$$

where $\theta_1 \in \mathbb{R}^{2l}, \theta_2, \theta_3, \theta_4 \in \mathbb{R}^{l \times l}$ are model parameters. Since the embeddings used to define $\hat{Q}(u, S_t; \Theta)$ are computed based on the parameters from ToupleGNN, $\hat{Q}(u, S_t; \Theta)$ will depend on $\{\theta_i\}_{i=1}^4$ and all parameters in ToupleGNN. We will train these parameters (denoted by Θ) end-to-end by RL.

6.5 Reinforcement Learning

6.5.1 RL Formulation

RL concerns about how the intelligent agent can take actions according to the current state when interacting with environment to maximize the total reward received. Why do we use RL model to learn the parameters in $\hat{Q}(u, S_t; \Theta)$? Actually, IM problem can be naturally formulated as a RL problem:

- Action: an action selects a node $u \in \bar{S}_t$ as the next seed, and we use u's node embedding to represent the action.

- State: a state $\mathcal{S}_t$ represents a sequence of actions of selecting nodes in the current seed set S_t. We use a $|V|$-dimensional vector to represent state $\mathcal{S}_t$, where the corresponding component of node u is 1 if $u \in S_t$, and 0 otherwise. For simplicity, we will use S_t instead of $\mathcal{S}_t$ to represent the state when there is no ambiguity. The terminal state S_b is the state after selecting b nodes.

- Transition: changing the activation state X_u from 0 to 1 when $u \in \bar{S}_t$ is selected as the seed.

- Reward: the reward $r(S_t, u)$ at state S_t is defined as the change of reward after selecting node u into the current seed set S_t and transition to a new state. That is, $r(S_t, u) = \sigma(S_t \cup \{u\}) - \sigma(S_t)$ and $r(\emptyset) = 0$. In this way, the cumulative reward R of a terminal state S_b coincides exactly with the influence spread of seed set S_b, i.e., $R = \sum_{t=0}^{b-1} r(S_t, u_t) = \sigma(S_b)$.

- Policy: policy maps a state to possibilities of selecting each possible action. That is, a policy tells the agent how to pick the next action.

If we denote by Q^* the optimal Q-function for this RL problem, then our embedding parameterized function $\hat{Q}(u, S_t; \Theta)$ will be a function approximator for it, which will be learned by DDQN.

6.5.2 Training via DDQN

We use DDQN (Van Hasselt et al., 2016) to perform end-to-end learning of parameters in $\hat{Q}(u_t, S_t; \Theta)$, which can avoid the over-optimistic issue of a simple DQN by adopting two networks: behavior network and target network, parameterized with Θ and Θ', respectively. The target network provides Q-values estimation of future states during training of the behavior network, and only updates parameters Θ' from the behavior network Θ every m episodes. The detailed training process is illustrated in Algorithm 18. We use the term *episode* to represent a complete sequence of node additions starting from an empty set until termination, and a single action (node addition) within an episode is referred to as a *step*. To collect a more accurate estimate of future rewards, n-step Q-learning (Sutton and Barto, 1998) is utilized to update the parameters, which is to wait n steps before updating parameters. Additionally, we apply the fitted Q-iteration (Riedmiller, 2005) with experience replay for faster learning convergence. Formally, the update is performed by minimizing the following square loss:

$$(y - \hat{Q}(u_t, S_t; \Theta))^2, \tag{6.17}$$

where $y = \sum_{i=0}^{n-1} \gamma^i r(S_{t+i}, u_{t+i}) + \gamma^n \max_v \hat{Q}(v, S_{t+n}; \Theta')$, and $\gamma \in [0, 1]$ is the discount rate, determining the importance of future rewards.

Specifically, we first apply the PDW method (Alg. 17) to obtain initial embeddings. Then for each episode (Lines 2-20), the seed set is initialized to empty set. For each step, ε-greedy policy is utilized to select a node, which selects a node randomly with probability ε and with $(1 - \varepsilon)$ probability selects the node with the maximum Q-value (Lines 5-14). If $t \geq n$, it will add the current sample $(S_{t-n}, u_{t-n}, \sum_{i=0}^{n-1} \gamma^i r(S_{t-n+i}, u_{t-n+i}), S_t)$ to the replay

Algorithm 18: Training of ToupleGDD

```
1  Obtain initial embedding for each u ∈ V by Alg. 17;
2  for episode e = 1 to D do
3      S_0 = ∅;
4      for t = 1 to b do
5          Uniformly sample a number c from [0, 1);
6          if c < ε then
7              Randomly select a node u_t ∈ V \ S_t;
8          else
9              for i = 1 to K do
10                 for u ∈ V do
11                     Update X_u^(i), S_u^(i), T_u^(i) by ToupleGNN;
12             for u ∈ V do
13                 Calculate Q̂(u, S_t; Θ) by (6.16);
14             Select u_t = arg max_{u∈S̄_t} Q̂(u, S_t; Θ);
15         S_t = S_{t−1} ∪ {u_t};
16         if t ≥ n then
17             (S_{t−n}, u_{t−n}, Σ_{i=0}^{n−1} γ^i r(S_{t−n+i}, u_{t−n+i}), S_t) to replay buffer M;
18     Sample random batch B ∼ M;
19     Update Θ by Adam optimizer over (6.17) with B;
20     Update Θ′ from Θ every m episodes;
21 return Θ;
```

buffer M. Instead of performing a gradient step with respect to the loss of the current example, the parameters are updated with a batch of random samples from the buffer (Lines 24-25). For each episode, we will perform b steps. At each step, node embeddings for each node will be updated for K times by ToupleGNN. At each layer of ToupleGNN, each node aggregates information from its in/out-neighborhood (overall $O(|E|)$). Therefore, the time complexity of each layer is $O(|V| + |E|)$. Putting it all together, the time complexity of Algorithm 18 is $O(|V||E| + DbK(|V| + |E|))$.

6.6 Experiments

In this section, we conduct several experiments on different datasets to validate the performance of our proposed ToupledGDD model. All experiments are conducted on a machine with Intel Xeon CPU (2.40 GHz, 28 cores), 512 GB of DDR4 RAM, Nvidia Tesla V100 with 16-GB HBM2 memory, running CentOS Linux 7. The source code is available at https://github.com/Dtrycode/ToupleGDD.

6.6.1 Experimental Setup

Datasets. To thoroughly evaluate the performance of the proposed model, both synthetic and real-world datasets are used for evaluation. We generate 20 random Erdős-Renyi (ER) graphs with node size varying from 15 to 50 for training and validation. Specifically, we first sample the number of nodes uniformly at random from 15 to 50, and then generate an ER graph with edge probability 0.15. Among those generated synthetic graphs, 15 graphs are used for training, and the others are used for validation with the soc-dolphins dataset (Rossi and Ahmed, 2015). The performance of the proposed model and baselines are tested on seven real-world datasets, whose detailed statistics are shown in Table 6.1. For the undirected graph, we replace each edge with two reversed directed edges. Among these datasets, Twitter, Wiki-1, caGr and Buzznet are from (Rossi and Ahmed, 2015), while Wiki-2, Epinions and Youtube are available on (Leskovec and Krevl, 2014).

Diffusion Models. Our model can be easily adapted to distinct diffusion models by revising the definition of reward function. In this paper, we report the results under the IC model here. Unless otherwise specified, the probability on edge (u, v) is set to $1/N_{in}(v)$ (in-degree setting), which is widely used in previous works about IM (Kempe et al., 2003) (Tang et al., 2014) (Tang et al., 2015) (Borgs et al., 2014) (Nguyen et al., 2016b). To fairly evaluate the performance of different methods, we first record the seed set obtained by different methods independently, and then perform 10,000 Monte Carlo simulations to estimate the expected

Table 6.1. Dataset characteristics

Dataset	n	m	**Type**	**Average degree**
soc-dolphins	62	159	directed	5
Twitter	0.8k	1k	directed	2
Wiki-1	0.9k	3k	directed	6
caGr	4.2k	13.4k	undirected	5
Wiki-2	7.1k	103.7k	directed	29
Epinions	76k	509k	directed	13
Buzznet	101k	3M	directed	55
Youtube	1.13M	3M	undirected	5

influence spread. All experiments are run 10 times and we report the average of the metric being measured.

Baselines. We compare the performance of ToupleGDD with the state-of-the-art approximation algorithm for IM problem, IMM (Tang et al., 2015) and OPIM-C (Tang et al., 2018), and the DRL methods S2V-DQN (Khalil et al., 2017) and GCOMB (Manchanda et al., 2020) for CO problem. Note that S2V-DQN is originally designed for CO problem, and we revised their code for maximum cut problem to solve the IM problem. Another baseline is PIANO (Li et al., 2022), which is modified from the S2V-DQN model for the IM problem. For all other baselines, we use the code shared by the authors. For IMM and OPIM-C, we set $\epsilon = 0.1$.

Training and testing details. For all training datasets, edge weights are set as in-degree setting. Edge weights on validation datasets and testing datasets have the same setting (so we will only specify the setting of testing datasets in the following), and may be set as one of the three settings: (1) in-degree setting; (2) set as 0.1 (0.1-setting); (3) set as 0.5 (0.5-setting). We set the budget b as 5 for all training datasets, while in validation setting 5 and 7 for ER graphs and soc-dolphins dataset, respectively. For each testing dataset, we vary budget b such that $b \in \{10, 20, 30, 40, 50\}$. For S2V-DQN and ToupleGDD, we use RIS method to estimate the influence spread for a given seed set in the training process. For

GCOMB, since their code is not able to deal with multiple training graphs, we follow the same instructions as in their paper and use the training graph shared by them by revising the edge weight to the in-degree setting.

6.6.2 Experimental Results

Ablation study. In the early version (called DISCO (Li et al., 2019)) of PIANO model, they have shown that the order of candidate nodes with respect to their Q values remains almost unchanged whenever we select a seed and recompute the network embeddings as well as the Q values. Therefore, instead of iteratively selecting and re-computing nodes embeddings and Q values according to each seed insertion (iterative operation), they simplified the process into only one iteration, by embedding only once and select the top-b nodes with the maximum Q (one-time operation). Inspired by this conclusion and operation, we compare the expected influence spread of seed sets obtained by our ToupleGDD model by these two operations. On the other hand, we also test the impact of the initial embedding to our model. Three groups of experiments are conducted: (1) both train and test with initial embedding (TIEI); (2) train with initial embedding but test without initial embedding (TIEN); (3) both train and test without initial embedding (TNEN). For all of these three types, the validation setting is same as the testing, and all experiments of this part use in-degree probability setting. Besides, for (1) and (2), they share the same training model, and validations are conducted independently for them. For each of the three groups, the iterative and one-time operations are performed at the same one experiment. That is, after computing the Q-values, we first output the top-b nodes with highest Q-values and then perform the iterative operation according to greedy strategy. Therefore, both of these two operations share the same initial embeddings if there is any.

The results are shown in Table 6.2. Note that TI and EI represent training and testing with initial embedding, respectively. Firstly, for the same dataset and seed selection operation (e.g., Twitter with iterative operation), comparing results of three groups, we see that

Table 6.2. Performance of ToupleGDD under different setting

Dataset	**Operation**	**TI**	**EI**	**Budget: 10** Influence spread (time:s)	**Budget: 20** Influence spread (time:s)	**Budget: 30** Influence spread (time:s)	**Budget: 40** Influence spread (time:s)	**Budget: 50** Influence spread (time:s)
Twitter	iterative	√	√	147.71 (9.17)	210.86 (16.41)	252.11 (23.42)	287.59 (29.84)	315.88 (38.33)
		√		146.93 (7.03)	210.66 (13.06)	252.12 (21.13)	287.95 (29.82)	315.99 (33.55)
				139.36 (5.69)	188.23 (13.80)	234.41 (19.56)	270.09 (27.22)	301.59 (33.03)
	one-time	√	√	146.87 (3.01)	210.86 (3.0)	252.11 (3.08)	288.08 (2.97)	316.60 (3.13)
		√		147.26 (0.67)	210.66 (0.58)	251.53 (0.64)	287.95 (0.62)	317.28 (0.63)
				139.36 (0.57)	188.29 (0.66)	232.85 (0.63)	270.11 (0.72)	300.34 (0.73)
caGr	iterative	√	√	213.13 (18.40)	368.74 (25.48)	489.15 (31.60)	602.95 (38.16)	696.95 (46.53)
		√		214.18 (6.09)	372.13 (12.24)	488.60 (18.15)	602.33 (27.53)	697.99 (37.62)
				210.56 (5.78)	368.28 (12.89)	489.33 (19.15)	605.18 (27.80)	699.82 (36.81)
	one-time	√	√	208.29 (12.09)	355.87 (11.99)	487.99 (11.65)	604.93 (11.77)	695.95 (11.77)
		√		210.10 (0.62)	372.79 (0.61)	488.27 (0.57)	608.25 (0.68)	704.79 (0.65)
				208.24 (0.57)	367.52 (0.63)	487.69 (0.63)	608.23 (0.66)	708.54 (0.67)
Wiki-2	iterative	√	√	290.48 (46.79)	423.96 (54.68)	521.79 (63.35)	601.39 (71.72)	669.43 (78.23)
		√		290.81 (7.39)	424.77 (15.79)	523.38 (22.82)	600.29 (31.01)	670.27 (40.04)
				285.10 (7.24)	422.81 (15.4)	510.53 (19.99)	585.87 (30.18)	647.11 (36.36)
	one-time	√	√	288.97 (39.94)	421.39 (40.46)	518.63 (39.05)	599.93 (39.57)	668.39 (39.16)
		√		290.22 (0.72)	424.56 (0.70)	516.09 (0.70)	599.04 (0.70)	666.61 (0.73)
				282.42 (0.67)	420.99 (0.68)	504.32 (0.66)	579.26 (0.72)	642.27 (0.69)

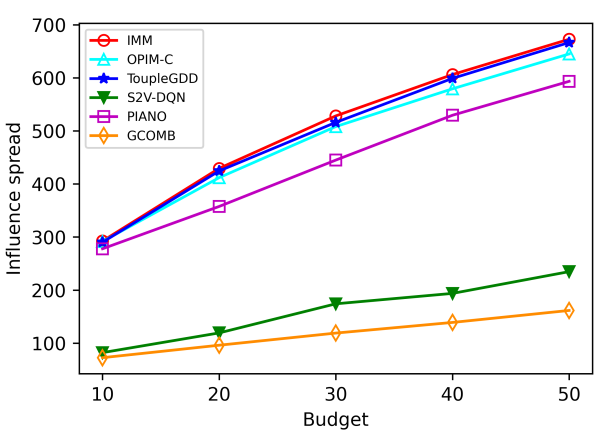
Table 6.2 continued

Dataset	**Operation**	**TI**	**EI**	**Budget: 10** Influence spread (time:s)	**Budget: 20** Influence spread (time:s)	**Budget: 30** Influence spread (time:s)	**Budget: 40** Influence spread (time:s)	**Budget: 50** Influence spread (time:s)
Epinions	iterative	√	√	6022.85 (2.77×10^3)	8303.34 (2.75×10^3)	9693.69 (2.81×10^3)	10866.88 (2.79×10^3)	11781.69 (2.8×10^3)
		√		6018.67 (13.4)	8295.06 (29.9)	9708.72 (44.96)	10853.63 (58.2)	11771.37 (75.61)
				6013.50 (13.51)	8300.42 (29.11)	9700.72 (42.07)	10840.46 (57.35)	11795.48 (70.3)
	one-time	√	√	6022.85 (2.76×10^3)	8300.36 (2.72×10^3)	9694.49 (2.76×10^3)	10832.49 (2.73×10^3)	11736.67 (2.73×10^3)
		√		6018.67 (1.36)	8315.56 (1.46)	9718.94 (1.42)	10829.7 (1.4)	11758.36 (1.46)
				6013.5 (1.38)	8310.07 (1.42)	9729.69 (1.42)	10864.27 (1.43)	11800.19 (1.34)

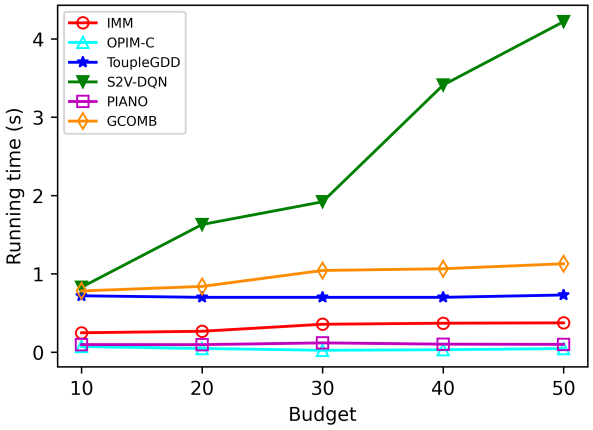
the expected influence spread of seed set obtained by TIEI and TIEN are very close. However, the results of TNEN have big differences from the other two under the same budget, and are not stable under different datasets, which indicates the necessity and importance of initial embedding in training. Secondly, the running time of TIEN and TNEN is less than that of TIEI for same dataset and seed selection operation, and this difference is significantly big for large datasets, like Epinions. This is because there is no initial embedding generation in TIEN and TNEN when testing, which can save much time especially for large datasets. Besides, the running time of iterative operation increases with the increase of budget due to more iterations and selections, and for one-time operation, there is no significant difference between different budgets. Thirdly, for same group of the experiment (e.g., Twitter under TIEI), comparing the expected spread obtained by iterative and one-time operation, we observe the difference between them is very small, but they actually don't share the exactly same seed set in most cases. However, we cannot figure out the reason causing this difference due to the machine accuracy configuration for very close values. Besides, one-time

operation can output the seed set faster than iterative selection, due to its less iteration and computation. From these results, it is convincing that we can use one-time operation for seed selection and TIEN setting to save time but without large decrease of influence spread. For ones who want to apply this algorithm in their problems, it is determined by the trade-off between accuracy and running time.

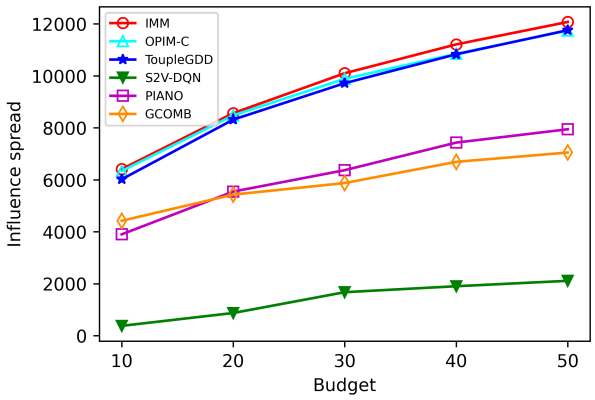
Influence spread. We test the performance of ToupleGDD and baselines on Wiki-1, Epinions, caGr, Buzznet and Youtube datasets with the in-degree probability setting. Fig. 6.3 and Fig. 6.4 draw the expected influence spread and running time produced by different models on these five datasets. Note that the results obtained by our model is from TIEN setting and one-time operation, which could not only provide close influence spread with corresponding iterative operation but also runs in less time. From the left column of Fig. 6.3 and Fig. 6.4, the expected influence spread increases with the increase of budget, which is consistent with the monotone increasing characteristic of influence spread under the IC model. Besides, the performance of ToupleGDD is very close to IMM and outperforms OPIM-C on Wiki-2, Buzznet and Youtube datasets, which proves the effectiveness of our model. Comparing the performance of all DRL-based models, ToupleGDD can outperform all other DRL-based models on all tested datasets, demonstrating the superiority of our model. And PIANO and S2V-DQN do not perform stably across different datasets, where S2V-DQN performs better than PIANO on undirected graph caGr, but worse than PIANO on other datasets. This may be because S2V-DQN is designed for undirected graph, and the original paper trained and tested the model on undirected graphs. Even though PIANO is revised from S2V-DQN, its performance is not close to S2V-DQN. This may be because code of PIANO is revised from the code for minimum vertex cover (MVC) problem in the shared S2V-DQN code, while our revised code is from maximum cut (MC) problem. Thus, we use different initial node features. The reason that we choose code of MC is that they have considered edge weight and edge features in MC but not in MVC. Additionally, PIANO and GCOMB have close performance on Epinions, caGr and Buzznet datasets.



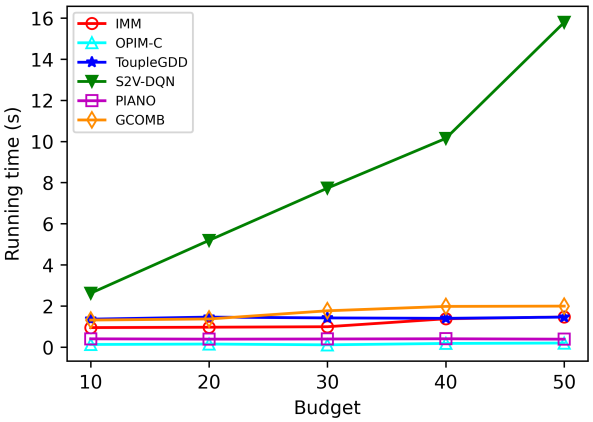
(a) Wiki-2, Performance



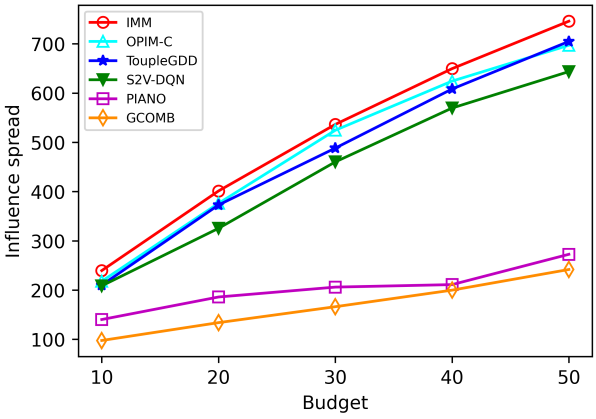
(b) Wiki-2, Running time



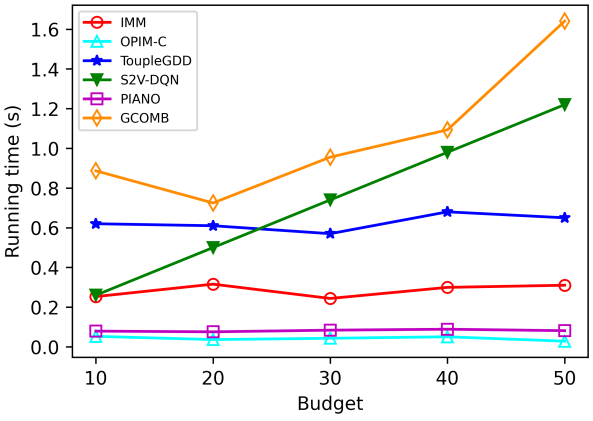
(c) Epinions, Performance



(d) Epinions, Running time

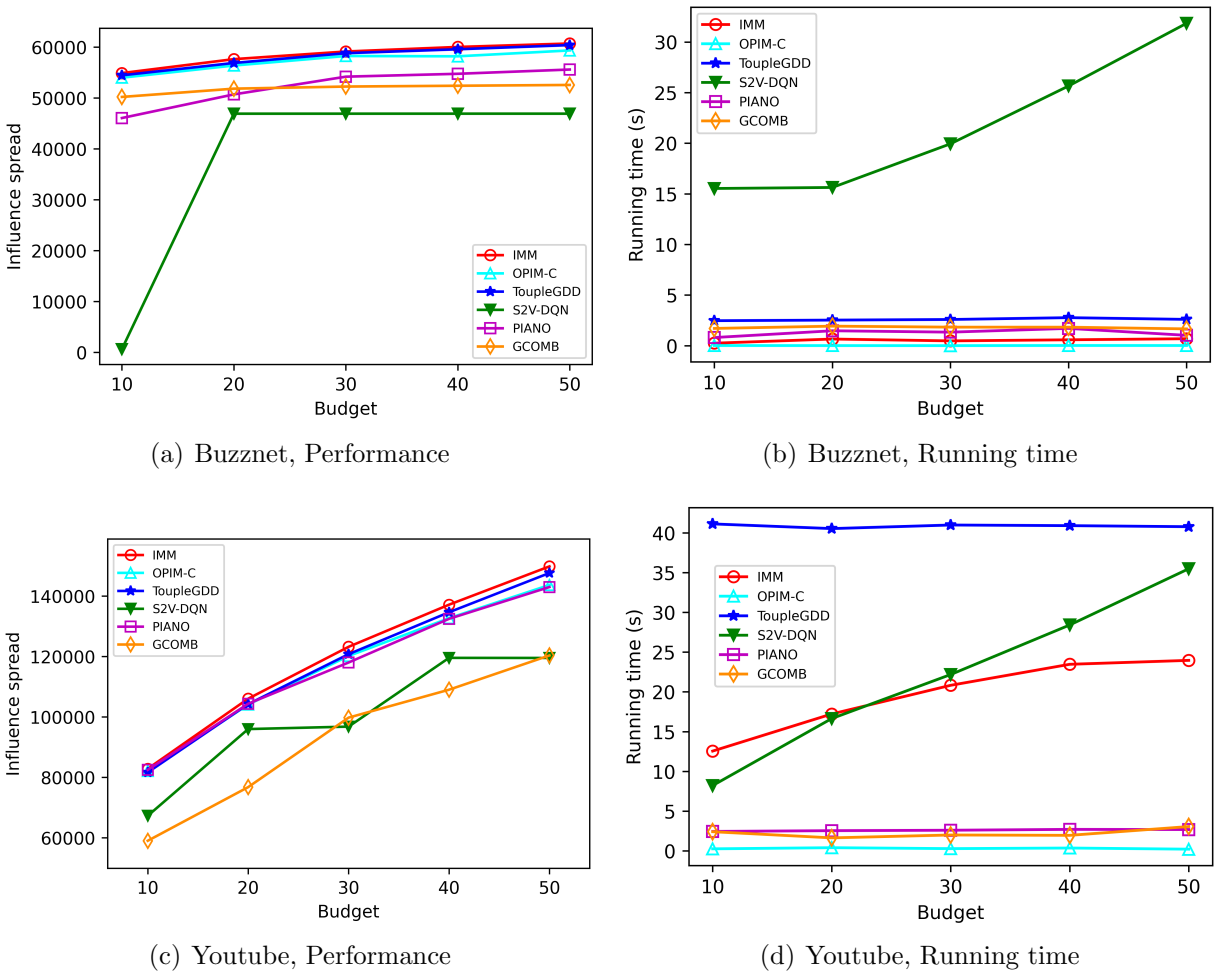


(e) caGr, Performance



(f) caGr, Running time

Figure 6.3. Performance and running time comparisons among different methods on Wiki-2, Epinions and caGr datasets.



(a) Buzznet, Performance

(b) Buzznet, Running time

(c) Youtube, Performance

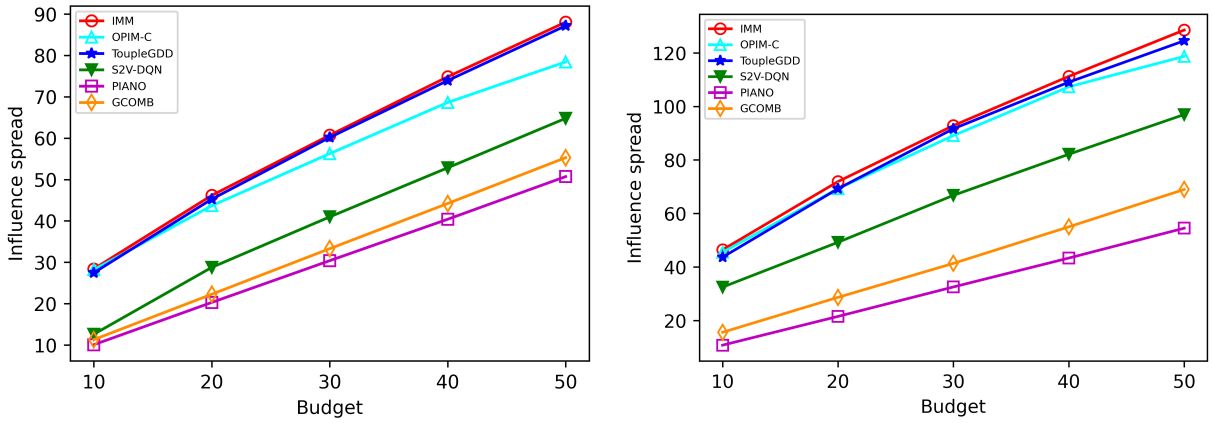
(d) Youtube, Running time

Figure 6.4. Performance and running time comparisons among different methods on Buzznet and Youtube datasets.

Running time. The right column of Fig. 6.3 and Fig. 6.4 draws the corresponding running time of different models to obtain the results in the left column. Note that we only record the time that the model needs to output the seed set for a budget not including the time to compute influence spread of the seed set. We observe that S2V-DQN needs more time to output the seed set than ToupleGDD on all datasets except Youtube. This may be because S2V-DQN use iterative not one-time manner to select seeds, which needs to update embedding and recompute Q-values for b times. Among all the tested methods, OPIM-C needs least time and ToupleGDD model runs a little slower. This may be because our

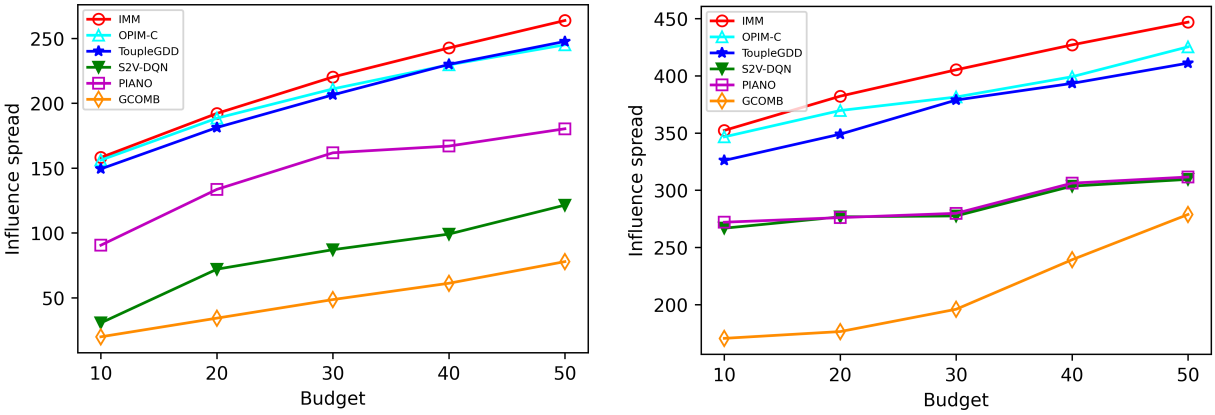
model has many parameters and need to compute the dynamic influence importance between nodes which is time-consuming. But our model's running time is acceptable since it is less than 3 seconds even for million-size dataset Buzznet. Note that this time difference also includes the effects of different implementation language, since ToupleGDD is implemented by Python, while IMM, OPIM-C and most part of PIANO are implemented by C++. We also observe that GCOMB runs slower than ToupleGDD and PIANO on Wiki-2, Epinions and caGr datasets. This may be because GCOMB is proposed for CO problem over very large networks and in their paper, they claimed GCOMB is hundreds of times faster than IMM on million-size datasets. However, from the results in Fig. 3, for small graphs Wiki-2, caGr and Epinions, the running time of GCOMB is longer than ToupleGDD and PIANO due to its extra computational overhead and efforts of hand-crafting the learning pipeline in the supervised learning part.

Generalization. To further validate ToupleGDD's generalization ability, we test the performance of the model trained under in-degree setting and tested on Twitter and Wiki-1 datasets with both 0.1-setting and 0.5-setting. Fig. 6.5 and Fig. 6.6 draw the results for 0.1-setting and 0.5-setting, respectively. From these results, the performance of ToupleGDD is almost equal to IMM and outperforms OPIM-C under 0.1-setting even though it is trained under in-degree setting. And ToupleGDD outperforms all other DRL-based models for both of the two edge probability settings on the two tested datasets. This demonstrates the robustness and generalization ability of the proposed ToupleGDD model. The performance of S2V-DQN, PIANO and GCOMB are not stable across different edge weight settings. S2V-DQN outperforms PIANO and GCOMB under 0.1-setting, while PIANO outperforms S2V-DQN and GCOMB under 0.5 setting. Furthermore, our model can obtain at least 33% gain of the expected influence spread than S2V-DQN under 0.1-setting, and at least 20% gain of the expected influence spread than PIANO under 0.5-setting.



(a) Twitter (b) Wiki-1

Figure 6.5. Performance comparisons among different methods under 0.1-setting.

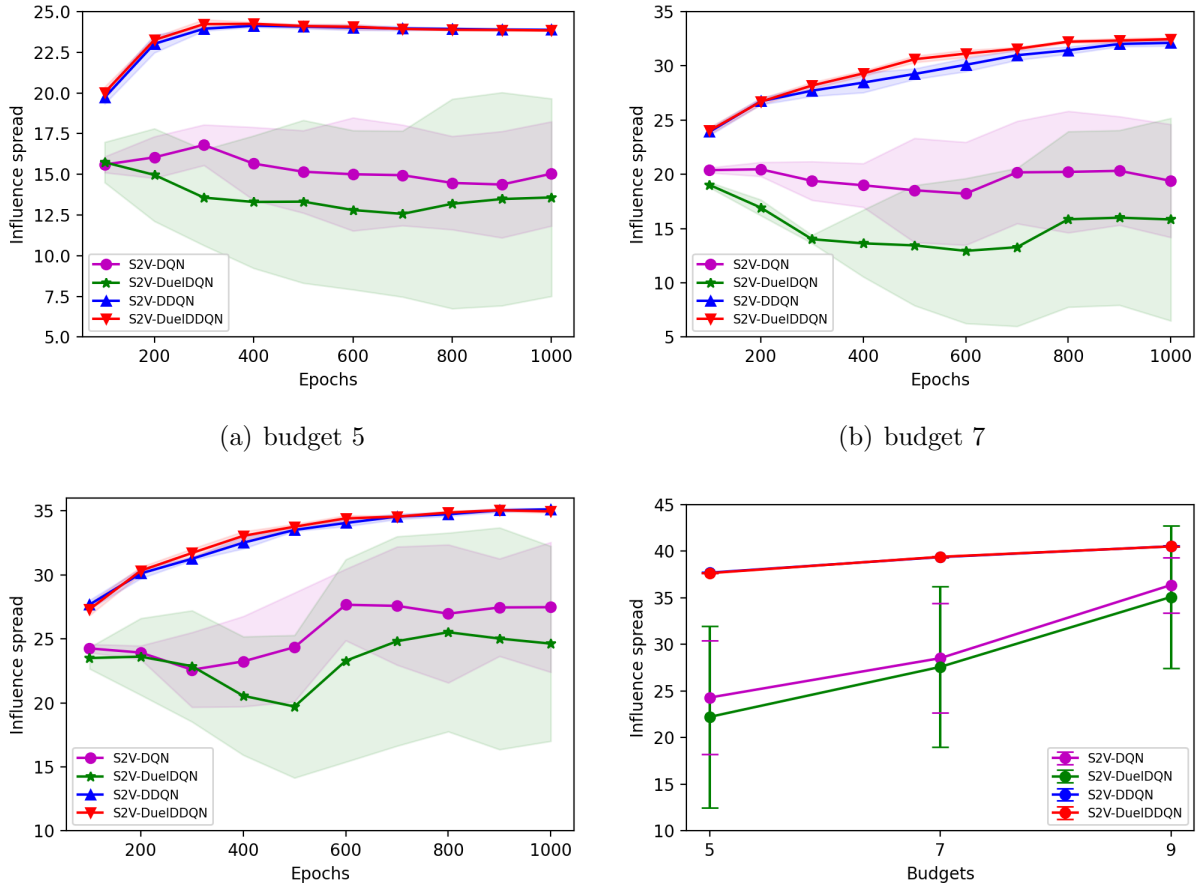


(a) Twitter (b) Wiki-1

Figure 6.6. Performance comparisons among different methods under 0.5-setting.

6.6.3 Intuition of Applying DDQN

Although DQN is an important milestone for deep learning, several limitations of this algorithm are now known. The improvement to DQN has blossomed in the last decades, such as Dueling DQN (DuelDQN), Double DQN (DDQN) and Duel Double DQN (DuelDDQN). How about the performance of S2V-DQN by replacing DQN with these improvements on IM? This is the main purpose of this group of experiments. We incorporate structure2vec method with these improved models, and compare their performance with S2V-DQN on IM.



(a) budget 5

(b) budget 7

(c) budget 9

(d) Testing result

Figure 6.7. Training and testing results for different models. (a) (b) and (c): learning curve with budget 5, 7 and 9, respectively; Solid line is average, and shadow is one standard deviation. (d) Testing result: dot is average, and bar shows one standard deviation.

We train the four models on soc-dolphins dataset with 0.5-setting for edge weight and the budget is taken from $\{5, 7, 9\}$. Here we keep the budget same for training and testing to avoid the effect of changing budget in the performance. For each budget, we train each framework 1000 epochs with exploration ratio ε starting from 1 and multiplied by a factor per epoch to balance exploration and exploitation. We run each framework 5 times to get the average and standard deviation.

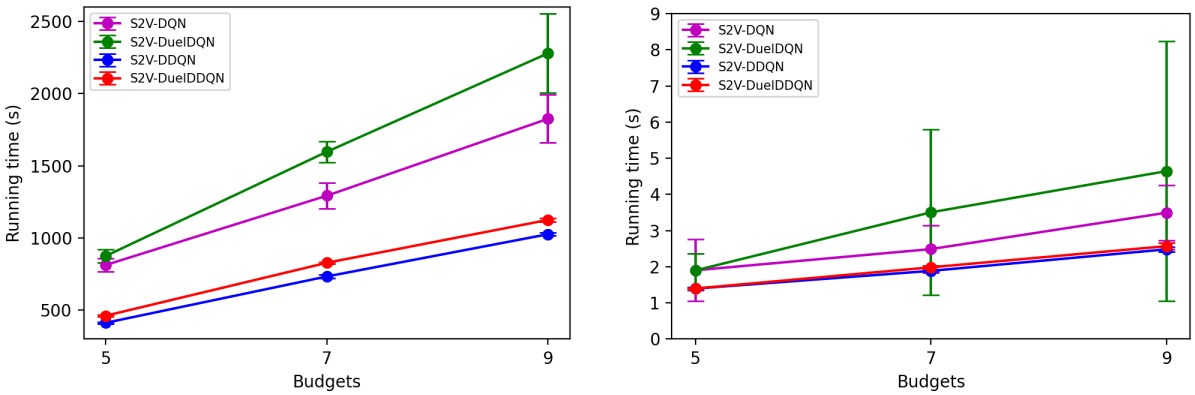
The learning curves of the four frameworks with budget 5, 7 and 9 are shown in Fig. 6.7 (a), (b) and (c), respectively. We expect S2V-DuelDQN to converge fast and S2V-DuelDDQN to perform the best. However, from the results, we observe that S2V-DuelDQN may not work and its advantage of fast convergence is not perceivable. The S2V-DDQN does perform well and DuelDDQN manages to make its influence score increase more in fewer epochs. The learning curves fluctuate more with the simple DQN and DuelDQN based models, while the DDQN based models maintain much more stable learning curves across multiple runs.

Table 6.3. p-value under different budgets (S2V is saved in methods' name for space)

Model	**DuelDQN**			**DDQN**			**DuelDDQN**		
Budget	5	7	9	5	7	9	5	7	9
DQN	0.3799	0.6611	0.4489	2.1029e-14	5.7903e-12	1.1655e-08	2.3740e-14	5.3526e-12	1.2368e-08
DuelDQN	-			4.4682e-10	2.0871e-08	0.0011	4.8612e-10	1.9727e-08	0.0011
DDQN	-			-			0.0082	0.0457	0.4111

We test the trained frameworks on a uniformly sampled graph with the same number of nodes and edges as in training dataset. For each model from training, we run 5 times on the testing graph to get its average performance and the seeds are selected with iterative operation. Fig. 6.7 (d) draws the expected spread of seed set obtained by different models. We observe that the DDQN based models still perform better than the simple DQN and DuelDQN model, and are much more stable. Furthermore, we use p-value to check significance of testing performance difference between frameworks as shown in Table 6.3. Generally, when p-value is less than 0.05, the performance difference of the two models is significant. The p-value results agree with our previous observation that DDQN based models perform significantly better than DQN and DuelDQN based models. Though the difference between S2V-DDQN and S2V-DuelDDQN is subtle in Fig. 6.7 (d), DuelDDQN does get significantly better performance when the budget is small.

Fig. 6.8 (a) and (b) draw the training and testing time averaged from 5 runs for each framework. The time usage is approximately proportional to the budget size. DDQN based



(a) Training time

(b) Testing time

Figure 6.8. Running time: dot is average, and bar shows one standard deviation.

models even maintain much lower time usage (both training and testing) compared to DQN based models, which demonstrates the efficiency of DDQN based models.

CHAPTER 7

CONCLUSION AND FUTURE WORK

This dissertation considers several variants of influence maximization problem in social networks, including continuous activity maximization problem, budget profit maximization with coupon advertisement problem, adaptive multi-Feature budgeted profit maximization problem, and learning-based influence maximization problem.

For the continuous activity maximization problem in Chapter 3 (see p. 9), we considered it as the maximization problem on lattice. We proved the hardness and gave a computing method for the objective function of the continuous activity maximization problem. This objective function is monotone but not DR-submodular and not DR-supermodular. We designed the unbiased sampling for it, its upper bound and lower bound. Adapted from IMM algorithm and sandwich approximation framework, a data-dependent approximation ratio can be obtained. The performance of the proposed algorithms is verified by experiments. The analysis of continuous activity maximization problem is applicable to others which is a branch of maximization problem on lattice.

For the budget profit maximization with coupon advertisement problem in Chapter 4 (see p. 40), we proved it can be classified as submodular maximization with cardinality constraint problem under the IC/LTmodel. By discretizing Continuous Double Greedy algorithm and combining with Random Greedy algorithm, an improved $h(G, k)$-approximation can be obtained. Then, in order to study its robustness, Robust-budget profit maximization with coupon advertisement problem is proposed, an $h(G, k) \cdot \alpha(\Theta)$- and $h(G, k) \cdot (1 - \epsilon)$-approximation can be obtained by Lower-Upper framework and uniform sampling. The performance of the proposed algorithms is verified by experiments. The analysis of budget profit maximization with coupon advertisement and Robust-budget profit maximization with coupon advertisement problem is applicable to others which is a branch of submodular maximization with cardinality constraint problem. The future work comes from the

Discretized-CDG, can we give an exact approximation ratio by connecting time step Δt and sampling number λ, we do not give the strict description how small Δt is and how large λ is. Next, the running time of Discretized-CDG needed to be improved further. We have given an initial attempt to replace Monte Carlo simulation by use of RIS, but do not bound the error theoretically. If you are interested in more about it, please read (Borgs et al., 2014) (Tang et al., 2014) (Tang et al., 2015).

For the adaptive multi-Feature budgeted profit maximization problem in Chapter 5 (see p. 68), we studied it under two models, oracle model and noise model. Specifically, a $(1-1/e)$ expected approximation policy was proposed in the oracle model. Under the noise model, we computed conditional expected marginal profit of a node under a partial realization by reverse influence sampling technique and proposed an efficient algorithm, which could achieve a $(1-e^{-(1-\epsilon)})$ expected approximation ratio, where $0 < \epsilon < 1$. To evaluate the performance of our algorithms, extensive experiments were done on six realistic datasets with the comparison of our proposed policies to their corresponding non-adaptive algorithms and some heuristic adaptive policies.

For the learning-based influence maximization problem in Chapter 6 (see p. 100), we presented a novel end-to-end framework, ToupleGDD, to address it by leveraging deep reinforcement learning technique. Specifically, we incorporated graph neural networks for network embedding and reinforcement learning technique, double deep Q-networks, for parameters learning. Compared to the state-of-the-art sampling-based approximation algorithms, ToupleGDD can avoid costly sampling of the diffusion paths. Compared to previous works using deep reinforcement learning method for the influence maximization problem, our model have a stronger generalization ability and show almost consistent performance across different social networks. We conducted extensive experiments to evaluate the performance of our proposed model. The empirical results show that ToupleGDD can achieve almost equal expected spread to that of IMM and outperform OPIM-C algorithm on several datasets,

which is much better than other learning based methods. This validates the effectiveness and efficiency of the proposed ToupleGDD model.

Submodular maximization problems on social networks have been extensively studied in the last few decades. However, the state-of-the-art methods, including heuristics and approximation algorithms, faced great difficulties such as theoretical guarantee, time efficiency, generalization, etc, which makes them unable to adapt to large-scale networks and more complex applications. On the other hand, with recent emerging applications, most of the objective functions of social network problems have been shown to be non-submodular or non-linear. Currently, there is no efficient algorithm for the problem with non-monotone and non-submodular objective function. Existing works are mainly focused on designing data-driven approximation algorithms, which suffer scalability and generalization issues. With the latest achievements of deep reinforcement learning in solving the combinatorial optimization problems, I intend to design the end-to-end deep reinforcement learning based model to solve more maximization problems on social networks efficiently and effectively, such as rumor blocking problem and competitive influence maximization, which can achieve good performance even on large-scale networks.

REFERENCES

Ali, K., Wang, C.-Y., and Chen, Y.-S. (2018). Boosting reinforcement learning in competitive influence maximization with transfer learning. In *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 395–400. IEEE.

Ali, K., Wang, C.-Y., Yeh, M.-Y., and Chen, Y.-S. (2020). Addressing competitive influence maximization on unknown social network with deep reinforcement learning. In *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 196–203. IEEE.

Arthur, D., Motwani, R., Sharma, A., and Xu, Y. (2009). Pricing strategies for viral marketing on social networks. In *International workshop on internet and network economics*, pages 101–112. Springer.

Banerjee, S., Jenamani, M., and Pratihar, D. K. (2020). Earned benefit maximization in social networks under budget constraint. *Expert Systems with Applications*, 169.

Bengio, Y., Lodi, A., and Prouvost, A. (2021). Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421.

Borgs, C., Brautbar, M., Chayes, J., and Lucier, B. (2014). Maximizing social influence in nearly optimal time. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 946–957. SIAM.

Borodin, A., Filmus, Y., and Oren, J. (2010). Threshold models for competitive influence in social networks. In *International workshop on internet and network economics*, pages 539–550. Springer.

Buchbinder, N., Feldman, M., Naor, J., and Schwartz, R. (2012). A tight linear time (1/2)-approximation for unconstrained submodular maximization. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 649–658.

Buchbinder, N., Feldman, M., Naor, J. S., and Schwartz, R. (2014). Submodular maximization with cardinality constraints. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1433–1452. Society for Industrial and Applied Mathematics.

Buchbinder, N., Feldman, M., Seffi, J., and Schwartz, R. (2015). A tight linear time (1/2)-approximation for unconstrained submodular maximization. *SIAM Journal on Computing*, 44(5):1384–1402.

Cai, Z., He, Z., Guan, X., and Li, Y. (2016). Collective data-sanitization for preventing sensitive information inference attacks in social networks. *IEEE Transactions on Dependable and Secure Computing*, 15(4):577–590.

Calinescu, G., Chekuri, C., Pál, M., and Vondrák, J. (2011). Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766.

Cao, Q., Shen, H., Gao, J., Wei, B., and Cheng, X. (2020). Popularity prediction on social platforms with coupled graph neural networks. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 70–78.

Chen, H., Qiu, W., Ou, H.-C., An, B., and Tambe, M. (2021). Contingency-aware influence maximization: A reinforcement learning approach. In *Uncertainty in Artificial Intelligence*, pages 1535–1545. PMLR.

Chen, T., Guo, J., and Wu, W. (2022). Adaptive multi-feature budgeted profit maximization in social networks. *Social Network Analysis and Mining*, 12(1):164.

Chen, T., Liu, B., Liu, W., Fang, Q., Yuan, J., and Wu, W. (2020a). A random algorithm for profit maximization in online social networks. *Theoretical Computer Science*, 803:36–47.

Chen, T., Liu, W., Fang, Q., Guo, J., and Du, D.-Z. (2019). Minimizing misinformation profit in social networks. *IEEE Transactions on Computational Social Systems*, 6(6):1206–1218.

Chen, W. (2018). An issue in the martingale analysis of the influence maximization algorithm imm. In *International Conference on Computational Social Networks*, pages 286–297. Springer.

Chen, W., Collins, A., Cummings, R., Ke, T., Liu, Z., Rincon, D., Sun, X., Wang, Y., Wei, W., and Yuan, Y. (2011). Influence maximization in social networks when negative opinions may emerge and propagate. In *Proceedings of the 2011 siam international conference on data mining*, pages 379–390. SIAM.

Chen, W., Lin, T., Tan, Z., Zhao, M., and Zhou, X. (2016). Robust influence maximization. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 795–804. ACM.

Chen, W. and Peng, B. (2019). On adaptivity gaps of influence maximization under the independent cascade model with full-adoption feedback. In *Proceedings of the 30th International Symposium on Algorithms and Computation (ISAAC'2019)*.

Chen, W., Peng, B., Schoenebeck, G., and Tao, B. (2020b). Adaptive greedy versus non-adaptive greedy for influence maximization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 590–597.

Chen, W., Wang, C., and Wang, Y. (2010a). 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.

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

Chen, W., Wu, R., and Yu, Z. (2018). Scalable lattice influence maximization. *arXiv preprint arXiv:1802.04555*.

Chen, W., Yuan, Y., and Zhang, L. (2010b). Scalable influence maximization in social networks under the linear threshold model. In *2010 IEEE international conference on data mining*, pages 88–97. IEEE.

Dai, H., Dai, B., and Song, L. (2016). Discriminative embeddings of latent variable models for structured data. In *International conference on machine learning*, pages 2702–2711. PMLR.

Domingos, P. and Richardson, M. (2001). 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.

Fan, C., Zeng, L., Sun, Y., and Liu, Y.-Y. (2020). Finding key players in complex networks through deep reinforcement learning. *Nature machine intelligence*, 2(6):317–324.

Feng, S., Cong, G., Khan, A., Li, X., Liu, Y., and Chee, Y. M. (2018). Inf2vec: Latent representation model for social influence embedding. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 941–952. IEEE.

Golovin, D. and Krause, A. (2011). Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42:427–486.

Goyal, A., Bonchi, F., and Lakshmanan, L. V. (2010). Learning influence probabilities in social networks. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 241–250. ACM.

Goyal, A., Lu, W., and Lakshmanan, L. V. (2011a). Celf++ optimizing the greedy algorithm for influence maximization in social networks. In *Proceedings of the 20th international conference companion on World wide web*, pages 47–48.

Goyal, A., Lu, W., and Lakshmanan, L. V. (2011b). Simpath: An efficient algorithm for influence maximization under the linear threshold model. In *2011 IEEE 11th international conference on data mining*, pages 211–220. IEEE.

Guo, J., Chen, T., and Wu, W. (2020a). Budgeted coupon advertisement problem: Algorithm and robust analysis. *IEEE Transactions on Network Science and Engineering*, 7(3):1966–1976.

Guo, J., Chen, T., and Wu, W. (2020b). Continuous activity maximization in online social networks. *IEEE Transactions on Network Science and Engineering*, 7(4):2775–2786.

Guo, J., Chen, T., and Wu, W. (2020c). A multi-feature diffusion model: Rumor blocking in social networks. *IEEE/ACM Transactions on Networking*, 29(1):386–397.

Guo, J., Li, Y., and Wu, W. (2019). Targeted protection maximization in social networks. *IEEE Transactions on Network Science and Engineering*, pages 1–1.

Guo, J. and Wu, W. (2019). A novel scene of viral marketing for complementary products. *IEEE Transactions on Computational Social Systems*, 6(4):797–808.

Guo, J. and Wu, W. (2020a). Adaptive influence maximization: If influential node unwilling to be the seed. *arXiv preprint arXiv:2005.08060*.

Guo, J. and Wu, W. (2020b). A k-hop collaborate game model: Adaptive strategy to maximize total revenue. *IEEE Transactions on Computational Social Systems*, 7(4):1058–1068.

Guo, Q., Wang, S., Wei, Z., and Chen, M. (2020d). Influence maximization revisited: Efficient reverse reachable set generation with bound tightened. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 2167–2181.

Han, K., Huang, K., Xiao, X., Tang, J., Sun, A., and Tang, X. (2018). Efficient algorithms for adaptive influence maximization. *Proceedings of the VLDB Endowment*, 11(9):1029–1040.

Hassani, H., Soltanolkotabi, M., and Karbasi, A. (2017). Gradient methods for submodular maximization. In *Advances in Neural Information Processing Systems*, pages 5841–5851.

Hatano, D., Fukunaga, T., and Kawarabayashi, K.-I. (2016). Adaptive budget allocation for maximizing influence of advertisements. In *IJCAI*, pages 3600–3608.

He, X. and Kempe, D. (2015). Stability of influence maximization. *arXiv preprint arXiv:1501.04579*.

He, Z., Cai, Z., Yu, J., Wang, X., Sun, Y., and Li, Y. (2016). Cost-efficient strategies for restraining rumor spreading in mobile social networks. *IEEE Transactions on Vehicular Technology*, 66(3):2789–2800.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*, pages 3215–3222.

Hoeffding, W. (1994). Probability inequalities for sums of bounded random variables. In *The Collected Works of Wassily Hoeffding*, pages 409–426. Springer.

Huang, K., Tang, J., Han, K., Xiao, X., Chen, W., Sun, A., Tang, X., and Lim, A. (2020). Efficient approximation algorithms for adaptive influence maximization. *The VLDB Journal*, 29(6):1385–1406.

Huang, K., Wang, S., Bevilacqua, G., Xiao, X., and Lakshmanan, L. V. (2017). Revisiting the stop-and-stare algorithms for influence maximization. *Proceedings of the VLDB Endowment*, 10(9):913–924.

Ireland, D. and Montana, G. (2022). Lense: Learning to navigate subgraph embeddings for large-scale combinatorial optimisation. In *International conference on machine learning*, pages 9622–9638. PMLR.

Jung, K., Chen, W., and Heo, W. (2011). Irie: A scalable influence maximization algorithm for independent cascade model and its extensions. Technical report.

Jung, K., Heo, W., and Chen, W. (2012). Irie: Scalable and robust influence maximization in social networks. In *2012 IEEE 12th International Conference on Data Mining*, pages 918–923. IEEE.

Kamarthi, H., Vijayan, P., Wilder, B., Ravindran, B., and Tambe, M. (2020). Influence maximization in unknown social networks: Learning policies for effective graph sampling. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 575–583.

Kempe, D., Kleinberg, J., and Tardos, É. (2003). 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.

Kempe, D., Kleinberg, J., and Tardos, É. (2015). Maximizing the spread of influence through a social network. *Theory OF Computing*, 11(4):105–147.

Khalil, E., Dai, H., Zhang, Y., Dilkina, B., and Song, L. (2017). Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30.

Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., Faloutsos, C., VanBriesen, J., and Glance, N. (2007). 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.

Leskovec, J. and Krevl, A. (2014). SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`.

Li, H., Xu, M., Bhowmick, S. S., Rayhan, J. S., Sun, C., and Cui, J. (2022). Piano: Influence maximization meets deep reinforcement learning. *IEEE Transactions on Computational Social Systems*.

Li, H., Xu, M., Bhowmick, S. S., Sun, C., Jiang, Z., and Cui, J. (2019). Disco: Influence maximization meets network embedding and deep learning. *arXiv preprint arXiv:1906.07378*.

Li, Z., Chen, Q., and Koltun, V. (2018). Combinatorial optimization with graph convolutional networks and guided tree search. *Advances in neural information processing systems*, 31:537–546.

Lin, S.-C., Lin, S.-D., and Chen, M.-S. (2015). A learning-based framework to handle multi-round multi-party influence maximization on social networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 695–704.

Lin, Y., Cai, Z., Wang, X., and Hao, F. (2019). Incentive mechanisms for crowdblocking rumors in mobile social networks. *IEEE Transactions on Vehicular Technology*, 68(9):9220–9232.

Liu, B., Li, X., Wang, H., Fang, Q., Dong, J., and Wu, W. (2020). Profit maximization problem with coupons in social networks. *Theoretical Computer Science*, 803:22–35.

Lu, W., Chen, W., and Lakshmanan, L. V. (2015). From competition to complementarity: comparative influence diffusion and maximization. *Proceedings of the VLDB Endowment*, 9(2):60–71.

Lu, W. and Lakshmanan, L. V. (2012). Profit maximization over social networks. In *2012 IEEE 12th International Conference on Data Mining*, pages 479–488. IEEE.

Lu, Z., Zhou, H., Li, V. O., and Long, Y. (2016). Pricing game of celebrities in sponsored viral marketing in online social networks with a greedy advertising platform. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE.

Maas, A. L., Hannun, A. Y., Ng, A. Y., et al. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer.

Maehara, T., Yabe, A., and Kawarabayashi, K.-i. (2015). Budget allocation problem with multiple advertisers: A game theoretic view. In *ICML*, volume 32, pages 428–437.

Manchanda, S., Mittal, A., Dhawan, A., Medya, S., Ranu, S., and Singh, A. (2020). Gcomb: Learning budget-constrained combinatorial algorithms over billion-sized graphs. *Advances in Neural Information Processing Systems*, 33:20000–20011.

Mazyavkina, N., Sviridov, S., Ivanov, S., and Burnaev, E. (2021). Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134:105400.

Minoux, M. (1978). Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization techniques*, pages 234–243. Springer.

Miyauchi, A., Iwamasa, Y., Fukunaga, T., and Kakimura, N. (2015). Threshold influence model for allocating advertising budgets. In *International Conference on Machine Learning*, pages 1395–1404.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

Moore, E. F. (1959). The shortest path through a maze. In *Proceedings of the International Symposium on the Theory of Switching*, pages 285–292.

Motwani, R. and Raghavan, P. (1995). *Randomized algorithms*. Cambridge university press.

Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. (1978). An analysis of approximations for maximizing submodular set functions—I. *Mathematical programming*, 14(1):265–294.

Netrapalli, P. and Sanghavi, S. (2012). Learning the graph of epidemic cascades. In *ACM SIGMETRICS Performance Evaluation Review*, volume 40, pages 211–222. ACM.

Nguyen, H. T., Dinh, T. N., and Thai, M. T. (2016a). Cost-aware targeted viral marketing in billion-scale networks. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE.

Nguyen, H. T., Thai, M. T., and Dinh, T. N. (2016b). Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. In *Proceedings of the 2016 International Conference on Management of Data*, pages 695–710.

Nguyen, H. T., Thai, M. T., and Dinh, T. N. (2017). A billion-scale approximation algorithm for maximizing benefit in viral marketing. *IEEE/ACM Transactions On Networking*, 25(4):2419–2429.

Peng, B. and Chen, W. (2019). Adaptive influence maximization with myopic feedback. In *NeurIPS*.

Richardson, M. and Domingos, P. (2002). Mining knowledge-sharing sites for viral marketing. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 61–70. ACM.

Riedmiller, M. A. (2005). Neural fitted Q iteration - first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer.

Rossi, R. and Ahmed, N. (2015). The network data repository with interactive graph analytics and visualization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

Saito, K., Nakano, R., and Kimura, M. (2008). Prediction of information diffusion probabilities for independent cascade model. In *International conference on knowledge-based and intelligent information and engineering systems*, pages 67–75. Springer.

Shan, X., Chen, W., Li, Q., Sun, X., and Zhang, J. (2019). Cumulative activation in social networks. *Science China Information Sciences*, 62(5):1–21.

Soma, T. (2017). Non-monotone dr-submodular function maximization. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 898–904. AAAI.

Soma, T., Kakimura, N., Inaba, K., and Kawarabayashi, K.-i. (2014). Optimal budget allocation: Theoretical guarantee and efficient algorithm. In *International Conference on Machine Learning*, pages 351–359.

Soma, T. and Yoshida, Y. (2015). A generalization of submodular cover via the diminishing return property on the integer lattice. In *Advances in Neural Information Processing Systems*, pages 847–855.

Soma, T. and Yoshida, Y. (2018). Maximizing monotone submodular functions over the integer lattice. *Mathematical Programming*, 172(1-2):539–563.

Sun, L., Huang, W., Yu, P. S., and Chen, W. (2018). Multi-round influence maximization. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2249–2258.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Sviridenko, M. (2004). A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43.

Tang, J., Sun, J., Wang, C., and Yang, Z. (2009). Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 807–816. ACM.

Tang, J., Tang, X., Xiao, X., and Yuan, J. (2018). Online processing algorithms for influence maximization. In *Proceedings of the 2018 International Conference on Management of Data*, pages 991–1005.

Tang, J., Tang, X., and Yuan, J. (2016). Profit maximization for viral marketing in online social networks. In *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, pages 1–10. IEEE.

Tang, Y., Shi, Y., and Xiao, X. (2015). Influence maximization in near-linear time: A martingale approach. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1539–1554.

Tang, Y., Xiao, X., and Shi, Y. (2014). 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.

Tian, S., Zhang, P., Mo, S., Wang, L., and Peng, Z. (2019). A learning approach for topic-aware influence maximization. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*, pages 125–140. Springer.

Tong, G., Wu, W., and Du, D.-Z. (2018). Coupon advertising in online social systems: Algorithms and sampling techniques. *arXiv preprint arXiv:1802.06946*.

Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, pages 2094–2100.

Vondrák, J. (2013). Symmetry and approximability of submodular maximization problems. *SIAM Journal on Computing*, 42(1):265–304.

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

Wang, Y., Cai, Z., Zhan, Z.-H., Gong, Y.-J., and Tong, X. (2019). An optimization and auction-based incentive mechanism to maximize social welfare for mobile crowdsourcing. *IEEE Transactions on Computational Social Systems*, 6(3):414–429.

Wang, Z., Yang, Y., Pei, J., Chu, L., and Chen, E. (2017). Activity maximization by effective information diffusion in social networks. *IEEE Transactions on Knowledge and Data Engineering*, 29(11):2374–2387.

Yadav, A., Noothigattu, R., Rice, E., Onasch-Vera, L., Soriano Marcolino, L., and Tambe, M. (2018). Please be an influencer?: Contingency-aware influence maximization. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1423–1431.

Yang, Y., Mao, X., Pei, J., and He, X. (2016). Continuous influence maximization: What discounts should we offer to social network users? In *Proceedings of the 2016 international conference on management of data*, pages 727–741. ACM.

Yang, Y. and Whinston, A. (2020). A survey on reinforcement learning for combinatorial optimization. *arXiv preprint arXiv:2008.12248*.

Zhang, H., Mishra, S., Thai, M. T., Wu, J., and Wang, Y. (2014). Recent advances in information diffusion and influence maximization in complex social networks. *Opportunistic Mobile Social Networks*, 37(1.1):37.

Zhang, H., Zhang, H., Kuhnle, A., and Thai, M. T. (2016). Profit maximization for multiple products in online social networks. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE.

Zhang, Y., Yang, X., Gao, S., and Yang, W. (2019). Budgeted profit maximization under the multiple products independent cascade model. *IEEE Access*, 7:20040–20049.

Zhou, F., Jiao, R. J., and Lei, B. (2015). Bilevel game-theoretic optimization for product adoption maximization incorporating social network effects. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(8):1047–1060.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI open*, 1:57–81.

BIOGRAPHICAL SKETCH

Tiantian Chen received her Bachelor of Science degree in Mathematics and Applied Mathematics, and Master of Science degree in Operational Research and Cybernetics from Ocean University of China in 2016 and 2019, respectively. She joined the Department of Computer Science, The University of Texas at Dallas, as a PhD student in 2019. She worked under the supervision of Dr. Weili Wu and Dr. Ding-Zhu Du. Her research interests include reinforcement learning, deep learning, social networks, blockchain, and design and analysis of approximation algorithms.

CURRICULUM VITAE

Tiantian Chen

May 1, 2023

Contact Information:

Department of Computer Science
The University of Texas at Dallas
800 W. Campbell Rd.
Richardson, TX 75080-3021, U.S.A.

Email: tiantian.chen@utdallas.edu

Educational History:

BS, Mathematics and Applied Mathematics, Ocean University of China, June 2016
MS, Operational Research and Cybernetics, Ocean University of China, June 2019
PhD, Computer Science, The University of Texas at Dallas, May 2023

Variant Influence Maximization: Approximation Algorithm and Deep Solution
PhD Dissertation
Department of Computer Science, The University of Texas at Dallas
Advisors: Dr. Weili Wu and Dr. Ding-Zhu Du

Employment History:

Teaching Assistant, The University of Texas at Dallas, August 2019 – present

Professional Recognitions and Honors:

Outstanding Academic Performance, and 2022 Grace Hopper Conference (GHC'22) Scholarship, The University of Texas at Dallas, 2022 – 2023
Outstanding Graduate Award of Shandong province, Outstanding Graduate Award and the Second Class University's Scholarship, Ocean University of China, 2018 – 2019
Second Prize in the Thirteenth National Post-Graduate Mathematical Contest in Modeling, Excellent Graduates Award, Excellent Graduate Scholarship, Ocean University of China, 2016-2017
Honorable Mention Award in 2015 Mathematical Contest in Modeling, National Scholarship, Outstanding Student Award of Shandong province, Outstanding Student Leader Award, and Outstanding Student Award of university, Ocean University of China, 2014-2015
National Scholarship, Provincial First Prize in 2014 China Undergraduate Mathematical Contest in Modelling, and Outstanding Student Award, Ocean University of China, 2013-2014
National Inspiration Scholarship, Ocean University of China, 2012-2013