Learning and Decision Making in Social Media Networks

Zhecheng Qiang
University of Central Florida

LEARNING AND DECISION MAKING IN SOCIAL MEDIA NETWORKS

by

ZHECHENG QIANG
M.S. University of Florida, 2015
B.S. Nanjing University, 2014

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Industrial Engineering and Management Systems
in the College of Engineering & Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2022

Major Professor: Qipeng Zheng

ABSTRACT

Social media is a virtual community where users share news, ideas, interests, and information. Learning the information diffusion dynamics and making decisions correspondingly, e.g., selecting the seed nodes to maximize the influence, have been widely applied to the areas of viral marketing and cyber security. In this dissertation, we study the problem of learning diffusion process, i.e., infection prediction, in social media networks utilizing both feature-based machine learning methods and mathematical model-based methods. For feature-based machine learning methods, the neighborhood information is treated as an important feature together with user profile and content similarity features. For model-based methods, two distinctive mathematical models, i.e., Linear Threshold Learning Model and Random Walk Learning Model, are proposed to learn the information diffusion dynamics. Neural networks are implemented to train the proposed models for all aforementioned methods. In this dissertation, we also study the problem of choosing seed nodes to maximize the influence in social media networks. In one project, the problem is addressed through solving the tiered influence and activation thresholds target set selection problem, which is to find the seed nodes that can influence the most users within a limited time frame. Both the minimum influential seeds and maximum influence within budget problems are considered in this study. In addition, we study the impacts arising from the uncertainties in network structures, user behavior and activation prices via two-stage stochastic optimization as well.

Keywords: Mathematical Optimization, Influence Maximization, Stochastic Programming, Infection Prediction, Machine Learning, Social Media, Network Optimization

ACKNOWLEDGMENTS

Throughout my career as a graduate student, I have received a great deal of supports. First and foremost, I would like to thank my advisor Dr. Qipeng Zheng for his insightful view in formulating the research topics. Dr. Zheng is an extremely kind, compassionate and supportive advisor that I could not have asked for more. He constantly believes in me, even though I don't always believe in myself. I would also like to thank all of my research colleagues in our group as well, who gave me a lot of support and help at various times. I feel very proud of being part of the group. We spent a lot of pleasant time cooperating and sharing experience with each other.

I gratefully acknowledge my cooperators Dr. Eduardo Pasiliao, Dr. Alexander Semenov and Lily Schleider. Thank you for your excellent cooperation and for all the opportunities I was given to conduct my research. In particular, Dr. Pasiliao - I cooperated with him for a couple of papers. He always has a high level view about the research field. His great passion in research always inspires me. I would like to extend my thanks to my committee members: Dr. Vladimir Boginski, Dr. Luis Rabelo, and Dr. Wei Zhang for their great efforts and support in my dissertation and defense.

I would also like to thank my parents for their unconditional love, my husband for his sympathetic ear and continued support, my daughter and son for those happy distractions to rest my mind outside the research. I am always grateful to my family for their love and support to make who I am today.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Motivation

Social media has become an important platform for people to connect with each other, share their opinions, news and spread the influence. Thus social media plays an important role in spreading the information. Understanding how the information propagates is essential for successful applications of viral marketing [1] and cyber security [2] in social media networks. To this end, researchers have defined various diffusion tasks to tackle the real world problems. The tasks include:

- Buzz prediction [3]: It predicts whether a topic is going to be a trend before it is actually declared to be a trend.
- Volume prediction [4]: It predicts the spread of an idea in a given time frame.
- Infection prediction [5]: It learns the diffusion parameters between two users under the propagation models.
- Link detection [6]: It predicts the edges of the information propagation network and estimates the transmission rates of each edge that best explain the observed data.
- Influence maximization [7, 8, 9, 10, 11]: It selects a limited number of nodes at the beginning that spread the most influence at the end.
- Firefighter problem [12]: It stops the infection by selecting the targeted vaccination nodes at the beginning when an infection is spreading through the network.

As we can see the diffusion tasks could be classified into either learning problems including buzz prediction, volume prediction, infection prediction and link prediction or decision making problems including influence maximization and firefighter problems in social media. In this thesis, we study the problem of learning and decision making in social media. In particular we focus on infection prediction and influence maximization problems. To predict the information diffusion between each pair of users is one of the most important tasks in prediction. On one hand, it applies to analyze human activities and explore the factors influencing the information diffusion, which helps business and organizations to make more attractive advertisements to boost the business. On the other hand, learning diffusion probability is the first step to tackle the influence maximization problem. Influence maximization problem involves finding a limited number of nodes that have the largest influence for the spread of information. Influence maximization is widely used in the areas of viral marketing and misinformation detection. In viral marketing, advertisers pay incentives to the selected influencers so that they could help to promote their products. In cyber security, we put the detectors on important users so that the misinformation could be detected in the early stage of spread. Besides, some users could be identified to be exposed to the true information before they hear the misinformation to limit the spread of misinformation in social media.

1.2 Methodologies

In the thesis, we investigate the infection prediction and influence maximization problems using both machine learning and mathematical optimization methods. Machine learning is the science of getting algorithms to find patterns or learn how to do tasks without telling computers the exact mathematical models. It has good performance but suffers in interpretability. Machine learning methods have been widely used in prediction tasks in social media. Mathematical optimization is more explainable and flexible for different requirements. In the thesis, we focus more on investigating the infection prediction and influence maximization problems using mathematical optimization methods.

1.2.1 Mathematical Optimization

Mathematical optimization selects the best solution from some set of available alternatives regarding to the problems of maximizing or minimizing an objective function subject to some constraints. The mathematical formula is shown below:

$$\begin{aligned} \min \quad & f(x) \\ s.t. \quad & g(x) \leq 0 \end{aligned} \tag{OBJ}$$

Where $f(x)$ and $g(x)$ correspond to objective function and constraints. x represents the decision variables. Mathematical optimization has wide applications in all quantitative disciplines in computer science and engineering, operations research and mathematics. Here we apply mathematical optimization to tackle the learning and decision making problems in social media considering different diffusion models.

Our mathematical optimization models are built based on the diffusion models. There are two widely used diffusion models in the literature, namely Independent Cascade Model (IC) [7] and Linear Threshold Model (LT) [13]. Independent Cascade Model assumes every node has one single chance to activate another node. Linear Threshold Model shown in Figure 1.1 assumes that a node is activated when the total weights of its neighbors' influence are at least θ_j. All of our proposed models are threshold based models.

1.2.1.1 Stochastic Programming

In the field of mathematical optimization, stochastic programming is a framework for modeling optimization problems that consider uncertainty. The uncertainty could lie in some or all problem parameters. The random parameters may appear in objective functions or constraints, but follow known probability distributions. Considering many real-world decisions involve uncertainty,

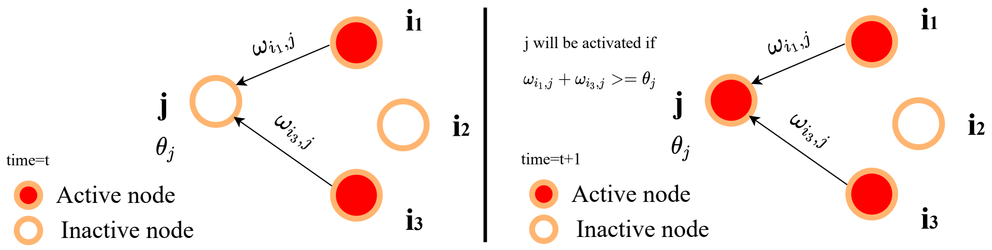


Figure 1.1: Linear Threshold Propagation Model

stochastic programming has wide applications in the areas of finance, energy and transportation.

The most widely used stochastic programming models are two-stage stochastic programs, where the decision maker has to make decisions in two stages(two different times) considering the uncertainty. At the first stage, the decision maker should make the decision based on the available data and cannot depend on future observations. Based on the observed random event and outcome of the first-stage decision, the recourse decision is made in the second stage. The general formulation of a two-stage stochastic programming problem is given by:

$$\min_{x \in X} \quad f(x) + E[Q(x,\xi)]$$

Where $Q(x,\xi)$ is the optimal value of the second-stage problem based on the solution of first stage decision $\hat{x}$.

$$Q(x,\xi) = \min\{q^T y | Wy = h - Tx, y >= 0\}$$

Where x represent the first stage decisions and y represent the second stage decisions.

1.3 Outline

The dissertation studies the learning and decision making problems in social media. It mainly focuses on solving the infection prediction and influence maximization problems. Different methods, e.g., mathematical optimization, machine learning, graph theory and stochastic programming are proposed to study the problems from different angles. Besides, several novel mathematical optimization models are presented and various advanced solution algorithms are developed to deal with the computational difficulty of each problem.

As we discuss, the thesis includes two themes: infection prediction and influence maximization. The outline of the thesis is summarized below.

Chapter 2 and Chapter 3 are tackling the infection prediction problem using either feature-based machine learning method or mathematical optimization method. In Chapter 2, we propose a new framework combining user profile, content similarity and the neighborhood information around each target link as input features to make the repost prediction. Here neighborhood information can be interpreted as the combination of neighbors' user profile. After collecting the input features, we implement the state-of-the-art machine learning methods, e.g., Logistic Regression, K-nearest Neighbors, Gaussian Naive Bayes, Deep Neural Network, Random Forest, XGBoosting and Stacking Model to predict repost probability. In Chapter 3, two learning models are proposed that are aimed at learning person-to-person influence in information diffusion from historical cascades based on the threshold propagation model. The first model is based on the linear threshold propagation model. In addition, by considering multi-step information propagation in one time period, this paper proposes a learning model for multi-step diffusion influence between pairs of users based on the idea of random walk. Mixed integer programs (MIP) have been used to learn these models by minimizing the prediction errors, where decision variables are estimations of the diffusion influence between pairs of users. For large-scale networks, this paper develops approximate methods for those learning models by using artificial neural networks to learn the pairwise influence.

Chapter 4 and Chapter 5 are solving the variants of the influence maximization problem. In Chapter 4, we address the problem through solving the tiered influence and activation thresholds target set selection problem, which is to find the seed nodes that can influence the most users within a limited time frame. Both the minimum influential seeds and maximum influence within budget problems are considered in this study. Besides, this study proposes several models exploiting different requirements on seed nodes selection, such as maximum activation, early activation and dynamic threshold. These time-indexed integer program models suffer from the computational difficulties due to the large number of binary variables to model influence actions at each time epoch. To address this challenge, this paper designs and leverages several efficient algorithms, i.e., Graph Partition, Nodes Selection, Greedy Algorithm, Recursive Threshold Back Algorithm and Two-stage Approach in Time, especially for large-scale networks. In Chapter 5, we tackle the influence maximization problem considering the uncertainties in network structures, user behavior and activation prices. We formulate the problems as two-stage stochastic optimization problems, and solve them via the Sample Average Approximation method. Chapter 6 concludes the dissertation.

CHAPTER 2: INCORPORATING NEIGHBORHOOD INFORMATION INTO A REPOST PREDICTION MODEL IN SOCIAL MEDIA NETWORKS

2.1 Introduction

Mass adoption of mobile devices forever changed the way people communicate and interact with each other. The planet suddenly became unexpectedly smaller for its seven billion residents, especially in the sense of communication. One irreversible result is that people have adopted social media platforms as the major venue for sharing their opinions and news, and for spreading their influence. This poses a series of challenges and opportunities. Understanding how the information propagates within social media platforms is essential for successful implementation of viral marketing [1] and cyber security [2] in social media networks. One of the most essential and fundamental information propagation tasks is to predict the information diffusion between each pair of users. On one hand, it has applications in the analysis of human activities and the exploration of the factors that influence information diffusion, which helps business and organizations in crafting more attractive advertisements to boost the business. On the other hand, learning diffusion probability is the first step in tackling the influence maximization problem. Influence maximization problem involves finding a limited number of nodes that have the largest influence for the spread of information. Influence maximization is widely used in the areas of viral marketing and misinformation detection.

Although social media platforms differ slightly from one another in information diffusion and cascade, the predominant dynamic for information diffusion is through the reposting or retweeting of messages. Unfortunately, the reposting probability for each user within a network is not readily available from the massive historical dataset of social media networks. To fill this gap, several research articles have studied the problem of learning repost probability in online social media networks.

Most of the machine learning models learned the repost probability through content and user profile features [14, 15, 16, 17, 18, 19]. According to previous research, user profile is a better indicator of information diffusion in social media [20, 21]. In addition, user preference for reposting may differ from post to post. Studies have shown that message propagation occurs more frequently when the posts are in line with user interests [22]. Zhu et al. [19] implemented logistic regression to model a user's repost decision. The article considered the content influence, including topic similarity, URLs, hashtags, mentions, user profile, and time influence, as factors for making a prediction. Lagnier et al. [17] proposed a logistic regression model that considers the similarity between content and user, the willingness to diffuse, and the time-decaying social pressure as important factors of repost prediction. Jiang et al. [16] studied the retweetability of users by incorporating interest similarity and social influence information into a one-class collaborative filtering model. Retweetability prediction models can be treated as binary classification models, which require positive and negative observations for learning the parameters. Usually, positive observations are easy to get; however, negative observations are not readily available. Therefore, researchers have to treat the missing data (no repost) as negative observations. The innovation of this approach lies in assigning the combined score of interest similarity and social influence information for each missing observation to give them different weights. Recently, Varshney, Kumar, and Gupta [14] proposed a Bayesian network based approach to predict the repost probability between each pair of users for messages of different topics. The approach considered diffusion history, user profile, topic information, network connection, activity and similarity as important features that influenced repost prediction in social media network.

For large scale social media networks, the relationship between important factors in a user's repost decision and repost probability is complex and nonlinear. Therefore, we implement several machine learning approaches to estimate the repost probability. We take all the essential factors that might affect the diffusion process into consideration, including user profile, neighborhood information, and content similarity. The contributions of the project are summarized as follows:

- We take user neighborhood information into consideration. Each user's friend circle is a good indicator of that user's repost decison. Users who have very active friends tend to be more active in reposting messages. To collect neighborhood information, we introduce two different combination models of neighbors' user profile from a graph theoretic perspective. We analyze both combination models and compare their performance in learning repost probability.

- Different state-of-the-art machine learning models are implemented to estimate the repost probability. In addition, we analyze the importance of different features based on tree-based ensemble prediction models as well.

- We use Bert Sentence Embedding, instead of the more often used LDA in research studies to extract the information embedding, which works better in generating dense vector representations for short sentences.

2.2 Repost Prediction Model

In social media, information propagates primarily through the reposting or retweeting of messages. Understanding repost probabilities contributes to better decision making in social media. We model the social media network as a directed graph $G = (V,E)$, where $V = \{v_1,\dots,v_n\}$ is the set of users of the network and $E \subseteq V \times V$ is the set of edges representing the friend relationships between users. Each user has a list of attributes $A = \{a_1,\dots,a_m\}$, which is referred to as the user profile in social media. And we define the content of each message or post using M.

Researchers have demonstrated that post content and user profile are major factors in determining a users' repost decisions [21]. Additionally, we think the neighborhood user profile also contributes to the prediction of the repost decisions as well. Active neighbors are expected to have a positive influence in the users, making the users more active in reposting when they are exposed to more posts from the neighbors. Thus we extract the features of user profile, neighborhood information and message embedding similarity from the original data user profile, graph topology and messages. Therefore, we aim to model the reposting or retweeting probability $p(i,j,m)$ between source user i and target user j, $i,j \in V$ for the message m as a function of user profile(A), neighborhood information(N) and content embedding similarity(S).

$$p(i,j,m) = f(A,N,S,\theta) \tag{2.1}$$

Considering the complexity of the function, we implement state-of-the-art machine learning methods to approximate the function and get the estimation of parameter θ. The repost prediction model is shown in Figure 2.1.

2.2.1 Feature Extraction

The user profile(A), network topology(G) and messages(M) are original data we could obtain from the dataset directly. Based on the original data, we extract three different kinds of features: user profile, neighborhood information and content embedding similarity as input features to make the prediction.

User Profile

User profile features are generally good predictors of the repost probability. We consider user profile of both source user and target user. The widely used user profile include the total number of followers, the total number of followees, the total number of posts or reposts, the total number of reciprocal relationships, gender and the created time.

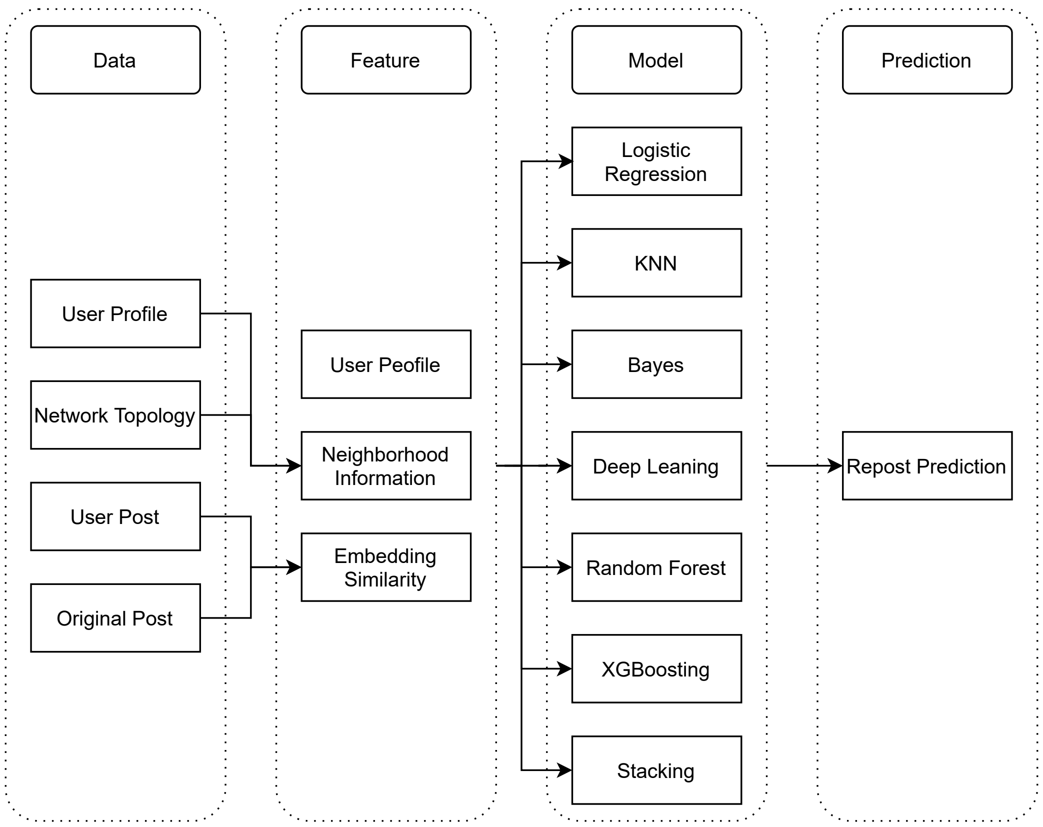


Figure 2.1: Repost Prediction Model

Neighborhood Information

Information from a user's circle of friends is a good indicator of repost decision. Commonly, a user with active friends tend to be more active in reposting. It is natural to expect that friends (reciprocal relationships) share similarity in behaviors with each other. Here the neighborhood information ($N \subseteq G \times A$) contains both the topology of neighborhood and the user profile of the neighbors.

Subgraph Extraction

To get the neighborhood for each pair of source user i and target user j, we extract an enclosing subgraph around them as the circle of friends of the pair. The subgraph is extracted from the network by the union of all user i and j' s friends nodes within 1 hop. The set of nodes in the 1-hop subgraph is defined as:

$$V_{i,j}^{1} = \{v | d(v,i) \leqslant 1 \quad \text{or} \quad d(v,j) \leqslant 1\} \tag{2.2}$$

We use the adjacency matrix $\tilde{A}$ to represent the structure of the subgraph, where $\tilde{A}_{i,j} = 1$ if $(i,j) \subset E$ and $\tilde{A}_{i,j} = 0$ otherwise. Considering the memory limitation and computation speed, we only consider the first two rows that contain the friendship relationship of source user i and target user j as matrix $\tilde{A}'$. Suppose the 1-hop subgraph contains n users in total, then the extracted adjacency matrix is $\tilde{A}' \subseteq \{1,0\}^{2\times n}$. The enclosing subgraph contains rich information about the pair's circle of friends. Figure 2.2 shows an illustration of subgraph extraction.

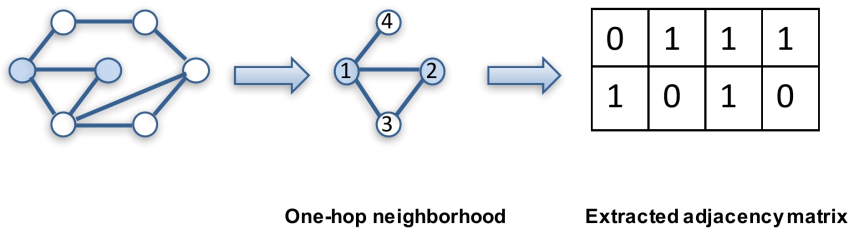


Figure 2.2: Illustration of Subgraph Extraction

Combination Models

After extracting the neighborhood of the pair, we combine user profile features of each user in the neighborhood. The neighborhood information features can be incorporated as a combination of circle of friends' user profile features. Here we introduce different combination models to get the neighborhood information.

Combination Model 1

The first combination model combines user profile of neighbors using extracted adjacency matrix with added self loop.Then the neighborhood information could be represented as:

$$N = (\tilde{A}' + I')X_G \tag{2.3}$$

where $\tilde{A}'$ represents the extracted adjacency matrix of subgraph, I' represents the first two rows of the identity matrix and $X_G \subseteq R^{n\times c}$ is the selected user profile of neighbors of the pair. From another prospective, the neighborhood information could also be interpreted as the sum of the users and their neighbors' user profile features.

Combination Model 2

The second combination model is inspired by graph Laplacian. The formula of graph Laplacian is shown below:

$$L = D - \tilde{A}, \tag{2.4}$$

where D is the degree matrix and $\tilde{A}$ is the adjacency matrix of the subgraph. Then we normalize the graph Laplacian. The random walk normalized Laplacian is:

$$L = D^{-1}(D - \tilde{A}) = I - D^{-1}\tilde{A} \tag{2.5}$$

To reduce the computation complexity, we introduce the following renormalization trick: $I - D^{-1}\tilde{A} \rightarrow D^{-1}(\tilde{A} + I)$ [23]. Then the neighborhood information can be represented as:

$$N = D'^{-1}(\tilde{A}' + I')X_G \tag{2.6}$$

Algorithm 1 Neighborhood Information

```
 1: procedure $N(i, j)$
 2:     $V^1_{i,j} = \{i, j\}$
 3:     friends= $\Gamma(i) \cup \Gamma(j)$
 4:     $V^1_{i,j} = V^1_{i,j} \cup$ friends
 5:     for $x = 1, 2$ do                                    ▷ x represents node i or j
 6:         for $y = 1, 2, \ldots, n$ do          ▷ n is the total number of nodes in $V^1_{i,j}$
 7:             if node i and node j are friends then:
 8:                 $\tilde{A}^{'}[x][y] = 1$
 9:             if y=1 or y=2 then:
10:                 $I^{'}[x][y] = 1$
11:                 $D^{'}[x][y] = |\Gamma(i)|\, or\, |\Gamma(j)|$
12:     for $y = 1, 2, \ldots, n$ do
13:         $X_G[y]$=UserFeatures
14:     return $D^{'-1}(\tilde{A}^{'} + I^{'})X_G$
```

where $D^{'-1}$ represents the inverse of first two rows of degree matrix, $I^{'}$ represents the first two rows of the identity matrix and $X_G \subseteq R^{n \times c}$ is the selected user profile of users of the neighborhood. From another angle, the neighborhood information could also be interpreted as the average of users and their neighbors' user profile features. Algorithm 1 shows the process of getting graph neighborhood information from the second combination model.

Content Embedding Similarity

Fei et al. [22] demonstrated that repost probability increased with an increase in the similarity between the post content and the user interests. User interests could be inferred from the user's historical posts. We define the post content user interest similarity as:

$$S_{p,u} = \frac{C_p I_u}{\| C_p \| \| I_u \|} \tag{2.7}$$

Here, C_p represents the content embedding of post and I_u represents the content embedding of user interests, i.e., content embedding of user's historical posts.

Peng et al. [20] and Suh et al. [21] have shown that content propagation across links occurs more frequently between the users sharing common interests. User interests could be inferred from their posts. We define the user and user interest similarity as:

$$S_{u1,u2} = \frac{I_{u1} I_{u2}}{\| I_{u1} \| \| I_{u2} \|} \tag{2.8}$$

Here, I_{u1} represents the content embedding of source user interests, i.e., content embedding of historical posts from source user $u1$ and I_{u2} represents the content embedding of target user interests, i.e., content embedding of historical posts from target user $u2$.

Content embedding is to map the social media posts of a user to dense vectors in a low dimensional embedding space to extract the important content and linguistic style expressed in the post. There are two widely used methods: Latent Dirichlet Allocation(LDA) topic model and the more recent neural network based methods.

Latent Dirichlet Allocation (LDA)

LDA is a common method for topic modeling. It's a generative model that represents document as a distribution of topics, and represents topic as a distribution of words. In social media, LDA finds topics posts belong to, based on the words in the posts. We could either train all the posts from each user as a single document and get the topic vector for each user. Or we could train each post separately and get all the topic vectors. Then all the per-post topic vectors are averaged to get the final topic vector for each user. LDA has the advantage of generating interpretable embeddings.

Neural Network Based

Word2Vec is a popular neural network based method to learn the embedding of content. Word2Vec uses two-layer neural network and contains two models: CBOW(Continuous bag of words) and Skip-gram model. CBOW predicts the target word from the surrounding words. Skip-gram predicts the surrounding words from the target word. After we get the embedding of the word from the trained model, we could use the average of the word embedding to represent the user embedding for each user.

Word2Vec embedding is context independent, which means each word has only one embedding vector. In reverse, Bidirectional Encoder Representations from Transformers(BERT) embedding is context dependent, which means for the same word, it could generate two different embedding vectors when the two words have different semantic meanings. BERT contains a stack of encoder transformers. Like Word2Vec, after we get the embedding of the word from the trained model, we could use the average of the word embedding to represent the user embedding.

Here we implement a multi-lingual sentence embedding model [24] using knowledge distillation to extract the content embedding. In the multi-lingual sentence embedding model, the sentences with similar meanings are close in vector space. The multi-lingual model uses an English Sentence Bert Embedding (SBERT) model [25] as a teacher model and uses XLM-RoBERTa (XLM-R) [26] as a multilingual student model. BERT generates the word embedding, and the multi-lingual sentence embedding model directly generates the sentence embedding. For experiments, we extract sentence embedding of the latest 50 posts of each user and get the average of the vector representations as the content embedding of user interests.

2.2.2 Prediction Models

To predict the repost probability, here we implement different machine learning methods, e.g., Logistic Regression, K-nearest Neighbors, Gaussian Naive Bayes, Deep Neural Network, Random Forest, XGBoosting and Stacking to make the prediction.

Logistic Regression

Logistic regression is a statistical model that can be used for binary classification. The posterior probability of class C can be represented by a sigmoid function acting on a linear function of the feature vector X. Here we denote the information diffusion probability given an observation X_i as:

$$p(y_i|X_i) = \sigma(\omega^T X_i) = \frac{1}{1 + exp^{-\omega^T X_i}} \tag{2.9}$$

Cross entropy error is applied to determine the parameters ω of the model. For a data set $\{X_i, y_i\}$, where $y_i \in \{0, 1\}$ and $i = 1, ..., N$. The cross entropy error can be written as:

$$E(\omega) = -\sum_{i=1}^{N} \{y_i \ln \sigma(\omega^T X_i) + (1 - y_i) \ln(1 - \sigma(\omega^T X_i))\} \tag{2.10}$$

The gradient of the error function with respect to ω is obtained:

$$\bigtriangledown E(\omega) = \sum_{n=1}^{N} (\sigma(\omega^T X_i) - y_i) X_n \tag{2.11}$$

Then we apply the technique of stochastic gradient descent to update the parameter ω iteratively until it converges. The stochastic gradient descent process is shown below:

$$\omega^{\tau+1} = \omega^{\tau} - \eta \bigtriangledown E(\omega) \tag{2.12}$$

K-nearest Neighbors

K-nearest Neighbors Classification is a type of instance-based learning. It doesn't generate a generalized learning model, but instead it stores instances of the training data. Then the classification of each unlabeled data is computed by the votes from the nearest neighbors of each data. The data is assigned the class which has the most representatives within the nearest neighbors. Here we set k as 5, which means we select 5 nearest neighbors. Then we assign the most voted class of the nearest neighbors to the data.

Gaussian Naive Bayes

Naive Bayes is a classification algorithm for binary and multi-class classification. It's called Naive Bayes because it has a strong assumption that attributes don't interact. Gaussian Naive Bayes assumes the data follows Gaussian distribution (normal distribution). Then the likelihood of each feature is assumed to be:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp(-\frac{(x_i - u_y)^2}{2\sigma_y^2}) \tag{2.13}$$

The parameters σ_y and u_y are standard deviation and the mean. Considering the naive conditional independence assumption, then the information diffusion probability given an observation X_i ($X_i = (x_{i,1}, x_{i,2}, ...x_{i,n})$) is:

$$P(y|x_1, x_2, ..., x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, x_2, ..., x_n)} \tag{2.14}$$

Deep Neural Network

Deep neural network is an artificial neural network with multiple layers [27, 28]. It could approximate complex mathematical function of either linear or nonlinear relationship from input to output. Specially, with the development of the optimization technology in recent years, the performance of deep neural network largely improves. Here we build a deep neural network shown in Figure 2.3 consisting of one input layer, three hidden layers and one output layer to predict the repost probability.

- Input Layer: The number of nodes in the input layer is determined by the number of features we define in the last subsection.
- Hidden Layer: The dimensionality (the number of nodes) of the hidden layer determines the complexity of the network. The more nodes we put the better we will be able to fit. However, higher dimensionality comes with computation costs and overfitting risks as well. We put 64 nodes in the first hidden layer, 32 nodes in the second layer and 16 nodes in the third layer.

The activation function of the hidden layer is ReLU function.

- Output Layer: The output layer contains one node giving the estimation of the repost probability. The activation function of the output layer is a sigmoid function making the output in the range between 0 and 1.

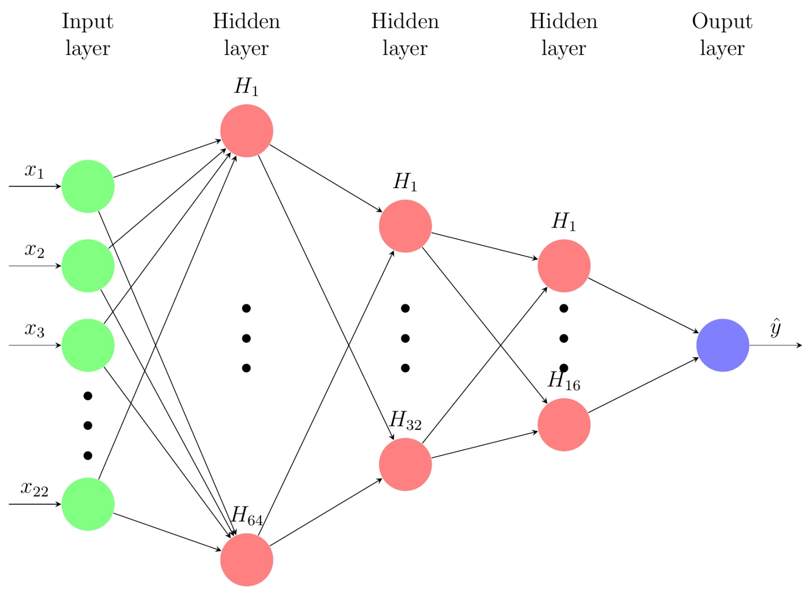


Figure 2.3: Deep Neural Network

The neural network makes prediction using forward propagation, which is a bunch of matrix multiplications and the application of the activation functions we defined above. The repost probability $\hat{y}$ can be represented as below:

$$\hat{y} = Sigmoid(ReLU(ReLU(ReLU(XW^0)W^1)W^2)W^3)$$

where X represents the input features defined in the subsection of feature extraction, W^0,W^1,W^2 and W^3 are parameters of the neural network, which are learned from our training data. To estimate the repost probability, we implement the binary cross-entropy loss function to train the model:

$$L = -\sum_i (y_i log(\hat{y}_i) + (1-y_i) log(1-\hat{y}_i)) \tag{2.15}$$

where y_i is the labeled repost action of observation i which is a binary variable and $\hat{y}_i$ is the predicted repost probability of observation i.

Random Forest

Random forest is an ensemble learning method based on decision trees for either classification or regression tasks. Random forest builds a multitude of decision trees parallelly at training time and outputs the class that is the mode for classification task or average prediction for regression task. It uses bootstrap samples, random selection of split variables and random thresholds to build each individual tree to reduce the variance.

XGBoosting

Gradient boosting decision tree is an ensemble learning method based on decision trees for either classification or regression tasks. Gradient boosting decision tree builds a multitude of decision trees sequentially by reducing the loss of last tree. XGBoosting is an implementation of gradient boosting decision tree algorithm. XGBoosting is developed with both deep consideration in terms of systems optimization and principles in machine learning. It has gained much attention and popularity recently for its fast speed and good performance.

Stacking Model

Stacking is an ensemble learning method that learns how to best combine the prediction from multiple well-performing machine learning models. The architecture of a stacking model includes both base models and meta model. We train base models using the training data and make predictions.

The meta model is trained on the predictions made by base models combined with the inputs to the base models. We adopt the k-fold cross-validation. Then the out-of-fold predictions are used as the basis for the training dataset for the meta-model. Then we retrain the base model on the entire training dataset.

2.3 Experimental Evaluation

We evaluate our proposed repost prediction model using the real world data set: Sina Weibo. Weibo is a large scale publicly available dataset released by Zhang et al. [29].

2.3.1 Datasets

Sina Weibo (http://www.weibo.com) is a Chinese microblogging website allowing users to follow other users and retweet the messages from the followed users similar to Twitter. Specifically, for experimental purpose, we randomly sample 2000 messages. The total number of involved reposts is 290250 by 228250 different users. The 290250 reposts are taken as positive observations for model training and testing. We also randomly sample the same amount of negative observations based on the absence of response from source user's followers for each post to train the model. The total negative observations we collect is 248770 because some of the posts don't have enough negative observations. In total, we collect 539020 observations for our experiment.

2.3.2 Experimental Set-up

User profile includes both poster and reposter features. Table 2.1 lists the selected user profile features of sina weibo dataset.

Messages propagate from users to users on Sina Weibo Dataset. We believe the neighborhood information of the pair is a significant indicator of the repost prediction. We obtain the

Table 2.1: User Profile Features

	User Profile Features
Post Features	Followers of poster
	Followees of poster
	Gender
	Total messages of poster
	Number of the reciprocal relationships
	Created time
Repost Features	Followers of reposter
	Followees of reposter
	Gender
	Total messages of reposter
	Number of the reciprocal relationships
	Created time

neighborhood information from the combination of neighbors' user profile including number of followers, number of followees, total number of messages and total number of the reciprocal relationships.

Regarding to the feature of similarity, we get the content embedding of both the post and user interests through using the multi-lingual sentence embedding model. Then we calculate the cosine similarity of content embedding as the similarity features.

For all kinds of features including user profile, neighborhood information and content similarity, we standardize all the features before training to give them equal importance. The normalization is shown below:

$$\hat{X} = \frac{X - \bar{X}}{\sigma(X)} \tag{2.16}$$

where X represents the feature of a sample, $\bar{X}$ represents the average value of the feature, $\sigma(X)$ is the standard deviation of the feature.

For training the deep neural network, we split the dataset into training set, validation set and testing set with 80% of training set and validation set, 20% of testing set. The validation set is

50000 for both datasets. ADAM optimization algorithm is implemented to update the parameters and get the function. We train the network in 250 epochs with the batch size of 50. In practice, we make use of Keras for a CPU-based implementation of the neural network. We used 6 i7 cores for training.

2.4 Results

In our experiments, we analyze the computational performance when we introduce the neighborhood information features. Besides, we also compare the performance and computational time of each prediction model as well. Lastly, we analyze the importance of each input feature by implementing tree-based ensemble prediction methods.

2.4.1 Combination Models Comparison

As we assume the neighborhood information collecting from the combination of user profile of neighbors is a good indicator of user's repost decision, thus we integrate neighborhood information into repost prediction. Then we come up with two different combination models to extract the neighborhood information. Table 2.2 shows the performance of different prediction models without neighborhood information and with neighborhood information using different combination models.

Table 2.2: Combination Models Comparison

Combination Model	LR	KNN	GB	NN	RF	XGB	Stacking
NA	68.44%	74.30%	64.67%	76.51%	82.41%	83.88%	84.56%
$(A^{'}+I^{'})X$	68.62%	74.74%	61.04%	76.72%	83.79%	85.17%	86.06%
$D^{'-1}(A^{'}+I^{'})X$	68.65%	73.97%	65.21%	77.04%	83.98%	85.40%	86.33%

We could conclude that models including the neighborhood information have much better performance especially for random forest, XGBoosting and Stacking models. In addition, the second combination model of $D'^{-1}(A' + I')X$ has better performance for all the models except K-nearest neighbor. Therefore, we take the neighborhood information getting from the second combination model as an input feature for future experiments.

2.4.2 Prediction Models Comparison

In this subsection, we compare the prediction performance of different prediction models, e.g., Logistic Regression, K-nearest Neighbor, Gaussian Bayesian, Deep Neural Network, Random Forest, XGBoosting and Stacking. The prediction results are shown in Table 2.3.

Table 2.3: Prediction Models Comparison

Model	Accuracy	Precision	Recall	F1	Time(s)
LR	68.65%	77.00%	68.60%	72.56%	1.08
KNN	73.97%	73.74%	76.95%	75.31%	925.51
GB	65.21%	71.19%	66.54%	68.78%	0.77
NN	77.04%	77.24%	79.58%	78.06%	672.66
RF	83.98%	84.98%	85.23%	85.10%	152.73
XGB	85.40%	86.08%	86.70%	86.39%	103.27
Stacking	86.33%	87.30%	87.30%	87.30%	1391.81

Overall, the results of different prediction models indicate that the tree-based ensemble models provide better prediction performance than the other models in terms of Accuracy, Precision, Recall and F1-score. XGBoosting provides slightly better prediction performance than the Random Forest Method with much less time. The Stacking method provides the best performance among all the prediction modes. However, it's more time consuming because it trains all the models as the base models.

2.4.3 Performance of Models using Different Features

In Figure 2.4, we demonstrate the importance of each input feature by implementing tree-based ensemble prediction methods, i.e., Random Forest and XGBoosting. Tree-based methods measure the importance of each feature by collecting how on average each feature decreases the impurity. The importance is calculated for each tree by the amount that each feature split point decreases the impurity. The feature importance are the average of all of the decision trees within the model.

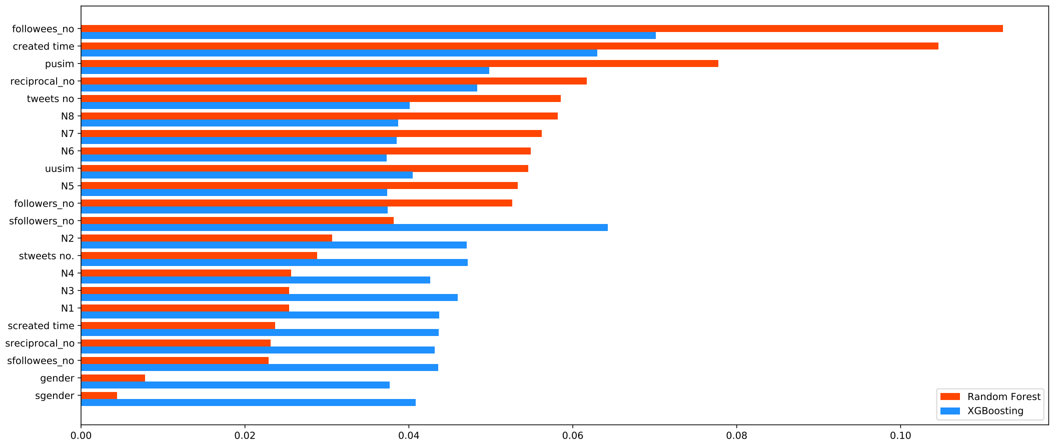


Figure 2.4: Feature Importance

We could conclude that followees number, created time (how long the user has been in social media) and post user similarity are the most important features in repost prediction for both Random Forest and XGBoosting. However, source followers number seems to be an inconsistent element for the tree-based learning models, i.e., XGBoosting learning method thinks it's a more important feature than Random Forest learning method. Regarding to the neighborhood information, target user neighborhood information features (i.e., N5, N6,N7 and N8) are more important than source user neighborhood information features (i.e., N1,N2,N3 and N4) for Random Forest learning model. In contrast, source user neighborhood information features are more important than target user neighborhood information features for XGBoosting learning model.

CHAPTER 3: MODEL-BASED LEARNING OF INFORMATION DIFFUSION IN SOCIAL MEDIA NETWORKS[1]

3.1 Introduction

The rapid development of the Internet and its mobile computing technologies in the past several decades contributes to form online virtual communities in social media networks. People share news, their ideas, interests and information in social media networks. Understanding how the information propagates is essential for successful applications of viral marketing [1] and cyber security [2] in social media networks. To this end, researchers have defined different problems such as the influence maximization problem [7] and contamination minimization problem [30]. Influence maximization problem involves finding a limited number of nodes which have the largest influence. Similarly, contamination minimization problem involves blocking a limited number of nodes which suppress the propagation of the rumors. However, all of the parameters in the diffusion models are assumed to be known. In reality, these parameters are not readily available from the massive historical datasets of social media networks. To fill this gap, this paper proposes several models to learn these influence parameters from historical cascades.

In the literature, researchers have tried to learn the information diffusion processes through different diffusion models. There are two most prevalent diffusion models - Independent Cascade Model (IC) [7] and General Threshold Model (GT) [13]. In the Independent Cascade Model, each active parent node i has a single chance to activate the child node j with a diffusion probability $p_{i,j}$. Saito, Nakano, and Kimura [31] studied the problem of learning influence probabilities based on the IC propagation model. They estimated the diffusion probabilities by expectation-maximization algorithm where the likelihood was maximized by iteratively updating the parameters. Besides,

[1]Qiang, Z., Pasiliao, E. L., & Zheng, Q. P. (2019). Model-based learning of information diffusion in social media networks. Applied Network Science, 4(1), 1-16.

there are some variants of Independent Cascade model based research, considering continuous time delays of infection as well [32, 6]. In contrast to IC models, General Threshold Model assumes that an inactive user j in social media networks is activated by all of its active neighbors when the total influence is larger than the target user's threshold. Linear Threshold Model is a special case of General Threshold Model and the total influence is the sum of the influence weights of activated neighbors. Goyal, Bonchi, and Lakshmanan [33] adopted the General Threshold diffusion Model and proposed three probabilistic models, which were Bernoulli distribution, Jaccard Index and Partical Credits to predict the diffusion influence weights.

In this project, we propose two different mixed-integer programming models to estimate the diffusion influence weights between pairs of users underlying the threshold model. We learn the diffusion influence weights through mining past diffusion cascades of posts. Given the historical information cascading data containing the initial states x_i^0 and final states x_i^T of all the nodes which are users in social media networks, we are trying to learn the diffusion influence weights $w_{i,j}$ between pairs of users in the Threshold Models. Two different learning models are built under the assumption. The first model is based on the Linear Threshold Model, where a user will be activated when the sum of its neighbors' influence weight is larger than the threshold of the user. The second model considers multi-step influence from the multi-hop neighborhood of the user. In the real dataset, the status of nodes at each time step could be collected. However, the time required for each propagation varies. When we know the status of users at two different time steps, the propagation steps could be more than one. Therefore, it's necessary to consider multi-step influence to better fit the real data. We summarize the contributions of this project as follows,

- We propose two mixed-integer programming models to learn diffusion influence between pairs of users. One is based on the popular propagation model, i.e., Linear Threshold Model, and another one is based on the idea of random walk considering multi-step influence.

- For large-scale network, we come up with approximate methods for the two learning models using artificial neural networks with no hidden layer.

- Experiments for assessing the validity of these learning models and approximate neural network methods are performed on both synthetic data and real data.

3.2 Learning Models of Information Diffusion Influence

We model a social media network as a directed network $G = (V, E)$, where $V = \{v_1, \dots, v_n\}$ is the set of users of the network and $E \subseteq V \times V$ is the set of edges representing the friend relationships between users. We observe multiple cascades spreading over it. For any information cascade propagating over the social media network, we observe the status x_i^t of each user i $(i \in V)$ at time $t = 0$ and $t = T$. The status of users is either active in reposting messages or inactive in reposting. Through mining the historical cascades, we could learn the diffusion influence weights $w_{i,j}$ representing the influence exerted by user i to user j in the social media network. Our model aims at learning the diffusion influence weights $w_{i,j}$ from user i to user j $(i, j \in V)$ based on transitions of activation status of all users, i.e., x_i^0 and x_i^T for all $i \in V$. To learn the diffusion influence weights, we can use a minimization model with the weights as decision variables and the objective function as the mean squared error between the predicted final status and the actual final status of users. Here we formulate two different mixed-integer programming learning models to make the prediction where the activation status is modeled by a binary variable.

3.2.1 Linear Threshold Learning Model

The first formulation of Linear Threshold Learning Model (LT) we come up with is underlying the Linear Threshold Propagation Model. In the propagation model of Linear Threshold Model [34, 7], initially, each node $j \in V$ independently selects a threshold θ_j uniformly at random in the range of $[0, 1]$. If the total influence weights of the arcs from its active in-neighbors $N_A^{in}(j)$ is at least θ_j, i.e., $\sum_{i \in N_A^{in}(j)} \omega_{i,j} \geq \theta_j$, then user j is activated. The diffusion influence weights are normalized in the Linear Threshold Model.

We formulate a mixed integer programming model shown below to learn the information diffusion influence between each pair of user i and user j based on the Linear Threshold Model. The objective is to minimize the mean squared error between actual final status of users and predicted final status of users. Here, the mean squared error is a quadratic function. It's hard for solver to handle quadratic function. We linearize the objective function by replacing mean squared error with mean absolute value of the error. The mean absolute value of the error equals to the mean squared error because the status of users is a binary variable. The constraints (3.1) and (3.2) represent the mean absolute error. The constraints (3.3) and (3.4) represent when the total influence from its active neighbors is larger than the threshold of target user, the target user will be activated. Otherwiese, the target user will remain inactive. The constraints (3.5) represent the sum of the diffusion influence from neighbors to user j is at most 1. Constraints (3.6) show when two users are connected, the value of diffusion influence is between 0 and 1. However, when they are not connected, the diffusion influence weights $w_{i,j}$ will be 0, which is shown in the constraints (3.7).

$$\min_{\omega,\mathbf{x},\mathbf{z}} \quad \frac{1}{NK}\sum_{k=1}^{K}\sum_{j=1}^{N} z_j^k \tag{OBJ}$$

$$s.t. \quad z_j^k \geq x_j^{k,T} - \hat{x}_j^{k,T} \qquad \forall k \in K, j \in J \tag{3.1}$$

$$z_j^k \geq \hat{x}_j^{k,T} - x_j^{k,T} \qquad \forall k \in K, j \in J \tag{3.2}$$

$$\sum_{i=1, i\neq j}^{N} \omega_{i,j}\hat{x}_i^{k,0} - \theta_j \geq \varepsilon - \theta_j\left(1 - x_j^{k,T}\right) \qquad \forall (k,j) \in \Xi \tag{3.3}$$

$$\sum_{i=1, i\neq j}^{N} \omega_{i,j}\hat{x}_i^{k,0} - \theta_j \leq (1-\theta_j)x_j^{k,T} \qquad \forall (k,j) \in \Xi \tag{3.4}$$

$$\sum_{i=1}^{N} \omega_{i,j} \leq 1 \qquad \forall j \in J \tag{3.5}$$

$$\varepsilon \leq \omega_{i,j} \leq 1 \qquad \forall (i,j) \in E, i \neq j \tag{3.6}$$

$$\omega_{i,j} = 0 \qquad (i,j) \in \bar{E} \tag{3.7}$$

$$x_i^{k,T}, z_j^k \in \{0,1\} \qquad \forall i,j \in J, k \in K \tag{3.8}$$

Where index k represents different observation, index i,j represent different users, $\omega_{i,j}$ represents the diffusion weight from user i to user j, θ_j represents the threshold of user j, $\hat{x}_i^{k,0}$ and $\hat{x}_j^{k,T}$ are the initial status of user i and final status of user j of observation k which are known already and $x_j^{k,T}$ represents the predicted final status of user j of observation k. Ξ is the set of (j,k), where the initial status of user j of observation k denoted as $\hat{x}_j^{k,0}$ should be 0. E represents the arcs of the social network.

3.2.2 Random Walk Learning Model

The second formulation is based on the idea of random walk. In comparison with the first formulation (LT) which considers the diffusion influence from one-step neighborhood only, Random Walk Learning Model (RW) introduces the diffusion influence from multi-step neighborhood. In reality, when we know the initial and final status of each user, we can't tell how many steps of information propagation occur. The target node j could be activated by either one step neighborhood or even multiple step neighborhood. Therefore, we come up with a learning model considering multi-step diffusion influence between pairs of users based on the idea of random walk. Here we consider the diffusion influence from two-hop neighborhood. The total diffusion influence matrix Y^k for observation k could be represented as:

$$Y^k = \left(A^1 \circ W_1 + A^2 \circ W_2\right) \hat{X}^{k,0} \tag{3.9}$$

Where $\circ$ represents the Hadamard product, A is the adjacency matrix of the social network, the (i,j) entry of A, denotes $a_{i,j,1}$, is the number of paths of length 1 starting at i and ending at j. The (i,j) entry of A^2, denotes $a_{i,j,2}$, is the number of paths of length 2 starting at i and ending at j. W_1

is the average diffusion influence weight matrix for one step diffusion, the entry $\omega_{i,j,1}$ denotes the diffusion influence weight from user i to user j for each one step path. W_2 is the average diffusion weight matrix for two step diffusion, the entry $\omega_{i,j,2}$ denotes the average diffusion influence weight from user i to user j for each two step path. $\hat{X}^{k,0}$ is a vector containing all the users' initial status from observation k, the i entry of $\hat{X}^{k,0}$, denotes $\hat{x}_i^{k,0}$, is the initial status of user i from observation k. The learning formulation based on Random Walk is shown below:

$$\begin{aligned}
\min_{\boldsymbol{\omega},\mathbf{x},\mathbf{z}} \quad & \frac{1}{NK}\sum_{k=1}^{K}\sum_{j=1}^{N} z_j^k && && \text{(OBJ)} \\
s.t. \quad & z_j^k \geq x_j^{k,T} - \hat{x}_j^{k,T} && \forall k \in K, j \in J && (3.10) \\
& z_j^k \geq \hat{x}_j^{k,T} - x_j^{k,T} && \forall k \in K, j \in J && (3.11) \\
& \sum_{i=1}^{N}\sum_{\ell=1}^{L} a_{i,j,\ell}\omega_{i,j,\ell}\hat{x}_i^{k,0} - \theta_j \geq \varepsilon - \theta_j\left(1 - x_j^{k,T}\right) && \forall (k,j) \in \Xi && (3.12) \\
& \sum_{i=1}^{N}\sum_{\ell=1}^{L} a_{i,j,\ell}\omega_{i,j,\ell}\hat{x}_i^{k,0} - \theta_j \leq \left(\sum_{i=1}^{N}\sum_{\ell=1}^{L} a_{i,j,\ell} - \theta_j\right)x_j^{k,T} && \forall (k,j) \in \Xi && (3.13) \\
& 0 \leq \omega_{i,j,l} \leq 1 && \forall i \in I, j \in J, l \in L && (3.14) \\
& x_i^{k,T}, z_j^k \in \{0,1\} && \forall i,j \in J, k \in K && (3.15)
\end{aligned}$$

The constraints (3.12) represent when the total influence from two step neighborhood is larger than the threshold of target user, the target user will be activated. Constraints (3.13) represent when the total influence from two step neighborhood is smaller than the threshold of user, the user will not be activated. Constraints (3.14) show that the average influence weights for different step of path are in the range of 0 and 1.

3.3 Approximate Approaches using Artificial Neural Networks

In the previous section, we formulate two different mixed integer programming learning models to learn the information diffusion influence weights. We could use solver Gurobi to solve the models and get a global optimal estimation of diffusion weights between pairs of users with the assumption. However, sometimes it doesn't seem to allow for an efficient solution for large network. Here we come up with approximate approaches of Linear Threshold Learning Model and Random Walk Learning Model using artificial neural networks.

We are using the simplest kind of neural network called a single-layer perceptron network consisting of a single layer of output nodes [35]. Then the inputs are fed directly to the outputs via a series of weights. To train the neural network and get the estimation of weights, the loss function we use is the mean squared error for both of the models. The optimization algorithm we use for training the neural network is gradient descent. Gradient descent starts with the initial values of parameters and iteratively moves toward the set of values minimizing the loss function through taking steps proportional to the negative of the gradient [36]. To compute the gradients, backpropagation algorithm is introduced [37]. However, the gradient descent with backpropagation is not guaranteed to find the global optimal of the loss function.

3.3.1 Artificial Neural Network for Linear Threshold Learning Model

We build the Linear Threshold Neural Network (LTNN) shown in Figure 3.1 to approximate the Linear Threshold Learning Model. The input $x_1^0, x_2^0, ..., x_n^0$ are the status of n users in social media networks at time $t = 0$. The output layer outputs $x_1^T, x_2^T, ..., x_n^T$, which are the status of n users in social media networks at time $t = T$.

There are several constraints to consider for the Linear Threshold Learning Model. Firstly, if the user is active in the beginning ($\hat{x}_j^0 = 1$), then they will stay active ($\hat{x}_j^T = 1$). To satisfy this constraint, we set the (i, j) entry of the neural network weights matrix W', denoted as $w_{i,j}^{'}$, as

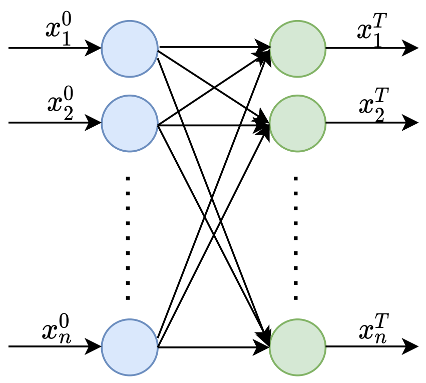


Figure 3.1: Neural Network for Linear Threshold Model

1 when i equals j. In this case, when user is active at the beginning, the total influence it gets at time $t = T$ will definitely be larger than its threshold and the user will stay active. Besides, the influence weights should be suppressed to 0 when the pair of users has no connection. Here, we use the Hadamard product of adjacency matrix A of the network and neural network weights matrix $W^{'}$ to suppress the influence weights matrix W. Thirdly, in the Linear Threshold Model, the diffusion influence weights should be normalized as well. All of the constraints above could be satisfied by designing the structure of neural network. However, the Linear Threshold Model has the activation function of step function. The gradients of step function are vanishing and the weights will not update at all [38]. The neural network is using gradient to update the estimation. Therefore, step function is not applicable to neural network. Instead, we approximate the step function using sigmoid function. Then the output $X^T = \{x_1^T, x_2^T, ..., x_n^T\}$ of neural network could be represented as:

$$X^T = Sigmoid\left(X^0\left((A+I)\circ W^{'}\right) - \theta\right)$$

Where θ is the threshold vector of users and I is the identity matrix. As mentioned before, the influence weights are normalized as well, thus the diffusion influence weight $\omega_{i,j}(\omega_{i,j} \in W)$ between user i and user j could be represented as:

$$\omega_{i,j} = \frac{a_{ij}\omega'_{ij}}{\sum_{i=1}^{n} a_{ij}\omega'_{ij}}$$

Where a_{ij} is the (i,j) entry of adjacency matrix A and ω'_{ij} is the (i,j) entry of artificial neural network weights matrix W'.

3.3.2 *Artificial Neural Network for Random Walk Learning Model*

The structure of neural network called Random Walk Neural Network (RWNN) built for Random Walk Learning Model is shown in Figure 3.2. Here we consider the diffusion influence could be from 2-step neighborhood. Therefore, the input is the two times combination of the initial status of n nodes, denoted as $x_1^0, \ldots, x_n^0, x_1^0, \ldots, x_n^0$. The output layer outputs $x_1^T, x_2^T, \ldots, x_n^T$, which are the status of users in the social media at time $t = T$.

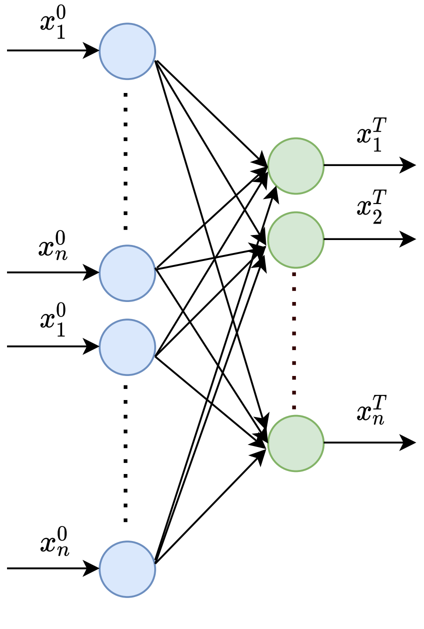


Figure 3.2: Neural Network for Random Walk Learning Model

To satisfy the constraints in Random Walk Learning Model, we make the following modifications to the artificial neural network. Considering the diffusion influence from two step neighborhood, the total influence in the Random Walk Model could be transferred as:

$$\left(A^1 \circ W_1 + A^2 \circ W_2\right) X^0 = \begin{bmatrix} X^0 & X^0 \end{bmatrix} \left(\begin{bmatrix} A^1 \\ A^2 \end{bmatrix} \circ \begin{bmatrix} W_1 \\ W_2 \end{bmatrix} \right)$$

Besides, the user will remain active when they are active in the beginning. Thus we add the identity matrix I to adjacency matrix A to satisfy the constraint. Then the output vector $X^T = x_1^T, x_2^T, \ldots, x_n^T$ of neural network could be represented as:

$$X^T = Sigmoid\left(\begin{bmatrix} X^0 & X^0 \end{bmatrix} \left(\begin{bmatrix} A^1 + I \\ A^2 \end{bmatrix} \circ \begin{bmatrix} W_1^{'} \\ W_2^{'} \end{bmatrix} \right) - \theta \right)$$

Where θ is the threshold of users, A is the adjacency matrix, A^2 is the square of adjacency matrix. $W_1^{'}$ and $W_2^{'}$ are weights matrix of neural network. Thus the diffusion influence weights matrix W could be represented as:

$$W = \begin{bmatrix} W_1^{'} - I \\ W_2^{'} \end{bmatrix}$$

3.4 Experimental Evaluation

In this section we conduct the experimental evaluation of our proposed models which are Linear Threshold Model, Random Walk Model, Linear Threshold Neural Network Model and Random Walk Neural Network Model to learn the information diffusion weights. We analyze the performance of proposed models on both synthetic data and real data.

3.4.1 Experiments on Synthetic Data

Firstly, we generate synthetic data to evaluate our models. The generation of synthetic data includes both network generation and cascade generation.

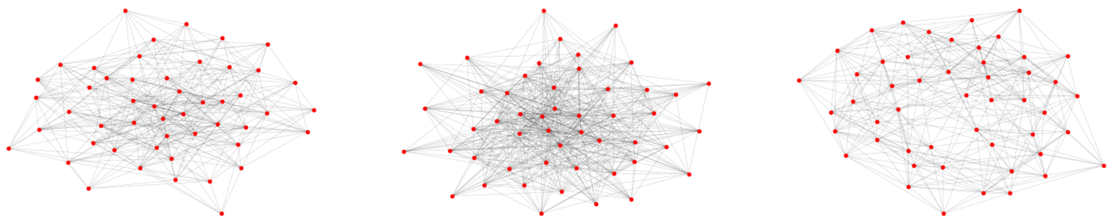
Network generation: In order to understand how the underlying network topology affects the performance of our learning models, we generate different well-known generative networks.

- **Erdos-Renyi Graph:** In the graph theory, Erdos-Renyi model generates random graph. Here we use the $G(n, p)$ model in which a graph is constructed by connecting nodes randomly by edges with probability p. For experimental purpose, we generate the network shown in Figure 3.3a containing 50 nodes and generate edges between each pair of nodes in the network with probability of 0.3.

- **Barabasi-Albert Graph:** Social media networks are thought to be scale-free. Barabasi-Albert graph generates random scale-free networks using a preferential attachment mechanism. Here we generate a graph of 50 nodes shown in Figure 3.3b which is grown by attaching new nodes each with 15 edges that are preferentially attached to existing nodes with high degree.

- **Watts-Strogatz Graph:** Watts-Strogatz model generates random graph with small-world properties, which are short average path lengths and high clustering. Here we generate a graph of 50 nodes shown in Figure 3.3c, and each of them connects to 15 nearest neighbors. In addition, the probability of rewiring each edge is 0.5.

Cascade generation: We learn the diffusion influence weights from mining historical diffusion cascades. Here, we generate the cascades for the synthetic data. We randomly select a set of seed nodes accounting for 35 % of total nodes . Then the seed nodes could propagate the information to other nodes. Here we assume the active nodes will remain active and the inactive nodes will have the chances to be activated only when they have connections to active nodes within 2 step



(a) Erdos-Renyi (b) Barabasi-Albert (c) Watts-Strogat

Figure 3.3: Generative Network Topology

neighborhood. The activation probability p for target node is defined as:

$$p = a + \frac{b}{N} * AN \tag{4}$$

Where N represents the total number of nodes and AN represents the active direct neighbors. We set a as 0.5 and b as 0.5 for the following experiments. We experiment with 200 cascades to compare the performance of different learning models.

Implementation Details and Evaluation: In order to learn the diffusion influence weights under different proposed learning models, we set the threshold of 0.5 for each user in the social media network. In order to provide an unbiased evaluation of a final model on the training dataset, we split the dataset into training set and testing set with 80% of training set and 20% of testing set. To validate the performance of the mixed integer programming learning models and the approximate neural network learning models, we evaluate the performance of accuracy. Accuracy is the ratio of correct predictions. Regarding to solve the problem using mixed-integer programming, we propose the decomposition method which is tractable to compute and to optimize for the original learning models. For each node, we apply the optimization model to get the diffusion weights separately because the diffusion influence from neighbors to each node is independent from each other. For the neural network learning models, we train in 10000 steps, with batch size of 10 and learning rate of 0.001.

Experimental Results: For different network types, we generate the same number of cascades using the same cascade generation strategy to compare the performance of different models. Table 3.1 shows the performance of synthetic data of 50 nodes.

Table 3.1: Performance of Synthetic Data

Network Type	Model	Nodes	Edges	Cascades	Train Accuracy	Test Accuracy
Erdos-Renyi	LT	50	335	200	76.4%	65.6%
	RW	50	335	200	98.9%	68.8%
	LTNN	50	335	200	72.0%	69.6%
	RWNN	50	335	200	76.6%	70.7%
Barabasi-Albert	LT	50	525	200	77.6%	63.1%
	RW	50	525	200	99.5%	68.3%
	LTNN	50	525	200	75.8%	71.8%
	RWNN	50	525	200	79.1%	73.1%
Watts-Strogatz	LT	50	350	200	76.6%	62.9%
	RW	50	350	200	98.6%	67.5%
	LTNN	50	350	200	72.6%	70.8%
	RWNN	50	350	200	76.6%	68.9%

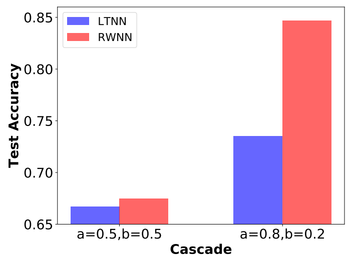
The performance of different learning models are consistent in different network topologies. For the same number of observations(cascades), approximate neural network learning methods even outperform the mixed integer programming learning methods. The mixed integer programming methods of Linear Threshold Model and Random Walk Model have severe overfitting problems. In addition, the computational time for mixed integer programming models is much longer than neural network based models.

Therefore, for larger scale networks we conduct the experimental evaluation just using approximate neural network learning models. The performance of different sizes of networks which are 100 nodes and 1000 nodes are shown in Table 3.2. The testing accuracy is consistent for different network types.

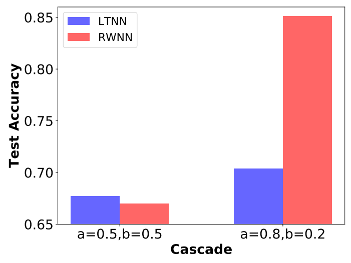
Table 3.2: Performance of Large-scale Synthetic Data

Network Type	Model	Nodes	Edges	Cascades	Training Accuracy	Testing Accuracy
Erdos-Renyi	LTNN	100	270	200	70.5%	66.7%
	RWNN	100	270	200	73.9%	67.5%
	LTNN	1000	24980	200	74.6%	67.6%
	RWNN	1000	24980	200	68.5%	68.0%
Barabasi-Albert	LTNN	100	291	200	69.6%	67.7%
	RWNN	100	291	200	75.1%	67.0%
	LTNN	1000	29100	200	74.5%	68.2%
	RWNN	1000	29100	200	68.7%	68.4%
Watts-Strogatz	LTNN	100	300	200	69.9%	66.9%
	RWNN	100	300	200	74.1%	67.0%
	LTNN	1000	3000	200	69.8%	67.5%
	RWNN	1000	3000	200	73.8%	67.9%

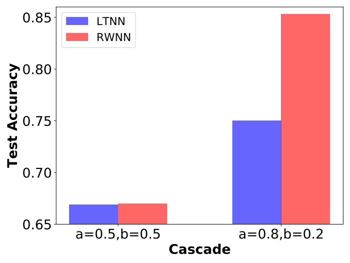
The performance of different models could be affected by the way of generating cascades. We test the performance of different generation methods by changing a and b parameters in Equation 4 using the same network topology. The performance of different network topologies are shown in Figure 3.4a, Figure 3.4b and Figure 3.4c. When a is larger which means getting more influence from two step neighborhood, Random Walk Neural Network Model outperforms Linear Threshold Neural Network Model.



(a) Erdos-Renyi



(b) Barabasi-Albert



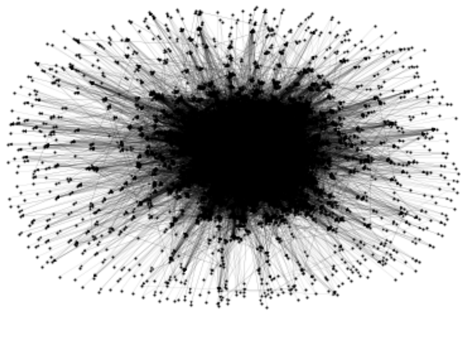
(c) Watts-Strogat

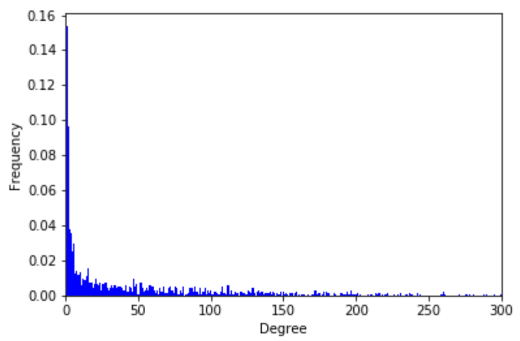
Figure 3.4: Performance of Different Network Topologies

3.4.2 Experiments on Real Data

We collect the real data from Lerman Digg 2009 dataset [39]. The Digg 2009 dataset contains data about stories promoted to Digg's front page over a period of a month in 2009. The dataset contains the time stamp of all of the users' vote for the stories. Both of the network topology and cascade information can be obtained from the dataset.

Dataset Preprocessing: Considering the computation limitation of mixed integer programming models, we extract a small size subgraph containing 1203 nodes shown in Figure 3.5a to compare the performance between mixed integer programming learning models and approximate neural network learning models. Our extracted subgraph contains 1203 users shown in Figure 3.5a. In Figure 3.5b, we could see the degree distribution of all the users follows power law distribution. The extracted network keeps the social network property of scale-free. In addition, to prove the effectiveness of the proposed models in large scale social network, we extract a network with 42259 nodes and conduct the experiments using neural network based models. In order to get the cascade information, we choose the most voted 600 stories. Then we collect all the users voting status at time T_0 and T_t for each story.


(a) Network Topology



(b) Degree Distribution

Figure 3.5: Extracted Subgraph

Implementation Details and Evaluation: To investigate the performance on real data, we set the threshold of 0.5 for each user in each of the proposed learning models.

Experimental Results of Real Data: We proceed with experiments on preprocessed real data of 1203 nodes to see the performance of different learning models. We collect the historical cascades of 650 stories and get 406 efficient cascades. We treat 300 observations as training samples and the rest of observations as testing samples. For the neural network models, we train in 10000 steps with batch size of 10 and learning rate of 0.001. The performance of the dataset is shown in Table 3.3. First, we evaluate the performance of different learning models based on test accuracy. Random walk learning model has the best performance in this preprocessed real dataset. It doesn't have any overfitting issue here which is likely due to the fact of sparseness of the network. In general, the random walk based learning models outperform the linear threshold based learning models in this preprocessed dataset. However, when it comes to the speed of computation, the approximate methods using neural network, i.e., LTNN and RWNN are much faster.

Table 3.3: Performance of Real Data

Model	Nodes	Edges	Cascades	Training Accuracy	Testing Accuracy	Time(s)
LT	1203	29166	406	82.8%	86.3%	1773
RW	1203	29166	406	88.3%	92.3%	32406
LTNN	1203	29166	406	85.5%	84.1%	137
RWNN	1203	29166	406	86.4%	85.6%	330

Then, we examine the performance of LTNN model and RWNN model using cascade information from different time intervals. We collect the data from different lengths of time intervals. One has the time interval of $4T_0$ and another has the time interval of $9T_0$. Figure 3.6 shows the performance comparison of two approximate models during different time intervals. For shorter time period, the LTNN model has significant improvement over RWNN model. In converse, RWNN model outperforms LTNN model in longer time period. This happens because in short time period, only one step propagation could happen. In longer time period, two-steps or multiple steps propagation could occur where random walk based model would better describe the information diffusion process.

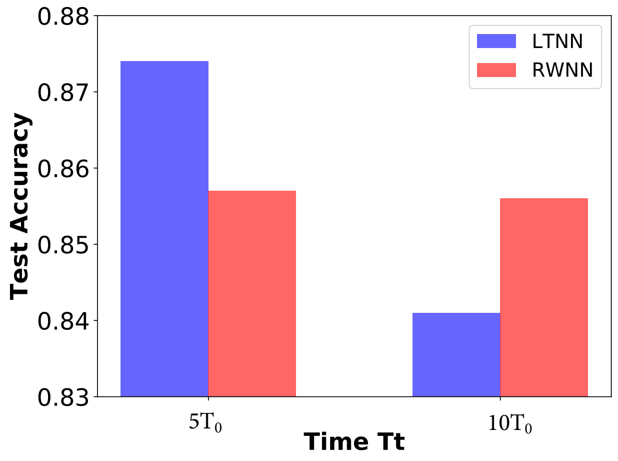


Figure 3.6: Performance of Different Time Intervals

For the large scale network containing 42259 users, we collect the historical cascades of 600 stories and get 566 efficient casacades for experimental purpose. We treat 440 cascades as training samples and the rest as testing samples. We train in 500 steps with batch size of 10 and learning rate of 0.001. The cascade has the time interval of $9T_0$. The experimental result of the large dataset is shown in Table 3.4. Random Walk Neural Network Model requires 380GB memory and it takes about 10 minutes for each step. It's time consuming and it gets stuck during the optimization process because of too many nodes. In comparison, Linear Threshold Neural Network Model converges in 500 steps and has very good performance.

Table 3.4: Performance of Large-scale Real Data

Model	Nodes	Edges	Cascades	Training Accuracy	Testing Accuracy	Time(s)
LTNN	42259	1559768	566	91.2%	90.4%	22782

CHAPTER 4: TARGET SET SELECTION IN SOCIAL NETWORKS WITH TIERD INFLUENCE AND ACTIVATION THRESHOLDS[1]

4.1 Introduction

Nowadays, the use of social media networks has become a necessary daily activity for people to interact with family and friends, access news, information and make decisions. Besides being a handy means for keeping in touch with friends and family, social media is more of a platform spreading the tremendous influence. Users of the social media tend to follow and adopt their friends or followers' thoughts and behaviors. Thus businesses pay incentives to social influencers using the social media platforms such as Youtube and Instagram to introduce their products and stimulate demands. Politicians begin to communicate their policy views and humanize themselves through their social media accounts. Politician campaigns boost their investment in social media ads to get votes as well. During the time of social isolation due to the COVID-19, people rely on social media for health and safety updates, entertainment and virtually interaction with family and friends. Social media has great impact on businesses, politics, disease control and others.

Here we focus on investigating the problem of target set selection. Formally, we define a social media network as a directed graph $G = (V, E)$, where users are defined as all nodes V and their friendships are defined as all edges E. Users are active when they repost the messages.

Target set selection problem refers to find a subset $S \subset V$, $|S| <= k$, then the nodes activated by S will be at least l. The target set selection problem has two optimization variants: minimum target set and maximum target set. The minimum target set refers to find the smallest set $S \subset V$, so that the nodes activated by S will be at least l. The maximum target set refers to find a set $S \subset V$ of size k, which has the largest activated nodes compared to any other subset $S^{'} \subset V$ of size k. The

[1]Qiang, Z., Pasiliao, E. L., & Zheng, Q. P. (2021, November). Target Set Selection in Social Networks with Influence and Activation Thresholds. In International Conference on Computational Data and Social Networks (pp. 371-380). Springer, Cham.

maximum target set problem is also called social influence maximization problem in social media [7]. The target set selection problem can be applied in the areas of viral marketing[1] and cyber security [2].

In the target set selection model, the seed nodes spread the influence until the diffusion process stops. The goal is to activate all the nodes or as many nodes as possible. In the target set selection model in social media, activated users normally refer to the users who repost the message. However, in reality, influenced users sometimes will repost the messages. But in most cases, even they are convinced or influenced by the message, users will not repost the message for certain reasons. In this case, influence not only refers to activation(repost) but also refers to the belief or like in the messages. Thus we build the tiered influence and activation thresholds target set selection models to describe the situation. Here we introduce two thresholds, one is activation threshold (ϕ) and another one is influence threshold (θ). Users will be influenced first before be activated, thus we define $\theta <= \phi$. The goal of the model is to influence all the nodes or as many nodes as possible.

Our models are time-indexed integer program models, which can be divided into two parts, the first part is the information propagation. There are two widely used propagation models, namely Independent Cascade Model(IC) [7] and Linear Threshold Model(LT) [13]. Independent Cascade Model assumes every node has a single chance to activate its neighbors. In Linear Threshold Model, each node will be influenced by each neighbor according to a weight. When the total weights from its neighbors is larger than a threshold ϕ, then the node will be activated. In this paper, we propose all the mathematical models based on the Linear Threshold Propagation Model. Here we set the weights as 1 for all the nodes. Thus an inactive node will become active if at least ϕ of its neighbors are active in the previous step. The second part is the influence dominating part, which means the users should be either active or influenced through having at least θ of activated neighbors at time T.

Similar to the variants of target set selection problem, we define the Dual Threshold Minimum Influential Seeds problem and its variants, Dual Threshold Maximum Influence with Budget and its variants. Minimum Influential Seeds problem refers to select the least nodes to influence all the nodes at time T. Maximum Influence with Budget refers to select the seed nodes within the budget to influence as many nodes as possible at time T.

To investigate the tiered influence and activation thresholds target set selection problem, the research of target set selection problem offers us some good insights. Kempe, Kleinberg, and Tardos [7] study the maximum active set (under the name target set selection) and show that it is NP-hard. They also provide a greedy algorithm within provable approximation guarantees based on the submodularity property of the objective function. Chen [8] study the minimum target set selection problem and show the problem is hard to approximate within a polylogarithmic factor. Besides, he comes up a polynomial-time algorithm to find an optimal solution when the underlying graph is a tree. Ackerman, Ben-Zwi, and Wolfovitz [40] propose a combinatorial model for the minimum target set selection and prove the combinatorial bounds for the perfect target set selection problem. Shakarian, Eyre, and Paulo [41] present a time-indexed formulation to find the minimum seeds for the target set selection and come up with a scalable heuristic based on the idea of shell decomposition. Spencer and Howarth [42] consider the problem of how to target individuals with subsidy in the network in order to promote pro-environmental behavior. It is also a target set selection problem and they use a time-indexed integer program formulation with as many time periods as the number of nodes in the network to tackle the problem. Günneç, Raghavan, and Zhang [43] study the variation of the target set selection problem called least cost target set selection on social networks, and they propose greedy algorithm and dynamic programming algorithm to solve the problem for the tree structure network. Raghavan and Zhang [44] develop and implement a branch-and-cut approach to solve the weighted target set selection problem on arbitrary graphs. To our knowledge, the previous research involving target set selection focuses on the single threshold (activation threshold) target set selection. In this project, we propose several practical tiered influence and activation thresholds target set selection mathematical models. The detailed contributions of the project is summarized below:

- We propose tiered influence and activation thresholds time-indexed integer program models to select the seed nodes which can be easily applied in practice. Both minimum target set selection and maximum target set selection are explored here. In addition, we propose models exploiting different requirements of seed nodes selection including the maximum activation, early activation and dynamic threshold.

- We conduct the sensitivity analysis to determine how the objective function will change if specified parameters, i.e, influence and activation thresholds deviate from their anticipated values.

- We compare between different mathematical models Minimum Influential Seeds model and its variants, Maximum Influence with Budget Model and its variants to draw conclusions regarding their differences and connections.

- We solve these novel models exactly by Gurobi for small datasets. Besides, we compare between some efficient computational algorithms, i.e., Graph Partition, Nodes Selection, Greedy Algorithm, Recursive Threshold Back Algorithm and Two-stage Approach in Time.

4.2 Tiered Influence and Activation Thresholds Target Set Selection Models

In this section, we propose eight time-indexed integer program models for identifying the seed nodes for the influence and activation thresholds target set selection problem. Both minimum influence and activation thresholds target set selection and maximum influence and activation thresholds target set selection with budget are considered here. In addition, the proposed models explore different requirements of seed nodes selection, which are maximum activation, early activation and dynamic threshold.

4.2.1 Minimum Influential Seeds

Firstly, we introduce a time-indexed integer program to find the minimum influential seeds for the influence and activation thresholds target set selection problem. An artificial time index t taking values from 0 to T is introduced to model the order in which nodes become active. The messages could propagate at varying distances through different forms of social media. Cha, Mislove, and Gummadi [45] observe that even for popular photos, only 19 percent of fans are more than 2 hops away from uploaders on Flickr.com. Ye and Wu [46] find that, on Twitter, 37.1 percent message flows spread more than 3 hops away from the originators. Thus here we set T as 0,1,2,3, which means we only consider the cascades less than or equal to three time steps. The formulation uses a binary variable $x_{i,t}$ to represent the status of node i at time t, which is 1 if node i reposts the message at time t and 0 otherwise. Here θ represents the influence threshold and ϕ represents the activation threshold. Nodes should always be influenced(like the message) first before be activated(repost the message), so we set $\theta <= \phi$. $N(i)$ represents the neighborhood of node i. The Minimum Influential Seeds Model is as follows:

$$\min_{\mathbf{x}} \quad \sum_{i \in V} x_{i,0} \tag{4.1}$$

$$s.t. \quad \theta x_{i,T} + \frac{1}{|N(i)|}\left\{\sum_{j \in N(i)} x_{j,T}\right\} - \theta \geq 0 \qquad \forall i \in V \tag{4.2}$$

$$\frac{1}{|N(i)|}\left\{\sum_{j \in N(i)} x_{j,t}\right\} + \phi x_{i,t} \geq \phi x_{i,t+1} \qquad \forall i \in V, t < T \tag{4.3}$$

$$\frac{1}{|N(i)|}\left\{\sum_{j \in N(i)} x_{j,t}\right\} + \phi x_{i,t} \leq \phi - \varepsilon + (1+\varepsilon) x_{i,t+1} \qquad \forall i \in V, t < T \tag{4.4}$$

$$\begin{aligned} \theta &\in (0,1] && \leftarrow \text{influence threshold} \\ \phi &\in (0,1] && \leftarrow \text{activation threshold} \\ \theta &\leq \phi && \leftarrow \text{node is influenced before activated} \\ N(i) &= \{\, j : (i,j) \in E \,\} && \leftarrow \text{Neighborhood, adjacent nodes} \end{aligned}$$

The objective function (4.1) aims to minimize the seed nodes activated at time 0. Constraints (4.2) are influential constraints, making sure that all the nodes should be either active or be influenced by at least θ of active neighbors at time T. Constraints (4.3) refer that a node i will stay inactive at time $t+1$ when it is not activated at time t. Constraints (4.4) restrict that a node will stay active if it is originally active, which means when $x_{i,t} = 1$, $x_{i,t+1}$ should be 1 as well. In addition, the constraints make sure that a node will become active at time $t+1$ when it is activated at time t, which means when $x_{i,t} = 0$, the influence from its neighbors is larger or equal than ϕ, then $x_{i,t+1} = 1$. Here we introduce two ε, the first ε restricts that node i should be active at time $t+1$ even if the influence from its neighbors is ϕ. The second ε confirms that when $x_{i,t} = 1$ and all the neighbors of i are active, the node i being active at time $t+1$ still holds.

When we set different values for different parameters θ and ϕ, the influential constraints (4.2) have different insights. When $\theta <= \frac{1}{|N(i)|}$, influential constraints (4.2) are identical to find the dominating set at time T, which means each node is either active or has at least one active neighbor. When $\theta = 1$, influential constraints (4.2) are identical to find the vertex cover set at time T, which means each edge has at least one active node.

4.2.2 *Minimum Influential Seeds with Maximum Activation*

Sometimes we not only want everyone to be convinced by the message but also want the message to be reposted by as many users. Then we propose a time-indexed integer program to find the minimum influential seeds that simultaneously maximize activation at time T. The objective function is different from 4.2.1, instead of only minimizing the number of seed nodes, we try to maximize the nodes reposting the message at time T as well. In objective function (4.5), we put more weights on minimum influential seeds and less weights on maximum activation at time T. Weight of $\frac{1}{|V|}$ is assigned to maximum activation, where $|V|$ represents the total number of nodes. The Minimum Influential Seeds with Maximum Activation model is as follows:

$$\min_{\mathbf{x}} \quad \sum_{i \in V} x_{i,0} - \frac{1}{|V|} \sum_{i \in V} x_{i,T} \tag{4.5}$$

$$s.t. \quad (2),(3),(4) \tag{4.6}$$

4.2.3 Minimum Influential Seeds with Early Activation

Instead of activating maximum nodes at time T, activating nodes as early as possible is necessary as well for some cases. Specifically for the health and disaster applications, users should be informed about the information and repost the message as early as possible to avoid the risks. Thus we come up with an integer program here to find the minimum influential seeds with early activation. The objective function (4.7) aims to minimize the seed nodes activated at time 0 and simultaneously maximize the number of activated nodes at each time step t. Here we could get rid of the constraints (4.4), because our objective function confirms that the node will prefer to stay or become active at each time step. Constraints (4.3) restrict that a node i will stay inactive at time $t+1$ when it is not activated at time t. The Minimum Influential Seeds with Early Activation model is as follows:

$$\min_{\mathbf{x}} \quad \sum_{i \in V} x_{i,0} - \frac{1}{T \cdot |V|} \sum_{t=1}^{T} \sum_{i \in V} x_{i,t} \tag{4.7}$$

$$s.t. \quad (2),(3) \tag{4.8}$$

4.2.4 Minimum Influential Seeds with Dynamic Activation Threshold

In the above models, we assume the activation threshold stays constant over the time. However, as time passes by, it will become more difficult to convince someone of another belief, i.e., the longer someone has a negative opinion, the more difficult it is to change it to positive. Thus we come up with the model of Minimum Influential Seeds with Dynamic Activation Threshold, where the activation threshold increases linearly over time. In the model, the activation threshold is $\frac{1}{T}\phi$ at time 0 and the activation threshold is ϕ at time $T-1$. The Minimum Influential Seeds with Dynamic Activation Threshold Model is as follows:

$$\min_{\mathbf{x}} \quad \sum_{i\in V} x_{i,0} \tag{4.9}$$

$$s.t. \quad \theta x_{i,T} + \frac{1}{|N(i)|}\left\{\sum_{j\in N(i)} x_{j,T}\right\} - \theta \geq 0 \qquad \forall i \in V \tag{4.10}$$

$$\frac{1}{|N(i)|}\left\{\sum_{j\in N(i)} x_{j,t}\right\} + \frac{t+1}{T}\phi x_{i,t} \geq \frac{t+1}{T}\phi x_{i,t+1} \qquad \forall i \in V, t < T \tag{4.11}$$

$$\frac{1}{|N(i)|}\left\{\sum_{j\in N(i)} x_{j,t}\right\} + \frac{t+1}{T}\phi x_{i,t} \leq \frac{t+1}{T}\phi - \varepsilon + (1+\varepsilon) x_{i,t+1} \qquad \forall i \in V, t < T \tag{4.12}$$

4.2.5 Maximum Influence with Budget

The problems discussed above are finding the minimum influential seeds under different circumstances, making sure all the nodes are influenced at the end. However, in real cases, when the social network is large scale, it is not practical to select the seed nodes to influence all the nodes. It is common that you have a budget for selecting the seed nodes. Here we come up with a model of Maximum Influence with Budget, which aims to find the seed nodes under budget σ that maximize the influenced nodes at the end. The formulation uses a binary variable $x_{i,t}$ to represent the status of node i at time t, which is 1 if node i is active at time t and 0 otherwise as well. In addition, it uses a binary variable y_i to represent whether the node i is influenced at time T. The Maximum

Influence with Budget model is as follows:

$$\max_{\mathbf{x}} \quad \sum_{i=1}^{n} y_i \tag{4.13}$$

$$s.t. \quad \sum_{i=1}^{n} x_{i,0} - \sigma \leq 0 \qquad \forall i \in V \tag{4.14}$$

$$\theta x_{i,T} + \frac{1}{|N(i)|}\left\{\sum_{j\in N(i)} x_{j,T}\right\} - \theta \geq y_i - 1 \qquad \forall i \in V \tag{4.15}$$

$$\frac{1}{|N(i)|}\left\{\sum_{j\in N(i)} x_{j,t}\right\} + \phi x_{i,t} \geq \phi x_{i,t+1} \qquad \forall i \in V, t < T \tag{4.16}$$

$$\frac{1}{|N(i)|}\left\{\sum_{j\in N(i)} x_{j,t}\right\} + \phi x_{i,t} \leq \phi - \varepsilon + (1+\varepsilon) x_{i,t+1} \qquad \forall i \in V, t < T \tag{4.17}$$

The objective function (4.13) aims to maximize the nodes influenced at time T. Constraints (4.14) set the budget of the nodes activated initially. Constraints (4.15) make sure that the node will be influenced when it is active already or have at least θ of active neighbors at time period T. Constraints (4.16) and Constraints (4.17) are cascade constraints. Constraints (4.16) refer that a node i will stay inactive at time $t+1$ when it is not activated at time t. Constraints (4.17) restrict that a node will stay active if it is originally active and a node will become active at time $t+1$ when it is activated at time t.

4.2.6 Maximum Influence and Activation with Budget

Then we propose a time-indexed integer program to find the limited influential seeds with maximum influence and activation at time T. The model differs from the Maximum Influence with Budget Model in the objective function, which not only maximizes the influenced nodes but also maximizes the nodes activated at time T. Here we put more weights on influenced nodes with the weight of 1 than activated nodes with the weight of $\frac{1}{|V|}$. The Maximum Influence and Activation with Budget model is as follows:

$$\max_{\mathbf{x}} \quad \sum_{i \in V} y_i + \frac{1}{|V|} \sum_{i \in V} x_{i,T} \tag{4.18}$$

$$s.t. \quad (14), (15), (16), (17) \tag{4.19}$$

4.2.7 Maximum Influence and Early Activation with Budget

Here we propose a time-indexed integer program to find the seed nodes with budget to have maximum influence and early activation as well. The objective function aims to choose the seed nodes with budget to maximize the nodes influenced at time T and maximize the activated nodes at each time step t which will force the nodes to be activated as early as possible. The Maximum Influence and Early Activation with Budget model is as follows:

$$\max_{\mathbf{x}} \quad \sum_{i \in V} y_i + \frac{1}{T \cdot |V|} \sum_{t=1}^{T} \sum_{i \in V} x_{i,t} \tag{4.20}$$

$$s.t. \quad (14), (15), (16) \tag{4.21}$$

4.2.8 Maximum Influence with Dynamic Activation Threshold

Same as in the Subsection 4.2.4, we also propose a maximum influence model with dynamic activation threshold. The dynamic activation threshold increases linearly over time. In the model, the activation threshold is $\frac{1}{T}\phi$ at time 0 and ϕ at time $T-1$. The Maximum Influence with Dynamic Activation Threshold model is as follows:

$$\max_{\mathbf{x}} \quad \sum_{i \in V} y_i \tag{4.22}$$

$$s.t. \quad \sum_{i=1}^{n} x_{i,0} - \sigma \leq 0 \qquad \forall i \in V \tag{4.23}$$

$$\theta x_{i,T} + \frac{1}{|N(i)|}\left\{\sum_{j\in N(i)} x_{j,T}\right\} - \theta \geq y_i - 1 \qquad \forall i \in V \tag{4.24}$$

$$\frac{1}{|N(i)|}\left\{\sum_{j\in N(i)} x_{j,t}\right\} + \frac{t+1}{T}\phi x_{i,t} \geq \frac{t+1}{T}\phi x_{i,t+1} \qquad \forall i \in V, t < T \tag{4.25}$$

$$\frac{1}{|N(i)|}\left\{\sum_{j\in N(i)} x_{j,t}\right\} + \frac{t+1}{T}\phi x_{i,t} \leq \frac{t+1}{T}\phi - \varepsilon + (1+\varepsilon) x_{i,t+1} \qquad \forall i \in V, t < T \tag{4.26}$$

4.3 Computational Algorithms

The time-indexed integer program models proposed in Section 4.2 are computationally intractable unless in very small instances because of the large number of binary variables. However, social media networks are usually in an extremely large scale. Thus we apply multiple computational algorithms to tackle the influence and activation thresholds target set selection models for larger scale networks in this manuscript. More details will be discussed in the rest of this section.

4.3.1 Graph Partition

When the social media network is large-scale, solving the models exactly through Gurobi is very difficult. The most intuitive way is to solve multiple smaller subgraphs instead of one large graph. Here we use techniques from *Modularity and Community Structure* [47] in networks to divide the large graph into several smaller subgraphs. Then we solve the models exactly separately for each subgraph.

4.3.2 Nodes Selection

When we're dealing with influence and activation thresholds target set selection problem for large-scale network, the large number of binary decision variables, which are $|V||T|$ in total, makes the problem difficult to solve. In order to accelerate the computational speed, we could reduce the decision variables through adding some constraints to restrict that some of the nodes are not selected or some of the nodes should be selected. Here we come up with two methods, one is to delete the leaf nodes, and another is to choose the nodes with high degree.

Leaf Nodes Deletion

Leaf nodes in a connected graph may not be seeded because they'll influence or activate at most one neighbor directly. Thus we add the constraints (4.27) to remove the option of activating leaf nodes. In other words, all the leaf nodes will not be seeded using the method.

$$x_{i,0} + 1 \leq |N(i)| \tag{4.27}$$

Degree Centrality Selection

Nodes with high degree have more potential to influence and activate other nodes. Therefore, we assume the high degree nodes must be seeded. Here $\frac{1}{|V|} \ll \rho < 1$ is defined as the criteria for choosing the seed nodes. When the total neighbors $|N(i)|$ of node i is larger than $\rho|V|$, the node i will be seeded. Thus we add the constraints (4.28) to the original models in order to choose the nodes with more than $\rho|V|$ neighbors as seed nodes.

$$x_{i,0} + \rho \geq \frac{|N(i)|}{|V|} \tag{4.28}$$

$$\rho = \varepsilon \quad \text{implies that} \quad N(i) = 1 \quad \text{for a connected graph}$$

$$\rho = 1 - \varepsilon \quad \text{implies that} \quad N(i) = |V| - 1$$

The larger the ρ, the nodes with higher degree will be selected as seed nodes. When ρ is ε, the nodes having neighbors will all be selected. When ρ is $1-\varepsilon$, only the node connecting to all the other nodes will be selected.

4.3.3 Greedy Algorithm

We propose the greedy algorithm for the Minimum Influential Seeds problem and Maximum Influence with Budget problem. The greedy algorithm selects the seed node with the largest number of inactive neighbors in each iteration and adds it to the seed node set S until the stop conditions have been met. Then we update the nodes threshold and active set in each iteration considering the propagation process. Here we come up with two different ways to update the threshold and active set, one is based on the depth-first search(DFS) and another is based on the breadth-first search(BFS).

For the DFS Greedy algorithm, the algorithm starts with empty seed set (S). Then at each iteration we select the seed node with the largest number of inactive neighbors and add it to the seed node set S. Next, we carry out the propagation process from this newly activated seed node and update the threshold and activated time step (T) of an inactive neighbor node by adding the influence from the activated seed node. When the threshold of the node is larger or equal than ϕ, the node is activated. Then we add this node to the active set A. We continue to carry out the same propagation process when the node is activated until certain time steps. DFS starts at the root node and explores as far as possible along each branch before backtracking. The iteration terminates when all the inactive nodes have at least θ of active neighbors for the Minimum Influential Seeds problem and when the budget is used up or all the inactive nodes have at least θ of active neighbors for the Maximum Influence with Budget problem. Algorithm 2, 3 show the DFS Greedy Algorithm for the Minimum Influential Seeds Problem.

Algorithm 2 DFS Greedy Algorithm

```
    Input: Graph $G = (V, E)$, Propagation step $P$
    Output: Seed node set $S$
1: $A \leftarrow \emptyset$, $S \leftarrow \emptyset$
2: $threshold(i) = 0$, $T(j) = 0$
3: while $\exists\ v \in V \setminus A$, $n_A(v) < \theta deg(v)$ do
4:     Pick $u \in V \setminus A$ with the most inactive neighbors
5:     $S = S \cup u$
6:     $A = A \cup u$
7:     $iteration = 0$
8:     Threshold-Update$(u, P, iteration, A)$
9: Return (S)
```

Algorithm 3 Threshold-Update$(u, P, iteration, A)$

```
    Input: Graph $G = (V, E)$, Node $u$,Propagation step $P$
    Output: Active set $A$
 1: $iteration = iteration + 1$
 2: for j in $N(u)$ do
 3:     if j not in A then
 4:         $threshold(j) = threshold(j) + \frac{1}{deg(j)}$
 5:         $T(j) = max(T(j), iteration)$
 6:         if threshold($j$)$>= \phi$ then
 7:             $A = A \cup j$
 8:             if T(j)>=P then
 9:                 Continue
10:             else
11:                 Threshold-Update$(j, P, iteration, A)$
```

For the BFS Greedy Algorithm for the Minimum Influential Seeds Problem shown in Algorithm 4, firstly we choose the seed node with the largest number of inactive neighbors and add it to the seed set S. Then we update the threshold and activation time step (T) of an inactive neighbor node by adding the influence sent from the activated seed node. Different from DFS, BFS starts at the tree root and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level. Here we use a queue Q to store the parent nodes which will spread the influence within propagation step P.

4.3.4 Recursive Threshold Back Algorithm

We also come up with the Recursive Threshold Back Algorithm especially for the Minimum Influential Seeds Problem. We decompose the Minimum Influential Seeds problem and tackle the

Algorithm 4 BFS Greedy Algorithm

```
    Input: Graph $G = (V,E)$, Propagation step $P$
    Output: Seed node set $S$
 1: $A \leftarrow \emptyset$, $S \leftarrow \emptyset$, $Q$ be queue
 2: $threshold(i) = 0$, $T(i) = 0$, $ite(i) = 0$
 3: while $\exists\ v \in V \setminus A$, $n_A(v) < \theta deg(v)$ do
 4:     Pick $u \in V \setminus A$ with the most inactive neighbors
 5:     $Q.enqueue(u)$
 6:     $S = S \cup u$
 7:     $A = A \cup u$
 8:     $iteration = 0$
 9:     while $Q$ is not empty do
10:         $v = Q.dequeue()$
11:         $iteration = ite(v) + 1$
12:         for all $\omega$ in $N(v)$ do
13:             if $\omega$ not in A then
14:                 $threshold(\omega) = threshold(\omega) + \frac{1}{deg(\omega)}$
15:                 $T(\omega) = max(T(\omega), iteration)$
16:                 if threshold($\omega$)>= $\phi$ then
17:                     $A = A \cup \omega$
18:                     if $T(\omega) < P$ then
19:                         $Q.enqueue(\omega)$
20:                         $ite(\omega) = iteration$
21: Return (S)
```

problem backwards with the tool of integer program. It is required that at time T, all the nodes would be either active or have θ of active neighbors. In other words, at time T the active nodes should dominate all the nodes. Here we assume the active nodes at time T as Minimum Positive Dominating Set. The Minimum Positive Dominating Set is the minimum set of nodes $S \subset V$ that all the nodes are either in the active set or have θ of neighbors in the active set. Then in order to minimize the seed nodes selected to activate at the beginning, we assume the number of nodes activated should be minimized at each time step. Thus we get the nodes activated at each time step backwards recursively. The pseudo code of the Recursive Threshold Back Algorithm is shown in Algorithm 5.

We use integer program model to solve both the Minimum Positive Dominating Set (MPDS) prob-

Algorithm 5 Recursive Threshold Back Algorithm

```
1: Find the Minimum Positive Dominating Set MPDS for all the nodes at time $T$
2: $time \leftarrow T-1$
3: for $t \leftarrow time$ to 0 do
4:     Find the status of nodes at time $t$: $x_{i,t} = \mathbf{TB}(x_{i,t+1})$
5: return $x_{i,0}$
```

lem and Threshold Back (TB) problem. The mathematical programming model for finding Minimum Positive Dominating Set (MPDS) is shown below:

$$\textbf{MPDS:} \min_{\mathbf{x}} \quad \sum_{i=1}^{n} x_{i,T} \tag{4.29}$$

$$s.t. \quad \theta x_{i,T} + \frac{1}{|N(i)|}\left\{\sum_{j\in N(i)} x_{j,T}\right\} - \theta \geq 0 \qquad \forall i \in V \tag{4.30}$$

Objective function (4.29) is to minimize the nodes activated at time T. Constraints (4.30) make sure that all nodes should be either active or have at least θ of active neighbors at time T. Then we get the status of users at different time t recursively by solving the following subproblem Threshold Back (TB).

$$\textbf{TB:} \min_{\mathbf{x}} \quad \sum_{i=1}^{n} x_{i,t} \tag{4.31}$$

$$s.t. \quad \frac{1}{|N(i)|}\left\{\sum_{j\in N(i)} x_{j,t}\right\} + \phi x_{i,t} \geq \phi x_{i,t+1} \qquad \forall i \in V, t < T \tag{4.32}$$

$$\frac{1}{|N(i)|}\left\{\sum_{j\in N(i)} x_{j,t}\right\} + \phi x_{i,t} \leq \phi - \varepsilon + (1+\varepsilon) x_{i,t+1} \qquad \forall i \in V, t < T \tag{4.33}$$

$$x_{j,t+1} \geq \hat{x}_{j,t+1} \tag{4.34}$$

Here $t = T-1, T-2, ..., 0$. $\hat{x}_{j,t+1}$ are solutions getting from the last ThresholdBack iteration. The objective function (4.31) is to minimize the nodes activated at each time period, so that the initial seed set will be minimized as well. Constraints (4.32), (4.33) refer that a node j will be activated in time period $t+1$ only if it is active in time period t, or if it has at least ϕ of active neighbors. Otherwise, the node j will remain inactive. Constraints (4.34) show that there could be more nodes active than the solutions getting from the last ThresholdBack iteration, which confirms the feasibility of the problem.

4.3.5 Two-stage Approach in Time

A Two-stage Approach in Time algorithm is proposed for the Minimum Influential Seeds problem. For the algorithm, we decompose the original Minimum Influential Seeds model into two stages. The first stage is to find the minimum positive dominating set by solving the Minimum Positive Dominating Set (MPDS) problem. The mathematical programming model for finding Minimum Positive Dominating Set is the same as the MPDS model presented in Subsection 4.3.4. The second stage is to obtain the minimum influential seeds through solving the following minimum target set selection problem (TS).

$$\textbf{TS:} \min_{\mathbf{x}} \quad \sum_{i=1}^{n} x_{i,0} \tag{4.35}$$

$$s.t. \quad \frac{1}{|N(i)|}\left\{\sum_{j\in N(i)} x_{j,t}\right\} + \phi x_{i,t} \geq \phi x_{i,t+1} \qquad \forall i \in V, t < T \tag{4.36}$$

$$\frac{1}{|N(i)|}\left\{\sum_{j\in N(i)} x_{j,t}\right\} + \phi x_{i,t} \leq \phi - \varepsilon + (1+\varepsilon) x_{i,t+1} \qquad \forall i \in V, t < T \tag{4.37}$$

$$x_{j,T} \geq \hat{x}_{j,T} \tag{4.38}$$

The objective function (4.35) is to minimize the selected seed set. Constraints (4.36), (4.37) refer that a node j will be activated in time period $t+1$ only if it is active in time period t, or if it has at least ϕ of active neighbors. Otherwise, the node j will remain inactive. Constraints (4.38) show that the node will be active if the node is a dominant node in the first stage.

4.4 Experimental Evaluation

We present computational results of the proposed influence and activation thresholds target set selection models and compare the performance of different computational algorithms in various sizes of datasets. The goals of the experiments are listed below:

- Conduct the sensitive analysis of different threshold parameters, i.e., influence threshold (θ) and activation threshold (ϕ) on synthetic datasets.
- Compare between different models to see their differences and connections.
- Test the performance and computational time of exact method and various computational algorithms.

4.4.1 Experimental Setting and Dataset

Our experiments are conducted on a Mac OS Catalina machine equipped with an Intel Core i7 2.6GHz processor, 16GB of RAM and 9.0.0 Gurobi Optimizer.

We consider two classes of datasets in our experiments. (1) Synthetic network of 50 nodes and 80 nodes using the Barabasi-Albert(BA) model [48], which is an algorithm for generating scale-free network with heavy-tailed degree distribution. (2) A subset of real-life social networks: Karate Club [49], Hamster Friendships Network [50], Facebook Network Dataset [51] and LastFM Social Network Dataset [52]. The detailed information of the networks is shown in Table 4.1.

4.4.2 Sensitivity Analysis

The modification of influence and activation thresholds will change the seeds required for minimum dual threshold target selection models and the nodes influenced for maximum dual threshold target selection models. Here we conduct numerous experiments using BA-50 and BA-80 networks to do the sensitive analysis to determine how the objective function will change if specified

Table 4.1: Real Datasets

Network	Nodes	Edges	Description
Karate	34	78	Contain social ties among members of a university karate club
Hamster	1858	12534	Contain friendships between users of hamsterster.com
Facebook1	2888	2981	Contain Facebook user-user friendships
LastFM	7624	27806	Contain mutual follower relationships of LastFM asian users

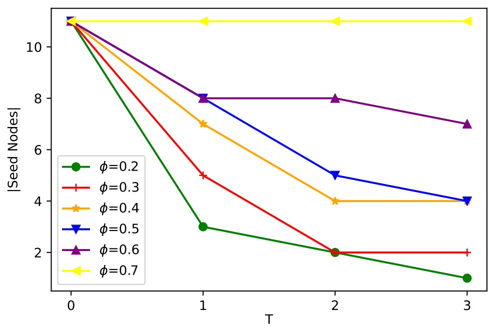
parameters, i.e, influence and activation thresholds, are permutated.

Sensitivity Analysis of Minimum Influential Seeds Model

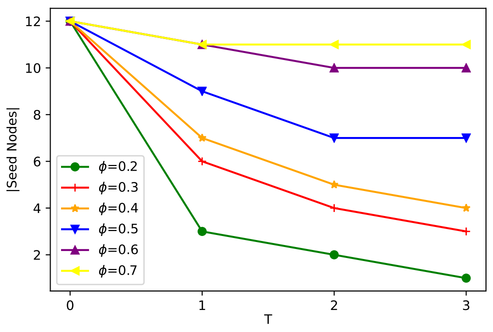
Firstly, we conduct the sensitivity analysis of different parameters θ and ϕ for both BA-50 and BA-80 datasets for the Minimum Influential Seeds Model.

Changes of Activation Threshold(ϕ):

Figure 4.1 shows the solutions of different activation thresholds with the same influence threshold θ of 0.2 for Minimum Influential Seeds Model. We could see that the seed nodes selected would be fewer over time as the activation threshold ϕ decreases. In other words, time contributes more in influence propagation when ϕ is small. It can be explained by that when the activation becomes easier, fewer seed nodes will be required to influence all the nodes over time. Another interesting finding is that when the activation threshold ϕ is large enough, i.e., $\phi >= 0.7$ for BA-50 network, the number of seed nodes selected will be constant over time, which can be explained by the fact that when the activation is quite hard, the seed nodes required will remain constant over time.



(a) BA-50

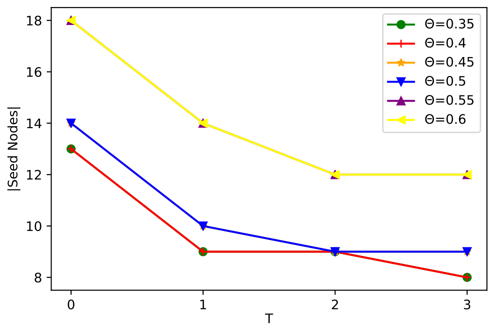


(b) BA-80

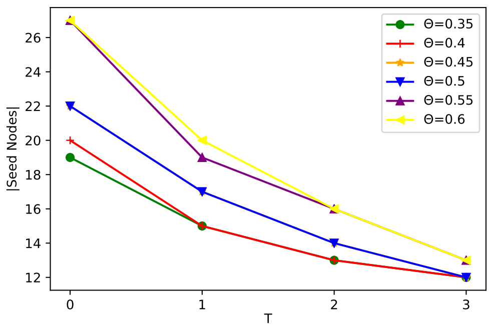
Figure 4.1: Influence Threshold θ=0.2

Changes of Influence Threshold(θ):

Figure 4.2 shows the solutions of different influence thresholds with the same activation threshold ϕ of 0.6 for Minimum Influential Seeds model. We could conclude that when the influence threshold θ increases, the more seed nodes will be selected, which can be explained by that when nodes are more difficult to be influenced, then more seed nodes will be selected at the beginning.



(a) BA-50



(b) BA-80

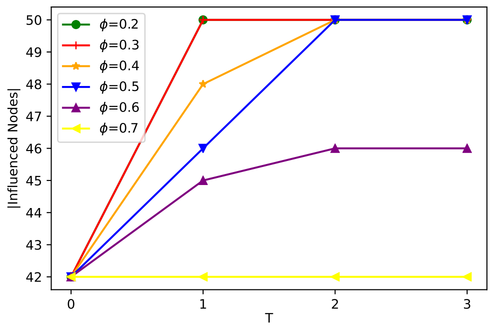
Figure 4.2: Activation Threshold ϕ=0.6

Sensitivity Analysis of Maximum Influence with Budget Model

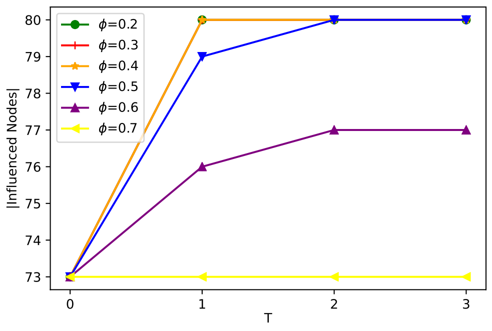
Then we conduct the sensitivity analysis of different threshold parameters θ and ϕ for both BA-50 and BA-80 datasets for the Maximum Influence with Budget Model.

Changes of Activation Threshold(ϕ):

Figure 4.3 shows the maximum influenced nodes with different activation thresholds ϕ and the same influence threshold θ of 0.2 for Maximum Influence with Budget Model. When ϕ is smaller, more nodes will be influenced over time. It can be explained by that when the activation is easy, more nodes will be activated and influenced. In addition, when ϕ is too large which means that the activation is too hard, then the influenced nodes will stay constant.



(a) BA-50

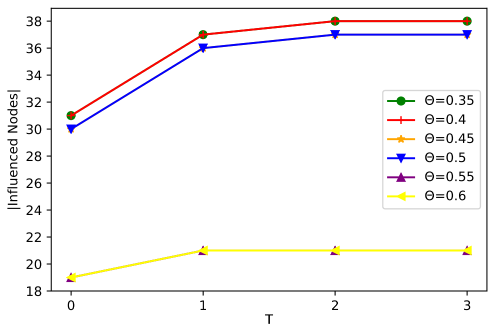


(b) BA-80

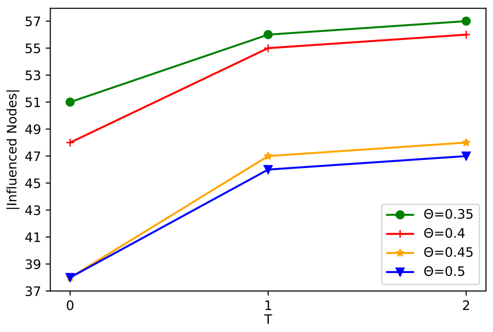
Figure 4.3: Influence Threshold θ=0.2

Changes of Influence Threshold(θ):

Figure 4.4 shows the maximum influenced nodes with different influence thresholds and the same activation threshold ϕ of 0.6. We could see that when θ is larger which means that it is more difficult to be influenced, then the fewer nodes will be influenced.



(a) BA-50



(b) BA-80

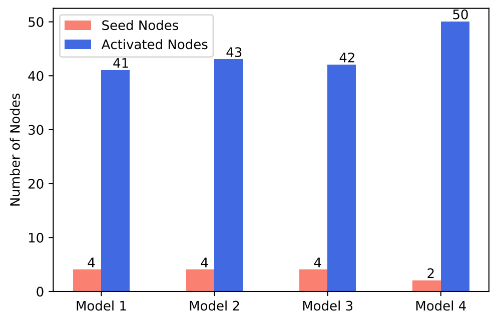
Figure 4.4: Activation Threshold ϕ=0.6

4.4.3 Model Comparison

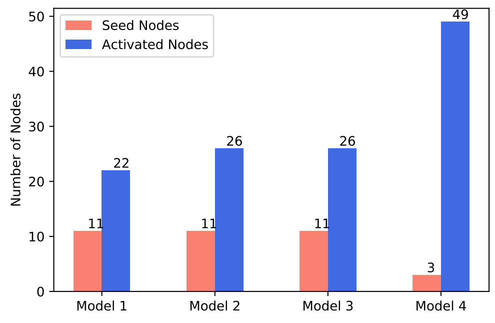
We propose Minimum Influential Seeds Model and its variants, i.e., Minimum Influential Seeds with Maximum Activation, Minimum Influential Seeds with Early Activation and Minimum Influential Seeds with Dynamic Activation Threshold models for the minimum dual threshold target selection problem. For the maximum dual threshold target selection problem, Maximum Influence with Budget model and its variants, i.e., Maximum Influence and Activation with Budget, Maximum Influence and Early Activation with Budget and Maximum Influence with Dynamic Threshold models are proposed. The original model and its variants of maximum activation, early activation and dynamic activation threshold models are different but there also exist some connections between the models. In this section, we test our proposed models and explore the connections through conducting experiments in BA-50 dataset.

Minimum Target Set Selection Models Comparison

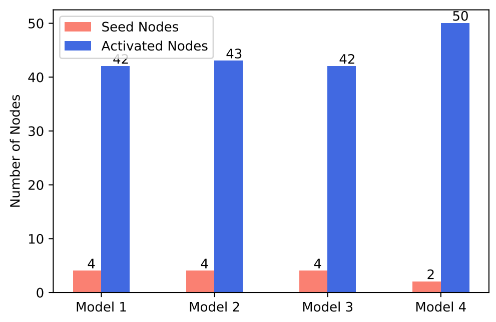
Here we compare the seed nodes required and activated nodes for different variations of dual threshold minimum target set selection models. Figure 4.5 shows the performance of models using different influence and activation thresholds in BA-50 dataset. Here model 1 refers to Minimum Influential Seeds model, model 2 refers to Minimum Influential Seeds with Maximum Activation model, model 3 refers to Minimum Influential Seeds with Early Activation model and model 4 refers to Minimum Influential Seeds with Dynamic Threshold model. We could verify that when the number of seed nodes is the same, Minimum Influential Seeds with Maximum Activation model will have more or at least the same number of activated nodes than Minimum Influential Seeds model and Minimum Influential Seeds with Early Activation model. Besides, Minimum Influential Seeds with Dynamic Threshold Model will require fewer or at least the equal number of seed nodes to influence all the nodes and normally it will activate more nodes compared with the other models.



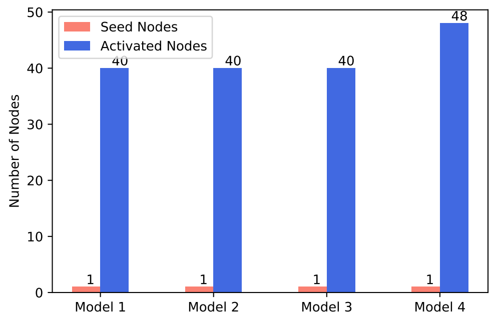
(a) θ=0.02, ϕ=0.5



(b) θ=0.02, ϕ=0.75



(c) θ=0.25, ϕ=0.5

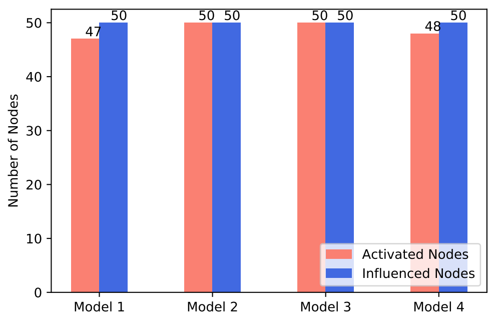


(d) θ=0.2, ϕ=0.25

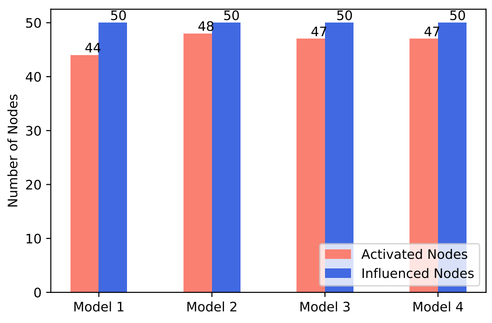
Figure 4.5: Minimum Target Selection Models Comparison of BA-50

Maximum Target Set Selection Models Comparison

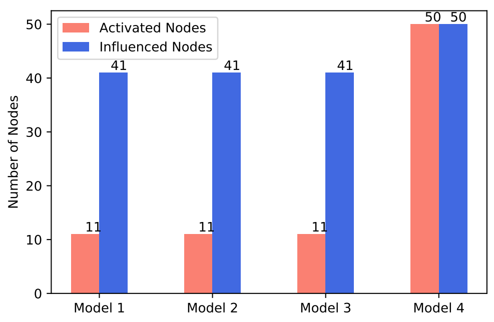
We also compare the activated nodes and influenced nodes for different variations of maximum target set selection models with budget. Figure 4.6 shows the performance of models using different influence and activation thresholds in BA-50 dataset with the budget of 5 seed nodes. Here model 1 represents Maximum Influence with Budget model, model 2 represents Maximum Influence and Activation with Budget model, model 3 represents Maximum Influence and Early Activation with Budget model and model 4 represents Maximum Influence with Dynamic Activation Threshold model. From the figure, we could conclude that Maximum Influence and Activation with Budget model will activate more or at least the same number of nodes in comparison with Maximum Influence with Budget model, Maximum Influence and Early Activation with Budget model. Furthermore, Maximum Influence with Dynamic Threshold model will influence more or at least the same number of nodes as other maximum target selection models. However, it may not always activate more nodes than the other models in all cases, such as the cases shown in Figure 4.6a and Figure 4.6b.



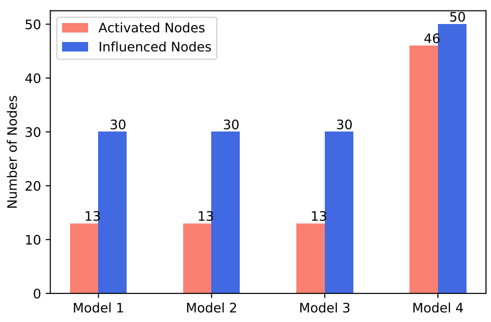
(a) θ=0.2, ϕ=0.25



(b) θ=0.25, ϕ=0.5



(c) θ=0.25, ϕ=0.75



(d) θ=0.5, ϕ=0.75

Figure 4.6: Maximum Target Selection Models Comparison of BA-50

4.4.4 Computational Algorithms Comparison

In this subsection, we assess and draw comparisons between different computational algorithms introduced in the Section 3. We implement these computational algorithms for both Minimum Influential Seeds Model and Maximum Influence with Budget Model.

Computational Algorithms Results Comparison For Minimum Influential Seeds Model

Firstly, we conduct numerous experiments in different datasets to investigate the performance of different computational algorithms. For Minimum Influential Seeds Model, we implement the original mathematical model using solver Gurobi and various computational algorithms, i.e., Graph Partition, Leafnode, Degree Centrality, DFSGreedy, BFSGreedy, Recursive Threshold and Two Stage in Time. We set the time limit of 3600 s for bold methods in the following experiments. For the method of degree centrality, we set the ρ as 0.2.

For the BA-50 network shown in Table 4.2, we could solve the model directly using Gurobi. But the degree centrality method is better which speeds up the computation without sacrificing the performance. For the BA-80 network shown in Table 4.3, we could see it is hard for Gurobi to solve the model directly for the network of 80 nodes. However, the Degree Centrality method could provide a good solution within much shorter time. The Karate network is a network of very small size and the result is shown in Table 4.4. For the Karate Club network, we could solve the model directly. However, the Leaf Node method could accelerate the computation slightly without sacrificing the performance. For larger size network of Hamster Dataset, the Graph Partition, Recursive Threshold, Two Stage in Time methods couldn't generate a solution in one hour, so we don't include here in Table 4.5. For Leaf Node method, the model is not feasible which means we couldn't exclude all the leaf nodes as seed nodes for Minimum Influential Seeds model. We could see from the table the Original Model even performs better than the Degree Centrality method within the time limit of 3600s. In addition, both the DFS Greedy method and BFS Greedy method offer good solutions within much less time compared to the Original Model. Facebook 1 is a dataset of low density. The Facebook 1 network has a large number of nodes with few friends. Thus it is easy for Gurobi to solve it directly. However, the Leaf node method has the shortest computation time for this dataset. The results of LastFM Asia Dataset are shown in Table 4.7, here Leaf Node performs the best compared with Original Model and Degree Centrality methods. For this dataset, BFS Greedy method has better performance than DFS Greedy method.

Table 4.2: BA-50 Network(Minimum Influential Seeds Model)

Method	θ	ϕ	T	Seeded	Activated	Influenced	Obj	Time
Original Model	0.4	0.6	3	8	37	50	8	2.63
Graph Partition	0.4	0.6	3	16	37	50	16	0.12
Leaf Node	0.4	0.6	3	8	37	50	8	1.62
Degree Centrality	0.4	0.6	3	8	37	50	8	0.22
DFS Greedy	0.4	0.6	3	10	43	50	10	0.001
BFS Greedy	0.4	0.6	3	10	43	50	10	0.002
Recursive Threshold	0.4	0.6	3	13	NA	50	13	0.05
Two Stage in Time	0.4	0.6	3	12	50	50	12	0.59

Table 4.3: BA-80 Network(Minimum Influential Seeds Model)

Method	θ	ϕ	T	Seeded	Activated	Influenced	Obj	Time
Original Model	0.4	0.6	3	12	63	80	12	3600.07
Graph Partition	0.4	0.6	3	19	63	80	19	0.37
Leaf Node	0.4	0.6	3	12	63	80	12	3600.11
Degree Centrality	0.4	0.6	3	12	54	80	12	39.13
DFS Greedy	0.4	0.6	3	14	71	80	14	0.004
BFS Greedy	0.4	0.6	3	14	71	80	14	0.004
Recursive Threshold	0.4	0.6	3	20	NA	80	20	0.15
Two Stage in Time	0.4	0.6	3	16	80	80	16	1585.26

In summary, for the small size datasets, we could solve the problem directly using Gurobi. For the network of low density, especially when large portion of the nodes have few neighbors(long-

Table 4.4: Karate Network(Minimum Influential Seeds Model)

Method	θ	ϕ	T	Seeded	Activated	Influenced	Obj	Time
Original Model	0.4	0.6	3	6	18	34	6	0.34
Graph Partition	0.4	0.6	3	7	32	34	7	0.09
Leaf Node	0.4	0.6	3	6	18	34	6	0.32
Degree Centrality	0.4	0.6	3	7	32	34	7	0.02
DFS Greedy	0.4	0.6	3	7	33	34	7	0.001
BFS Greedy	0.4	0.6	3	7	33	34	7	0.001
Recursive Threshold	0.4	0.6	3	9	NA	34	9	0.05
Two Stage in Time	0.4	0.6	3	8	34	34	8	0.12

Table 4.5: Hamster Dataset(Minimum Influential Seeds Model)

Method	θ	ϕ	T	Seeded	Activated	Influenced	Obj	Time
Original Model	0.4	0.6	3	282	1543	1858	282	3600.47
Degree Centrality	0.4	0.6	3	291	1529	1858	291	3600.56
DFS Greedy	0.4	0.6	3	330	1753	1858	330	16.22
BFS Greedy	0.4	0.6	3	327	1766	1858	327	15.05

tailed network), we could consider the Leaf Node method and Degree Centrality method. For the larger size datasets, normally the DFS Greedy and BFS Greedy will have better performance. The Graph Partition, Recursive Threshold and Two Stage in Time algorithms have poor performance and long computational time for the selected social media networks. For the Graph Partition method, it could result from the structure of network which is hard to divide into subgraphs. Furthermore, even it is divided properly, sometimes the size of the subgraph is still hard to solve directly. For the Recursive Threshold and Two Stage in Time methods, the problem lies in the first step of solving the Minimum Positive Dominating Set problem, which is also a complicated NP complete problem and very difficult to be solved by Gurobi directly for large size data sets.

Table 4.6: Facebook 1 Dataset(Minimum Influential Seeds Model)

Method	θ	ϕ	T	Seeded	Activated	Influenced	Obj	Time
Original Model	0.4	0.6	3	10	2888	2888	10	16.66
Graph Partition	0.4	0.6	3	10	2888	2888	10	20.17
Leaf Node	0.4	0.6	3	10	2888	2888	10	0.55
Degree Centrality	0.4	0.6	3	10	2888	2888	10	4.40
DFS Greedy	0.4	0.6	3	10	2888	2888	10	0.75
BFS Greedy	0.4	0.6	3	10	2888	2888	10	1.11
Recursive Threshold	0.4	0.6	3	10	NA	2888	10	0.76
Two Stage in Time	0.4	0.6	3	10	2888	2888	10	7.25

Table 4.7: LastFM Asia Dataset(Minimum Influential Seeds Model)

Method	θ	ϕ	T	Seeded	Activated	Influenced	Obj	Time
Original Model	0.4	0.6	3	2850	6682	7624	2850	3601.25
Leaf Node	0.4	0.6	3	1498	6453	7624	1498	3601.52
Degree Centrality	0.4	0.6	3	3427	7624	7624	3427	3601.42
DFS Greedy	0.4	0.6	3	1697	7276	7624	1697	879.06
BFS Greedy	0.4	0.6	3	1675	7255	7624	1675	868.62

Computational Algorithms Results Comparison For Maximum Influence with Budget Model

We conduct numerous experiments in different datasets to investigate the performance of different computational methods for Maximum Influence with Budget Model as well. For Maximum Influence with Budget Model, we implement the Original Model using solver Gurobi, Leaf Node, Degree Centrality, DFS Greedy and BFS Greedy computational methods correspondingly. We set the time limit of 3600s for bold methods in the following experiments. The budget of seed nodes is 10% of the total nodes. For the method of degree centrality, we set the ρ as 0.2 for BA-50, BA-80 and Facebook1 datasets. For Karate, Hamster and LastFM Asia datasets, ρ is set as 0.3.

For the BA-50 network, the results are shown in Table 4.8. The model could be solved by Gurobi directly. However, Degree Centrality method works faster with the same performance.

From the results of BA-80 network shown in Table 4.9, we could see it is already very hard for the original model to solve the network of 80 nodes. Here Degree Centrality Method could have the same performance with much less time. The result of Karate Network is shown in Table 4.10, solving the original model directly works well for Karate Network dataset. From the results of Hamster Dataset shown in Table 4.11, we could see DFS Greedy and BFS Greedy work much better than the other methods. Facebook 1 is a dataset with several users of high degree. In this case, when these leader users are activated, then all the users will be activated and influenced. From the result shown in Table 4.12, DFS Greedy and BFS Greedy methods are much better because they will select as few nodes as possible to activate and influence all the nodes without using all the budget. From the results of LastFM Asia Dataset shown in Table 4.13, BFS Greedy and DFS Greedy outperform the other methods in both computational time and performance.

Table 4.8: BA-50 Network(Maximum Influence with Budget Model)

Method	θ	ϕ	T	Seeded	Activated	Influenced	Obj	Time
Original Model	0.4	0.6	3	5	18	38	38	9.27
Leaf Node	0.4	0.6	3	5	18	38	38	9.70
Degree Centrality	0.4	0.6	3	5	18	38	38	0.18
DFS Greedy	0.4	0.6	3	5	21	36	36	0.001
BFS Greedy	0.4	0.6	3	5	21	36	36	0.001

Table 4.9: BA-80 Network(Maximum Influence with Budget Model)

Method	θ	ϕ	T	Seeded	Activated	Influenced	Obj	Time
Original Model	0.4	0.6	3	8	21	56	56	3600.01
Leaf Node	0.4	0.6	3	8	21	56	56	3600.02
Degree Centrality	0.4	0.6	3	8	27	56	56	4.05
DFS Greedy	0.4	0.6	3	8	24	53	53	0.002
BFS Greedy	0.4	0.6	3	8	24	53	53	0.002

Table 4.10: Karate Network(Maximum Influence with Budget Model)

Method	θ	ϕ	T	Seeded	Activated	Influenced	Obj	Time
Original Model	0.4	0.6	3	3	9	24	24	1.15
Leaf Node	0.4	0.6	3	3	9	24	24	2.04
Degree Centrality	0.4	0.6	3	3	12	21	21	0.01
DFS Greedy	0.4	0.6	3	3	12	21	21	0.001
BFS Greedy	0.4	0.6	3	3	12	21	21	0.001

Table 4.11: Hamster Dataset(Maximum Influence with Budget Model)

Method	θ	ϕ	T	Seeded	Activated	Influenced	Obj	Time
Original Model	0.4	0.6	3	185	999	1509	1509	3600.11
Leaf Node	0.4	0.6	3	185	959	1476	1476	3600.08
Degree Centrality	0.4	0.6	3	185	893	1419	1419	3600.08
DFS Greedy	0.4	0.6	3	185	1147	1552	1552	11.72
BFS Greedy	0.4	0.6	3	185	1178	1546	1546	10.20

In summary, the advantages of DFS Greedy and BFS Greedy are more obvious compared to other computational algorithms for the Maximum Influence with Budget problem. On one hand, DFS Greedy and BFS Greedy could save the budget when the budget can not be fully used. On the other hand, when the budget could be completely used, the DFS Greedy and BFS Greedy will work much better in both efficiency and computing time in comparison with the other methods for larger size networks. For small size networks, we could solve it directly. Or we could also consider the Leaf Node method and Degree Centrality method to speed up the computation. In this subsection, we assess and draw comparisons between different computational algorithms introduced in the Section 3. We implement these computational algorithms for both Minimum Influential Seeds Model and Maximum Influence with Budget Model.

Table 4.12: Facebook1 Dataset(Maximum Influence with Budget Model)

Method	θ	ϕ	T	Seeded	Activated	Influenced	Obj	Time
Original Model	0.4	0.6	3	288	2888	2888	2888	2.84
Leaf Node	0.4	0.6	3	12	2888	2888	2888	0.12
Degree Centrality	0.4	0.6	3	60	2888	2888	2888	0.92
DFS Greedy	0.4	0.6	3	10	2888	2888	2888	0.78
BFS Greedy	0.4	0.6	3	10	2888	2888	2888	1.13

Table 4.13: LastFM Asia Dataset(Maximum Influence with Budget Model)

Method	θ	ϕ	T	Seeded	Activated	Influenced	Obj	Time
Original Model	0.4	0.6	3	762	1907	3220	3220	3600.21
Leaf Node	0.4	0.6	3	762	2023	3542	3542	3600.16
Degree Centrality	0.4	0.6	3	762	1907	3220	3220	3600.15
DFS Greedy	0.4	0.6	3	762	3923	5673	5673	382.64
BFS Greedy	0.4	0.6	3	762	3951	5697	5697	383.08

CHAPTER 5: INCORPORATING UNCERTAINTY INTO INFLUENCE MAXIMIZATION PROBLEM IN SOCIAL MEDIA NETWORKS

5.1 Introduction

Social media has become an important platform for people to spread the influence and get influenced. In marketing, business use social media as a marketing platform by paying incentives to influencers to advertise or selling products directly through social media. In politics, politicians use social media to influence, attract, and convince people with their ideas and policies. In disaster management, social media helps to provide the up-to-the-minute news information — road closure updates, evacuation routes, designated help areas, shelter locations, and more. Recognizing the influential users is essential for all the applications of social media. The problem of recognizing the influential users in the social media is called influence maximization problem (IM) [1]. IM problem refers to selecting the seed nodes(users) at the beginning that spread the most influence at the end.

Most of the research focusing on solving the influence maximization problem in a deterministic way. In reality, there exist various uncertainties due to the evolutionary dynamics of social networks, changing user behavior and ads price over time. Therefore, we build two-stage stochastic programming models considering the uncertainties in network topology, activation thresholds and activation prices. As far as we know, this is the first attempt to use two-stage stochastic optimization to solve the influence maximization problem considering the uncertainties of network topology and ad price. Although Tanınmış, Aras, and Altınel [53] consider the uncertainty of threshold as well, but they mainly focus on solving the competitive influence maximization where two decision makers are involved.

To investigate the influence maximization problem considering the uncertainty, the research of influence maximization problem offers us some good insights. The influence maximization problem has been solved using discrete optimization approach and has been classified as NP-hard [7]. They also provide a greedy algorithm achieving $(1 - 1/e)$ approximation guarantee based on the submodularity property of the influence function focusing on both Linear Threshold Model and Independent Cascade Model. Cost Effective Lazy Forwarding(CELF) [51] addresses the run time of greedy algorithm, achieved by exploiting submodular features. CELF++ [54] improves the CELF algorithm achieved by avoiding recomputing of the marginal gain with respect to already selected node. Although greedy algorithms provide approximation guarantees for Influence maximization problem, they still suffer from the runtime issue. Therefore, heuristics based on the network structural features are proposed. Heuristics based on centrality ratings like degree, betweenness, closeness, and eigenvector are mostly used to identify influential users based on their location.

In contrast to deterministic influence maximization models, there are also some researches focusing on stochastic influence maximization problem. Song and Dinh [55] focus on the problem of identifying the ideal subset of links whose removal reduces the propagation of disinformation and rumors by relying solely on actual stochastic cascades that occurred in the network. Wu and Kucukyavuz [56] solve the influence maximization problem in a stochastic way which they refer to as two-stage stochastic submodular optimization model. They also propose a delayed constraint generation algorithm to find the optimal solution. Güney [57] develops a binary integer program that approximates the influence maximization problem by monte carlo sampling the cascade process, then they propose SAA and a linear programming relaxation based method with a provable worst case bound to solve it. Tanınmış, Aras, and Altınel [53] consider a variation of the well-known Influence Maximization Problem including two decision makers. The leader strives to increase total influence spread by picking the most influential nodes, while the follower tries to reduce it by deactivating some of these nodes. The lower level problem is a stochastic programming problem considering the uncertainty of threshold in Linear Threshold Propagation Problem.

5.2 Models

Influence Maximization Problem refers to find a set of k seed nodes to start a cascade process so that the nodes activated at the end is maximized. Here we're solving the Influence Maximization Problem considering various uncertainties. The uncertainties include dynamic network, changing user behavior and activation price.

Two-stage stochastic program is introduced to model the problem considering uncertainties. The main idea behind two-stage stochastic programming is that optimal decisions should be made using data accessible at the moment they are made, rather than relying on future observations. In the Influence Maximization Problem considering uncertainties, instead of just selecting the seed nodes to activate at the beginning, here we select nodes to activate in two stages(two different times) considering the uncertainty. A general two-stage stochastic binary program could be represented as:

$$\begin{aligned} \max \quad & c^T x + \sum_{\omega \in \Lambda} \rho_\omega \sigma_\omega(x) && \text{(OBJ)} \\ s.t. \quad & x \in \chi \\ & x \in \{0,1\}^n \end{aligned}$$

where $c \in R^n$ is a given objective vector, the set χ represents the constraints on the first-stage variables x and $\sigma_\omega(x)$ is the objective function of the second-stage problem for scenario $\omega \in \Lambda$ solved as a function of first-stage decisions.

For our problem, at the first stage, we should make decisions which users to choose as seed nodes here and now. At the second stage, after knowing the uncertainty, we optimize our behavior by solving an appropriate optimization problem.

Sets and Parameters:	
c_i :	cost of selecting seed node i
d_i :	cost of activating node i
s :	total budget
Decision Variables:	
x_i :	1 if node i is selected as seed node; 0 otherwise
y_i :	1 if node i is activated; 0 otherwise

Based on the notations, the stochastic binary program is shown below:

$$\begin{aligned} \max \quad & E_\omega[g(x,y,\omega)] && \text{(OBJ)} \\ s.t. \quad & c^T x + d^T y \le s \\ & x, y \in \{0,1\}^n \end{aligned}$$

Where $g(x,y,\omega)$ represents the number of influenced nodes for a scenario ω corresponding to a given seed selection strategy x at the beginning and node activation strategy y at the second stage. Then $E_\omega[g(x,y,\omega)]$ denotes the total expected number of influenced nodes, where the expectation is taken with respect to the probabilistic scenario which could be network topology, threshold or activation price.

5.2.1 Influence Maximization Model Considering Stochastic Network Topology

The existing literature has almost always focused on solving the influence maximization problem for a deterministic network where all connections between users are known ahead of time. However, in reality, the network topology of social media is dynamic. Users add new friends or break the friendship everyday. To better fit the reality, we assume the topology of the network is uncertain: some connections may or may not exist. This makes it a better representation of, for instance, emerging social networks where the connections between users are not yet well-defined. Thus, we propose the influence maximization model considering dynamic network topology shown below to select the seed nodes at the beginning that spread the most influence at the end.

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{n} \rho_\omega (z_i^\omega + y_i^\omega) & & \text{(OBJ)} \\
s.t. \quad & \sum_{i=1}^{n} (c_i x_i + d_i^\omega y_i^\omega) \leq s & \forall \omega \in S & \quad (5.1) \\
& \frac{1}{|N(i)^\omega|} \sum_{j \in N(i)^\omega} x_j + \theta x_i \geq \theta z_i^\omega & \forall i \in V, \omega \in S & \quad (5.2) \\
& z_i^\omega + y_i^\omega \leq 1 & \forall i \in V, \omega \in S & \quad (5.3) \\
& c_i = c|N(i)| & \forall i \in V & \quad (5.4) \\
& d_i = d|N(i)^\omega| & \forall i \in V, \omega \in S & \quad (5.5) \\
& x, y, z \in \{0,1\}^n & & \quad (5.6)
\end{aligned}
$$

Here x_i refers to the initial status of node i. y_i^ω refers to whether to activate the node i under scenario ω. z_i^ω refers to the final status of node i under scenario ω. ρ_ω represents the probability of different network topology. θ represents threshold in our experiment. $N(i)$ represents the neighbors of user i. $N(i)^\omega$ represents the neighbors of user i under scenario ω. s represents the total budget. c_i represents the cost of selecting seed node i at the beginning, where c represents the unit cost per neighbor of selecting seed node i. d_i represents the cost of activating node i, where d represents the unit cost per neighbor of activating node i.

The objective function aims to maximize the nodes influenced at the end. Constraints (5.1) set the budget of total activated nodes. Constraints (5.2) make sure that the node will be influenced when it is active already or have at least θ of active neighbors. Constraints (5.3) confirm that the nodes will not be activated again at the second stage when the nodes are already activated.

5.2.2 *Influence Maximization Model Considering Stochastic Thresholds*

When we promote messages in social media, users may have different reactions to the messages depending on the time they see the messages and their mood. For example, when it's holiday season or weekends, users may be more active in re-posting the message because they have more time and they are more relaxed. Here we consider the Linear Threshold Propagation Model, therefore we could introduce stochastic thresholds to describe the changing user behavior. The following model is the influence maximization model considering stochastic thresholds.

$$\begin{aligned}
\max \quad & \sum_{i=1}^{n} \rho_\omega (z_i^\omega + y_i^\omega) && && \text{(OBJ)} \\
s.t. \quad & \sum_{i=1}^{n} (c_i x_i + d_i y_i^\omega) \le s && \forall \omega \in S && (5.7) \\
& \frac{1}{|N(i)|} \sum_{j \in N(i)} x_j + \theta_i^\omega x_i \ge \theta_i^\omega z_i^\omega && \forall i \in V, \omega \in S && (5.8) \\
& z_i^\omega + y_i^\omega \le 1 && \forall i \in V, \omega \in S && (5.9) \\
& x, y, z \in \{0,1\}^n && && (5.10)
\end{aligned}$$

Here x_i refers to the initial status of node i. y_i^ω refers to whether to activate the node i under scenario ω. z_i^ω refers to the final status of node i under scenario ω. ρ_ω represents the probability of different propagation thresholds combination. θ_i^ω represents the threshold under scenario ω for user i. $N(i)$ represents the neighbors of user i.

5.2.3 *Influence Maximization Model Considering Stochastic Price*

When advertisers launch an advertising campaign in social media, the ad price may change over time. We need to adjust the money in a reasonable way so that we could maximize the influence under the budget. Therefore, we build the influence maximization model considering stochastic price in the future as follows.

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{n} \rho_\omega (z_i + y_i^\omega) && && \text{(OBJ)}\\
s.t. \quad & \sum_{i=1}^{n} (c_i x_i + d_i^\omega y_i^\omega) \le s && \forall \omega \in S && (5.11)\\
& \frac{1}{|N(i)|} \sum_{j \in N(i)} x_j + \theta x_i \ge \theta z_i && \forall i \in V && (5.12)\\
& z_i + y_i^\omega \le 1 && \forall i \in V, \omega \in S && (5.13)\\
& x, y, z \in \{0,1\}^n && && (5.14)
\end{aligned}
$$

Here x_i refers to the initial status of node i. y_i^ω refers to whether to activate the node i under scenario ω. z_i refers to the final status of node i. ρ_ω represents the probability of different activation price combination. θ represents the threshold. $N(i)$ represents the neighbors of user i.

5.3 Sample Average Approximation Method

For the influence maximization model considering stochastic network topology, the scenarios of network topology could be very large, which is $2^{\frac{|V|(|V|-1)}{2}}$. Besides, for the influence maximization model considering stochastic thresholds, the threshold θ_i follows a continuous uniform distribution under the Linear Threshold Propagation Model. Even if θ_i follows a discrete distribution, the scenario is still large which is $m^{|V|}$, where m refers to the discrete values of threshold. In both cases, the scenario is too large that provides a poor solution for a large-sized integer program. Therefore, we utilize the sample average approximation(SAA) to reformulate the model. The process of SAA is summarized below:

Step 1. Generate M independent samples each of size N, i.e., $\omega_j^1, \omega_j^2, ..., \omega_j^N$ for $j = 1, ..., M$. Solve the SAA problem that corresponds to each sample.

$$\max \frac{1}{N} \sum_{n=1}^{N} \left[g\left(x, y, \omega_j^n\right) \right]$$

Let f_N^j and $\hat{x}_N^j$, $j = 1, ..., M.$, be the corresponding optimal objective value and an optimal solution, respectively.

Step 2. Compute

$$\bar{f}_{N,M} = \frac{1}{M} \sum_{j=1}^{M} f_N^j$$
$$\sigma_{\bar{f}_{N,M}}^2 = \frac{1}{(M-1)M} \sum_{j=1}^{M} (f_N^j - \bar{f}_{N,M})^2$$

It's proved that the expected value of f_N is greater than or equal to the optimal value f^* of the problem [58]. $\bar{f}_{N,M}$ is an unbiased estimator of $E[f_N]$, we could conclude that $E\left[\bar{f}_{N,M}\right] \geq f^*$. Thus $\bar{f}_{N,M}$ provides a higher statistical bound for the optimal value f^* of the true problem, and $\sigma_{\bar{f}_{N,M}}^2$ is an estimate of the estimator's variance.

Step 3: Choose $\hat{x} \in X$ as one of the calculated $\hat{x}_N^j$ which has the largest true objective function. The true objective function value $\tilde{f}(\bar{x})$ as follows:

$$\tilde{f}_{N'}(\bar{x}) = \frac{1}{N'} \sum_{n=1}^{N'} \left[g\left(x, y, \omega^n\right) \right]$$

Here $\omega^n, ..., \omega^{N'}$ is a sample of size $N^{'}$, where $N^{'} > N$, generated independently. Here $\tilde{f}_{N'}(\bar{x})$ is an unbiased estimator of $f(\bar{x})$. Since $\bar{x}$ is a feasible solution to the true problem, we have $f(\bar{x}) \leq f^*$. Thus $\tilde{f}_{N'}(\bar{x})$ is an estimate of a lower bound on f^*. When the sample $\omega^1, ..., \omega^{N'}$ is independent identically distributed, then the variance of the estimate can be estimated as

$$\sigma^2_{N'}(\bar{x}) = \frac{1}{(N'-1)N'} \sum_{n=1}^{N'} \left(g(\bar{x}, y, \omega^n) - \tilde{f}_{N'}(\bar{x}) \right)^2$$

Step 4: Using the upper bound estimate and the objective function value estimate from Steps 2 and 3, compute an estimate of the optimality gap of the solution $\hat{x}$ as follows:

$$gap_{N,M,N'}(\bar{y}) = \bar{f}_{N,M} - \tilde{f}_{N'}(\bar{y})$$

The estimated variance of the above gap estimator is then calculated as follows:

$$\sigma^2_{gap} = \sigma^2_{N'}(\bar{x}) + \sigma^2_{\bar{f}_{N,M}}$$

5.4 Experimental Evaluation

We conduct several numerical experiments using the SAA method for solving the proposed stochastic influence maximization models. We first conduct the computational experiments for different network topology including Barabasi-Albert, Watts-Strogatz and Erdos-Renyi networks. Then we evaluate the quality of the stochastic programming solution.

5.4.1 Network Topology Comparison

Here we use three different network models to test our stochastic influence maximization models: Barabasi-Albert, Watts-Strogatz, and Erdos-Renyi models. The Barabasi-Albert model generates networks with the scale-free property using a preferential attachment mechanism. Here we generate a graph of 20 nodes grown by attaching new nodes each with 2 edges that are preferentially attached to existing nodes with high degree. The Watts-Strogatz model generates a random graph with small-world properties, including short average path lengths and high clustering. We generate a Watts-Strogatz graph with 20 nodes and each node is connected to 6 nearest neighbors in ring topology with 0.1 probability of rewiring. The Erdos-Renyi model also has the property of having short average path lengths but with low clustering. We generate a Erdos-Renyi graph with 20 nodes where we choose each of the possible edges with probability $p = 0.1$.

As mentioned in Section 4.1, for the SAA method, we generate M independent samples involving each of size N sampled scenarios. Then we evaluate the candidate solution by evaluating the object function using $N^{'}$ sampled scenarios. In our implementation, we use $M = 20, N = 50, N^{'} = 2000$.

For the influence maximization model considering stochastic network topology, we set the budget as 30% of total edges, the activation threshold as 0.3. The cost of seeding node i at time 0 is directly proportional to the total number of neighbors of node i and the cost of activating node i at time 1 is proportional to 80% of the total number of neighbors of node i. To generate the stochastic network topology, we assume there's 10% probability to create an edge and 1% probability to delete an edge. The computational results of different network topology are shown in Table 5.1.

Table 5.1: Computational Results for Stochastic Network Topology Model

Network	$\bar{f}_{N,M}$	$\tilde{f}_{N'}(\bar{x})$	$gap_{N,M,N'}$	$\sigma^2_{\bar{f}_{N,M}}$	$\sigma^2_{N'}(\bar{x})$	σ^2_{gap}
BA-20	5.9970	5.6250	0.3720	0.0003	0.0004	0.0007
ER-20	9.6370	9.2775	0.3595	0.0007	0.0007	0.0014
Watts-20	6.6610	6.2415	0.4195	0.0008	0.0006	0.0014

For the influence maximization model considering stochastic thresholds, we set the budget as 30% of total edges. The cost of seeding node i at time 0 is proportional to the total number of neighbors of node i and the cost of activating node i at time 1 is proportional to 80% of the total number of neighbors of node i. To generate the stochastic thresholds, we assume the threshold of each user follows a uniform distribution between 0.001 and 1. The computational results of stochastic network thresholds are shown in Table 5.2.

Table 5.2: Computational Results for Stochastic Thresholds Model

Network	$\bar{f}_{N,M}$	$\tilde{f}_{N'}(\bar{x})$	$gap_{N,M,N'}$	$\sigma^2_{\bar{f}_{N,M}}$	$\sigma^2_{N'}(\bar{x})$	σ^2_{gap}
BA-20	7.6830	7.3650	0.3180	0.0009	0.0005	0.0014
ER-20	8.3280	8.0050	0.3230	0.0015	0.0007	0.0022
Watts-20	6.3900	5.9390	0.4510	0.0009	0.0011	0.0020

For the influence maximization model considering stochastic price, we set the budget as 30% of total edges, the activation threshold as 0.3. The cost of seeding node *i* at time 0 is proportional to the total number of neighbors of node *i*. To generate the stochastic price at the second stage, we assume the price could have three different scenarios: 0.5$*$total number of neighbors, 1.0$*$total number of neighbors and 1.5$*$total number of neighbors. For the Influence Maximization Model Considering Stochastic Price, the scenarios are not large, thus we could easily get the global optimal. The computational results of stochastic price are shown in Table 5.3 and the seed selection strategy is shown in Figure 5.1.

Table 5.3: Computational Results for Stochastic Price Model

Network	f_1^*	f_2^*	f_3^*	f^*
BA-20	11	10	10	10.33
ER-20	10	10	10	10
Watts-20	8	8	8	8

For all the results, we find that only BA-20 network could generate second stage seed selection, which can be explained by that BA-20 network has a long-tailed distribution. There's a small number of celebrities(large influencers) but a large number of micro influencers. Then in this case there will be more needs to activate nodes at the second stage.

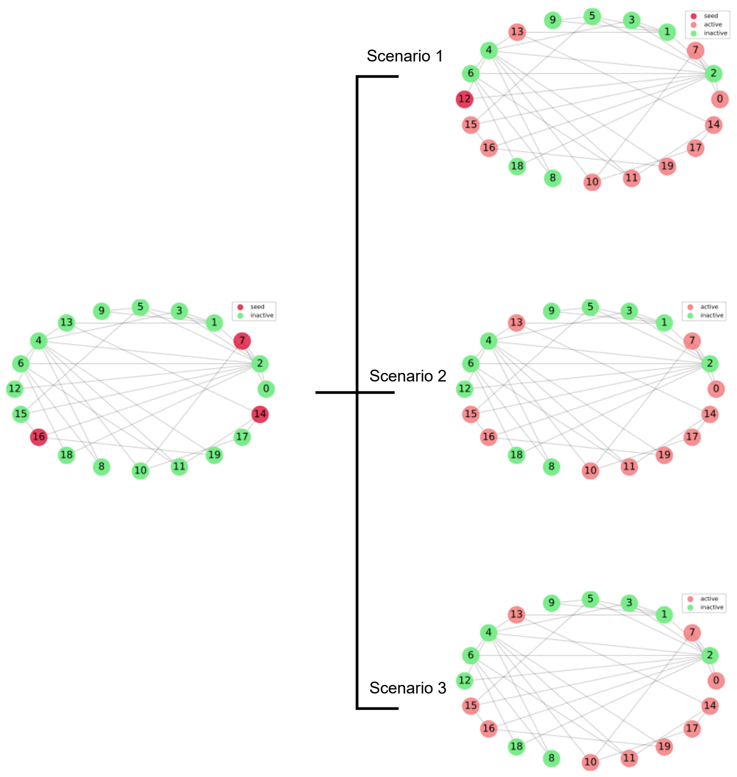
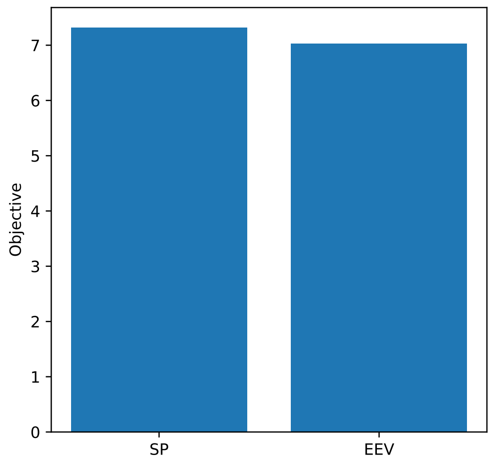


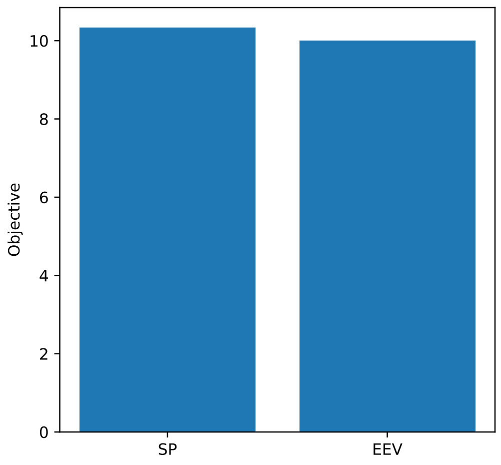
Figure 5.1: Stochastic Price Model Seed Selection for BA-20 Network

5.4.2 *Quality of Stochastic Programming Solution*

Firstly we evaluate the value of the stochastic solution(VSS). The VSS is defined as the difference between the stochastic programming model(SP) and the expected result of using the mean values of the uncertain problem parameters(EEV). Figure 5.2 shows the SP and EEV For different stochastic models. Here we get the results of SP and EEV using 3000 scenarios based on the stochastic programming solution getting from SAA and the deterministic solution getting from the expected value problem. In the stochastic threshold model, there's a differnce between EEV and SP, therefore, we could conclude there's a value of stochastic solutions. In the stochastic price model, the value of the stochastic solution is larger than the expected value of the mean solution as well. Therefore, we could conclude there's a value of stochastic solutions for stochastic price model.



(a) Stochastic Threshold Model



(b) Stochastic Price Model

Figure 5.2: The Value of Stochastic Solutions

Table 5.4, 5.5 present the optimality gap estimates identified by the SAA method for different sample size N. We could conclude that SAA model with a modest number of sampled scenarios could provide very high quality solutions to the stochastic models involving potentially infinite number of scenarios. The more scenarios included in the SAA problem, the smaller the gap.

Table 5.4: Optimality Gap Estimates for the Stochastic Network Model

N	$gap_{N,M,N'}$	σ^2_{gap}
5	1.884	0.0170
10	1.5240	0.0053
20	1.0475	0.0046
50	0.3720	0.0007

Table 5.5: Optimality Gap Estimates for the Stochastic Threshold Model

N	$gap_{N,M,N'}$	σ^2_{gap}
5	1.3635	0.0165
10	0.6940	0.0043
20	0.6930	0.0028
50	0.3180	0.0014

CHAPTER 6: CONCLUSIONS

The dissertation studies both the learning and decision-making problems in social media, which mainly focuses on the information diffusion learning and influence maximization problems.

In Chapter 2, we integrate neighborhood information and embedding similarity into repost prediction in social media. We come up with different machine learning prediction models, and among them stacking model outperforms the other models in Sina Weibo dataset. In addition, XGBoosting model is also a good choice with comparable performance but much less computational time. When extracting the features, we take neighborhood information into consideration as well. Friend circle is a good indicator of user's repost decison. Users who have very active friends tend to be more active in reposting messages. To collect neighborhood information, we come up with two different combination models of neighbors' user profile inspired by the graph theory. We analyze two combination models and compare their performance in learning repost probability. Here we use an efficient way to extract the neighborhood information by aggregating the neighbors' user profile. It would be interesting to explore whether using the graph embedding or graph convolutional neural network will help to improve the repost prediction.

In Chapter 3, we investigate the problem of learning diffusion influence weights between pairs of users in social media networks from mining historical cascades. We formulate two different mixed-integer programming models to learn the diffusion influence weights which could be used to predict the status of users in the future. The first mixed integer programming model is based on the popular propagation model, i.e., Linear Threshold Model. The second mixed integer programming model is based on the idea of random walk considering the influence from multi-step neighborhood. For larger network, we introduce approximate approaches for both models using neural network. We bring marriage between optimization-based diffusion models and neural network through approximating optimization models using a single layer neural network. Therefore, the parameters in the neural network are explainable, which are diffusion weights related parameters in our models. The approximate approaches using artificial neural networks have relatively good performance and fast computational speed. There are several interesting directions for future works. Here we just learn the information diffusion influence through the historical cascades. It would be more interesting to consider other features such as the similarity between posters and reposters, the user profiles or even the topology information of the social media networks to make predictions with higher accuracy. Besides, here we have the predefined thresholds, we could try to learn the influence weights and thresholds together in the future study.

In Chapter 4, we develop the influence and activation thresholds target set selection models including both the minimum influence and activation thresholds target set selection models and maximum influence and activation thresholds target set selection models. Our models allow us to find the minimum seed nodes that influence all the nodes at time T, and determine the seed nodes under budget that maximize the influence. In addition, to appeal to various applications, different forms of seed nodes selection models are proposed, which are maximum activation, early activation and dynamic activation threshold. We provide different computational algorithms to tackle the various datasets as well. They are Graph Partition, Leaf Node, Degree Centrality, DFS Greedy, BFS Greedy, Recursive Threshold and Two Stage in Time computational algorithms. Experiements in various datasets show that DFS Greedy and BFS Greedy are much more efficient than the other methods for large size datasets. Besides, leaf node deletion and degree centrality selection perform better in terms of long-tailed network. While we already consider the maximum activation, early activation and dynamic threshold models in the manuscript, we could still customize the dual threshold target set selection models for different applications for future study. Furthermore, we could investigate comprehensively various computational algorithms with regard to different network topologies.

In Chapter 5, our research focuses on tackling the influence maximization problem considering uncertainties of network topology, user behavior and activation price. We formulate a two-stage stochastic optimization model to solve the problem. Considering the large scenarios of network topology and activation thresholds, Sample Average Approximation method is used to approximate the objective value. Experiments show that Sample Average Approximation method with a modest number of sampled scenarios could provide very high quality solutions. Considering the propagation process could last a long time horizon, we think the multi-stage stochastic optimization would help to improve the performance of tackling the influence maximization problem considering uncertainties. In the future research, we could formulate the problem using multi-stage stochastic optimization model.

In summary, the dissertation applies different novel methodologies, i.e., mathematical programming, stochastic programming, machine learning, graph theory, or even their combination to better address the learning and decision making problems in social media. It will definitely inspire other researchers to approach the problems in social media from different angles and to solve the problems utilizing different tools. We would like to see that these novel models and efficient methodologies could be further applied and developed in industry applications in the future.

LIST OF REFERENCES

[1] Pedro Domingos. “Mining social networks for viral marketing”. In: *IEEE Intelligent Systems* 20.1 (2005), pp. 80–82.

[2] Ceren Budak, Divyakant Agrawal, and Amr El Abbadi. “Limiting the spread of misinformation in social networks”. In: *Proceedings of the 20th international conference on World wide web*. ACM. 2011, pp. 665–674.

[3] George H Chen, Stanislav Nikolov, and Devavrat Shah. “A latent source model for nonparametric time series classification”. In: *Advances in Neural Information Processing Systems*. 2013, pp. 1088–1096.

[4] Oren Tsur and Ari Rappoport. “What’s in a hashtag?: content based prediction of the spread of ideas in microblogging communities”. In: *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM. 2012, pp. 643–652.

[5] Simon Bourigault, Sylvain Lamprier, and Patrick Gallinari. “Representation learning for information diffusion through social networks: an embedded cascade model”. In: *Proceedings of the Ninth ACM international conference on Web Search and Data Mining*. 2016, pp. 573–582.

[6] Manuel Gomez Rodriguez, David Balduzzi, and Bernhard Schölkopf. “Uncovering the temporal dynamics of diffusion networks”. In: *arXiv preprint arXiv:1105.0697* (2011).

[7] David Kempe, Jon Kleinberg, and Éva Tardos. “Maximizing the spread of influence through a social network”. In: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2003, pp. 137–146.

[8] Ning Chen. “On the approximability of influence in social networks”. In: *SIAM Journal on Discrete Mathematics* 23.3 (2009), pp. 1400–1415.

[9] Mengnan Chen, Qipeng P Zheng, Vladimir Boginski, and Eduardo L Pasiliao. "Reinforcement Learning in Information Cascades Based on Dynamic User Behavior". In: *International Conference on Computational Data and Social Networks*. Springer. 2019, pp. 148–154.

[10] Guanxiang Yun, Qipeng P Zheng, Vladimir Boginski, and Eduardo L Pasiliao. "Information Network Cascading and Network Re-construction with Bounded Rational User Behaviors". In: *International Conference on Computational Data and Social Networks*. Springer. 2019, pp. 351–362.

[11] Cheng-Lung Chen, Eduardo L Pasiliao, and Vladimir Boginski. "A Cutting Plane Method for Least Cost Influence Maximization". In: *International Conference on Computational Data and Social Networks*. Springer. 2020, pp. 499–511.

[12] Elliot Anshelevich, Deeparnab Chakrabarty, Ameya Hate, and Chaitanya Swamy. "Approximation algorithms for the firefighter problem: Cuts over time and submodularity". In: *International Symposium on Algorithms and Computation*. Springer. 2009, pp. 974–983.

[13] Mark Granovetter. "Threshold models of collective behavior". In: *American journal of sociology* 83.6 (1978), pp. 1420–1443.

[14] Devesh Varshney, Sandeep Kumar, and Vineet Gupta. "Predicting information diffusion probabilities in social networks: A Bayesian networks based approach". In: *Knowledge-Based Systems* 133 (2017), pp. 66–76.

[15] Kazumi Saito, Kouzou Ohara, Yuki Yamagishi, Masahiro Kimura, and Hiroshi Motoda. "Learning diffusion probability based on node attributes in social networks". In: *International Symposium on Methodologies for Intelligent Systems*. Springer. 2011, pp. 153–162.

[16] Bo Jiang, Jiguang Liang, Ying Sha, Rui Li, Wei Liu, Hongyuan Ma, and Lihong Wang. "Retweeting behavior prediction based on one-class collaborative filtering in social net-

works". In: *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM. 2016, pp. 977–980.

[17] Cédric Lagnier, Ludovic Denoyer, Eric Gaussier, and Patrick Gallinari. "Predicting information diffusion in social networks using content and user's profiles". In: *European conference on information retrieval*. Springer. 2013, pp. 74–85.

[18] Adrien Guille and Hakim Hacid. "A predictive model for the temporal dynamics of information diffusion in online social networks". In: *Proceedings of the 21st international conference on World Wide Web*. ACM. 2012, pp. 1145–1152.

[19] Jiang Zhu, Fei Xiong, Dongzhen Piao, Yun Liu, and Ying Zhang. "Statistically modeling the effectiveness of disaster information in social media". In: *Global Humanitarian Technology Conference (GHTC), 2011 IEEE*. IEEE. 2011, pp. 431–436.

[20] Huan-Kai Peng, Jiang Zhu, Dongzhen Piao, Rong Yan, and Ying Zhang. "Retweet modeling using conditional random fields". In: *2011 11th IEEE International Conference on Data Mining Workshops*. IEEE. 2011, pp. 336–343.

[21] Bongwon Suh, Lichan Hong, Peter Pirolli, and Ed H Chi. "Want to be retweeted? large scale analytics on factors impacting retweet in twitter network". In: *2010 IEEE Second International Conference on Social Computing*. IEEE. 2010, pp. 177–184.

[22] Hongliang Fei, Ruoyi Jiang, Yuhao Yang, Bo Luo, and Jun Huan. "Content based social behavior prediction: a multi-task learning approach". In: *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM. 2011, pp. 995–1000.

[23] Muhan Zhang and Yixin Chen. "Link prediction based on graph neural networks". In: *Advances in neural information processing systems* 31 (2018).

[24] Nils Reimers and Iryna Gurevych. “Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Nov. 2020. URL: https://arxiv.org/abs/2004.09813.

[25] Nils Reimers and Iryna Gurevych. “Sentence-bert: Sentence embeddings using siamese bert-networks”. In: *arXiv preprint arXiv:1908.10084* (2019).

[26] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. “Unsupervised cross-lingual representation learning at scale”. In: *arXiv preprint arXiv:1911.02116* (2019).

[27] Jürgen Schmidhuber. “Deep learning in neural networks: An overview”. In: *Neural networks* 61 (2015), pp. 85–117.

[28] Yoshua Bengio et al. “Learning deep architectures for AI”. In: *Foundations and trends® in Machine Learning* 2.1 (2009), pp. 1–127.

[29] Jing Zhang, Jie Tang, Juanzi Li, Yang Liu, and Chunxiao Xing. “Who Influenced You? Predicting Retweet via Social Influence Locality”. In: *ACM Trans. Knowl. Discov. Data* 9.3 (Apr. 2015), 25:1–25:26. ISSN: 1556-4681. DOI: 10.1145/2700398. URL: http://doi.acm.org/10.1145/2700398.

[30] Masahiro Kimura, Kazumi Saito, and Hiroshi Motoda. “Blocking links to minimize contamination spread in a social network”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 3.2 (2009), p. 9.

[31] Kazumi Saito, Ryohei Nakano, and Masahiro Kimura. “Prediction of information diffusion probabilities for independent cascade model”. In: *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer. 2008, pp. 67–75.

[32] Kazumi Saito, Masahiro Kimura, Kouzou Ohara, and Hiroshi Motoda. "Learning continuous-time information diffusion model for social behavioral data analysis". In: *Asian Conference on Machine Learning*. Springer. 2009, pp. 322–337.

[33] Amit Goyal, Francesco Bonchi, and Laks VS Lakshmanan. "Learning influence probabilities in social networks". In: *Proceedings of the third ACM international conference on Web search and data mining*. ACM. 2010, pp. 241–250.

[34] Wei Chen, Laks VS Lakshmanan, and Carlos Castillo. "Information and influence propagation in social networks". In: *Synthesis Lectures on Data Management* 5.4 (2013), pp. 1–177.

[35] Christopher M Bishop. "Pattern recognition and machine learning". In: (2006).

[36] Mordecai Avriel. "Nonlinear programming: analysis and methods". In: (2003).

[37] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. "Deep learning". In: (2016).

[38] Abram L Friesen and Pedro Domingos. "Deep learning as a mixed convex-combinatorial optimization problem". In: *arXiv preprint arXiv:1710.11573* (2017).

[39] Kristina Lerman and Rumi Ghosh. "Information contagion: An empirical study of the spread of news on digg and twitter social networks". In: *Fourth International AAAI Conference on Weblogs and Social Media*. 2010.

[40] Eyal Ackerman, Oren Ben-Zwi, and Guy Wolfovitz. "Combinatorial model and bounds for target set selection". In: *Theoretical Computer Science* 411.44-46 (2010), pp. 4017–4022.

[41] Paulo Shakarian, Sean Eyre, and Damon Paulo. "A scalable heuristic for viral marketing under the tipping model". In: *Social Network Analysis and Mining* 3.4 (2013), pp. 1225–1248.

[42] Gwen Spencer and Richard Howarth. “Maximizing the spread of stable influence: Leveraging norm-driven moral-motivation for green behavior change in networks”. In: *arXiv preprint arXiv:1309.6455* (2013).

[43] Dilek Günneç, Subramanian Raghavan, and Rui Zhang. “Least-cost influence maximization on social networks”. In: *INFORMS Journal on Computing* (2019).

[44] Subramanian Raghavan and Rui Zhang. “A branch-and-cut approach for the weighted target set selection problem on social networks”. In: *INFORMS Journal on Optimization* 1.4 (2019), pp. 304–322.

[45] Meeyoung Cha, Alan Mislove, and Krishna P Gummadi. “A measurement-driven analysis of information propagation in the flickr social network”. In: *Proceedings of the 18th international conference on World wide web*. 2009, pp. 721–730.

[46] Shaozhi Ye and S Felix Wu. “Measuring message propagation and social influence on Twitter. com”. In: *International conference on social informatics*. Springer. 2010, pp. 216–231.

[47] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. “Finding community structure in very large networks”. In: *Physical review E* 70.6 (2004), p. 066111.

[48] Albert-László Barabási and Réka Albert. “Emergence of scaling in random networks”. In: *science* 286.5439 (1999), pp. 509–512.

[49] Wayne W Zachary. “An information flow model for conflict and fission in small groups”. In: *Journal of anthropological research* 33.4 (1977), pp. 452–473.

[50] Jérôme Kunegis. “Konect: the koblenz network collection”. In: *Proceedings of the 22nd international conference on world wide web*. 2013, pp. 1343–1350.

[51] Jure Leskovec and Julian J Mcauley. “Learning to discover social circles in ego networks”. In: *Advances in neural information processing systems*. 2012, pp. 539–547.

[52] Benedek Rozemberczki and Rik Sarkar. *Characteristic Functions on Graphs: Birds of a Feather, from Statistical Descriptors to Parametric Models*. 2020. arXiv: 2005 . 07959 [cs.LG].

[53] Kübra Tanınmış, Necati Aras, and IK Altınel. "Influence maximization with deactivation in social networks". In: *European Journal of Operational Research* 278.1 (2019), pp. 105–119.

[54] Amit Goyal, Wei Lu, and Laks VS Lakshmanan. "Celf++ optimizing the greedy algorithm for influence maximization in social networks". In: *Proceedings of the 20th international conference companion on World wide web*. 2011, pp. 47–48.

[55] Yongjia Song and Thang N Dinh. "Optimal containment of misinformation in social media: A scenario-based approach". In: *International Conference on Combinatorial Optimization and Applications*. Springer. 2014, pp. 547–556.

[56] Hao-Hsiang Wu and Simge Kucukyavuz. "Maximizing Influence in Social Networks: A Two-Stage Stochastic Programming Approach That Exploits Submodularity". In: *arXiv preprint arXiv:1512.04180* (2015).

[57] Evren Güney. "An efficient linear programming based method for the influence maximization problem in social networks". In: *Information Sciences* 503 (2019), pp. 589–605.

[58] Wai-Kei Mak, David P Morton, and R Kevin Wood. "Monte Carlo bounding techniques for determining solution quality in stochastic programs". In: *Operations research letters* 24.1-2 (1999), pp. 47–56.