Influence-oriented Community Analysis in Social Networks

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

Xinjue Wang
B.Sc.,
Department of Computer Science and Technology,
Zhuhai College of Jilin University,
Jinwan, Zhuhai, China.
M.Sc.,
School of Science,
College of Science, Engineering and Health
RMIT University,
Melbourne, Victoria, Australia.

School of Science
College of Science, Engineering and Health
RMIT University

March 20, 2018

For my loving parents,
and my dear maternal and paternal grandparents.

Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program; and, any editorial work, paid or unpaid, carried out by a third party is acknowledged.

Xinjue Wang
School of Science RMIT University
March 20, 2018

Acknowledgments

I would like to express my special thanks of gratitude to my supervisors Dr. Ke Deng and Dr. Jianxin Li who generously gave me guidance and helped me through the Ph.D. candidature. Secondly, I would also like to thank Assoc. Prof. Xiuzhen Zhang and Prof. Timos Sellis who guided me into the realm of research and gave me the golden opportunity to do wherever my passion lies. Finally, I would like to thank my parents and friends who supported me along this journey.

Credits

Portions of the material in this thesis have previously appeared in the following publications:

- J. Li, X. Wang, K. Deng, X. Yang, T. Sellis, and J. X. Yu. Most influential community search over large social networks. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*, pages 871–882, 2017. doi: 10.1109/ICDE.2017.136. URL https://doi.org/10.1109/ICDE.2017.136
- X. Wang, K. Deng, J. Li, J. X. Yu, C. S. Jensen, and X. Yang. Targeted influence minimization in social networks. In *22nd Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2018, Melbourne, Australia, Proceedings*
- J. Li, X. Wang, K. Deng, T. Sellis, J. X. Yu, and F. Xia. Efficient diverse influence maximization in social networks. *submitted to IEEE Transaction on Knowledge and Data Engineering, under review*
- X. Wang, K. Deng, J. Li, J. X. Yu, C. S. Jensen, and X. Yang. Efficient targeted influence minimization in big social networks. *submitted to World Wide Web Journal, under review*

Contents

List of Figures

List of Tables

Abstract

The emergence of online social networks has fundamentally changed the way people communicate with each other. Scholars have never ceased devoting their time and energy to the phenomenon since its emergence. Among researches around the social network,

- One line of study that draws a large amount of attention recently is the discovery of communities, i.e. relatively densely connected sub-networks. Discovering such structures or communities provides insight into the relationship between individuals and composition of a social network. However, these studies mainly focus on the inner connection between individuals inside a community structure and neglect the external influence of a community as a whole.

- Another line of study in the field of the social network is influence analysis which analyze the ability of individuals to convince other users to adopt a new product (or an innovative idea, a service, a political opinion, etc.) with word-of-mouth effect which propagates information through network structures that can trigger cascades of further adoptions. However, these studies mainly focus on the relationship between individuals and the information diffusion process and neglect the community structures in a social network.

There is a lack of studies that analyze the social influence of communities, which is fundamentally important for understanding the relationship between network structures and the information diffusion among it and has many practical applications. For example, a company may try to find the most influential community to advertise their products; an organization may intend to initiate a campaign in hope to attract more diverse customers, i.e., maximizing the number of influenced communities instead of customers; an association may hope to minimize the influence of a malicious information spread by one of its opponents, so that the community consisted of its core customers would be affected the least.

To fill in this meaningful blank, in this thesis, we intend to analyze communities on the aspect of social influence and solve three research questions as follows. First, how to identify the communities with the dense intra-connections and the highest outer influence on the users outside the communities? Second, how to maximize both the spread and the

diversity of the diffusion at the end of the information propagation by selecting a fixed number of influential users from a social network to spread the information. The higher diversity means more communities are influenced. Third, how to minimize the influence of a set of initial active nodes, which has been infected by a piece of malicious information, over a target community? The aim is to protect from this disinformation, by deleting a fixed number of edges in a social network.

To address the first research question, we propose a new metric to measure the likelihood of the community to attract the other users outside the community within the social network, i.e., the community's outer influence. There are lots of applications that need to rank the communities using their outer influence, e.g., Ads trending analytics, social opinion mining and news propagation pattern discovery by monitoring the influential communities. We refer to such problem as *Most Influential Community Search*. While the most influential community search problem in large social networks is essential in various applications, it is mostly ignored by the academic research community. In this work, we systematically investigate this problem. Firstly, we propose a new community model, maximal kr-Clique community, which has desirable characters, i.e., *society*, *cohesiveness*, *connectivity*, and *maximum*. And then, we developed a novel tree-based index structure, denoted as C-Tree, to maintain the offline computed r-cliques. To efficiently search the most influential maximal kr-clique communities with the maximum outer influence, we developed four advanced index-based algorithms, which can improve the search performance of non-indexed solution by about 200 times. The efficiency and effectiveness of constructing index structure and evaluating the search algorithms have been verified using six real datasets including Facebook, Google+, Gowalla, Twitter, Youtube, and Amazon. A small case study shows the value of the most influential communities using DBLP data.

To solve the second research question, we investigate *Diverse Influence Maximization* (DIM) to efficiently find k nodes which, at the end of propagation process, can maximize the number of activated nodes and the diversity of the activated nodes. In this work, an evaluation metric has been proposed to balance the two objectives. To address the computational challenges, we develop two efficient algorithms and one advanced *PSP-Tree* index. The effectiveness and efficiency of our DIM solution are verified by the extensive experimental studies on five real-world social network datasets.

To address the last research question, we study the community-targeted influence minimization problem. Unlike previous influence minimization work, this study considers the influence minimization concerning a particular group of social network users, called *targeted influence minimization*. Thus, the objective is to protect a set of users, called *target nodes*, from malicious information originating from another group of users, called *active nodes*. This study also addresses two fundamental, but largely ignored, issues in different influence minimization problems: (**i**) the impact of a budget on the solution; (**ii**) robust sampling. To this end, two scenarios are investigated, namely unconstrained

and constrained budget. Given an unconstrained budget, we provide an optimal solution; Given a constrained budget, we show the problem is NP-hard and develop a greedy algorithm with an $(1 - \frac{1}{e})$-approximation. More importantly, to solve the influence minimization problem in large, real-world social networks, we propose a robust sampling-based solution with a desirable theoretic bound. Extensive experiments using real social network datasets offer insight into the effectiveness and efficiency of the proposed solutions.

CHAPTER 1

Introduction

The emergence of online social networks has fundamentally changed the way people communicate with each other. Social networks, such as Facebook, Twitter, and Google Plus, not only bring people together but also carry all kinds of social entities into an ever narrow space. Two most important social entities in our modern life are buyer and seller or, in a contemporary term, customer, and company. Social networks have revolutionized the way industries maintain relationships with their (potential) customers. Since the appearance of social networks, we have seen fewer and fewer street questionnaires being handed out and focus groups being held. Meanwhile, more and more companies turn to social networks when it comes to collecting feedback of a certain commodity. In addition, businesses are able to effectively campaign for their new products on social networks thanks to the study of *influence maximization* (IM) problem.

In a social network, people tend to form into small groups out of their similar interest or property in certain social aspect. People inside these small groups tend to have a closer relationship than ones who are outside of these groups. Scholars call these small groups communities or relatively densely connected sub-networks [Newman 2006]. Discovery of such network structures or communities becomes more and more important in recent years since they offer much more valuable information for companies than individuals in a social network alone. With community information, businesses are able to identify social groups and target different groups of people for their different products.

However, these are just a taste of what social networks can provide for our modern life. The much more impactful applications in this area have yet to be unearthed. Both influence and community are important aspects of a social network. Nonetheless, like two sides of a coin, they are essential to be studied together. Analyzing communities on the aspect of social influence is fundamentally important for understanding the relationship between network structures and the information diffusion among it, and would enable various practical applications. For example, in a social network, with the information of

community and information diffusion, a company could find the most influential community to advertise their products; an organization is able to initiate a campaign in hope to attract more diverse customers, i.e. a large number of communities instead of a large number of customers; an association is capable of minimizing the influence of a malicious information spread by one of its opponents, so that the community consisted of its core customers would be affected the least.

Unfortunately, none of the above scenarios are able to be implemented yet, as there is a lack of studies that analyze communities from the aspect of social influence.

1.1 Literature Review

1.1.1 Influence Maximization

Kempe et al. [Kempe et al. 2003b] proposed two discrete influence spread models, Independent Cascade (IC) model and Linear Thresholds model. Based on the two widely-accepted models, there are lots of work focusing on influence maximization problem, e.g., [Domingos and Richardson 2001, Richardson and Domingos 2002, Barbieri et al. 2012, Chen et al. 2014, Aslay et al. 2014, Chen et al. 2015]. Their target is to select a limited number of nodes from a social network where the selected nodes can influence the maximal number of nodes in the social network. The relationship between the selected nodes is not required, i.e., it is very likely the users represented by the selected nodes don't know the existence of the others.

The influence maximization problem was proposed in [Domingos and Richardson 2001, Richardson and Domingos 2002]. The two proposed methods are probabilistic and had no bounded influence spread guarantee. Kempe et. al. [Kempe et al. 2003b] proposed two discrete influence spread models, Independent Cascade (IC) model and Linear Thresholds model. They proved the influence maximization problem using the two models can be solved by a greedy algorithm with $1 - \frac{1}{e}$ approximation ratio. Recently, the problem of influence maximization are also investigated in the other different aspects. E.g., [Barbieri et al. 2012, Chen et al. 2014, Aslay et al. 2014, Chen et al. 2015] studied the problem of topic-aware influence maximization. the assumption is that the influence of social users may be weighted differently for different topics, which would produce different selections of the k seed users. [Li et al. 2014a] studied the location-aware influence maximization problem, i.e., given a query region, how to find the k seed nodes that can influence the maximum number of nodes in the query region. [Feng et al. 2014] considered novelty decay in influence maximization and [Liu et al. 2012, Gomez-Rodriguez and Schölkopf 2012] considered time constraint in influence maximization. Both of them are trying to modify the influence propagation model, e.g., IC model. The most relevant work is [Tang et al. 2014a] that is to seek diverse seed nodes. It assumes that if the k seed nodes are diverse, then their influenced nodes would be also diverse. This assumption is questionable

because two neighbor nodes may influence different sets of nodes due to the direction of social influence in a social network.

1.1.2 Social Community Detection

A great deal of work has been devoted to find communities in large networks, and much of this has been devoted to formalize the intuition that a community is a set of nodes that has more and/or better links between its members than with the remainder of the network. To design effective community discovery models, Newman and Girvan in [Newman and Girvan 2004] proposed a quantitative measure, called modularity, to assess the quality of community structures, and formulated community discovery as an optimization problem. Its key idea is similar to graph partitioning, which iteratively removes the edge with the highest betweenness score. Betweenness based community detection metric was also studied by Girvan and Newman in [Girvan and Newman 2002]. Ruan and Zhang [Ruan and Zhang 2007] proposed a more efficient spectral algorithm to find high quality communities by applying k-way partitioning and recursive 2-way partitioning strategies [White and Smyth 2005]. Satuluri and Parthasarathy in [Satuluri and Parthasarathy 2009] developed efficient Markov clustering algorithms to identify communities by using stochastic flow technique. The key idea in it is to enhance flow to well-connected nodes, i.e., rich get richer and poor get poorer. All the above implicit community detection models focused on the global connectivity of the social network, by which a specified number of partitions (communities) are generated. The discovered communities often have small cohesiveness. Another line of work is to discover communities based on explicit community model like k-core [Li et al. 2014b] and k-truss [Huang et al. 2014]. The k-core of a graph is the largest subgraph within which each node has at least k connections. In the induced subgraph (i.e., a k-core community), since it only requires each node has k neighbors, two nodes may have large hops (i.e., less cohesive). Given a graph G, the k-truss of G is the largest subgraph in which every edge is contained in at least $(k-2)$ triangles within the subgraph. The k-truss is a type of cohesive subgraph defined based on triangle which models the stable relationship among three nodes. With edge connectivity constraints, the induced subgraph (i.e., a k-truss community) is connected and cohesive. But the connectivity is so strong that k-truss can only be used to discover communities of very small size (i.e., not a society). LPA [Raghavan et al. 2007] has been evaluated and recommended to be the better choice as an accurate and efficient community detection technique in the recent studies [Wang et al. 2015] as well as [Leung et al. 2009]. It works as follows. Each node in a social network is first given a unique label. At every iteration, each node is updated by choosing the label which most of its neighbors have. If a node happens to be multiple labels, then one of these would be selected randomly. After several iterations, the communities will be uncovered via the labels where each label represents a community. Besides LPA, there is a considerable amount of research devoting towards community detection

models and algorithms, e.g., with edge content consideration in [Qi et al. 2012], with clique definition and parallel algorithm in [Gregori et al. 2013]. More details can be referred to the survey works in [Leskovec et al. 2010, Wang et al. 2015]. In this work, we propose a new explicit community model, called k-r Maximal Cliques (krMC), which is based on the concept of maximal cliques [Bron and Kerbosch 1973]. Compared to k-core and k-truss, the krMC community model concurrently has the desirable characters including *society, cohesiveness, connectivity, and maximum.*

1.1.3 Diversified Influence Analysis

The database community has recently studied the keyword search result diversification problem in [Demidova et al. 2010, Drosou and Pitoura 2009, Liu et al. 2009, Vieira et al. 2011b;a, Yu et al. 2009, Zhao et al. 2011]. Given a very large database, an exploratory query can easily lead to a vast answer set. Typically, an answer's relevance to the query is based on top-k or tf-idf. As a way of increasing user satisfaction, different result diversification techniques have been proposed including some system based on ones taking into account query parameters, evaluation algorithms, and dataset properties. For many of these, a max-sum type objective function is usually used. Other than those discussed above, there are many recent works studying result diversification in different settings via different approaches and through different perspectives. The reader is referred to [Agrawal et al. 2009, Drosou and Pitoura 2010] for a good summary of this field. All the above diversification works take a query and the query-relevant result candidates as input and select a diverse set as output. However, the input of our proposed diversified influence maximization problem is the whole social network and a user-specified budget k. In this case, every node in the social network may be a candidate. In addition, our problem is to address the diversification of influenced node sets activated by the k-vertex set, rather than measuring the diversity of the k-vertices as [Tang et al. 2014a]. Furthermore, our work is also different from minimum dominating set problem [Basuchowdhuri and Majumder 2014, Zhao et al. 2014] that selects a minimum number of nodes from a network so that any node in the network is either in this minimum node set or is adjacent to at least one node of this set. Different from [Basuchowdhuri and Majumder 2014, Zhao et al. 2014], we can only select a limited number of nodes (e.g., k is often much small than the minimum dominating set size) and need to evaluate their influenced nodes' diversity via an information diffusion model. Therefore, our investigated problem in this thesis is more challenging than the diversified result search problem and the minimum dominating set problem.

1.1.4 Influence Minimization

In this section, we briefly introduce the related work in influence minimization which has attracted the attention of research community in the past decade. While the influence maximization is irrelevant to this work, we concisely discuss the targeted influence maximization which concerns the influence to a targeted set of nodes in social networks.

Overall Influence Minimization

The first track of studies aims to find a certain number of edges in social networks such that, by deleting these edges, the influence of any information in social networks is minimized [Kimura et al. 2007] [Kimura et al. 2008] [Khalil et al. 2013]. No source nodes and targeted nodes are specified.

Kimura et al. have introduced the influence minimization problem [Kimura et al. 2008]. They define *contamination degree* of social networks as the average influence of some information to each individual node. Given a budget, they aim to find the set of edges is social networks such that the number of edges is no more than budget and, if the selected edges are deleted, the *contamination degree* of the social networks is minimized where information diffusion model is *Independent Cascade* (IC) [Kempe et al. 2003a]. They have proposed a greedy algorithm which iteratively selects the next best edge to be removed base on the *contamination degree* reduced. To improve the processing efficiency, they estimate the *contamination degree* by adapting the bond percolation method [Kimura et al. 2007].

In [Khalil et al. 2013], Khalil et al. define *spread susceptibility* of social networks as $\sum_{i \in V} f_i(S)$ where $f_i(S)$ is the number of nodes influenced by node i after deleting a set of edges, denoted as S, in social networks. Given a vector of information propagation probabilities and a positive integer as budget, they aim to select a set of edges in social networks such that the number of edges is no more than budget and, if the selected edges are deleted, the *spread susceptibility* is minimized. They have proposed a greedy algorithm which computes the loss of susceptibility by removing each edge; delete the one leading to the maximum loss; this operation is performed iteratively until the budget limit is reached. They have discussed two information diffusion models, i.e., *Independent Cascade* (IC) and *Linear Threshold* (LT) [Kempe et al. 2003a]. If LT is applied, $f_i(S)$ is a monotone and supermodular function; but it is not true for IC. Therefore, if LT is applied, the greedy algorithm is $(1 - 1/e)$ of the optimal according to [Nemhauser et al. 1978]. Wang et al. solve the similar problem using IC with the aim to minimize the influence of negative information in the network by deleting nodes [Wang et al. 2013].

Source Influence Minimization

The second track of studies assume that initially, a specific set of nodes have some information to be spread; the aim is to delete a certain number of edges social networks such that the influence of information in the social networks is minimized. In particular, the topics of information is considered in [Yao et al. 2015] and the spread of counter-information from competitors in the same period of time is considered in [Luo et al. 2014] [Song et al. 2017].

Luo et al. have investigated a different influence minimization problem considering the influences of two opposite campaigns in a given time period [Luo et al. 2014]. They assume once a node becomes active to the information from one campaign, it will not change back to be inactive and will not be active to the information from another campaign. Given a positive integer as budget and a set of nodes in a social network which are active for campaign A, they aim to select another set of nodes R where the number of nodes in R is subject to the budget, and the number of nodes activated by campaign A is minimized. A greedy algorithm has been proposed with a new time-aware influence diffusion model CTMCDM (Continuous-Time Multiple Campaign Diffusion Model) which is adapted from the diffusion model introduced in [Rodriguez et al. 2014]. The greedy algorithm iteratively adds into R (empty initially) the next best node according to the objective function, i.e., selecting this node for campaign B will lead to the number of nodes activated for campaign A minimized. They prove the objective function is a monotone and submodular. Therefore, the solution greedy algorithm is $(1-1/e)$ of the optimal [Nemhauser et al. 1978].

Yao et al. solve the influence minimization problem under the Topic-aware Independent Cascade (TIC) diffusion model [Yao et al. 2015]. Given a set of infected nodes by a textual message in social networks and a budget, they aim to select a set of uninfected nodes where the number is not more than the budget; if the set of selected nodes are deleted, the number of ultimately infected nodes by the message in the social network is minimized. Specifically, the probability that passing the message from an infected node a to an uninfected node b considers whether b are interested in the message based on the log of past propagation. They have proposed to iteratively delete the node which has the current highest score. The score is measured with either betweenness or out-degree.

With both source and targeted nodes specified, the targeted influence minimization can be viewed as a new version of influence minimization where source nodes are specified as discussed in section 1.1.4. The most relevant work is [Yao et al. 2015] where the edges in social networks have varying weights for different textual messages to be spread. That is, when the message is given, the social network is fixed. In this situation, this situation is same as a special case of our problem where the targeted nodes are all other nodes besides the source nodes. The following two points make this study different from existing studies. First, the existing studies including [Yao et al. 2015] assume that the budget is insufficient even though it is not always true. This study addresses this issue for solving the targeted

influence minimization. Second, the existing studies including [Yao et al. 2015] directly apply greedy algorithm which can also be used to solve our problem; however, we observe that the greedy algorithm is hard to handle large social media. This motivates sampling techniques in our solution in targeted influence minimization.

1.2 Research Questions

In this thesis, we intend to conduct community analysis from the aspect of social influence and find solutions to three research questions as follows.

Given a social network,

1. How to identify the communities with the dense intra-connections and the highest outer influence to the users outside the communities?

2. How to maximize both the spread and the diversity of the diffusion at the end of the information propagation by selecting a fixed number of influential users from a social network to spread the information? (A higher diversity means more communities are influenced at the end of the information propagation.)

3. How to minimize the influence of a set of initial active nodes, which has been infected by malicious information, over a target community, which we aim to protect from this disinformation by deleting a fixed number of edges in a social network?

1.2.1 Most Influential Community Search

First, we try to study communities from the perspective of influence maximization. Precisely, we target to address a significant and novel problem of identifying a number of closely related nodes (i.e., at least k) that can influence the maximal number of outer nodes in the social network. It is denoted as *most influential community search*. The selected k nodes must be cohesive enough, i.e., they could constitute a community. A new explicit community model is defined based on the concept of maximal cliques [Bron and Kerbosch 1973], called maximal kr-Clique community. The maximal kr-Clique community model has the desirable characters, i.e., *society, cohesiveness, connectivity, and maximum*. In specific, a maximal kr-Clique community is defined as a connected and induced subgraph that consists of at least k nodes (*society*), the shortest path of any two nodes is not larger than r (*cohesiveness* and *connectivity*), and it is not a subgraph of another community (*maximum*).

Even though discovering the most influential communities are important in various applications, it is largely ignored by the research community. In this work, we study, for the first time, the influential community search problem in large social networks. The influence of a community is measured by its influence on the users in the social network

outside the community based on the widely-accepted influence propagation model, i.e., the *independent cascading* model (IC-model) [Kempe et al. 2003b]. Our proposed most influential community search problem is fundamentally different from the classic *influence maximization* and the typical *community detection*. First, the problem of influence maximization aims to select a limited number of nodes (i.e., at most k) from a social network where the selected nodes can influence the maximal number of nodes in the social network [Chen et al. 2010, Kimura and Saito 2006, Leskovec et al. 2007]. They did not consider the social distance among the selected k nodes. But in our problem, we require the selected nodes must be within r-hop distance in the social network, which provides a new way to understand the social influence at the community level, rather than at the individual node level. It can help a group of people to know their collaborative effect on influencing the external members. Second, several explicit models can be used to define communities, e.g., k-core [Li et al. 2014b] and k-truss [Huang et al. 2014]. For k-core, the community may be lack of cohesiveness because the number of hops of two members of a community may be enormous. For k-truss, its connectivity requirement of the community is too strong to be practical in discovering reasonably large communities. But our proposed maximal kr-clique community model can address all the shortcomings well.

It is an NP-hard problem to search the most influential communities. To improve the search efficiency, this work has deliberately developed index structure and search algorithms. We would like to point out that our designed index structure and search algorithms are independent of community models, which is further verified in our experiments. That is to say, our index would support the most influential community search where other explicit community models such as k-core and k-truss are applied.

1.2.2 Diverse Influence Maximization

The second problem we intend to solve is diversified influence maximization, in which we contemplate influence maximization problem from the aspect of the community. More specifically, the aim is to select k nodes such that the number of activated nodes and the diversity of the activated nodes can be maximized. Figure 1.1 shows an example

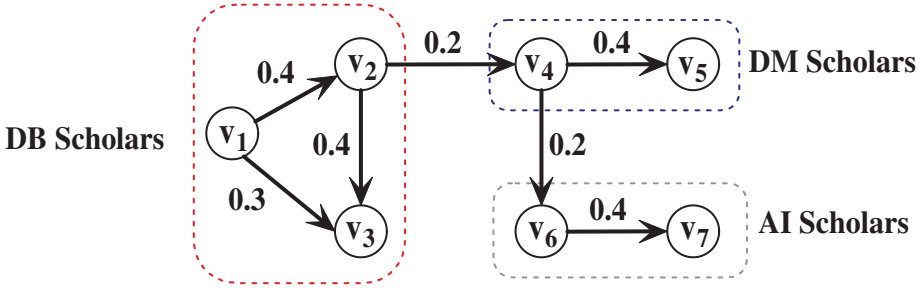


Figure 1.1: An example illustrating diverse influence maximization.

where nodes represent the scholars in different research communities, e.g., v_1, v_2, and v_3 in Database (DB), v_4 and v_5 in Data Mining (DM), v_6 and v_7 in Artificial Intelligence (AI). The weight of an (DB), edge represents the influence probability of connected nodes. The weight of edges within the same community is higher than that across communities. Assume we want to find one scholar (i.e., $k = 1$) to advertise call-for-paper for a conference. Based on IC model, v_1 will be selected as the seed for IM problem. However, for the DIM problem, it is easy to see that selecting v_2 is better than selecting v_1 due to considering the diversity of influenced nodes in the social network. If we take v_2 as the seed, v_2's influence can be propagated into community DB and DM because v_3 in community DB can be influenced by v_2 with the maximum probability 0.4 and v_4 in community DM can get the influence from v_2 with the maximum probability 0.2. If we take v_1 as the seed, v_1 can also successfully influence two nodes v_2 and v_3. But v_1's influence is constrained in community DB only. The maximum probability that v_1 influences v_4 is $0.4 \times 0.2 = 0.08$. With such low probability, it is unlikely for v_4 to adopt the information from v_1. In other words, the information of call-for-paper cannot be propagated from v_1 to community DM. This work is the first effort to formally investigate diverse influence maximization problem.

1.2.3 Targeted Influence Minimization

Finally, we consider the influence minimization problem from the perspective of the community. We propose, define, and solve a new problem of so-called targeted influence minimization. This problem and its solutions are relevant to many applications. For example, a government agent may want to shield young social network users from pornography or recruitment to terrorism; or a company may initiate a campaign to protect their customers from defamatory information spread by their competitors. The targeted influence minimization problem can be briefly described as follows: given a set of source nodes I with information to be spread and a set of target nodes T in a social network, the aim is to find the minimum set of edges under a budget constraint such that deleting these edges minimizes the influence from I to T. The deletion of an edge (u_1,u_2) can be considered as persuading u_1 does not spread any information to u_2, or u_2 does not accept any information from u_1. Note that T may include all nodes other than I in a social network in the extreme case. Suppose a set of nodes I regularly spread information for business B_1. A competitor B_2 may initiate a campaign to prevent such information from a set of target nodes T, such as the customers of B_2. To do that, it needs to find a set of edges under the campaign budget such that these edges will not pass any information related to B_1. As a consequence, the influence from I to T can be reduced to the minimum level.

All existing studies on influence minimization simply assume the budget is insufficient and provide a greedy algorithm. However, this assumption is not always true. We develop an optimal solution to completely block propagated information for the target users if the budget is sufficient. Otherwise, the problem is proved to be NP-hard, and a greedy

algorithm is developed. To meet the time requirement in handling large social network data, a novel sampling-based solution is provided.

1.3 Thesis Organization

The thesis is organized as follows: Chapter 2 discusses the technologies required and used for solving for the most influential community search problem. Chapter 3 provides depth analysis of the diversified influence maximization problem. Chapter 4 discusses the answer to the targeted influence minimization problem in detail. Chapter 5 conclude the thesis.

CHAPTER 2

Most Influential Community Search over Large Social Networks

2.1 Problem Definition

A social network is modeled as a directed graph $G = (V, E, P)$, where V is a set of nodes representing a set of social users, E is a set of edges representing user-to-user friendship relationships, and P_{uv} is a set of weights associated with edges in E, each of which represents the influence probability that user u can influence user v in G.

Definition 1. (kr-Clique Community) Given a social network G=(V, E,P), and integers k and r, a kr-clique community is an *induced* subgraph C^{kr} = (V^{kr}, E^{kr}) of G that meets the following constraints:

- Society - C^{kr} contains at least k nodes;
- Cohesiveness and Connectivity - any two nodes in C^{kr} can be reached at most r hops via their shortest path in the subgraph C^{kr}. Thus, it always holds that C^{kr} must be connected.

A kr-clique community C^{kr} is a *maximal* kr-clique community if there is no existing another kr-clique community $C^{kr'}$ that is a subgraph of C^{kr}.

Example 1. Consider a small social network shown in Figure 2.1. Suppose, for instance, that $r = 2$ and $k = 4$, then by definition the subgraph induced by node set {3, 4, 5, 13, 14} is a maximal kr-clique community. Although the subgraph induced by node set {3, 4, 5, 13} satisfies the criteria of k and r, it is not a maximal kr-clique community because it is contained in another kr-clique community induced by the node set {3, 4, 5, 13, 14}. There are 19 maximal kr-clique communities, among which six communities contain five

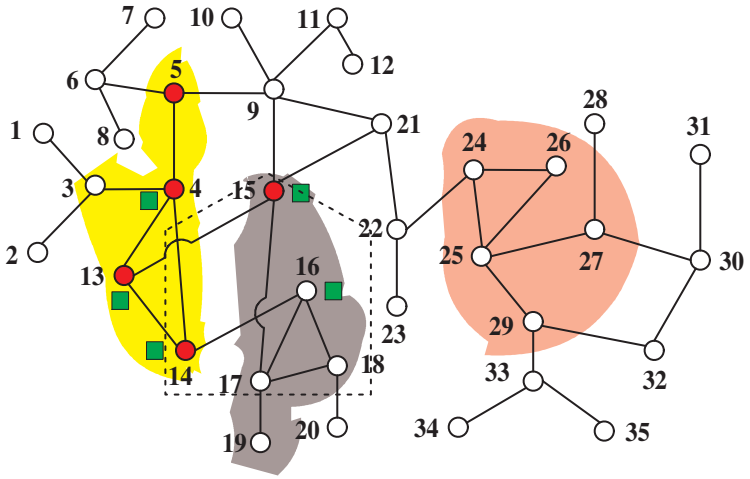


Figure 2.1: An Example Social Network Graph

nodes *in their communities*. We have marked the six communities with different symbols in Figure 2.1 (the six communities have been annotated by different colors or lines).

There are many methods to model the process of influence propagation. In this paper, we adopt the widely-accepted the Linear Threshold (LT) model [Chen et al. 2010, Kempe et al. 2003b]. To measure the influence of a community, we first take all the community members as activated nodes. We assume that an outer node of a community C^{kr} (i.e. outside of C^{kr}) can be activated if and only if the nodes in the community have an *aggregated influence probability* no less than a threshold Δ to the outer node.

Definition 2. (Aggregated Influence Probability)

$$Pr(v|V(C)) = 1 - \prod_{u \in V(C)} (1 - P_{u \to v})) \tag{2.1}$$

where $P_{u \to v}$ is the influence probability of the path from u to v. It is computed by multiplying the probabilities on the edges along the maximum influence path from u to v where the maximum influence path has been evaluated in [Lee and Chung 2015].

Definition 3. (The Outer Influence of a Maximal kr-Clique Community) Given a social network $G = (V, E, P)$ and a maximal kr-clique community C, the influence *score* of C is measured by the number of outer nodes to be successfully activated by C in the social network G. It is defined as below:

$$Score(C) = |\{v | v \in V(G) \setminus V(C) \wedge Pr(v|V(C)) \geq \Delta\}| \tag{2.2}$$

where $Pr(v|V(C))$ is the aggregated influence probability that $V(C)$ can influence to v, and Δ is a threshold parameter to judge if a node can *or cannot* be activated. *It is learnt from historical actions the social users taken in the social network based on the work in [Goyal et al. 2010]. Thus, Δ is a system setting parameter in this paper.*

Example 2. Consider a maximal kr-clique community induced by {3, 4, 5, 13, 14} in Figure 2.1. To make the example concise, we assume that each edge has equal influence probability (e.g., 0.5). For a node, we ignore its influenced nodes if its influence probability less than a value (e.g., 0.15) to the nodes, which just use to simplify the example here. If we say a node can be influenced by a community, then the community should have influence probability to the node by at least a threshold value (e.g., Δ=0.4). So, we have the influenced_list for each node as follows.

Node	Influenced Nodes with Probability
3	(1, 0.5), (2, 0.5)
4	(1, 0.25), (2, 0.25), (6, 0.25), (9, 0.25), (15, 0.25), (16, 0.25)
5	(6, 0.5), (7, 0.25), (8, 0.25), (9, 0.5), (10, 0.25), (11, 0.25), (15, 0.25), (21, 0.25)
13	(9, 0.25), (15, 0.5), (16, 0.25), (17, 0.25), (21, 0.25)
14	(15, 0.25), (16, 0.5), (17, 0.25), (18, 0.5)

Since Δ is set as 0.4, we can see nodes {1, 2, 6, 9, 15, 16} can be successfully influenced by the community induced by the node set {3, 4, 5, 13, 14}. Besides those, nodes {17, 21} can also be influenced based on the aggregated influence, i.e., node 17 (or 21) can be influenced with probability $1-(1-0.25)(1-0.25)=0.44$ by nodes 13 and 17 (or nodes 5 and 13). Therefore, the influence of the community can be scored as eight.

Problem statement. Given a social network $G=(V,E,P)$, and an integer k, our target is to find the maximal kr-clique community C satisfying:

$$\arg\max_{V(C)\subseteq V(G)} Score(C) \tag{2.3}$$

subject to

$$|V(C)| \geq k, \forall u,v \in V(C) \quad |sp(u,v)| \leq r$$

where $|V(C)|$ is the number of vertices in C, $|sp(u,v)|$ represents the length of the shortest path between the vertices u and v, and r is a system setting parameter. Based on our statistic on the datasets as shown in Figure 2.3, most communities only contain less than 10 nodes when r = 1and the duplicate ratio between the nodes in any two communities is nearly 95% when $r=3$. It only makes much sense for r to take the value of 2 in such settings. It is meaningless to ask users to specify this parameter, which is demonstrated in our experiments - Figure 2.3.

Therefore, the target of this work is to search the maximal kr-cliques that have the maximum outer influence score from the given social network. We refer the problem as *most influential community search* in this paper. The returned result may be more than one most influential communities if there are several maximal kr-cliques where their maximum outer influence scores are same. As discussed in [Feige 2004, Wang et al. 2016b], the maximal clique problem (MCP) has been proved to be an NP-hard problem.

Our most influential community search problem is more complicated and thus difficult to solve because we not only need to identify the maximal cliques with their size no less than k, but also computing their outer influenced node sets. Therefore, our problem in this paper is also an NP-hard problem.

2.2 Baseline Solution

To address the most influential community search problem, the basic solution is to first sort the nodes in $V(G)$ by their *individual influence score* and then check the sorted nodes one by one in the descending order. For each node, we need to compute its maximal kr-cliques and calculate the outer influence score of each found maximal kr-cliques. After all the nodes have been checked, we compare their outer influence scores and return the maximal kr-clique with the maximum score as the most influential community.

Algorithm 1 Basic Solution

Input: A social graph $G = (V, E, P)$ and an integer k
Output: A set $\mathbb{C}$ of *Most* influential maximal kr-clique *communities*

```
1: Offline compute the influence score of each node in V;
2: Get a list L with the nodes sorted by their influence scores;
3: Transform G to G_r using the system setting r;
4: for all u ∈ L do
5:    Compute the maximal kr-cliques containing u from G_r;
6:    Calculate the outer influence score of each maximal kr-clique;
7:    Update ℂ by writing the maximal kr-cliques with the maximum score;
8:    next_k_score ← add the next k nodes' individual influence score in L;
9:    if the score in ℂ > next_k_score then
10:      return ℂ;
11:   end if
12: end for
13: return ℂ;
```

The baseline algorithm is presented in Algorithm 1. Here, the influenced node list and their score for each node have been offline computed and maintained. As such, the time complexity of the algorithm mainly consists of the following parts. The first part is to list all maximal cliques that would be $O(3^{n/3})$ using the state-of-the-art algorithm in [Tomita et al. 2006]. Based on the community size distribution as shown in Figure 2.3, the number of generated k-communities would be in the complexity of $O(log_k^n)$. To evaluate the *outer influence score of each community*, we need to find the influence path from every community node to the outer nodes as discussed in [Lee and Chung 2015]. The computational complexity would be $O(kn)$. Therefore, the least time complexity of Algorithm 1 would be $O(3^{n/3} + knlog_k^n)$. From the analytics, we can see that the main bottleneck of the algorithm is the maximal clique generation part and the second main cost is spent on the outer influence computation part. This two challenging parts are addressed by the our designed tree-based index and advanced algorithms, respectively.

2.3 Index-based Influential Community Search

In this section, we first propose a tree structure, called C-tree, to index maximal r-cliques induced from $V(G)$. We show that C-tree is compact and any kr-cliques can be generated efficiently by using the C-tree.Then we propose four efficient search algorithms based on C-tree, which are *sequential-order based* (SO) search, *improved sequential-order based* (SO+) search, *best-first based* (BF) search, and *fast best-first based* (BF+) search. The first two algorithms evaluate the community candidates supported by C-Tree in a sequential order. They can avoid some common nodes' influence computation. The last two algorithms probe the community candidates in a priority of their upper bound scores, by which they can filter much more nodes in their influence computation.

2.3.1 Indexing Maximal r-Cliques

We can pre-store all maximal r-cliques, which can avoid the online computation of maximal r-cliques in baseline solution. However, if we simply store all these maximal cliques as the maximal kr-clique influential community candidates to be searched by users, the space cost would be expensive because generally there are a large number of maximal cliques in a social network. But we know, these maximal cliques may contain lots of overlaps or duplicate nodes. Therefore, to effectively maintain the communities and support efficient search, we propose an effective tree index based on the relations of the communities, which is called as *C-Tree*. Our tree-based index can also significantly reduce the computational cost because it can reduce a large number of influence evaluation for the duplicate nodes among overlapped communities.

Definition 4. (C-Tree) Given a social network G and a system setting parameter r, we have all its maximal r-cliques $\mathcal{C} = \{C_1, C_2, ..., C_{n_c}\}$ that may contain many duplicate nodes. The C-Tree $T(\mathcal{C})$ of $\mathcal{C}$ is defined as:

- The root of $T(\mathcal{C})$ is a virtual node;
- The node in $T(\mathcal{C})$ is a node set or a single node. An internal node is the maximum intersection of all the maximal r-cliques in the subtree rooted at the internal node. For any leaf node, it does not have intersection with its siblings;
- Each path from the root of $T(\mathcal{C})$ to a leaf node is a maximal r-cliques community.

Figure 2.2 presents the C-Tree for the small social network in Figure 2.1. For instance, the community induced by {3, 4, 5, 13, 14} can be easily identified from the most left path in Figure 2.2.

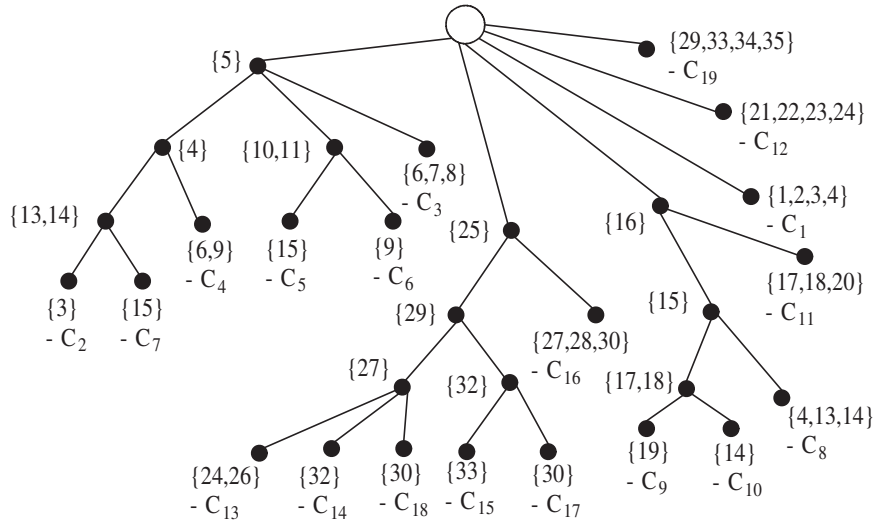


Figure 2.2: Example of C-Tree

2.3.2 Sequential-Order based Search

Since each path from the root to the leaf node represents a community, we can directly access each path and find all the nodes in the corresponding community of the path. Obviously, some nodes will be re-accessed in many times because they are shared by multiple communities. To avoid this, we propose a stack-based algorithm to access the nodes in the C-Tree index, which maintains the shared nodes in a stack until all the communities containing them are dealt with. As such, we can retrieve each community from the index tree in the depth-first-order and calculate its influential score based on Definition 3. Therefore, we call the community search method as *Sequential Order based* search, denoted as *SO*.

The detailed procedure is presented in Algorithm 2. Here, we utilize two stacks to implement the depth-first tree traversal. The tree traversal starts from the root node (i.e., v_{root}) of the index tree T. At Line 5, we get the node u at the top position of Stack A. And then we check if u can be added into Stack B, as shown in Line 7-Line 20. If u is a child of the top node in B and it is also an internal node, i.e., u.children() $\neq$ null, then we just explore u's children nodes in the tree T, as shown in Line 8-Line 11. Here, the node set in Stack B is only a subset of a community or multiple communities. If u has been identified as a leaf node, i.e., u.children()==null, then we can see that the nodes in B and node u are on the same path, i.e., it is a complete community C_{temp} based on Definition 4. At the moment, we also check the community size. When the size of the community C_{temp} is no less than the required size k, we compute its influential Score(C_{temp}) by calling Function Computing_Score(C_{temp}) and record the more influential community $\mathbb{C}$ via comparison, which is described in Line 13-Line 17. When u is not a child of the node $B.top()$, it says that all communities containing the node $B.top()$ have

Algorithm 2 SO Algorithm

Input: C-Tree T of a social network G w.r.t. r, an integer k
Output: A set $\mathbb{C}$ of *Most* influential maximal kr-clique *communities*

```
1: New two stacks A and B, set Score(C_max) as zero;
2: Get the root node v_root of the C-Tree T;
3: A.push(v_root);
4: while A is not empty or B is not empty do
5:    u = A.pop();
6:    if B is not empty then
7:       if u is a child of B.top() and u.children()≠null then
8:          B.push(u);
9:          for v ∈ u.children() do
10:            A.push(v);
11:         end for
12:      else if u is a child of B.top() and u.children()==null then
13:         if B.size≥ k-1 then
14:            C_temp = B ∪ {u};
15:            Score(C_temp) = Computing_Score(C_temp);
16:            ℂ ← C_temp if Score(C_max) < Score(C_temp);
17:         end if
18:      else
19:         B.pop();
20:      end if
21:   else
22:      B.push(u);
23:      for v ∈ u.children() do
24:         A.push(v);
25:      end for
26:   end if
27: end while
28: return ℂ;
```

been processed. In this case, we only need to pop it out from Stack B. The algorithm will be terminated until all communities in the index tree are visited. At the end, the most influential community $\mathbb{C}$ will be returned.

2.3.3 Improved Sequential-Order based Search

In this section, we propose two upper bound based pruning properties, which can improve the efficiency of sequential-order based search by pruning more insignificant community candidates, which is denoted as *SO+*.

Before providing the first upper bound, let's review the important property in influence maximization problem. This is because the influence of a community in this work is calculated by the maximum influence spread of the community minus the community members. The following property has been proved in many works [Kempe et al. 2003b, Chen et al. 2012; 2009].

Property 1. The influence maximization $f()$ is a monotonic and submodular function. I.e., for any two sets S and X, $f(S \cup X) + f(S \cap X) \leq f(S) + f(X)$ always holds.

Now, we can prove Property 2. Consider two communities C_1 and C_2. Their maximum common part is denoted as $C_{1,2} = C_1 \cap C_2$. To be simple, we let $S_1 = C_1 \setminus C_{1,2}$, $S_2 = C_2 \setminus C_{1,2}$, and $X = C_{1,2}$. Our target is to look for the condition that guarantees $f(S_1 \cup X)$ - $(S_1 \cup X)$ is equal to or larger than $f(S_2 \cup X)$ - $(S_2 \cup X)$ with less computation.

Property 2. (Tight Upper Bound) If $f(S_1|X)$ - $S_1 \geq f(S_2)$-S_2, then we can directly conclude that $f(S_1 \cup X)$ - $(S_1 \cup X) \geq f(S_2 \cup X)$ - $(S_2 \cup X)$ without computing $f(S_2 \cup X)$. That is to say, C_1 is more influential than C_2. That is to say, we only maintain C_1 as the most influential community candidate w.r.t. C_2.

Proof. Since X is a subset of S_1, there is no overlap between S_1 and X. So $S_1 \cup X$ can be rewritten as $S_1 + X$. $f(S_1 \cup X)$ - $(S_1 \cup X)$ - $[f(S_2 \cup X)$ - $(S_2 \cup X)]$ can be rewritten as $f(S_1 \cup X)$ - $(S_1 + X)$ - $[f(S_2 \cup X)$ - $(S_2 + X)] = f(S_1 \cup X)$ - S_1 - $f(S_2 \cup X) + S_2$. Based on Property **??**, we have $f(S_2 \cup X) \leq f(S_2) + f(X)$ - $f(S_2 \cap X) = f(S_2) + f(X)$ where $f(S_2 \cap X)$ is zero because $S_2 \cap X$ is null. Therefore, we have $f(S_1 \cup X)$ - S_1 - $f(S_2 \cup X)$ $+ S_2 \geq f(S_1 \cup X)$ - S_1 - $(f(S_2) + f(X)) + S_2 = f(S_1 \cup X)$ - $f(X)$ - S_1 - $f(S_2) + S_2 = f(S_1|X)$ - S_1 - $(f(S_2)$-$S_2)$, which satisfies our pre-condition. □

Property 3. (Loose Upper Bound) Consider any set of nodes, e.g., $S_2 = \{u_{ij}\}$ and the reachable node set $R(u_{ij})$ of node u_{ij}. The upper bound of the influence $(f(S_2) - S_2)$ for S_2 is $|\{\cup R(u_{ij})\} \setminus C_i|$, denoted as $\mathrm{UB}(S_2)$.

According to Property 2 and Property 3, sometimes we do not need to evaluate the influence spread of the vertice for a whole community. Based on the partial evaluation, we can filter more insignificant community candidates.

The detailed procedure of the improved sequential-order based search is presented in Algorithm 3. Different from Algorithm 2, Algorithm 3 will check the pruning conditions, i.e., the loose upper bound and the tight upper bound. If the new generated community can be filtered by using the upper bounds, then we don't need to compute its influential score. The efficiency can be improved due to the following two main reasons: (1) if we know some nodes can be activated above a threshold by $C_{temp} \setminus X$, these nodes must be successfully activated above the same threshold by its super-set C_{temp}. Therefore, we can exclude these nodes from the computing process of Computing_Score(C_{temp}); (2) Even if we cannot filter the community C_{temp} by the partial result of Computing_Score($C_{temp} \setminus X$), the intermediate results of Computing_Score($C_{temp} \setminus X$) can continuously be used to make calculation for Computing_Score(C_{temp}). We don't need to do Computing_Score(C_{temp}) from scratch.

Algorithm 3 SO+ Algorithm

```
1: The codes same to Line 1-Line 13 in Algorithm 2;
2: if B.size≥ k-1 then
3:    C_temp = B ∪ {u};
4:    if Score(ℂ) ≥ UB(C_temp) then
5:       Do nothing;
6:    else
7:       for all C_max ∈ ℂ do
8:          X = C_max ∩ C_temp;
9:          if Score(C_max) ≥ Computing_Score(C_temp \ X) then
10:            Do nothing;
11:         else
12:            Score(C_temp) = Computing_Score(C_temp);
13:            ℂ: C_max ← C_temp if Score(C_max) < Score(C_temp);
14:         end if
15:      end for
16:   end if
17: end if
18: The codes same to Line 18-Line 27 in Algorithm 2;
19: return ℂ;
```

2.3.4 Best-First based Search

The performance of SO+ is very sensitive to the influential score of the first probed community, it has to evaluate the communities in the C-Tree in a sequential order. If the first community is lucky to be the most influential one, then it can improve the efficiency a lot. Otherwise, it cannot prune any community, e.g., when the most not-influential one appears at the first place. To effectively access C-Tree with a "good" order, we develop an efficient best-first search algorithm (denoted as *BF*) in this section. From Property 3, we know that for node u_{ij} in a community C_i and the reachable node set $R(u_{ij})$ of u_{ij}, the influence for C_i is bounded by $|\{\cup R(u_{ij})\} \setminus C_i|$, denoted as $\text{UB}(C_i)$.

Lemma 1. *Assume we have computed the influence $Score(C_1)$ of a community C_1 based on Definition 3 at one moment. For any other communities C_i that have not been visited, we can skip C_i without computation if $Score(C_1) \geq UB(C_i)$.*

The key idea of the BF approach is to find all community candidates with their size no less than k from the C-Tree index. And then it sorts these communities by their upper bound values and maintain them in an ordered queue Q. Every time, we pop the community candidate C_x from the top position of Q, which should have the highest upper bound value. We then calculate its influential score. If the computed score of C_x is no less than the upper bound of the next community in Q, then we can say C_x would be the most influential community. Otherwise, we need to use the calculated real score to replace the upper bound value of C_x, and add it back to Q. The process will be repeated until the most influential community is found.

Algorithm 4 BF Algorithm

Input: C-Tree T of a social network G w.r.t. r, an integer k
Output: A set $\mathbb{C}$ of *Most* influential maximal kr-clique *communities*

```
1: New two stacks A and B, and new a queue Q;
2: The codes same to Line 2-Line 13 in Algorithm 2;
3: if B.size≥ k-1 then
4:    C_temp = B ∪ {u};
5:    C_temp.ub = UB(C_temp);
6:    C_temp.score = null;
7:    Add C_temp into Q and sort the communities by upper bounds;
8: end if
9: The codes same to Line 18-Line 27 in Algorithm 2;
10: found = false, ℂ = null;
11: repeat
12:    C_x = Q.pop();
13:    if C_x.score is not null then
14:       ℂ ← C_x;
15:       found = true;
16:    else
17:       C_x.score = Computing_Score(C_x);
18:       if C_x.score ≥ Q.top().ub then
19:          ℂ ← C_x;
20:          found = true;
21:       else
22:          C_x.ub = C_x.score;
23:          Add C_x back to the ordered queue Q;
24:       end if
25:    end if
26: until found
27: return ℂ;
```

The detailed procedure is presented in Algorithm 4. In Line 1-Line 9, we access the C-Tree index and generate an ordered queue containing all the community candidates with their size no less than the parameter k. In Line 10-Line 26, we check the community candidates based on their upper bound values in the queue. For a probed community candidate C_x, we compute its influential score at Line 17. Line 9 is used to update the upper bound value of a community using its real influential score that has been computed.

2.3.5 Fast Best-First based Search

To further accelerate the efficiency of community search, in this section we develop a more fast best-first based approach, which is denoted as *BF+*.

The key idea of *BF+* is to maintain a pre-computed list. The list contains the C-Tree leaf node IDs where each leaf node ID represents the corresponding community of the path from root to the leaf node. And the list is sorted by the upper bound value of the influence of the community. We compute and sort the list through offline computation. When a user issues a community search query by specifying the parameter k, we first

access the list in the ascending order. For each leaf node, we can locate its corresponding community by traversing the path from the tree root to the leaf node in the C-Tree. Thus, the first visited community should have the highest upper bound value on the influence. And then, we calculate the real influential score of the community if the visited community contains no less than k members. Otherwise, we discard the community and probe the next community in the ordered list. After that, we measure the real influential score of the community with the upper bound value of the next community. If the real influential score of the community is no less than the upper bound value of the next one, then the current community is the most influential community, i.e., the algorithm can be terminated. Otherwise, we use the real score of the community to replace its upper bound value. The correctness has been proved in Lemma 1. We repeat the above procedure until the termination condition is true. The algorithm is similar to the procedures in Line 10-Line 26 in Algorithm 4.

2.4 C-Tree Index Construction

Now we show how to efficiently construct C-Tree index. We first show the procedure of enumerating all the maximal cliques for a social graph by revisiting the work in [Tomita et al. 2006]. And then, we present the algorithm of index construction and discuss the complexity of building and maintaining the C-Tree index.

2.4.1 Revisiting Maximal Cliques Enumeration

To generate maximal r-cliques from a graph G, we first transform the graph into r-hop based graph G_r where we create a direct edge for any two nodes if their shortest distance is no more than r in G. Then, we recursively call a function Clique_Generation to produce the maximal r-cliques. We implement Clique_Generation by revisiting the state-of-the-art algorithm for maximal clique enumeration proposed in [Tomita et al. 2006]. Its efficiency has been further verified in [Eppstein et al. 2010].

The detailed procedure is presented in Algorithm 5. In Function Clique_Generation, it maintains three node sets: R, P and X. The nodes in R form a partial clique to be expanded. P contains the nodes that are adjacent to all the nodes in R in the new graph G_r and are potential candidates to be added to the maximal clique. X contains the nodes such that (1) they are adjacent to all the nodes in R in the new graph G_r , and (2) they have been traversed to form maximal cliques in any previous level of recursion. Adding any node in X to the current partial clique R will result in duplicate cliques. Therefore, nodes in X must be excluded from R. P and X together cover the nodes that are adjacent to all nodes in R in the new graph G_r. When both P and X become empty (Line 4), R cannot be further expanded. Thus, we output R as a maximal clique (Line 5-Line 7). To make sense, we are only interested in the communities with one or two nodes. If R

Algorithm 5 EnumkrMC()

Input: A social graph $G = (V, E)$ and a system setting parameter r
Output: maximal r-clique set $\mathcal{C}$

1: Transform G to G_r regards r;
2: $\mathcal{C} \leftarrow$ Clique_Generation(ϕ, V, ϕ);
3: Function Clique_Generation(node set R, node set P, node set X)
4: **if** $P == \phi$ and $X == \phi$ **then**
5: **if** $|R| \geq 3$ **then**
6: add R as a maximal clique into $\mathcal{C}$;
7: **end if**
8: **else**
9: $u \leftarrow argmax_{v \in P \cup X}\{|P \cap N_r(v)|\}$;
10: **for all** $v \in P \setminus N_r(u)$ **do**
11: num = num$_{ini}$;
12: Clique_Generation($R \cup \{v\}$, $P \cap N_r(v)$, $X \cap N_r(v)$);
13: $P \leftarrow P \setminus \{v\}$;
14: $X \leftarrow X \cup \{v\}$;
15: **end for**
16: **end if**
17: **return** $\mathcal{C}$;

can be further expanded, then we recursively adds a node from the candidate nodes in P to expand the current partial clique R. Each time a vertex v is added into R, we refine P and X by keeping only the nodes that are also adjacent to v (Line 12) and invoke the function Clique_Generation recursively. P and X are updated after each recursive call (Line 13-Line 14). The pivot node u in Line 9 is used to reduce the computational cost of Clique_Generation. In principal, every node in P or X can be chosen as a pivot node. Here, we choose the one with the maximum number of neighbors in P because such a pivot node is shown to be computation effective in [Eppstein et al. 2010, Yuan et al. 2015].

2.4.2 Algorithm of Constructing C-Tree

Theorem 2. *Constructing C-Tree is an NP-Hard problem.*

Proof. Consider we have a community tree T. Constructing C-Tree can be equivalently transformed to a minimization problem, i.e., minimizing $\sum_{n_1,n_2 \in T} |n_1 \cap n_2|$ with regard to $\cup_{n_i \in T}\{n_i\} = V(G)$ and $\cup_{v_j \in path(n_i)}$ is a community. Therefore, the problem of constructing C-Tree can be reduced to the generalized assignment problem that is a NP-hard problem. □

By reviewing the procedure of generating maximal cliques in [Tomita et al. 2006], we can see the maximal cliques are generated in the recursive process and the nearly generated cliques may have high chance to be overlapped. As such, the overlapped part is the frequent node set in the local social network portion covering the nearly maximal cliques. However, it cannot provide direct help for us because the sequence of generated

cliques are very sensitive to the selection of the nodes to be probed in the procedure of clique generation [Tomita et al. 2006]. We can not assume it can probe the best node every time.

To address this challenge, we propose a novel technique to identify the local-frequent nodes, which is then used to construct the C-Tree with regards to a set of communities.

Definition 5. (Node Duplicate Frequency NDF) Consider all the maximal r-cliques $\mathcal{C} = \{C_1, C_2, ..., C_{n_c}\}$ in a social network G. For any node $u \in C_i$, its node duplicate frequency $NDF(u)$ is the number of communities in $\mathcal{C}$ that contain node u.

Property 4. (Local Most Duplicate Node) Given a subset of communities $\mathcal{C}_{sub} \subseteq \mathcal{C}$, the local most duplicate node u in $\mathcal{C}_{sub}$ is the node with the maximum frequency occurring in $\mathcal{C}_{sub}$, denoted as $\text{NDF}_{\mathcal{C}_{sub}}(u)$.

The above property is easy to be concluded. We do not prove it in this paper. Assume we maintain the community information by using node list data structure, i.e., community C_i: members $u_1, u_2, ...$ For each community, we can sort its node list based on the NDF score of the nodes. After that, we can first identify the most duplicate nodes via the NDF scores of nodes. In the other words, if a node is highly duplicated in communities, then the node will appear at the top position in their community node lists. By doing this, we can select the node with the most duplicates and add it as an internal node into the C-Tree. We then exclude the node from the related communities and deal with these communities with the remaining nodes under the subtree rooted at the added internal node in C-Tree. For the other communities that do not contain the node at their top positions, we will deal with them based on their shared nodes at their top positions in a similar way. At the next iteration, we re-calculate the local most duplicate node for each branch of communities based on Property 4. The local most duplicate node is taken as a new internal node grouping such communities. As such, the C-Tree can be constructed level by level, while the duplicates can be maximally reduced. This is because the local most duplicate nodes have high chance to be shared by multiple communities.

Example 3. (C-Tree) Figure 2.2 presents the C-Tree for the small social network in Figure 2.1 based on Property 4. Here, we only maintain the communities with no less than four members. Firstly, since node 25 in Figure 2.2 has the most duplicate frequency, we create a new internal node {25} including the communities {C_{13}, C_{14}, C_{18}, C_{15}, C_{17}, C_{16}}. And then, we deal with node 5 in Figure 2.2 due to its high duplicate frequency, which results in a new addition of node {5}. The new internal node includes the communities {C_2, C_7, C_4, C_5, C_6, C_3}. Similarly, we can calculate the duplicate frequencies of the nodes and select the new internal nodes for the remaining communities. After that, the first level of the tree is constructed. In the following iterations, we can repeat the above procedures for each subset of communities and build the tree level by level. The

construction algorithm can be terminated until there is no duplicate among sibling nodes in the tree.

Algorithm 6 CTree()

Input: Maximal r-clique community set $\mathcal{C} = \{C_1, C_2, ..., C_{n_c}\}$ and a tree root v_{root}
Output: a C-Tree T

1: **while** $\mathcal{C}$ is not empty **do**
2: Calculate the NDF score for each node in $\mathcal{C}$;
3: Sort the nodes in each community by their NDF scores;
4: Get node u where $NDF_{\mathcal{C}}(u) \geq NDF_{\mathcal{C}}(u'|u' \in \backslash u)$;
5: Add u as a child node into v_{root} in T;
6: Get the subset $\mathcal{C}_u$ of communities with u at their top positions and remove u from $\mathcal{C}_u$; {Incur a recursive process for each subset of communities}
7: u.CTree($\mathcal{C}_u$, u);
8: $\mathcal{C} \leftarrow \mathcal{C} \setminus \mathcal{C}_u$;
9: **end while**
10: **return** The tree T with the root node v_{root};

The detailed procedure is presented in Algorithm 6. The key idea of this algorithm is to select the most duplicate node u for classifying the communities to be processed. Obviously, the communities can be splitted into two sets: one set of communities contains u at their top positions, and another set of communities does not. For the previous one set, we take u and add it as a child node of the current root, as shown in Line 5. And we remove u from the set $\mathcal{C}_u$ of communities. After that, we re-call a recursive process to deal with the new community subset $\mathcal{C}_u$ where u is taken as the new root. For the latter set, i.e., $\mathcal{C} \setminus \mathcal{C}_u$, we repeat the steps in Line 2-Line 8. When all subsets become empty, the whole recursive algorithm can be terminated. At the end, the C-Tree can be obtained via the root node v_{root}.

2.4.3 Optimizing C-Tree

To further reduce the duplicate nodes in C-Tree, we develop an iterative procedure to optimize the C-Tree. The key idea is to check the nodes with high duplicate frequency in the C-Tree built in Section 2.4.2 and identify each node that the optimal benefit of changing it is larger than the cost. For each node u to be checked, its optimal benefit can only be calculated based on the branches containing node u.

Example 4. (C-Tree Optimization) Take node 4 in Figure 2.2 as an example. We can see the related branches are C_2, C_7, C_4, C_8 and C_1 in Figure 2.2. If we group these branches based on node 4, then the optimal benefit is that we can reduce two duplicates of node 4 (coming from C_8 and C_1) and reduce one duplicate for nodes 13, 14, 15. Therefore, the total optimal benefit of changing node 4 is weighted as 5. Note that the optimal benefit is only the upper bound value of changing the node. Similarly, we also need to calculate the cost based on the branches containing node u. Consider node 4 in Figure 2.2 as an

example again. If we group these branches based on node 4, then the cost is that we need to bring new duplicates for nodes 5, 15 and 16 because they are the ancestors of the tree nodes containing node 4. Therefore, the cost of changing node 4 is weighted as 3. Since the optimal benefit is larger than the cost, we will try to change the C-Tree by adjusting the position of node 4. If the exact benefit is still larger than the exact cost, then we update and generate a new version of C-Tree. Otherwise, we still keep the current version of the C-Tree.

Algorithm 7 CTreeOptimization()

Input: A C-Tree T
Output: An optimized C-Tree T'

1: $S \leftarrow$ Get the nodes with duplicates in T;
2: **while** S is not empty **do**
3: $u = argmax_{v \in S}\{Benefit(v,T) - Cost(v,T)\}$;
4: **if** $Benefit(u,T) - Cost(u,T) > 0$ **then**
5: $T' \leftarrow Refine(T,u)$;
6: $S' \leftarrow$ Get the nodes with duplicates in T';
7: $S = S' \cap S$;
8: **end if**
9: $S = S \setminus \{u\}$;
10: **end while**
11: **return** The refined tree T';

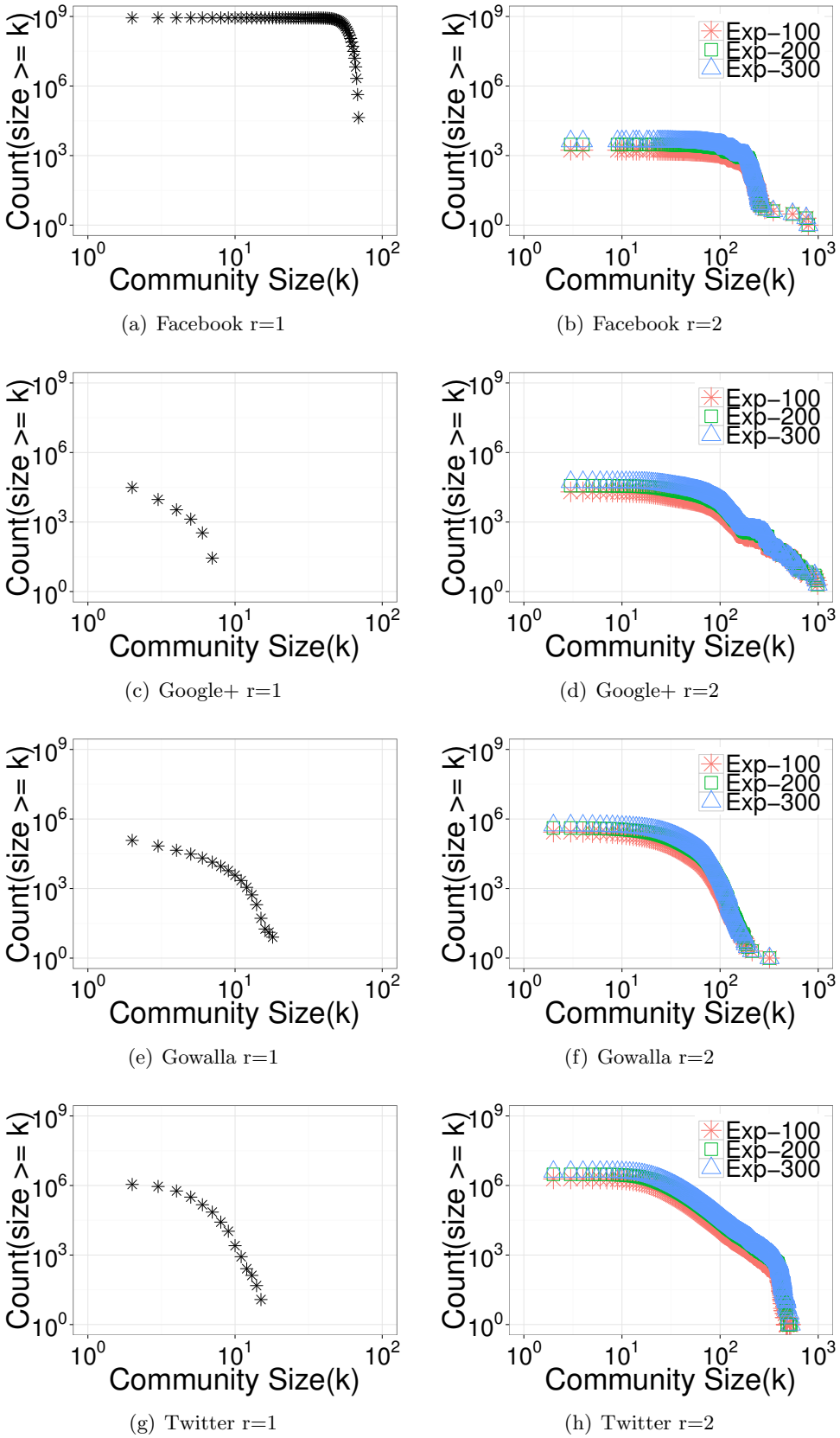
Algorithm 7 presents the detailed procedure of optimizing C-Tree. Here, we use a candidate set S to maintain the nodes that need to be refined. At the beginning, S only contains the nodes with duplicates in the generated C-Tree T. We will reduce the candidate set S through iterations. At each iteration, we select node u that can bring the maximal gain to the tree refinement (Line 3). We refine the tree T into T' based on the adjustment of node u (Line 5). At the next iteration, we only need to consider the nodes that appear as the duplicate nodes in T' and T (Line 6-Line 7). The algorithm can be terminated until the candidate set becomes empty.

2.5 Experimental Study

All experiments have been conducted on a Red Hat Enterprise Linux Server (7.2), with 792GB RAM and Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00GH shared by the school of science, RMIT. The algorithms are implemented using Python 2.7. The average experimental results of 200 runs with different data inputs at the same settings are reported.

2.5.1 Data Sets and Parameter Settings

To evaluate the proposed new community model, index and search algorithms, we selected four real social network datasets of different sizes. The basic information is shown in



(a) Facebook r=1 (b) Facebook r=2

(c) Google+ r=1 (d) Google+ r=2

(e) Gowalla r=1 (f) Gowalla r=2

(g) Twitter r=1 (h) Twitter r=2

Figure 2.3: Community Size Distribution.

Datasets	#nodes	#Edges	**A**vg Degree
Facebook	4,039	176,468	43.69
Google+	23,628	78,388	3.32
Gowalla	69,097	351,452	5.09
Twitter	486,879	4,293,610	8.82
Youtube	52,675	636,864	12.10
Amazon	317,194	1,745,870	5.50

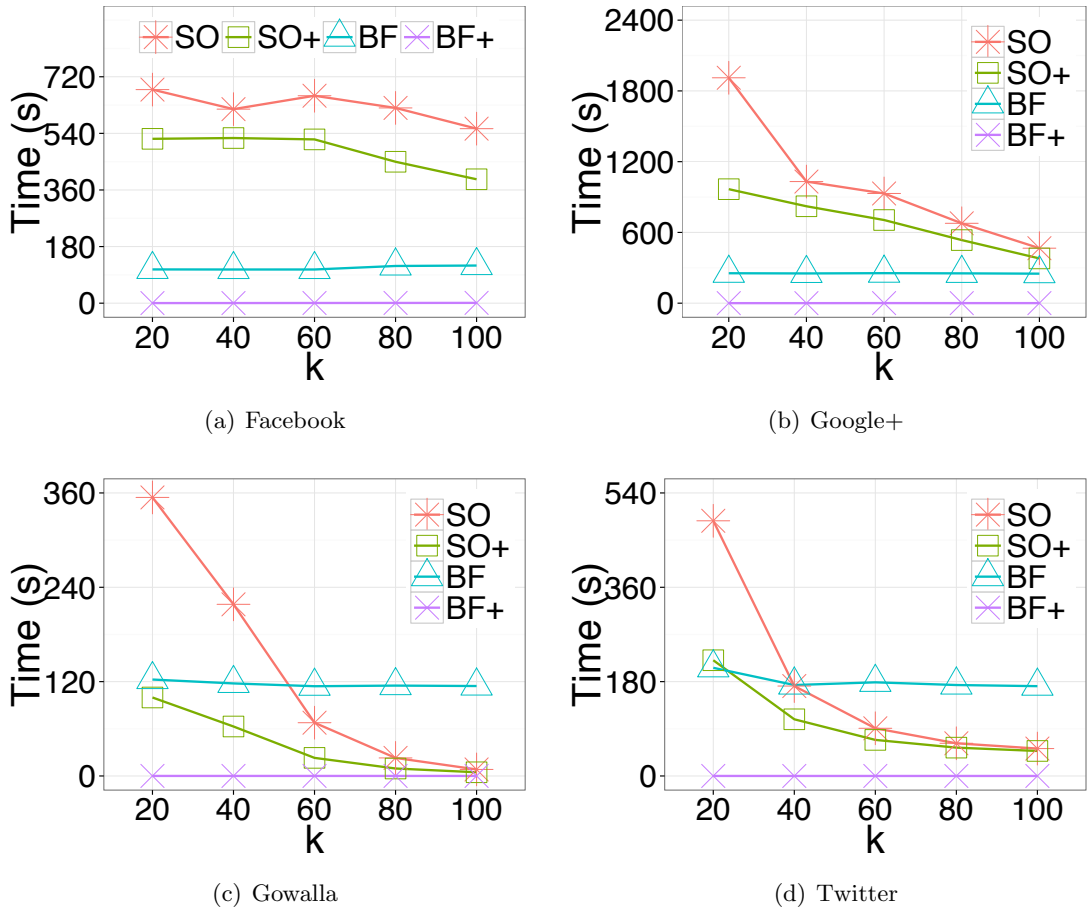
Table 2.1: Statistical Information of the Datasets

Table 3.1. Figure 2.3 illustrates the distribution of community size where x-axis is the community size and y-axis indicates the number of communities whose size is greater than the corresponding value in x-axis. For each dataset, we have produced three sets of communities, denoted as Exp-100, Exp-200 and Exp-300, where each node is constrained to be included in at most 100, 200 and 300 communities respectively. When $r = 1$, the size of all communities is less than 80. When $r = 2$, the size of most communities (99%) is in between 100 and 500. The proposed methods aim to handle the complex situation where the search space consists of a large number of large communities. So, the experiments focus on the situation when $r = 2$.

2.5.2 Efficiency Evaluation of Influential Community Search

In this subsection, we test the efficiency of the proposed search algorithms (i.e., SO, SO+, BF and BF+) based on the C-tree ($k_{tree} = 20$). Figure 2.4 illustrates the test results when the community size $k = \{20, 40, 60, 80, 100\}$. When $k = 20$, it means the query concerns the communities of size no less than 20 only. The communities with smaller size (< 20) will be ignored. According to the test results in Figure 2.4, BF and BF+ are much faster than SO and SO+ for Facebook and Google+ datasets. This is because BF and BF+ first probe the community with the highest upper bound value. The probed community with the higher bound may have a relatively big influential score and thus it can be used to prune lots of communities with smaller upper bound. In contrast, SO and SO+ probe the communities maintained in C-tree in a sequential order. Interestingly, however, BF consumes the similar time as SO+ for Gowalla and Twitter. Look closely, this is because BF needs to sort all the community candidates first based on their upper bounds and then compute the most influential community. Different from BF, SO+ only scans the C-tree once.

As the improved version of BF, BF+ pre-sorts the communities in the C-tree by their



(a) Facebook

(b) Google+

(c) Gowalla

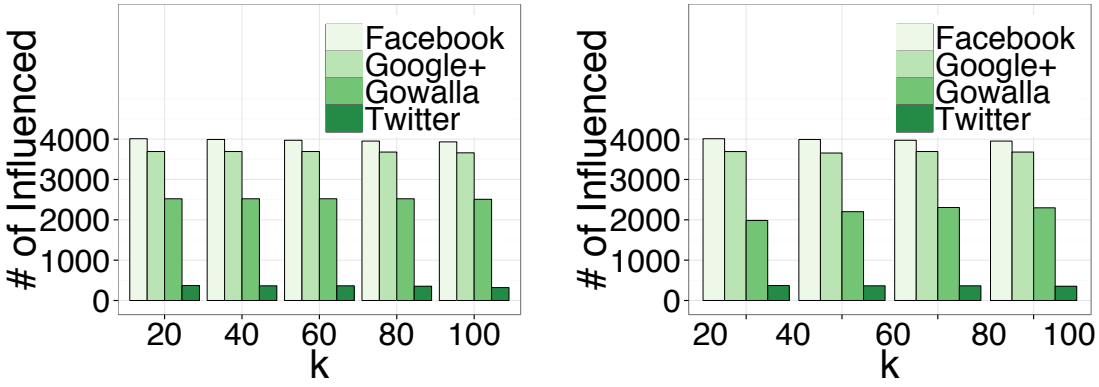
(d) Twitter

Figure 2.4: Search time.

upper bounds where each community is represented by a C-tree node ID. For each ID, we online obtain its corresponding community by accessing its subtree and its ancestor nodes in the C-tree. As a result, we have a sorted ID list to guide the search of the most influential community. The test results show that BF+ is much faster than BF in most cases. In particular, when $k = 20$, the time consumed by BF is about 622 times of that by BF+ for Facebook, 28 times for Twitter data, and 5.7-8.2 times for Google+ and Gowalla.

2.5.3 Effective Evaluation of Community Influence Spread

In this subsection, we conducted two studies on the influence spread of returned influential communities. The first study is to report the influence of the most influential community whose size is no less than a given parameter k (i.e., $\geq k$). The second study is to report the influence of the most influential community whose size is in a range $[k_1, k_2)$ (i.e., $\geq k_1$ and $< k_2$). In Figure 2.5, the test results for $k = \{20, 40, 60, 80, 100\}$ are presented in (a) and the results for four ranges [20,40), [40,60), [60,80) and [80,100] are presented in (b). Surprisingly, we notice that the numbers of nodes influenced at different settings of k and



(a) Normal Query (b) Range Query

Figure 2.5: Influential Scores over Different Datasets

$[k_1, k_2)$ are almost same. It shows that the most influential community can influence 3932-4010 persons in Facebook, 3661-3693 persons in Google+, 2509-2522 persons in Gowalla, and 320-371 persons in Twitter. This result delivers two important messages: (**i**) the influence spread of a community is very sensitive to the types of social networks; and (**ii**) the communities of smaller size can have the similar influence spread as the ones of larger size.

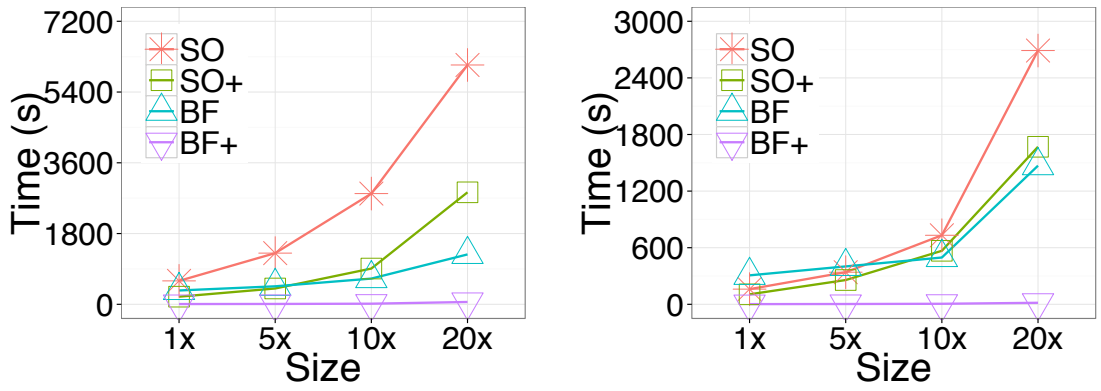
2.5.4 Effective Evaluation of maximal kr-clique Community Model

The recent study in [Wang et al. 2015] comprehensively assessed eight community detection models over different types of datasets. It concluded that LPA (Lable Propagation Algorithms) is the most reliable model in generating the desirable communities. In this subsection, we aim to verify the robustness of krMC community. For this purpose, we identify communities using LPA in the same datasets and treat those communities as the desirable communities. To check to which extent the krMC community are desirable, we compare our krMC communities against the communities using LPA based on *Normalized Mutual Information (NMI)*. The score of NMI stands for the agreement of two sets of results and is defined as

$$NMI = \frac{-2\sum_{i,j} N_{ij} log \frac{N_{ij}N_t}{N_{i*}N_{*j}}}{\sum_i N_{i*} log \frac{N_{i*}}{N_t} + \sum_j N_{*j} log \frac{N_{*j}}{N_t}}$$

N is the confusion matrix whose element N_{ij} is the number of the shared members between a krMC community C_i and a LPA community C_j. N_{i*} and N_{*j} are the sum over row i and column j respectively, and $N_t = \sum_i \sum_j N_{ij}$. The experimental results show NMI score can achieve about 99% for Twitter, 89% for Gowalla, 70% for Facebook and 63% for Google+. The higher scores tested in different datasets verify that our proposed krMC community model is reliable and desirable.

2.5.5 Efficiency Evaluation of Scalability



(a) k=50 (b) k=100

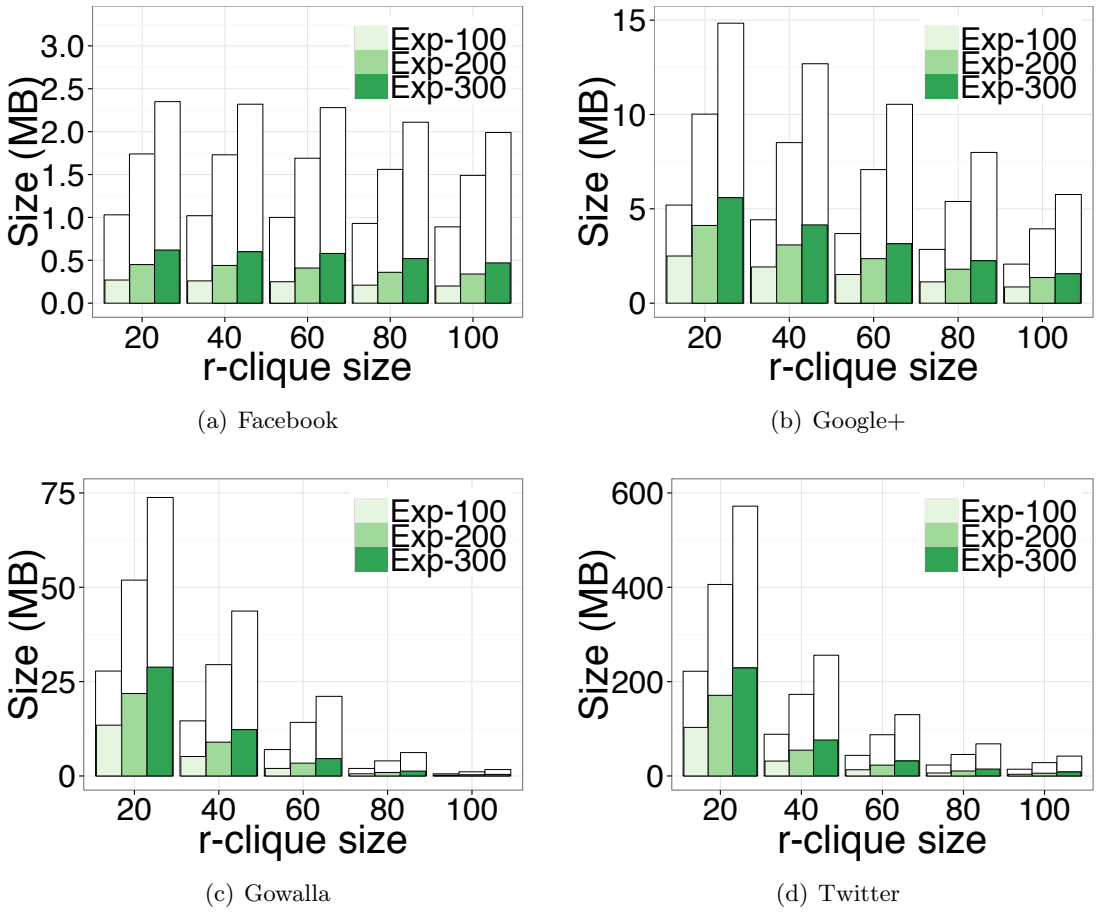
Figure 2.6: Scalability Evaluation

To further evaluate the performance of our proposed index and algorithms, We tested the time cost when we increased the data size by 5 times, 10 times and 20 times on Twitter dataset, as shown in Figure 2.6. When k is 50, the time cost of SO was increased linearly. But the increasing trend lines of the other three algorithms grow slowly. When k becomes 100, the three algorithms SO, SO+ and BF performed similary when the data size is 1 time, 5 times and 10 times, respectively. But when the data size is 20 times, SO+ and BF were much better than SO. In all the situations, BF+ performed the best in a stable status. This is because BF+ is able to filter lots of time-consuming community identification and influence spread evaluations.

2.5.6 Time and Space Cost Evaluation of Building C-Tree

Given a social network, the minimal duplicate community tree (C-tree) is generated for maintaining communities with less storage requirement. The key optimization to C-tree is to reduce the duplicates of nodes appearing in different communities. As discussed in section 2.4, C-tree is constructed using a greedy algorithm which processes nodes in the order of their duplicate level. A node has the higher duplicate level if it is the member of more communities. C-tree is further refined by giving the higher processing order to the node which has the relatively lower duplicate level but can more reduce duplicate nodes.

Figure 2.7 compares the storage requirements of maintaining communities with C-tree and without C-tree. The x-axis indicates the minimum community size that the C-tree supports, denoted as k_{tree}, i.e., only the communities containing more than k_{tree} nodes are retrievable from the C-tree; y-axis indicates the storage requirement with C-tree (color section of each bar) and without C-tree (the entire bar). In Figure 2.7, we observe that the storage requirement can be reduced about 75% for Facebook, 65% for Google+,



(a) Facebook

(b) Google+

(c) Gowalla

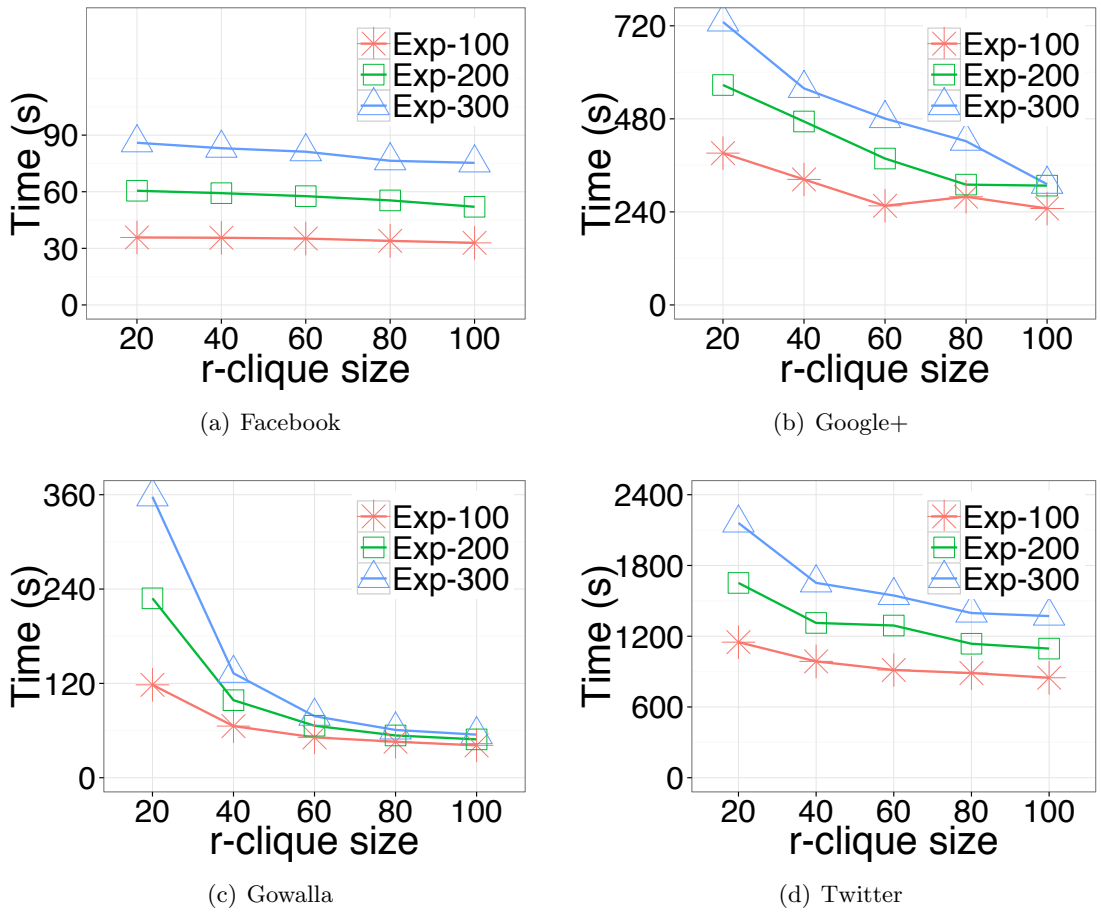
(d) Twitter

Figure 2.7: C-tree storage requirement.

66% for Gowalla and 65% for Twitter using C-tree. Clearly, the storage requirement decreases when k_{tree} increases because the quantity of communities decreases. Similarly, the quantity of communities decreases from Exp-300 to Exp-100 and thus the storage requirement decreases from Exp-300 to Exp-100. Less storage requirement means less computation time is requirement. Figure 2.8 reports the construction time of C-tree. When k_{tree} increases from 20 to 100 for a given social network, it consumes less time to construct C-tree.

2.5.7 Additional Evaluation

In this subsection, we tested the outer influence spread of communities by using two ground truth datasets - Youtube and Amazon. They have already presented the ground truth communities in the datasets. The evaluation tells us that how much percentage the ground truth communities can influence the outer node set that exceeds their own sizes by 1 times (1x), 5 times (5x), 10 times (10x) and 50 times (50x). The majority of the communities can influence their outer nodes between 1 time and 5 times. There are



(a) Facebook

(b) Google+

(c) Gowalla

(d) Twitter

Figure 2.8: C-tree construction time.

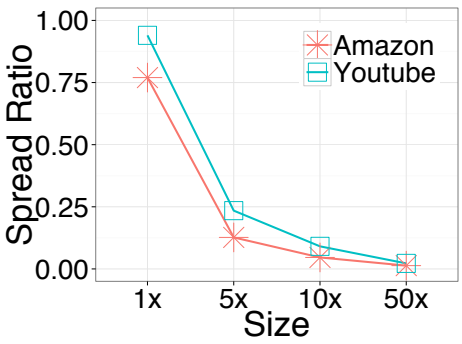


Figure 2.9: Influential Ratio in Ground truth Community Datasets

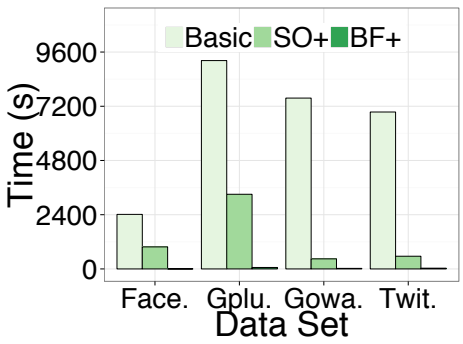


Figure 2.10: Time Cost of Basic vs. Index Methods

5%-12% communities that can influence 5-10 times of their community sizes. Only a few communities can influence up to 50 times. For SO and SO+, they have to check every maximal kr-clique communities, but for BF and BF+, it only checks a few communities. This is the reason that BF and BF+ performed much better than SO and SO+.

Since the basic algorithm needs to calculate the communities, it is clear for us to know that it is much slower than the other algorithms. To verify this, we compared the time cost of the basic algorithms with that of SO+ and BF+ over the four real datasets. Figure 2.10 reports their time costs when k is 20. The basic algorithm is slower than SO+ by about 5-7 times on Gowalla and Twitter, about 1.5 times on Google+ and Facebook. This is mainly because Google+ and Facebook have more edges with high bidirected influence probability. In this situation, SO+ also needs to consume lots of time cost on the influence computation. As such, the acceleration of SO+ is just 1.5 times regards Basic. In all the datasets, BF+ outperformed the other two algorithms greatly, which is faster than Basic by about 200 times and SO+ by about 50-100 times.

2.5.8 Case Study

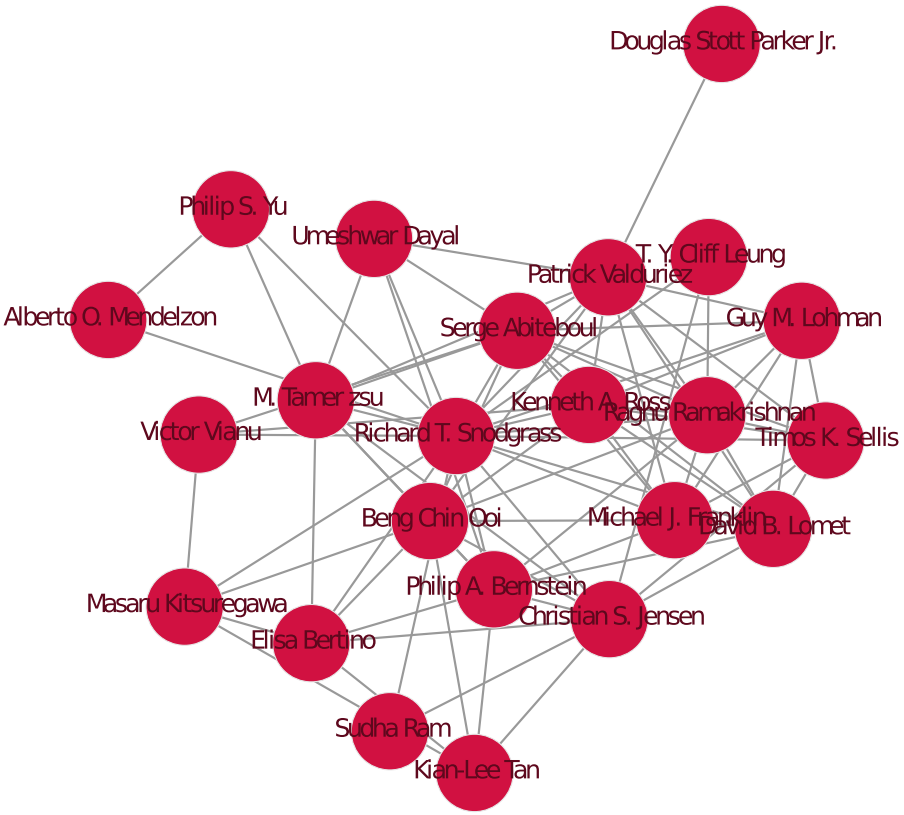
We conducted a case study based on the co-authorship network and citation network in database research areas[1]. Our study focuses on the 19,853 papers published by 22,250 authors in the top-10 DB conferences/journals (i.e., SIGMOD, VLDB, PVLDB, ICDE, EDBT, CIKM, TODS, VLDB J., SIGMOD Rec., and TKDE). The citations in 4,943 papers involves 4,558 authors. The influence of a researcher to another one is learnt by the citation relationships and the order of their names appearing in the cited papers.

Figure 2.11 presents the top-2 most influential communities where $k = 2$ and $r = 2$. The results illustrates that Philip S. Yu and Beng Chin Ooi have strong relation via M. Tamer Zsu, which cannot be detected by k-truss. The small case study also verifies that the most influential communities identified did report the collaborative groups with the well recognized influence in the real world. We highlighted the top-six contributors in each community. For instance, Philip A. Bernstein can independently influence 621 authors and Umeshwar Dayal can independently influence 515 authors, which brings benefit to Top-1 community; David J. DeWitt can independently influence 734 and Michael J. Carey can independently influence 722, which brings benefit to Top-2 community.

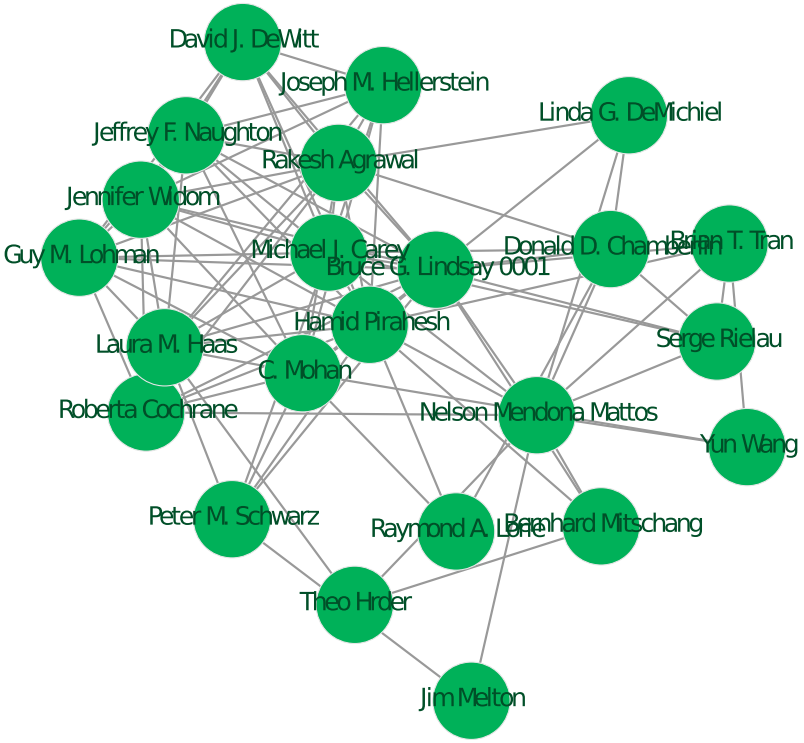
2.6 Conclusions

In this work, we investigated a novel and significant problem of searching the most influential maximal kr-clique communities in a large social network, which is a NP-hard problem and has various important applications in real life. This is the first work to discover communities via their outer influence. Compared with the existing community models,

[1] http://dblp.uni-trier.de/xml/



(a) Top-1



(b) Top-2

Figure 2.11: The top 2 influential co-authoring groups

such as k-core and k-truss, our proposed maximal kr-clique community model has more desirable characters. These modelling contributions provide a new aspect for users and companies to understand the communities and their influence in the social network at the community level. In addition, we has developed tailored C-Tree index and efficient search algorithms to enable the most influential community search in large social networks. The robustness of the maximal kr-clique community model has been verified through the case study of a real world application. The efficiency of the C-Tree index and search algorithms have been tested on six real world social networks.

CHAPTER 3

Efficient Diverse Influence Maximization

3.1 Problem Definition

3.1.1 Preliminary

A social network is modeled as a directed graph $G = (V, E, w)$, where a node in V represents one social media user, the edge (u, v) in E represents the (*follower*, *followee*) relationship, and $w_{u,v}$ represents the propagation probability along edge (u, v). Note edge (u, v) is directional, v is an out-neighbor of u and u is an in-neighbor of v. There are different diffusion models used to define influence propagation process. Without loss of generality, we adopt the Independent Cascade (IC) diffusion model [Chen et al. 2010, Kempe et al. 2003b]. Initially, every node is inactive. If a node u is selected as a seed, u becomes active and attempts to activate one of its inactive out-neighbors. The newly activated nodes will attempt to activate their inactive out-neighbors. Regardless of success or not, the same node will never get second chance to activate the same inactive out-neighbor. This process terminates when no more inactive nodes can be activated. In particular, we say a node v is successfully activated by a set S of seeds if and only if the overall influence from S to v is above a given threshold. In addition, the success of node u in activating out-neighbor v only depends on $P_{u,v}$ which allows us to evaluate the influence of seeds to other nodes via the maximum influence path [Liu et al. 2014].

In social networks, the subset of nodes $S \subseteq V$, which are active initially before influence propagation process, are known as *seeds*. Each seed spreads influence to inactive nodes in the social networks. For an inactive node v, we define the aggregated probability that v is activated by the seeds in S:

Definition 6. (Aggregated Influence Probability)

$$Pr(v|S) = 1 - \prod_{u \in S} (1 - Pr(p_{u,v})). \tag{3.1}$$

where $p_{u,v}$ is the maximum influential path from u to v. Suppose $p_{u,v}$ is $\{u, v_i, \cdots, v_j, v)\}$. $Pr(p_{u,v})$ is the probability that u can influence v along the path $p_{u,v}$, i.e., $Pr(p_{u,v}) = w_{u,v_i} \times \cdots \times w_{v_j,v}$. Since $p_{u,v}$ is the maximum influential path, $Pr(p_{u,v})$ is greater than that along any other path from u to $v$in the social network.

Definition 7. (Activated Nodes) Given a set of seeds S, the nodes activated by S are a subset of nodes:

$$\sigma(S) = \bigcup_{v \in V, Pr(v|S) \geq \delta} \{v\}. \tag{3.2}$$

where δ is the activation threshold.

Definition 8. (Influence Maximization (IM)) Given a social network $G = (V, E)$ and an integer k, the influence maximization is to find a set of nodes $S \subseteq V$, known as *seeds*, such that, if only the nodes in S are active initially, the number of nodes activated by S, at the end of information propagation process following one diffusion model, is maximized.

$$\arg\max_{S \subseteq V, |S| \leq k} \{|\sigma(S)|\}. \tag{3.3}$$

3.1.2 Diverse Influence Maximization

Given a set of seeds S, we define the diversity of the nodes activated by S. As we know, the nodes in social networks can be grouped into different communities. The community can be defined differently in different applications such as based on the research field of researchers in the example in Figure **??** or based on the connection density in the social networks. The communities are unnecessary to be disjoint. If the activated users are from more communities, it implies the higher diversity. The diversity is evaluated as follows:

Definition 9. (Diversity Function) Suppose the nodes in a social network $G = (V, E)$ have been organized into m communities, denoted as $\mathbb{C} = \{C_1, ...C_m\}$. Given a set of seeds S, the diversity of nodes activated by S is defined as:

$$D(S) = \sum_{C_i \in \mathbb{C}} \sqrt{\sum_{v_j \in C_i \cap \sigma(S)} r(v_j)} \tag{3.4}$$

where v_j is a node activated by S (i.e., $v_j \in \sigma(S)$) and a member in community C_i, $r(v_j)$ represents the importance of v_j in social networks.

$D(S)$ is greater when the diversity of activated nodes increases. Specifically, when activating a node from a new community (i.e., this community does not have any activated node yet), the higher score is awarded. For the other nodes from the same community, the award for activating them decreases by applying the square root operator. The similar idea has been used in document summarization [Lin and Bilmes 2011]. For node v_j, the importance in social networks $r(v_j)$ can be the degree of v_j, or the PageRank score of v_j, or any other user-defined score function. In this work, $r(v_j)$ is set as 1 by default.

Definition 10. (Diverse Influence Maximization (DIM)) Given a social network $G = (V, E)$ and an integer k, the DIM query is to find a set of seeds $S \subseteq V$ satisfying:

$$\phi(S) = \underset{S \subseteq V, |S| \leq k}{\arg\max} \{(1-\lambda)\frac{|\sigma(S)|}{|V|} + \lambda \frac{D(\sigma(S))}{D(V)}\}. \tag{3.5}$$

where $\sigma(S)$ represents the set of nodes activated by S, $D(\sigma(S))$ represents the diversity of $\sigma(S)$; $\lambda \in [0, 1]$ is the trade-off parameter to balance the two objectives, i.e., the number of activated nodes and the diversity of the activated nodes; $|V|$ and $D(V)$ are the constants for normalization.

3.2 Monotone and Submodularity

The evaluation metric $\phi(.)$ of diverse influence maximization is monotonous and submodular. To prove this, we show that $|\sigma(.)|$ and $D(.)$ in Equation 3.5 are monotonous and submodular respectively. Given any tradeoff parameter $\lambda \geq 0$, the aggregation function of two monotonous and submodular functions is still monotonous and submodular.

Lemma 3. $|\sigma(.)|$ *is monotonous and submodular.*

The influence maximization using LT models has been proved (Theorem 2.2 in [Kempe et al. 2003b], Theorem 2 in [Liu et al. 2012]). it is not obvious for the adapted function $|\sigma(.)|$ to be true. Therefore, we summarize the proof as below.

Each social network can be treated as a random graph. Each edge $(u, v) \in E$ is associated with a random Bernoulli variable governed by $w_{u,v}$, which controls the likelihood u activates v. Let X denote the entire probability space constituting all possible determined influence propagation graphs. A determined influence propagation graph is generated by flipping a coin of bias $w_{u,v}$ for every edge $(u, v) \in E$ to determine if (u, v) exists in the determined graph. Then we have $Pr(v|S) = \sum_{x \in X} P(x)I(S, v, x)$, where $P(x)$ is the probability of a possible determined graph x, and $I(S, v, x)$ is an indicator to say if v can be reached from one of nodes in S in the determined graph x. If the indicator is true, then $I(S, v, x)$ equals 1. Otherwise, $I(S, v, x)$ equals 0. As $\sigma(S)$ is a node set $\bigcup_{v \in V, Pr(v|S) \geq \delta}\{v\}$ based on Definition 3.2, the size of the node set $|\sigma(S)|$ is equivalent to $\sum_{v \in V}\{1|\sum_{x \in X} P(x)I(S, v, x) \geq \delta\}$.

We can safely say the function $|\sigma(.)|$ is monotone if the inequality $|\sigma(S \cup \{u\})| \geq |\sigma(S)|$ holds. It is easy to verify the inequality via comparing their alternatives $\sum_{v \in V}\{1|\sum_{x \in X} P(x)I(S \cup \{u\}, v, x) \geq \delta\}$ and $\sum_{v \in V}\{1|\sum_{x \in X} P(x)I(S, v, x) \geq \delta\}$. Here, if $I(S, v, x)$ equals 1, i.e., v is reachable from S in the possible graph x, then $I(S \cup \{u\}, v, x)$ must be 1. Conversely, it doesn't hold, i.e., if $I(S \cup \{u\}, v, x)$ is 1, then $I(S, v, x)$ may be 0 or 1. Since $P(x) \in (0, 1]$, $\sum_{x \in X} P(x)\ I(., v, x)$ is monotonous. Thus, $\sum_{v \in V}\{1|\sum_{x \in X} P(x)\ I(S \cup \{u\}, v, x) \geq \delta\}$ is always no less than $\sum_{v \in V}\{1|\sum_{x \in X} P(x)\ I(S, v, x) \geq \delta\}$.

Let $S \subseteq T \subseteq V$, $u \in V$ and $u \notin T$. We first consider a determined graph $x \in X$. $\mathbb{R}_x(S \cup \{u\}) - \mathbb{R}_x(S)$ is the set of nodes reachable from u, but not reachable from S, in the determined graph x. As $S \subseteq T$, we have $\mathbb{R}_x(S \cup \{u\}) - \mathbb{R}_x(S)$ must have equal or more additional reachable nodes than $\mathbb{R}_x(T \cup \{u\}) - \mathbb{R}_x(T)$. Thus $|\mathbb{R}_x(.)|$ is a submodular function. Noticing that $|\sigma(.)|$ is a non-negative linear combination of submodular functions $\mathbb{R}_x(.)$ over the determined graph space X with the threshold δ. Thus $|\sigma(.)|$ is also submodular. Thus, the proof can be concluded.

Lemma 4. *$D(.)$ is monotonous and submodular.*

Since $\sigma(.)$ is a monotone and submodular function to be proved in Lemma 3, we have $\Delta(u|S) \geq \Delta(u|T)$ for any $S \subseteq T \subseteq V$ and $u \in V \setminus T$ where $\Delta(u|S) = \sigma(S \cup \{u\}) - \sigma(S)$ representing the set of nodes that are activated by u, but not by S.

Since $D(S)$ is defined as $\sum_{i=1}^{m} \sqrt{\sum_{v_j \in C_i \cap \sigma(S)} r(v_j)}$, if we suppose $D_i = \sqrt{\sum_{v_j \in C_i \cap \sigma(S)} r(v_j)}$, then $D(S)$ can be expressed as $\sum_{i=1}^{m} D_i$. If we can prove that $D_i(.)$ is a monotone and submodular function, then $D(.)$ must be based on the general submodular property. We can prove $D_i(.)$ being a monotone and submodular function by proving $\sum_{v_j \in C_i \cap \sigma(S)} r(v_j)$ because applying the square root to a monotone submodular function yields a submodular function, and summing them all together retains submodularity.

From $\Delta(u|S) = \sigma(S \cup \{u\}) - \sigma(S)$, we can get $\sum_{v_j \in C_i \cap \sigma(S \cup \{u\})} r(v_j) - \sum_{v_j \in C_i \cap \sigma(S)} r(v_j) = \sum_{v_j \in C_i \cap \Delta(u|S)} r(v_j)$. Similarly, we can get that $\sum_{v_j \in C_i \cap \sigma(T \cup \{u\})} r(v_j) - \sum_{v_j \in C_i \cap \sigma(T)} r(v_j) = \sum_{v_j \in C_i \cap \Delta(u|T)} r(v_j)$. Because $\Delta(u|S) \geq \Delta(u|T)$ holds, we have that $\sum_{v_j \in C_i \cap \Delta(u|S)} r(v_j) \geq \sum_{v_j \in C_i \cap \Delta(u|T)} r(v_j)$ for the same community C_i. Thus, for any $S \subseteq T \subseteq V$ and $u \in V \setminus T$, it can conclude that $\sum_{v_j \in C_i \cap \sigma(S \cup \{u\})} r(v_j) - \sum_{v_j \in C_i \cap \sigma(S)} r(v_j) \geq \sum_{v_j \in C_i \cap \sigma(T \cup \{u\})} r(v_j) - \sum_{v_j \in C_i \cap \sigma(T)} r(v_j)$. Therefore, we can see that $\sum_{v_j \in C_i \cap \sigma(S)} r(v_j)$ satisfies the submodular property. Obviously, it also satisfies the monotone property. Lemma 4 can be proved.

3.3 Solution Frameworks

This section proposes two solutions of DIM problem.

Greedy Algorithm

The monotone and submodularity property of $\phi(.)$ shown in Section 3.2 guarantees that the greedy algorithm of DIM problem is with $(1 - \frac{1}{e} - \epsilon)$-approximation.

Suppose there are m communities in social networks. Initially, the seed set S is empty. The greedy algorithm runs by k iterations. At iteration i, if u leads to the maximal *diverse influence gain*, denoted as $\Delta(u)$, a node u is selected as a seed and inserted into S (denoted as S_{i-1} before inserting the new seed at iteration i). The *diverse influence gain* is defined

as

$$\Delta(u|S_i) = \phi(S_{i-1} \cup \{u\}) - \phi(S_{i-1}). \tag{3.6}$$

In this work, $\phi(.)$ is calculated based on the sampling technique discussed in [Chen et al. 2010, Kempe et al. 2003b]. The time complexity of the greedy algorithm is $O(kn^2 \cdot \frac{1}{2\epsilon^2} log\frac{n}{\eta})$ where n is the number of nodes in the social network, ϵ and η are two sampling parameters in [Chen et al. 2010, Kempe et al. 2003b]. The complexity consists of two parts: the first one $O(kn)$ means that the algorithm needs to run k iterations and, at each iteration, it requires to probe each node in the social network; the second part $O(n \cdot \frac{1}{2\epsilon^2} log\frac{n}{\eta})$ means that the estimation of $\phi(S)$ needs to check each node in the social network to determine whether it can be activated in the sampled graphs of size $\frac{1}{2\epsilon^2} log\frac{n}{\eta}$. The error bound of the greedy algorithm is $(1 - \frac{1}{e} - \epsilon)$ where $(1 - \frac{1}{e})$ comes from the greedy approximation and ϵ comes from the sampling approximation.

Upper Bound Algorithm

To improve the efficiency of the greedy algorithm, we develop an upper bound based approach in order to reduce the unnecessary computations as much as possible. Next, we show the existence of upper bound.

Lemma 5. *Given any node u, if it is selected as a seed at one of k iterations, the diverse influence gain cannot exceed the diverse influence gain if u is the first selected seed.*

Since $\phi(.)$ has been proved to be monotonous and submodular in Section 3.2, we can derive that $\Delta(u|S_{i-1}) \geq \Delta(u|S_i)$ for any node $u \in V \setminus S_i$ where $S_{i-1} \subseteq S_i$. Let $\Delta_i(u)$ denote the diverse influence gain of u at iteration i. If $\Delta_i(u)$ is greater than $\Delta_i(v)$ for any $v \in V \setminus S_i, v \neq u$, u is selected as a seed at iteration i. Thus, we have $\Delta_{i-1}(u) \geq \Delta_i(u)$. It means that the diverse influence gain by selecting a node as a new seed in the earlier steps must be not less than that by selecting it in the later steps. In addition, it is easy to see $\Delta_0(u) = \phi(u)$. So $\Delta_0(u)$ is the upper bound of the diverse influence gain by selecting u as a seed at any of the k iterations.

According to Lemma 5, if the diverse influence gain of node u at iteration i, denoted as $\Delta_i(u)$, is known, we can safely prune any node if the upper bound of its diverse influence gain is less than $\Delta_i(u)$. Furthermore, the upper bounds provide the probing priority for the nodes not pruned. That is, the nodes with higher upper bounds should be evaluated earlier.

The time complexity of the upper bound based greedy algorithm is $O(k\#num \cdot n \cdot \frac{1}{2\epsilon^2} log\frac{n}{\eta})$ where $\#num$ is the maximum number of nodes to be evaluated until a successful seed is selected at each iteration and $\#num$ is often much smaller than n. The upper bound based greedy algorithm is more efficient while it also keeps the same $(1 - \frac{1}{e} - \epsilon)$-approximation. The detailed algorithm is omitted here due to space restrictions.

3.4 Quick Aggregated Influence Calculation

The tree-based model has been used in solving influence maximization [Chen et al. 2010, Li et al. 2014a, Lee and Chung 2015]. In this work, we adopt this model to calculate the aggregated influence of seeds to any other node in social networks. Before introducing our model, we first transform the most influential path to the equivalent shortest path.

3.4.1 Path Transformation

Given two nodes u, v over a directed graph G, there is no existing work that clearly shows how to find the most influential path from u to v. One way is to search from u in the best first fashion until v is reached. However, it requires visiting a lot of irrelevant nodes. In addition, it is challenging to maintain the most influential paths between all pairs of nodes in a large graph. To address the two issues, we transform the most influential paths to the equivalent shortest paths. As such, we can utilize the existing techniques dealing with shortest paths between nodes in graph database.

To guarantee the equal transformation, we make mathematical conversion of the weights associated with the edges. For each edge (u, v), we transform the original weight $w_{u,v}$ to $\log(w_{u,v})$. By doing this, we can calculate the influence probability along a path p by computing the sum of transformed weights of the edges in the path, $A = \sum_{(i,j)\in p} \log(w_{i,j})$, and the influence probability is e^A. This transformation makes the conversion from the multiplication operation to addition operation.

Most techniques regarding the shortest paths in graph database prefer to locate the path with the minimum weight. However, the most influential path is the path with the maximum value. Furthermore, as the edge weight is in (0, 1], its log value is negative. Therefore, a further adaption is to change the negative value to a positive value by adding a minus sign. As a result, finding the most influential path from u to v is equivalent to look for the shortest path from u to v.

3.4.2 PSP-Tree

To estimate the influence of a seed set S, a shortest path tree can be constructed for every node v in the social network G. The root of the tree is v. For any node u in the tree, the path to the root is unique. This path corresponds to the shortest path from u to v. So, the influence of u to v (i.e., $Pr(p_{u,v})$) can be calculated. The shortest path tree can be viewed as the compressed version of shortest paths from all other nodes to v by merging the same node appearing in different paths. Given a set of seeds S and a node v, the aggregated influence of S to v can be computed by finding seeds in the tree. For each seed, the influence to v is calculated and then compute the aggregated influence according to Equation 3.1.

Building and maintaining complete shortest path trees is time and space consuming. To handle this issue, we develop an adapted shortest path tree. The idea is to partition social networks into disjoint subgraphs, and then construct the shortest path trees off-line within each subgraph. As a result, the shortest path between nodes within a subgraph can be obtained directly while the shortest path between nodes in different subgraph is reconstructed on-the-fly. Note that the subgraphs are unnecessarily equivalent to communities. Since the adapted tree is a fraction of the complete shortest path tree, it is called *partial shortest path tree* (PSP-Tree).

Definition 11. (Boundary Nodes and Inner Nodes) Given any subgraph g_i with nodes V_i, the boundary nodes of g_i are the subset of nodes in V_i having direct connections with the nodes in other subgraphs. The remaining nodes in V_i are the inner nodes that are not boundary nodes.

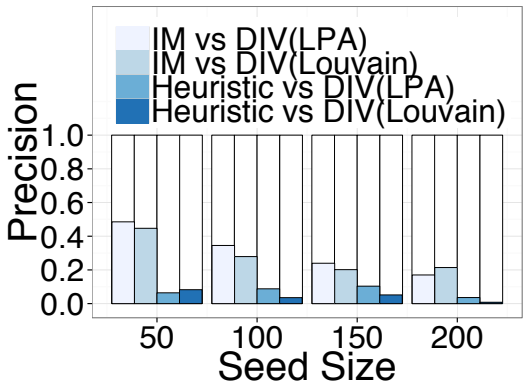
PSP-Tree Construction

The PSP-Tree can be constructed in the following steps. First, all edges in social network G change the direction and the resultant subgraph is denoted as G'; for each node u, the shortest paths from u to all other nodes in the same subgraph can be identified using single source shortest path algorithm; it is easy to prove the shortest path from u to v in G' is the same path from v to u is G. Second, the social network is partitioned into a number of subgraphs by cutting edges [Buluç et al. 2016]. Third, for each node v in each subgraph g, the shortest paths from all other nodes to v obtained in the first step are processed to construct a shortest path tree with v as root. This is correct in concept but not convenient in practice. In this work, the shortest path tree is constructed by using single source shortest path algorithm in G' to find all shortest paths from v to all other nodes in C.

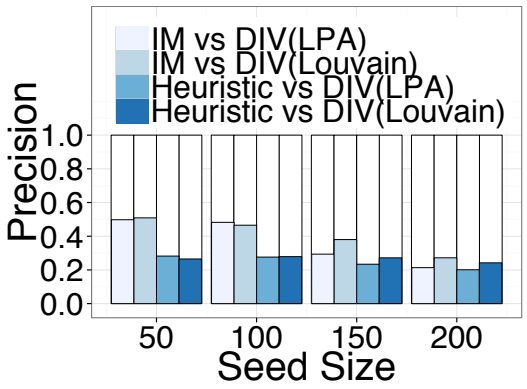
Aggregated Influence Computation with PSP-Tree

Once PSP-Trees have been constructed, it can be used to support computing the aggregated influence from any set of seeds S to any node v in social networks. For each seed $u \in S$, $Pr(p_{u,v})$ is computed. Two situations need to be considered:

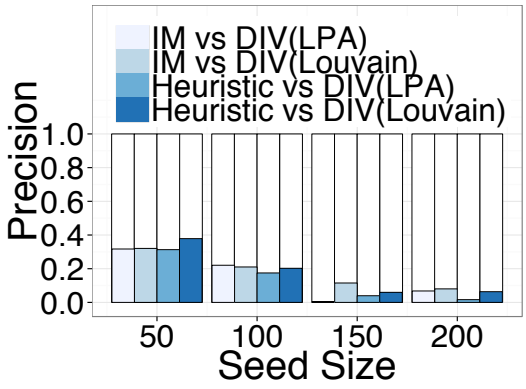
- If u and v are in the same subgraph, the shortest path from u to v can be directly retrieved from the PSP-Tree with root v and thus $Pr(p_{u,v})$ is obtained.
- If u and v are in different subgraphs, the shortest path from u to v is reconstructed by visiting a derived graph. Specifically, the derived graph consists of u, v and the boundary nodes of all subgraphs. For a pair of boundary nodes in the same subgraph, an edge is created and the weight is the shortest path distance between them. For a pair of boundary nodes in different subgraphs, if there is an edge between them in



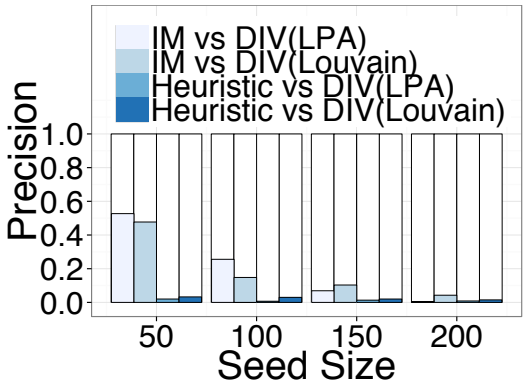
(a) Facebook-Seeds



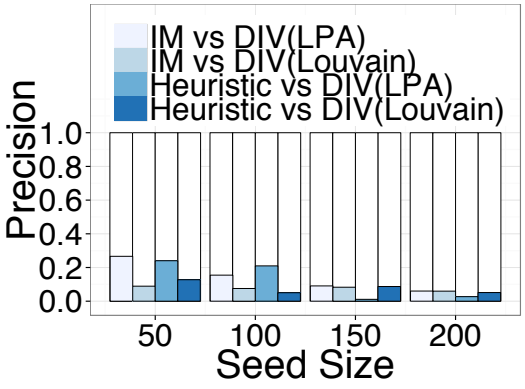
(b) Citation-Seeds



(c) Gowalla-Seeds

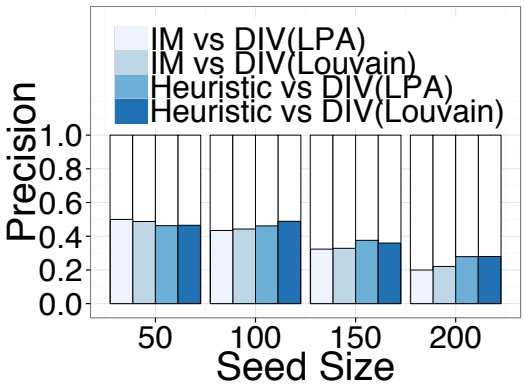


(d) Youtube-Seeds

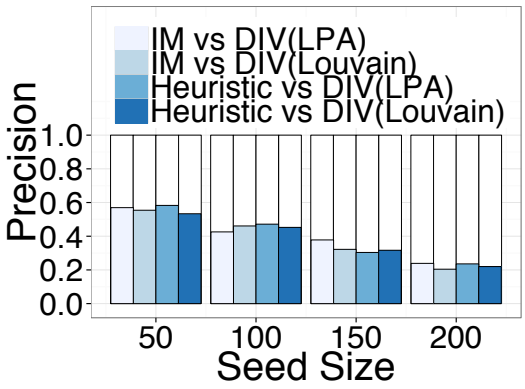


(e) Amazon-Seeds

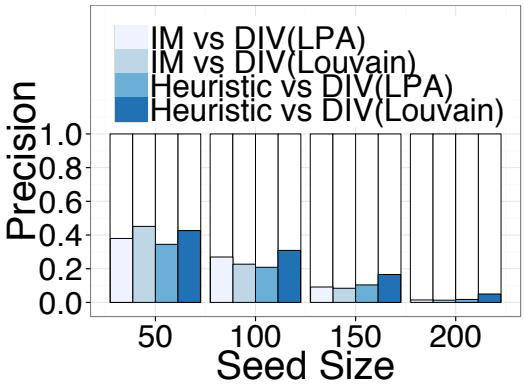
Figure 3.1: Measuring precision of seeds.



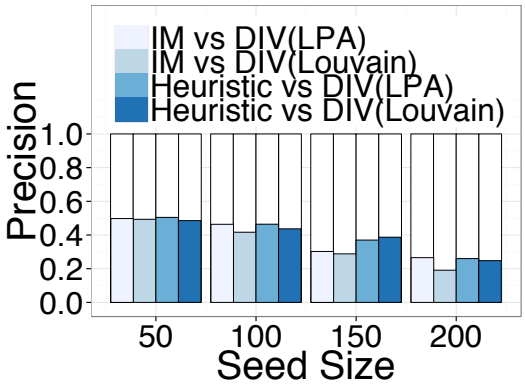
(a) Facebook-Influencees



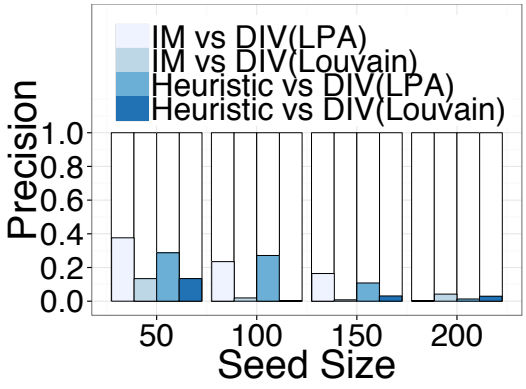
(b) Citation-Influencees



(c) Gowalla-Influencees



(d) Youtube-Influencees



(e) Amazon-Influencees

Figure 3.2: Measuring precision of nodes activated by the seeds.

the original social network, the edge is retained; otherwise, there is no edge between them in the derived graph. From u to each boundary node of the same subgraph as u, an edge is created and the weight is the shortest path distance from u to the boundary node. From each boundary node of the same subgraph as v to v, an edge is created and the weight is the shortest path distance from the boundary node to v. In the derived graph, the shortest path from u to v is identified and thus $Pr(p_{u,v})$ is obtained.

Once $Pr(p_{u,v})$ is computed for every $u \in S$, the aggregated influence from S to v, i.e., $Pr(v|S)$, can be obtained by applying Equation (3.1).

3.5 Experimental Study

We have conducted extensive experimental study to evaluate the effectiveness and efficiency of the proposed solution of DIM problem. All these experiments are tested on a Red Hat Enterprise Linux Server (7.2), with 792GB RAM and Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00GH shared by the school of science, RMIT. The algorithms are implemented using Python 2.7.

Data Sets.

Five real-world social network data sets are tested. They are downloaded from *Stanford Large Network Dataset Collection* [1]. The statistics of the data sets are shown in Table 3.1.

Data Sets	#nodes	#Edges	Avg Degree
Facebook	4,039	88,234	21.8
Citation (DBLP)	4,558	217,984	47.8
Gowalla	69,097	351,452	5.1
Youtube	52,675	636,864	12.1
Amazon	317,194	1,745,870	5.5
DBLP	260,998	1,900,118	7.3

Table 3.1: Statistics of data sets.

[1] http://snap.stanford.edu/data/index.html

Baselines and Our Solutions

We have implemented two baseline algorithms, i.e., IM and Heuristic, and the proposed algorithm, i.e., DIV(.).

- *IM* is the solution developed for influence maximization problem [Chen et al. 2010, Kempe et al. 2003b]. Given a seed u, the set of activated nodes by u is denoted as *Inf(u)*. Given another node u', the new nodes activated by u' is *Inf(u')/Inf(u)*. For the node selected as the next seed, it must be able to activate the maximum number of new nodes in social networks.

- *Heuristic* is an adapted version of *IM*, which checks the nodes with one more hop. Given a seed u, the set of activated nodes by u is denoted as *Inf(u)* and the unactivated neighbors of *Inf(u)* is denoted as *NB(Inf(u))*. Given another node u', the new nodes purely activated by u' is *Inf(u')/{NB(Inf(u)),Inf(u)}*. For the node selected as the next seed, it must be able to purely activate the maximum number of new nodes in social networks.

- *DIV(LPA)* and *DIV(Louvain)* is our DIM solutions using community detection method *LPA* and *Louvain* respectively. Note that community detection method is orthogonal to DIM solution and thus does not impact the evaluation. No matter how the communities are specified/defined, our DIM solution is aware of the communities and the baselines are not.

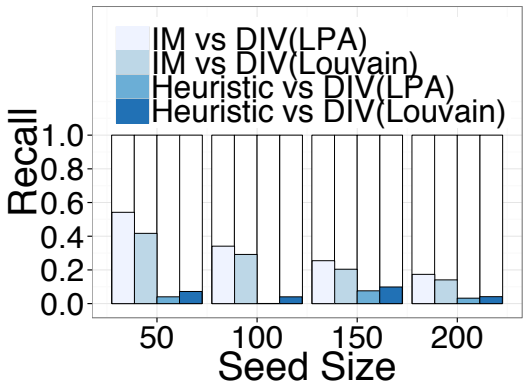
We compare the seeds and the nodes activated by the seeds using our solution and the baselines in terms of effectiveness. We also test the efficiency with and without PSP-Tree. The performance is reported when parameter λ in Equation 3.5 and δ in Equation 3.2 vary. By default, parameter λ is set as 0.5 and parameter δ is set as 0.2.

3.5.1 Evaluation of Effectiveness

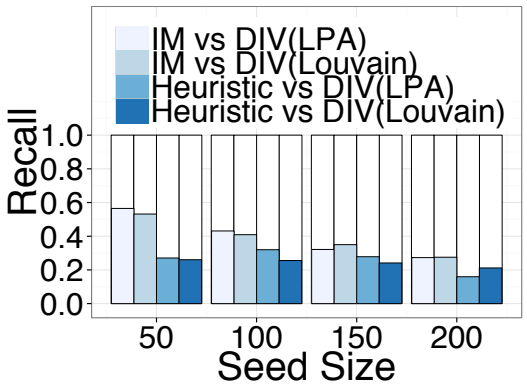
The objective is to find k seeds that can activate the maximum number of nodes in social networks and the activated nodes are as diverse as possible. The effectiveness test is to check whether our solutions show advantage towards the objective compared with the baselines. .

Precision

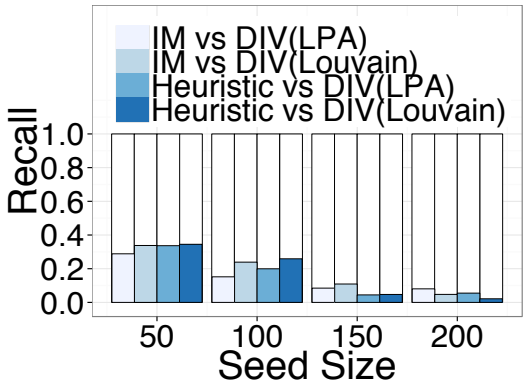
The precision is measured by comparing the seeds returned by *IM* and *Heuristic* against that of *DIV(LPA)* and *DIV(Louvain)* respectively. For example, the seed set returned by *IM* is *im* and the seed set returned by *DIV(.)* is *div*, the precision of *IM* vs. *DIV(LPA)* is $\frac{|im \cap div|}{|im|}$. If the precision is smaller, only a smaller fraction of IM solution (i.e., baselines)



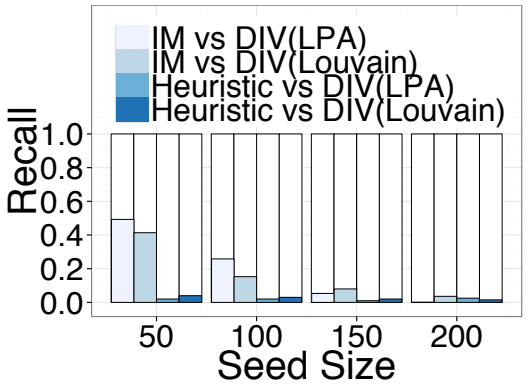
(a) Facebook-Seeds



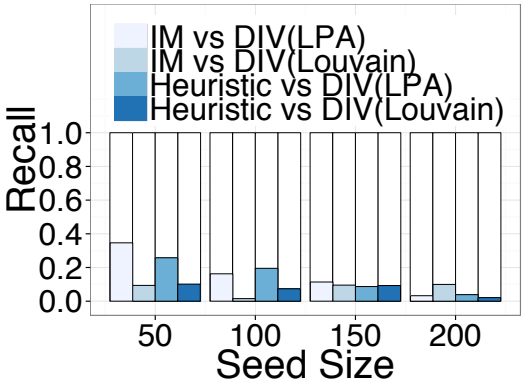
(b) Citation-Seeds



(c) Gowalla-Seeds

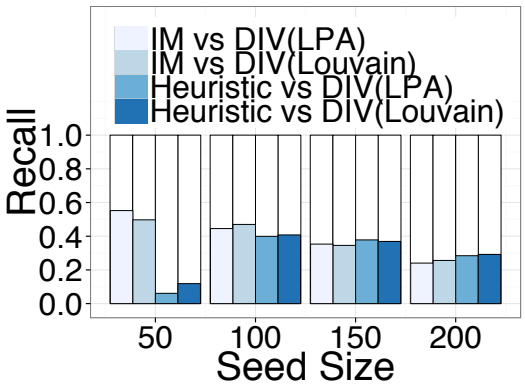


(d) Youtube-Seeds

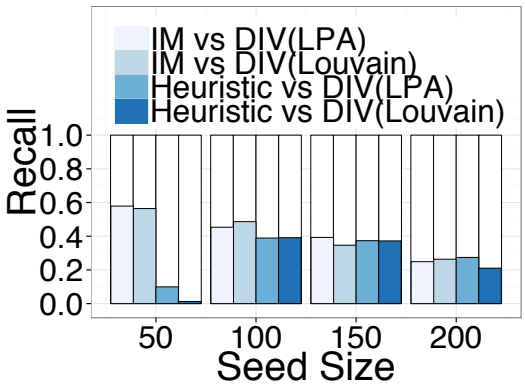


(e) Amazon-Seeds

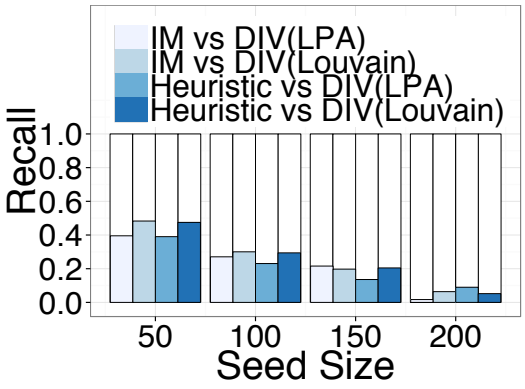
Figure 3.3: Measuring recall of seeds.



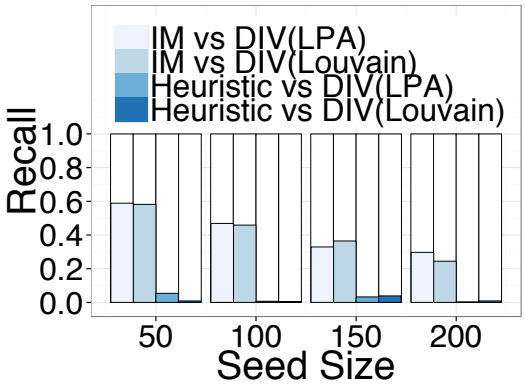
(a) Facebook-Influencees



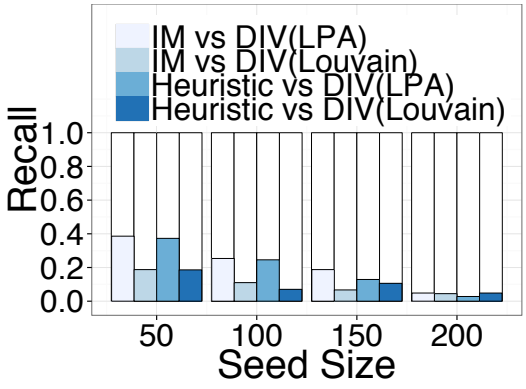
(b) Citation-Influencees



(c) Gowalla-Influencees



(d) Youtube-Influencees



(e) Amazon-Influencees

Figure 3.4: Measuring recall of nodes activated by the seeds.

overlaps the DIM solution (i.e., our methods); it implies IM problem is less similar to DIM problem.

For each data set, the seed sets returned by *IM*, *Heuristic*, *DIV(LPA)* and *DIV(Louvain)* are compared, and the test results are presented in Figure 3.1 where k varies from 50 to 200. In Facebook, *IM* can identify 30%-50% of seeds returned by *DIV(.)* when k is 50 or 100. When k is 150 or 200, *IM* can identify about 20% of seeds returned by *DIV(.)*. In all settings of k, *Heuristic* can identify about 10% of seeds returned by *DIV(.)*. The similar trend can be observed for other three data sets. The activated nodes by seeds using *IM* and *Heuristic* are compared with that using *DIV(LPA)* and *DIV(Louvain)* respectively. The test results are presented in Figure 3.2. When k is 50 or 100, the precision of *IM* and *Heuristic* for Facebook, Citation and Youtube reaches 50%; the precision is about 20%-40% for Gowalla and about 30% for Amazon. When k is 150 or 200, the precisions are becoming much smaller for all data sets. The test results clearly indicate that DIM problem is significantly different from IM problem.

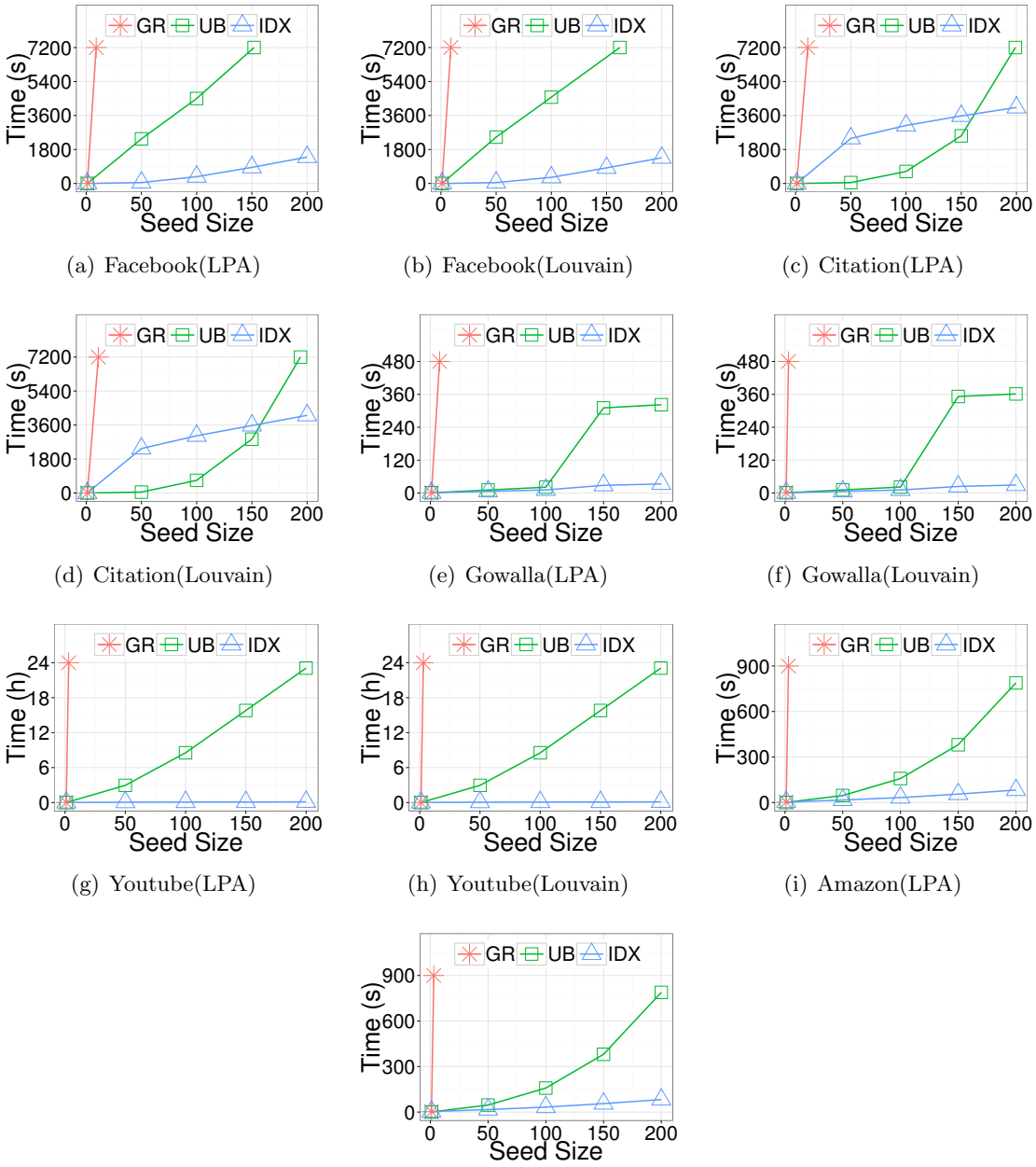
Recall

The recall is measured by comparing the seeds returned by *IM* and *Heuristic* against that of *DIV(LPA)* and *DIV(Louvain)* respectively. The recall of *IM* vs. *DIV(LPA)* is $\frac{|im \cap div|}{|div|}$. If the recall is smaller, IM solution (i.e., baselines) returns a smaller fraction of DIM solution (i.e., our methods); it implies IM solution is less effective to solve DIM problem.

The recall in terms of seeds is presented in Figure 3.3. For all five data sets, the recall of *IM* is lower than 55% and the recall of *Heuristic* is lower than 30% except Citation dataset. The recall in terms of the nodes activated by the seeds is presented in Figure 3.4. In the dense data sets Facebook and Citation, *IM* can activate about 50% nodes among all nodes activated by *DIV(.)*. *Heuristic* can activate about 40% except k=50. For the other three data sets, *IM* and *Heuristic* have less recall values. This test results demonstrates the significance of specifically designed solutions for DIM, that is, *DIVs* cannot be replaced by IM solution, i.e., *IM* and *Heuristic*.

3.5.2 Evaluation of Efficiency

We evaluate the scalability of *GR* (greedy algorithm), *UB* (upper bound algorithm), and *IDX* (PSP-Tree supported upper bound algorithm) at different settings of k. Figure 3.5 reports the time consumed using different algorithms for solving DIM problem. The LPA generated communities and the Louvain generated communities are tested. However, the reported time does not include time for community detection since it is orthogonal to DIM solution. That is, community detection algorithm LPA and Louvain have been performed offline and the generated communities have been maintained.



(a) Facebook(LPA) (b) Facebook(Louvain) (c) Citation(LPA)

(d) Citation(Louvain) (e) Gowalla(LPA) (f) Gowalla(Louvain)

(g) Youtube(LPA) (h) Youtube(Louvain) (i) Amazon(LPA)

(j) Amazon(Louvain)

Figure 3.5: Efficiency of proposed algorithms.

As shown in Figure 3.5, *IDX* outperforms the other two algorithms by about 8-40 times on all data sets (except Citation). The next one is *UB*. *GR* performance is the worst in all situations. For Citation, *UB* is better than *IDX* when k is 50, 100, and 150. But when k increases up to 160, *IDX* performs much better than *UB*. When k is 200, *IDX* is about 2.5 times faster than *UB*. This is because Citation is a dense social network. The dense social network tends to have more computation for intra-community nodes. Additionally, we observe that each algorithm performs similarly over the same data set with LPA and Louvain.

3.5.3 Other Evaluations

This subsection presents the evaluation results when we vary the tradeoff parameter λ, the threshold parameter δ and the data size. Due to the limited space, we just report partial results in this paper.

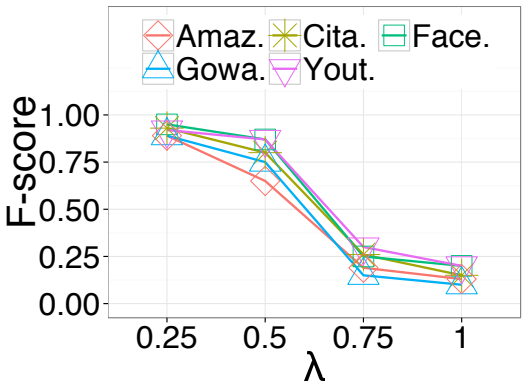
Varying λ

In this test, the accuracy of the activated nodes using *DIV(LPA)* and *IM* is evaluated. The accuracy is measured by F-measure where $F = 2 * \frac{precision*recall}{precision+recall}$. The higher F-measure means the activated nodes are more diverse. The test results are shown in Figure 3.6(a) when λ varies from 0.25 to 1. In the situation $\lambda = 1$, it means the diversity is the only factor to be considered. When $\lambda = 0.25$, it means the diversity is considered less while the number of nodes activated is considered more. By default, $k = 100$ and $\delta = 0.2$. F-score decreases with the increase of λ value for all five data sets. And the decreasing trend goes slowly before $\lambda = 0.5$ and goes sharply after $\lambda = 0.5$.

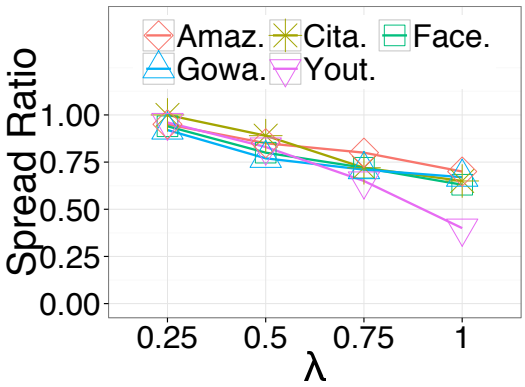
IM aims to activate maximum number of nodes in social networks and *DIV* aims to activate maximum number of nodes which are from more communities. Compared with *IM*, *DIV* has to sacrifice a certain number of activated nodes to ensure they are from more communities. The ratios between the number of activated nodes using *DIV* and that using *IM* at different settings are reported in Figure 3.6(b). When λ is between 0.25 and 0.75, the number of activated nodes using *DIV* is 75% of that using *IM* for all data sets. When λ is 1, the ratio is 75% in four data sets and 40% in Youtube.

Varying δ

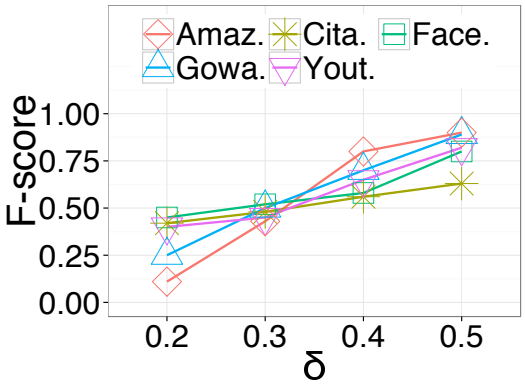
In the test, $k = 100$ and $\lambda = 0.5$ are set. As shown in Figure 3.6(c), F-score increases when δ varies from 0.2 to 0.5. The results depict that it is easy for *IM* to activate diverse nodes when the higher value is set to δ. We also evaluate the time consumed by different algorithms over Amazon when δ changes. The test results are shown in Figure 3.6(d). The performance of *UB* is accelerated with higher δ value, but the performance of *IDX* is affected slightly by δ. Clearly, *IDX* always outperforms *UB*.



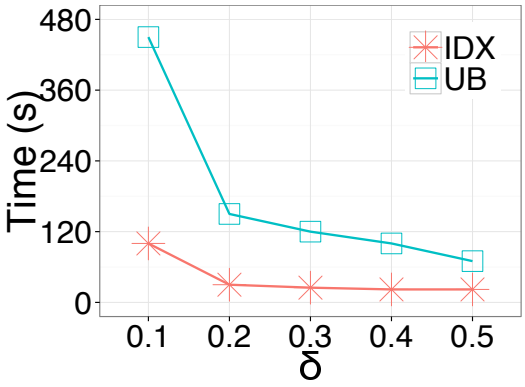
(a) F-Score of IM vs. DIV



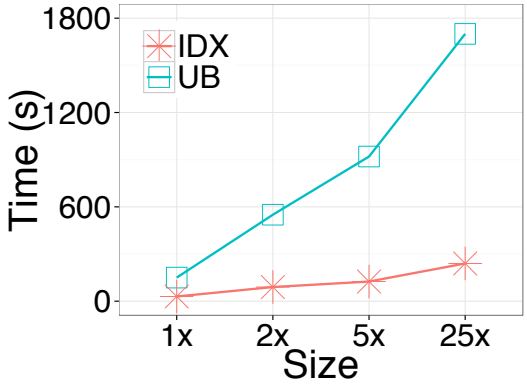
(b) Influence ratio of DIV vs. IM



(c) F-Score of IM vs. DIV



(d) Time cost with δ



(e) Varying data size

Figure 3.6: Impact of λ, δ and data size.

Varying Data Size

We generate three synthetic datasets based on Amazon data in order to evaluate the scalability of the proposed algorithms in terms of the size of social networks. The three data sets have size 2 times, 5 times and 25 times of the size of the original Amazon data set shown in Table 3.1. Figure 3.6(e) illustrates *UB* consumes 600 seconds when data size is 2 times and it consumes 1800 seconds when the data size is 25 times. Obviously, while the data size increases by about 12 times, the consumed time increases by 3 times only. *IDX* performs much better and the trending slope is very small.

3.6 Conclusions

This work has studied diverse influence maximization (DIM) problem to find k influential users as seeds from social networks such that diverse influence, i.e., the number of influenced users and the number of influenced communities, is maximized at the end of information propagation process. DIM provides a new perspective to evaluate the influence over social networks, that is, the information spread is not once-off process because some influenced users may become new seeds later to actively spread information. To handle the conflicting nature of the number of influenced users and the number of influenced communities, we have provided an evaluation metric to balance the weight between them. More importantly, we prove that the evaluation metric possesses the monotone and submodularity property. It allows greedy algorithm to be applied with reasonable approximation bound. To enable quick response to DIM query, the upper bound of diverse influence has been found and explored to minimize the seed candidates. In particular, we have designed PSP-Tree index to quickly compute the diverse influence of seed candidates. On the five real world data sets, the effectiveness of DIM has been verified by comparing with IM and efficiency of our solution has been demonstrated by varying a set of parameters. In short, this study has filled the gap in influence maximization research by breaking through the current limit and it sheds light to social network research where information diffusion is essential.

CHAPTER 4

Targeted Influence Minimization in Social Networks

4.1 Problem Definition

A social network is modeled as a directed graph $G = (V, E)$, where V is a set of nodes and $E \subseteq V \times V$ is a set of edges. A set of nodes $I \subseteq V$ are called active nodes and have information to be diffused in the social network. Another set of nodes $T \subseteq V$, $I \cap T = \emptyset$, are called target nodes and are the recipients of interest.

4.1.1 Diffusion Model

We assume the Linear Threshold (LT) diffusion model [Kempe et al. 2003a]. Thus, each edge (u, v) comes with a weight $b_{u,v} \in [0, 1]$ to represent the influence u has on v. If a message is from u, the influence of this message on v is added by $b_{u,v}$. If the message is from all neighbors of v, denoted as $Adj(v)$, then influence, $v.inf$ of the message on v is $\sum_{u \in Adj(v)} b_{u,v}$. An activation threshold, $v.\tau$, is associated with v. If $v.inf \geq v.\tau$, v is activated; otherwise, v is not activated.

It has been shown that diffusion in the LT model is equivalent to the process of reachability under random choice of live edges in graph instances [Kempe et al. 2003a]. Given a graph $G = (V, E)$, each node $v \in V$ selects at most one of its incoming edges at random, choosing the edge connecting u to v with probability $b_{u,v}$ and not choosing any other edge with probability $1 - \sum_{u \in Adj(v)} b_{u,v}$; the chosen edge is called *live*. After processing each node in V this way, a graph instance G_x containing only the live edges and all the nodes in G is generated. In G_x, suppose a set of nodes I are active initially; an inactive node $u \in V$ ends up as active if and only if G_x contains a path from any node in I to u.

The set of all graph instances that can be generated from G is denoted as χ_G. The influence of I to a set of nodes $T \subseteq (V \setminus I)$ in graph G under the LT diffusion model is defined as follows:

$$\Lambda_G(I,T) = \sum_{G_x \in \chi_G} Prob[G_x] r_{G_x}(I,T), \tag{4.1}$$

where $r_{G_x}(I,T)$ is the number of nodes in T reachable from any node in I in graph instance G_x, and $Prob[G_x]$ is the probability of graph instance G_x.

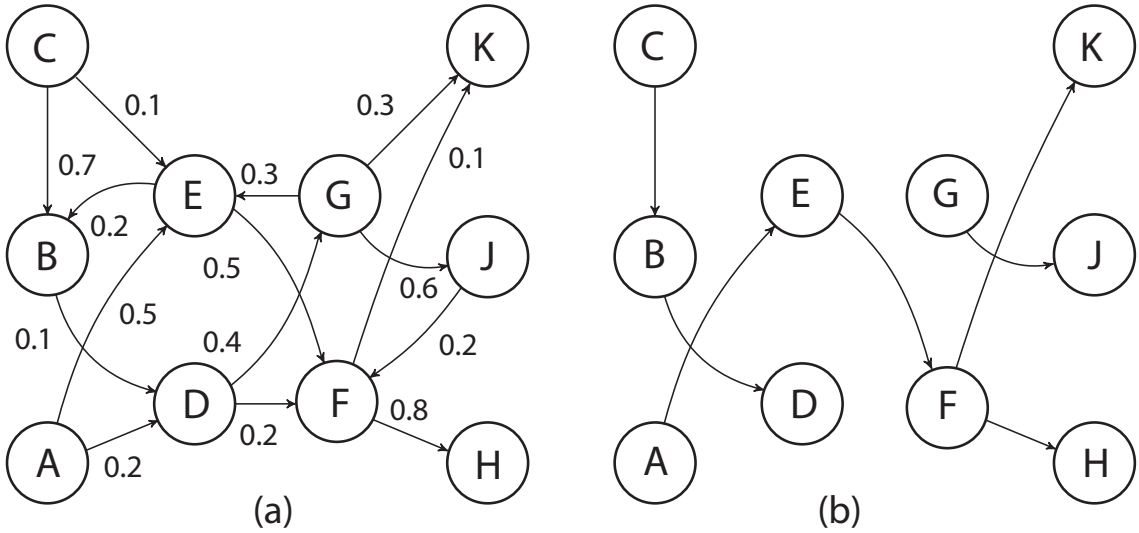


Figure 4.1: A social network and an instance graph.

Figure 4.1(a) illustrates a social network, and an instance graph using the LT diffusion model is shown in Figure 4.1(b). The probability of the instance graph is 0.000504. Suppose $I = \{A, B, C\}$ and $T = \{K, J, H\}$. Then K and H are reachable from A, while J is not reachable from any node in I.

4.1.2 Targeted Influence Minimization

A social network $G = (V, E)$ from which a subset of edges $S \subseteq E$ has been deleted is denoted as $G(S)$.

Definition 12 (Targeted Influence Minimization (TIMin))**.** Given a social network $G = (V, E)$, a set of active nodes $I \subseteq V$, a set of target nodes $T \subseteq \{V \setminus I\}$ and a positive real number k as a budget, suppose $\mathbb{S} = \{S_1, S_2, \cdots, S_n\}$ contains all possible sets of edges where $|S_i| \leq k$, $1 \leq i \leq n$;

- if there does not exist $S_i \in \mathbb{S}$ such that $\Lambda_{G(S_i)}(I,T) = 0$, TIMin aims to find the set $S_* \in \mathbb{S}$ such that $\Lambda_{G(S_*)}(I,T)$ is minimal;

- if a set $S_i \in \mathbb{S}$ exist such that $\Lambda_{G(S_i)}(I,T) = 0$, TIMin aims to find a set $S_* \in \mathbb{S}$ such that $\Lambda_{G(S_*)}(I,T) = 0$ and $|S_*|$ is minimal.

In the former case, the budget is insufficient to completely block the information propagation from I to T. In the latter case, the budget is sufficient to do so. As an example, consider Figure 4.1, where $I = \{A, B, C\}$ and $T = \{K, J, H\}$. A budget $k = 2$ is insufficient to completely block the information propagation from I to T. Thus, TIMin aims to find the set of edges S_* such that $\Lambda_{G(S_*)}(I,T)$ is minimized. Given a budget of $k = 10$, there are many sets of edges that, if deleted, will completely block the information propagation from I to T. In this situation, among all such sets of edges, TIMin aims to find one with the minimum number of edges.

Given active nodes I and target nodes T, we initially need to determine whether the budget k is sufficient or not since this is not known in advance. This leads to the following processing framework.

1. The first stage solves the influence minimization with am unconstrained budget, defined as follows.

$$\begin{aligned} \min \quad & |S_i| \\ s.t. \quad & \Lambda_{G(S_i)}(I,T) = 0 \wedge S_i \subset \mathbb{S} \end{aligned} \tag{4.2}$$

   If $|S_i| \leq k$, the problem is solved by returning S_i because the budget is sufficient to completely block the information propagation from I to T; otherwise, we go to the second stage.

2. The second stage solves the influence minimization with a budget k, defined as follows.

$$\begin{aligned} \min_{S_i} \quad & \Lambda_{G(S_i)}(I,T) \\ s.t. \quad & |S_i| \leq k \wedge S_i \subset \mathbb{S} \end{aligned} \tag{4.3}$$

4.2 Budget Unconstrained Solution

We first examine whether the budget is sufficient to completely block the information propagation from I to T. For this purpose, TIMin with unconstrained budget (i.e., $k = \infty$) is solved as a *minimum cut* or *maximum flow* problem. Let s and t be a source node and sink node in a flow network, respectively. In optimization theory, the *max-flow min-cut theorem* states that the maximum amount of flow passing from the source to the sink is equal to the total weight of the edges in the minimum cut, i.e., equal to the smallest total weight of the edges that, if removed, would disconnect the source from the sink [Papadimitriou and Steiglitz 1998]. If multiple sources and multiple sinks exist, the problem is transformed into a single-source and single-sink maximum flow problem by

adding two new nodes: one connecting all source nodes and the other connecting all sink nodes; the weights of the new edges connected to the two new nodes are ∞.

Lemma 6. *Given a social network $G = (V, E)$, a set of source nodes $I \subseteq V$, and a set of target nodes $T \subseteq \{V \setminus I\}$, the influence minimization is equivalent to the minimum cut problem if budget $k = \infty$.*

Proof. By Definition 12, influence minimization with an unconstrained budget is to identify the minimum set of edges S_* to delete such that by deleting which $\Lambda_{G(S_*)}(I, V) = 0$. As introduced in Section 4.1, the influence from source nodes is computed using Equation 4.1. If there is path from any node in I to any node in T, a graph instance exists where $Prob[G_x] > 0$ and $r_{G_x}(I, T) > 0$, and thus $\Lambda(I, V) > 0$. So, $\Lambda(I, V) = 0$ holds only if all possible paths from any node in I to any node in T are blocked. In this situation, no instance graph may have a path from any node in I to any node in T such that $\Lambda_{G(S_*)}(I, V) = 0$.

Therefore, influence minimization with an unconstrained budget is to find a minimum set of edges that, if deleted, disconnect I and T. This is equivalent to the single-source, single-sink minimum cut problem if I and T each contain one node; otherwise, it is equivalent to the multi-source, multi-sink minimum cut problem, which can be transformed into a single-source, single-sink minimum cut problem as discussed above. □

In Figure 4.2, influence minimization with an unconstrained budget is modeled as a single-source, single-target minimum cut problem. Specifically, a node s is added and linked to all the active nodes in I, and a node t is added and linked to all the target nodes in T. The weight of each edge is infinity.

The *minimum cut* or *maximum flow* problem is well studied [Papadimitriou and Steiglitz 1998]. We adopt Dinic's algorithm to solve this problem [**?**]. Dinic's algorithm uses the concept that a flow is maximum if no path from s to t exists in the residual graph. Given a flow network, if there exists an s-t path, then the algorithm constructs a residual graph based on by applying for the weight reduction on each edge in the s-t path. The weight to be reduced is the smallest weight on the edges in the path and the updated weights are called *forward capacity*. Meanwhile, the residual graph also records a *backforward capacity* for each edge in the s-t path. The *backforward capacity* for an edge increments by 1 if its *forward capacity* decreases by 1 where the initial *backforward capacity* on each edge is zero. To improve the efficiency, Dinic's algorithm further proposes the concept of level graph. Each node u in the level graph has an attribute with its shortest distance to s in the residual graph, which maintains information to accelerate the computation of s-t path. If there exists an s-t path left in the residual graph, then it updates the residual graph as well as the level graph. The algorithm stops when no s-t path is left in the residual graph.

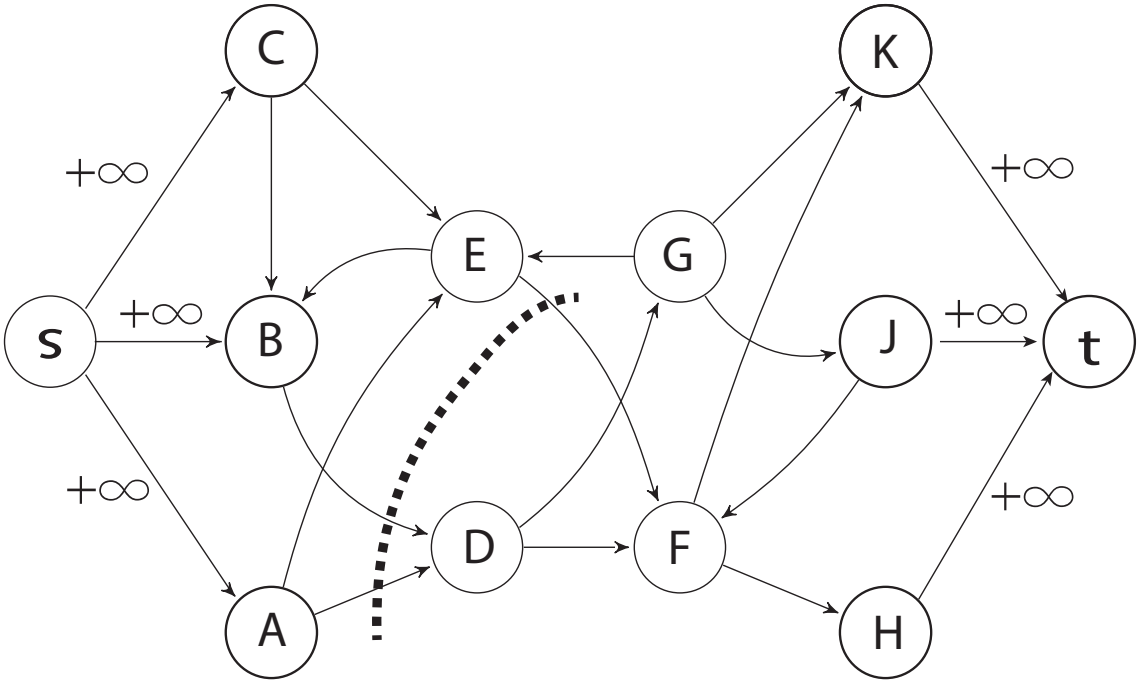


Figure 4.2: Influence minimization, unconstrained budget.

The complexity of Dinic's algorithm is $\mathcal{O}(min\{V^{2/3}, E^{1/2}\}E)$ if I and T each contains only one node; otherwise, it is $\mathcal{O}(E^{3/2})$. In Figure 4.2, the set of edges returned is $S_* = \{(A, D), (B, D), (E, F)\}$ and $|S_*| = 3$. If budget $k \geq |S_*|$, the budget is sufficient to completely block the influence from I to T.

4.3 Budget Constrained Solution

Theorem 7. *TIMin with an insufficient budget k is NP-hard.*

Proof. TIMin with an insufficient budget k is an instance of the *maximum coverage* problem that is known to be NP-hard [Vazirani 2013]. Given a number k and a number of sets (the sets may have elements in common), the *maximum coverage* problem aims to select at most k sets such that the maximum number of unique elements are covered, i.e., the cardinality of the union of the selected sets is maximized. In TIMin, we take each edge e as a distinct set number. And the set contains the corresponding paths passing e from the source nodes I to the target nodes T in all instance graphs. The generated sets may have duplicate paths. Suppose the corresponding sets of all edges are known and the paths have the uniform probability. TIMin with insufficient budget k aims to select at most k edges which, if deleted, the maximum number of paths from I to T are blocked. Therefore, TIMin with insufficient budget k is the same as identifying the best k sets in the *maximum coverage* problem. So, TIMin with budget k can be proved to be an NP-hard problem using a reduction from the well-known NP-hard problem. □

Due the result in Theorem 7, we provide a greedy algorithm to solve targeted influence minimization with an insufficient budget.

4.3.1 Greedy Algorithm

The greedy algorithm searches for a set of edges $S \subseteq E$ such that $|S| \leq k$ and the following objective function is maximized.

$$f(S) = \Lambda_G(I,T) - \Lambda_{G(S)}(I,T), \tag{4.4}$$

where $\Lambda_G(.,.)$ is computed using Equation 4.1.

The greedy algorithm proceeds iteratively. Initially, S is empty. In each iteration, it computes the value of each edge e in $G(S)$ as follows.

$$value(e) = \Lambda_{G(S)}(I,T) - \Lambda_{G(S')}(I,T), \tag{4.5}$$

where $S' = S \cup \{e\}$. The value of e, $value(e)$, is the reduction of influence from I to T with and without e in $G(S)$. Among all edges, the one with the maximum value, say e_*, is deleted. Then e_* is inserted into S, and the remaining budget is decremented by 1. The process terminates when the remaining budget reaches 0. The greedy algorithm is an $(1 - \frac{1}{e})$-approximation ($\approx$0.632-approximation) since the objective function is non-negative, monotonous, and submodular [Khalil et al. 2013].

4.4 Sampling-based Solution

It is prohibitively expensive to directly generate all graph instances and compute the value of each edge in each iteration. Therefore, we devise a sampling-based solution. The solution is inspired by a recent influence maximization study [Tang et al. 2014b], but significant adaptions are required.

Reverse Influence Set (RIS)

Tang et al. [2014b] aim to select at most k nodes with maximum influence in a social network. The method is based on RIS that computes the influence of nodes using graph instances. Specifically, the reverse reachable (RR) node set for each node in each graph instance is generated. Given a node v in graph instance G_x, the RR set contains all nodes in G_x that can reach v. Using the sampling method, a number of nodes are randomly selected from V; the RR set for each node is generated using a randomly selected graph instance. So, a number of random RR sets are obtained. If a node u has a great impact on other nodes, u will have high probability of appearing in the random RR sets. As a result, the problem is transformed to the *maximum coverage problem* of identifying at most k nodes that cover the maximum number of the random RR sets. It has been shown

that if the number of random RR sets θ is no less than $(8+2\varepsilon)|V|\frac{\ln|V|+\ln\binom{|V|}{k}+\ln 2}{OPT_k\varepsilon^2}$, then RIS returns an $(1-1/e-\epsilon)$-approximate solution with at least $1-|V|^{-1}$ probability ($\epsilon \in (0,1)$) [Borgs et al. 2014].

4.4.1 Minimum Influence Path

RIS cannot be applied to our problem without significant modification due to two reasons.

- The random RR set is about node-to-node reachability. In our problem, however, we delete the edges to make reachable-nodes unreachable. While it is straightforward to determine node-to-node reachability, it is more difficult to identify edges the deletion of which makes reachable-nodes unreachable. The reason is that there may be many different paths between two reachable nodes, so deleting an edge does not necessarily block the reachability.

- The random RR set is for the reachability of any node. In our problem, however, only the source nodes I and the target nodes T are relevant.

We propose a novel sampling-based method called *Minimum Influence Path* (MIP) to solve TIMin. The idea is to exploit the fact that each node in a graph instance under the LT diffusion model has at most one incoming edge. Specifically, each node $v \in V$ in the graph instance generation process picks at most one of its incoming edges at random, selecting the edge from $w \in Adj(v)$ with probability $b_{w,v}$, and selecting no edge with probability $1-\sum_{w\in Adj(v)} b_{w,v}$. Figure 4.1 (b) shows an example.

As a result, for two nodes v and u, if v is reachable from u in the graph instance, it is easy to observe that the following properties hold: (**i**) there is one and only one path from u to v in the graph instance, and (**ii**) the path is acyclic. Therefore, the information propagation from u to v in this graph instance can be blocked by removing any edge in the path. On the other hand, if v is not reachable from u in the graph instance by deleting an edge e, this does not indicate that v is not reachable from u in other graph instances. However, if v is not reachable from u in many graph instances by deleting e, this implies that the information propagation from v to u is less likely to happen even though it is not impossible. So, the problem is to delete those edges that block the paths from source nodes to target nodes are blocked in many graph instances.

On the other hand, if v is not reachable from u in the graph instance, the information propagation is blocked without deleting any edge. This may occur for two reasons. First, v is not reachable from u in graph G. Second, v is not reachable from u in this graph instance. If v is not reachable from u in many graph instances, this implies that the information propagation from v to u is less likely to happen even though it is not impossible.

Given a node in $v \in T$, the *minimum influence path* in a graph instance is the path to v from any node $u \in I$ with the fewest edges. Figure 4.3 (a) shows a graph instance where $I = \{u_1, u_2, u_3\}$ and $T = \{v_1, v_2, v_3, v_4\}$. The minimum influence path from I to each target node is shown in Figure 4.3 (b). The minimum influence path to v_1 is (e_1, e_2, e_3). Cutting any edge in the minimum influence path will prevent I from reaching v_1 in this graph instance. Intuitively, the edge appearing in more minimum influence paths is more likely to, if deleted, lead to the more influence reduction. In this graph instance, edge e_5 appears in the minimum influence paths of v_2 and v_3 such that deleting e_5 prevents I from reaching two nodes. If deleting e_5 prevents I from reaching many nodes in T in other graph instances, e_5 is likely to be the edge in the solution of MIP.

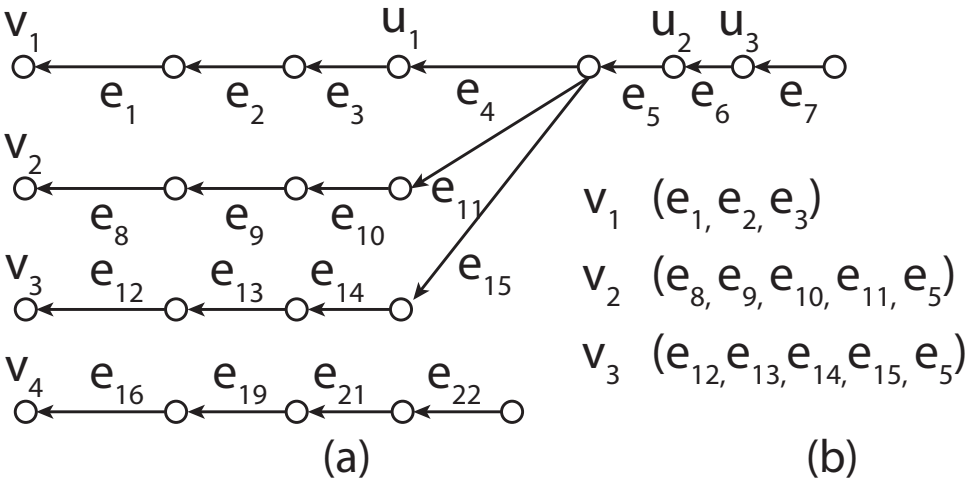


Figure 4.3: Reverse influence paths.

4.4.2 Sampling-based Greedy Algorithm

The pseudo-code of the sampling-based greedy algorithm is presented in Algorithm 8. First, we randomly generate a graph instance in lines 8–8. One node in T is selected randomly in line 8, and the minimum influence path of this node is generated in line 8. This way, θ nodes have been sampled, and the minimum influence path is generated for each of them. Note that a graph instance is more likely to be selected if the probability of the graph instance is high. If deleting an edge can prevent I from reaching many nodes in T in many graph instances, this edge is more likely to appear in the minimum influence paths. So, the problem is transformed to the *maximum coverage problem* of selecting at most k edges to cover the sampled nodes as many as possible. In our solution, we assume that the specified budget is sufficient, otherwise, the budget unconstrained solution is applied. To this end, the incremental solution of maximum coverage problem, known as incrementalMC(M), is applied in line 8.

Algorithm 8 Sampling-based Solution

Input: $G = (V, E)$, I, T, k, θ
Output: S_*

```
i ← 0
M ← ∅
while i ≤ θ do
    j ← 0
        // generate a graph instance i
    foreach v ∈ V do
        if generateEdge() then
            randomly select w ∈ Adj(v) with probability b_{w,v}
        end
    end
    u ← randomly select a node in V
    u.M ← minInfPath(u)
    M ← M ∪ u.M
    i ← i + 1
end
S_* ← incrementalMC(M)
return S_*
```

The maximum coverage problem is solved using an adapted greedy algorithm that is aware of the budget sufficiency. The pseudo-code is presented in Algorithm 9. The generated minimum influence paths and the corresponding reverse minimum influence paths are used. For each minimum influence path, the algorithm maintains a node $v \in I$ and the list of the edges in the path. For each reverse minimum influence path, it maintains an edge e and a list of the nodes each of which has e in its minimum influence path. The reverse influence minimum paths are constructed while the influence minimum paths are generated (line 9). First, the edge with the longest reverse minimum influence paths is moved to solution S_* (lines 9–9). Then, the nodes in the reverse minimum influence path are processed by finding their minimum influence paths and removing them (line 9); for any edge in the minimum influence paths, its reverse minimum influence paths is found and updated (lines 9–9). The process is repeated until k edges are selected (line 9) or no complete path exists in the remaining influence minimum paths (line 9). The budget sufficiency awareness is implemented by checking whether no complete path exists.

Theorem 8. *If $|S_*| \leq k$, the probability that the information propagation from I to T is completely blocked is at most $\frac{1}{n}$; the $|S_*|$ is an $\frac{1}{n}$-approximation of the optimal solution.*

Proof. The proof is based on Theorem 3.1 provided by **?**. In the theorem, consider an arbitrary unweighted multigraph $G = (V, E)$ with edge connectivity λ and choose a subset

Algorithm 9 Incremental Maximum Coverage

Input: `Mip`
Output: S_*

$i = 0$

```
RMip ← construct reverse minimum influence path
S* ← ∅
while i ≤ k do
    if no complete path in Mip then
        break;
    end
    rp_e ← the longest path in RMip
    delete rp_e from RMip
    S* ← S* ∪ e
    foreach v ∈ p_e do
        delete path p_v from Mip
        foreach e ∈ p_v do
            rp_e ← delete v from rp_e
        end
    end
end
return S*
```

$S \subseteq E$ by indicating each edge $e \in E$ in set S independently with probability p. If $p \geq \frac{20logn}{\lambda}$ then the sampled subgraph $G' = (V, S)$ is connected with probability at least $1 - \frac{1}{n}$.

In this work, each reverse influence path can be modeled as a small graph. Given sets I and T of source and sink nodes, we build a multigraph. As such, the targeted influence minimization from I to T can be transformed into reducing the connectivity of the sampled subgraph G'. Cutting the selected subset S can guarantee that G' is connected with probability at most $\frac{1}{n}$ if $|S| \leq k$. Otherwise, the probability of being disconnected is at most $1 - \frac{k}{|S|}(1 - \frac{1}{n})$. □

4.5 Experimental Study

We evaluate the effectiveness and efficiency of our proposed algorithms by comparing with two heuristic algorithms called *Random* and *Weight*. *Random* selects edges randomly until budget k is used. *Weight* selects edges with largest edge weights. Their performance are evaluated in different parameter settings using three real-world networks: *Wiki* with 7,115 nodes and 103,689 edges, *Ego-twitter* with 23,370 nodes and 33,101 edges, and *Epinions* with 75,879 nodes and 508,837 edges. All the three datasets are downloaded from the

Stanford Dataset Collection[1].

4.5.1 Evaluation of Effectiveness

Varying k: Figure 4.4 shows the experimental results when varying k while the source node set I and the target node set T are fixed in size at 500 unless stated otherwise. The source and target nodes are selected randomly. The study shows that *Greedy* and

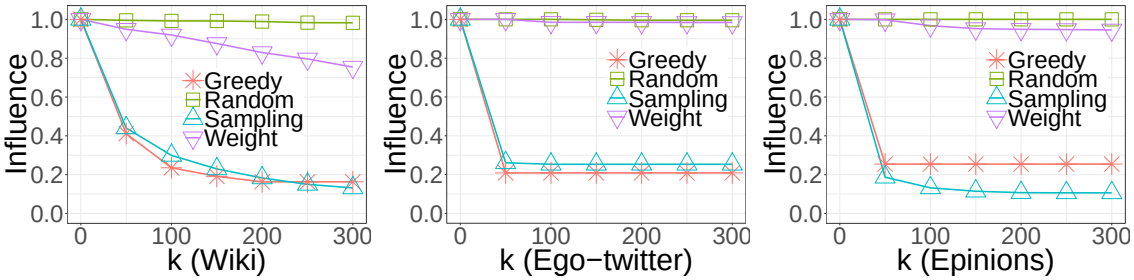


Figure 4.4: Remaining influence from I on T when varying k.

Sampling are able to greatly reduce the influence of I on T for all three datasets given a sufficiently large value of k. When k is above 100, both solutions are able to reduce the influence by up to 80%. Next, *Random* and *Weight* can slightly reduce the influence in *Wiki*. They do not work for *Ego-twitter* and *Epinions*. *Random* and *Weight* cannot block the influence well because the selection of their deleted edges are not relevant to target users. However, this matter is taken into account in *Greedy* and *Sampling*. So the influences minimized by *Greedy* or *Sampling* are always larger than that of *Random* or *Weight*.

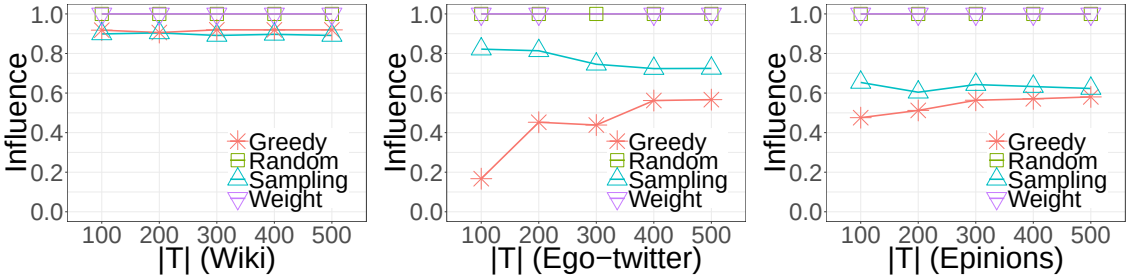


Figure 4.5: Remaining influence of I on T when varying $|T|$.

Varying T: We randomly select 500 nodes as the source node set I and set k as 500. Figure 4.5 shows the results when we increase the target node set from 100 to 500 nodes. *Greedy* and *Sampling* are still able to reduce the remaining influence from I on T by deleting at most 500 edges. *Random* is the worst for all datasets. *Weight* performs better than *Random* in *Wiki* only. The resultant observation is quite interesting. Our proposed

[1]http://snap.stanford.edu/data/

solutions *Greedy* and *Sampling* are quite stable at blocking the influence of the source nodes on the target nodes at a certain level.

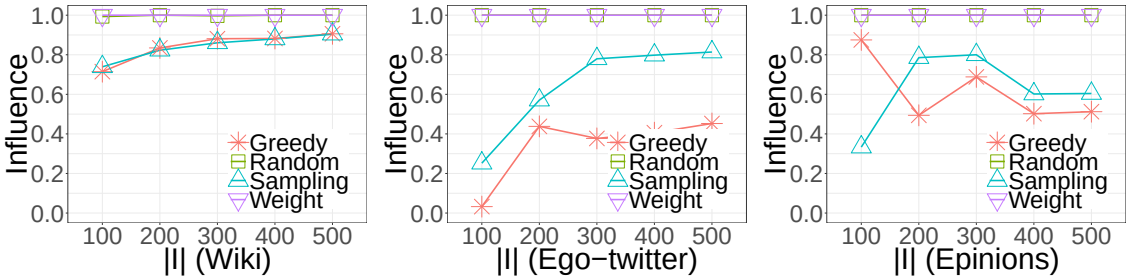


Figure 4.6: Remaining influence of I on T when varying $|I|$

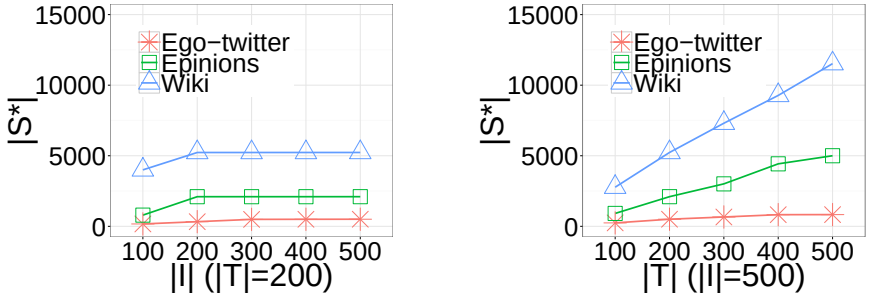


Figure 4.7: #edges deleted for unconstrained budget.

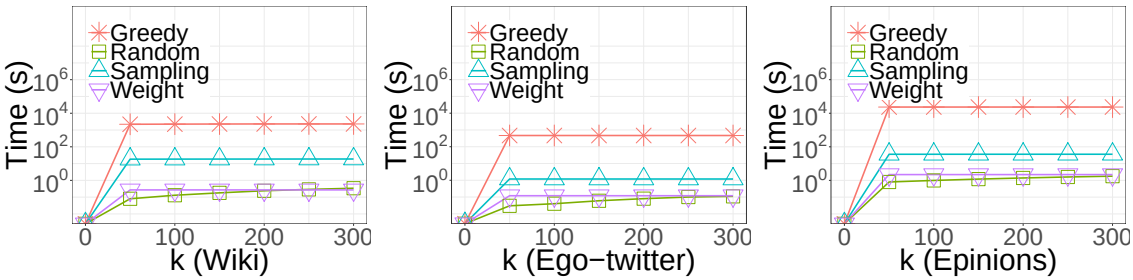


Figure 4.8: Time cost when varying k.

Varying I: Figure 4.6 shows the results when we vary the size of the source set I for $k = 500$ and $|T| = 200$. In this study, *Greedy* and *Sampling* can reduce the remaining influence to 0.2 in *Wiki*, which is a dense graph. For *Ego-twitter* and *Epinions*, their performance varies more. Thus, *Greedy* performs better on *Ego-twitter*, and *Sampling* does well on *Epinions*. However, *Random* and *Weight* have the worst performance in all three datasets.

Budget Unconstrained Evaluation: As shown in Figure 4.7, we can see that the influence from I to T can be blocked completely by deleting a certain number of edges. When $|I| = 500$, $|T| = 100$, it requires 243 edges for *Ego-Twitter*. But more edges must be deleted for *Epinions* and *Wiki* because *Ego-Twitter* dataset is much sparse than *Epinions*

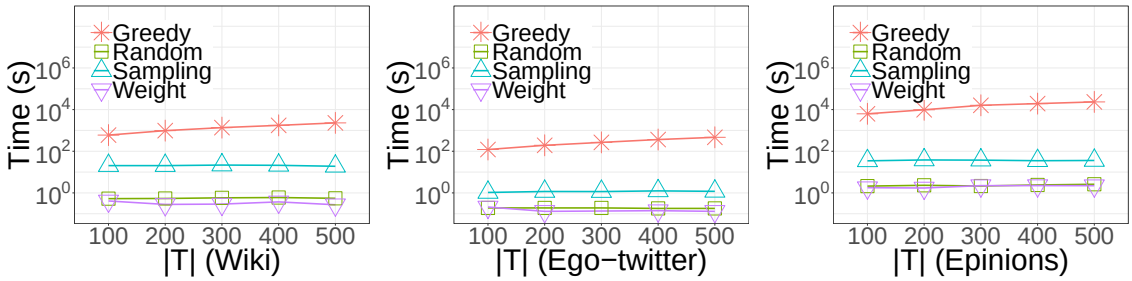


Figure 4.9: Time cost when varying $|T|$.

or *Wiki* datasets. In order to minimize the influence of I on T in the same parameter settings, it has to delete more edges so that all the paths connecting from I to T can be disconnected. However, when $|T|$ becomes large, it is quite challenging to completely block the initial users' influence on the target users because a large number of edges need to be deleted. Our sampling solution can be applied to block the majority of the influence.

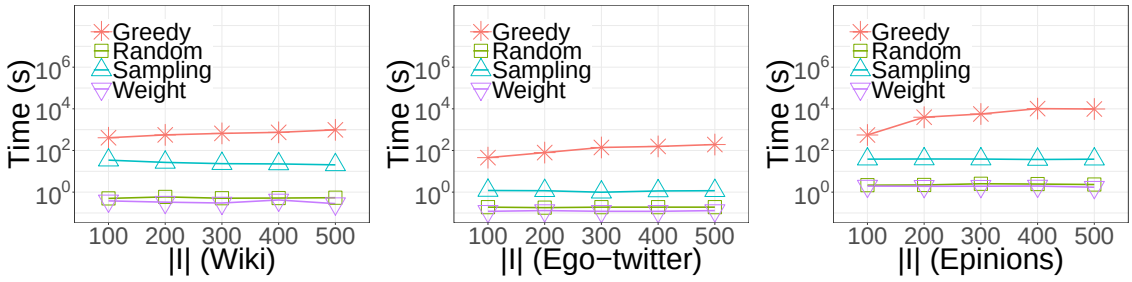


Figure 4.10: Time cost when varying I.

4.5.2 Evaluation of Efficiency

We evaluate the efficiency of the four solutions when varying k, T, and I. Figures 4.8–4.10 present the results. Our sampling solution is capable of outperforming the greedy solution by 2 orders of magnitude in all datasets. Both solutions are stable in performance when we increase k. But the time cost of *Greedy* grows with the increase of T or I. Compared with *Greedy* and *Sampling*, *Random* and *Weight* have the best efficiency because their deleted edges can be found without too much computation. But, as we have seen, their lack of effectiveness render them of little use. Therefore, the sampling solution is the best choice for targeted influence minimization in terms of effectiveness and efficiency.

4.6 Conclusion

In this work, we propose and formalize the problem of targeted influence minimization in social networks that has not previously been studied. We present different solutions that address the computational challenges associated with this problem. We report on

empirical studies showing that the proposed solution is capable of quickly blocking 80% or more the influence of source users on target users. The proposed sampling-based solution is efficient when applied to large scale social networks. This is very important because system need to be able to quickly identify the set of edges to be deleted in order to block the source users' influence. A less efficient solution may enable the source users to activate additional users as new source users, who can then spread the malicious information and this way influence the target users.

CHAPTER 5

Conclusion and Future Works

"We know very little, and yet it is astonishing that we know so much, and still more astonishing that so little knowledge can give us so much power."

–Bertrand Russell

In this thesis, we analyze communities where social influence plays an essential role and solve three research questions as follows.

First, we have investigated a novel and significant problem of searching the most influential maximal kr-clique communities in a vast social network, which is an NP-hard problem and has various essential applications in real life. This is the first work to discover communities via their outer influence. Compared with the existing community models, such as k-core and k-truss, our proposed maximal kr-clique community model has more desirable characters. These modelling contributions provide a new aspect for users and companies to understand the communities and their influence in the social network at the community level. In addition, we have developed tailored C-Tree index and efficient search algorithms to enable the most influential community search in large social networks. Interestingly, once the community model has been explicitly defined, no matter the maximal kr-clique communities or any previous model, the proposed index and the search algorithms still work with easy adaption. The robustness of the maximal kr-clique community model has been verified through the case study of a real-world application. The efficiency of the C-Tree index and search algorithms have been tested on six real world social networks.

Secondly, we have studied diverse influence maximization (DIM) problem to find k influential users as seeds from social networks such that diverse influence, i.e., the number of influenced users and the number of influenced communities, is maximized at the end of information propagation process. DIM provides a new perspective to evaluate the influence over social networks, that is, the information spread is not once-off process

because some influenced users may become new seeds later to actively spread information. To handle the conflicting nature of the number of influenced users and the number of influenced communities, we have provided an evaluation metric to balance the weight between them. More importantly, we prove that the evaluation metric possesses the monotone and submodularity property. It allows a greedy algorithm to be applied with reasonable approximation bound. To enable quick response to DIM query, the upper bound of diverse influence has been found and explored to minimize the seed candidates. In particular, we have designed PSP-Tree index to quickly compute the diverse influence of seed candidates. On the five real-world data sets, the effectiveness of DIM has been verified by comparing with IM and efficiency of our solution has been demonstrated by varying a set of parameters. In short, this study has filled the gap in influence maximization research by breaking through the current limit and it sheds light to social network research where information diffusion is essential.

Finally, we have proposed and formalize the problem of targeted influence minimization in social networks which has not previously been studied. We present different solutions that address the computational challenges associated with this problem. We report on empirical studies showing that the proposed solution is capable of quickly blocking 80% or more the influence of source users on target users. The proposed sampling-based solution is efficient when applied to large-scale social networks. It is critical because the system needs to be able to quickly identify the set of edges to be deleted to block the influence of source users. A less efficient solution may enable the source users to activate additional users as new source users, who can then spread the malicious information and this way influence the target users.

As to future works, we would like to consider the topic information along with the community structure when searching for the most influential community, maximizing the diversity of the social influence, and minimizing the influence on a target community. Since the influence propagation can be measured accurately with the topic information of a spreading message.

Bibliography

R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *Proceedings of the Second International Conference on Web Search and Web Data Mining, WSDM 2009, Barcelona, Spain, February 9-11, 2009*, pages 5–14, 2009. doi: 10.1145/1498759.1498766. URL http://doi.acm.org/10.1145/1498759.1498766.

Ç. Aslay, N. Barbieri, F. Bonchi, and R. A. Baeza-Yates. Online topic-aware influence maximization queries. In *EDBT*, pages 295–306, 2014.

N. Barbieri, F. Bonchi, and G. Manco. Topic-aware social influence propagation models. In *ICDM*, pages 81–90, 2012.

P. Basuchowdhuri and S. Majumder. Finding influential nodes in social networks using minimum k-hop dominating set. In *Applied Algorithms - First International Conference, ICAA 2014, Kolkata, India, January 13-15, 2014. Proceedings*, pages 137–151, 2014. doi: 10.1007/978-3-319-04126-1_12. URL http://dx.doi.org/10.1007/978-3-319-04126-1_12.

C. Borgs, M. Brautbar, J. T. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 946–957, 2014. doi: 10.1137/1.9781611973402.70. URL http://dx.doi.org/10.1137/1.9781611973402.70.

C. Bron and J. Kerbosch. Finding all cliques of an undirected graph (algorithm 457). *Commun. ACM*, 16(9):575–576, 1973.

A. Buluç, H. Meyerhenke, I. Safro, P. Sanders, and C. Schulz. Recent advances in graph partitioning. In *Algorithm Engineering - Selected Results and Surveys*, pages 117–158. 2016. doi: 10.1007/978-3-319-49487-6_4. URL http://dx.doi.org/10.1007/978-3-319-49487-6_4.

S. Chen, J. Fan, G. Li, J. Feng, K. Tan, and J. Tang. Online topic-aware influence maximization. *PVLDB*, 8(6):666–677, 2015.

W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *SIGKDD*, pages 199–208, 2009.

W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *SIGKDD*, pages 1029–1038, 2010.

W. Chen, T. Lin, and C. Yang. Efficient topic-aware influence maximization using preprocessing. *CoRR*, abs/1403.0057, 2014.

Y. Chen, W. Peng, and S. Lee. Efficient algorithms for influence maximization in social networks. *Knowl. Inf. Syst.*, 33(3):577–601, 2012.

E. Demidova, P. Fankhauser, X. Zhou, and W. Nejdl. *DivQ*: diversification for keyword search over structured databases. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, Geneva, Switzerland, July 19-23, 2010*, pages 331–338, 2010. doi: 10.1145/1835449.1835506. URL http://doi.acm.org/10.1145/1835449.1835506.

P. M. Domingos and M. Richardson. Mining the network value of customers. In *SIGKDD*, pages 57–66, 2001.

M. Drosou and E. Pitoura. Diversity over continuous data. *IEEE Data Eng. Bull.*, 32(4): 49–56, 2009. URL http://sites.computer.org/debull/A09dec/drosou-paper1.pdf.

M. Drosou and E. Pitoura. Search result diversification. *SIGMOD Record*, 39(1):41–47, 2010. doi: 10.1145/1860702.1860709. URL http://doi.acm.org/10.1145/1860702.1860709.

D. Eppstein, M. Löffler, and D. Strash. Listing all maximal cliques in sparse graphs in near-optimal time. In *Algorithms and Computation - Part I*, pages 403–414, 2010. doi: 10.1007/978-3-642-17517-6_36. URL http://dx.doi.org/10.1007/978-3-642-17517-6_36.

U. Feige. Approximating maximum clique by removing subgraphs. *SIAM J. Discrete Math.*, 18(2):219–225, 2004. doi: 10.1137/S089548010240415X. URL http://dx.doi.org/10.1137/S089548010240415X.

S. Feng, X. Chen, G. Cong, Y. Zeng, Y. M. Chee, and Y. Xiang. Influence maximization with novelty decay in social networks. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 37–43, 2014. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8485.

M. Girvan and M. Newman. Community structure in social and biological networks. *PNAS*, 99(12):7821–7826, 2002. doi: 10.1073/pnas.122653799.

M. Gomez-Rodriguez and B. Schölkopf. Influence maximization in continuous time diffusion networks. In *ICML*, 2012.

M. Gomez-Rodriguez, D. Balduzzi, and B. Schölkopf. Uncovering the temporal dynamics of diffusion networks. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 561–568, 2011.

A. Goyal, F. Bonchi, and L. V. S. Lakshmanan. Learning influence probabilities in social networks. In *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010*, pages 241–250, 2010. doi: 10.1145/1718487.1718518. URL http://doi.acm.org/10.1145/1718487.1718518.

E. Gregori, L. Lenzini, and S. Mainardi. Parallel (k)-clique community detection on large-scale networks. *IEEE Trans. Parallel Distrib. Syst.*, 24(8):1651–1660, 2013.

X. Huang, H. Cheng, L. Qin, W. Tian, and J. X. Yu. Querying k-truss community in large and dynamic graphs. In *SIGMOD*, pages 1311–1322, 2014. doi: 10.1145/2588555.2610495. URL http://doi.acm.org/10.1145/2588555.2610495.

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

D. Kempe, J. M. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *SIGKDD*, pages 137–146, 2003b.

E. Khalil, B. Dilkina, and L. Song. CUTTINGEDGE: Influence Minimization in Networks. *Cc.Gatech.Edu*, pages 1–13, 2013. URL http://www.cc.gatech.edu/grads/e/ekhalil3/pdfs/CuttingEdge.pdf.

M. Kimura and K. Saito. Tractable models for information diffusion in social networks. In *PKDD*, pages 259–271, 2006.

M. Kimura, K. Saito, and R. Nakano. Extracting influential nodes for information diffusion on a social network. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 1371–1376, 2007. URL http://www.aaai.org/Library/AAAI/2007/aaai07-217.php.

M. Kimura, K. Saito, and H. Motoda. Minimizing the Spread of Contamination by Blocking Links in a Network. *Aaai*, pages 1175–1180, 2008. URL http://www.aaai.org/Papers/AAAI/2008/AAAI08-186.pdf.

J. Lee and C. Chung. A query approach for influence maximization on specific users in social networks. *IEEE Trans. Knowl. Data Eng.*, 27(2):340–353, 2015. doi: 10.1109/TKDE.2014.2330833. URL http://dx.doi.org/10.1109/TKDE.2014.2330833.

J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. M. VanBriesen, and N. S. Glance. Cost-effective outbreak detection in networks. In *SIGKDD*, pages 420–429, 2007.

J. Leskovec, K. J. Lang, and M. W. Mahoney. Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 631–640, 2010. doi: 10.1145/1772690.1772755. URL http://doi.acm.org/10.1145/1772690.1772755.

I. X. Leung, P. Hui, P. Lio, and J. Crowcroft. Towards real-time community detection in large networks. *Physical Review*, 79(6), 2009.

G. Li, S. Chen, J. Feng, K. Tan, and W. Li. Efficient location-aware influence maximization. In *SIGMOD*, pages 87–98, 2014a.

J. Li, X. Wang, K. Deng, T. Sellis, J. X. Yu, and F. Xia. Efficient diverse influence maximization in social networks. *submitted to IEEE Transaction on Knowledge and Data Engineering, under review.*

J. Li, X. Wang, K. Deng, X. Yang, T. Sellis, and J. X. Yu. Most influential community search over large social networks. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*, pages 871–882, 2017. doi: 10.1109/ICDE.2017.136. URL https://doi.org/10.1109/ICDE.2017.136.

R. Li, J. X. Yu, and R. Mao. Efficient core maintenance in large dynamic graphs. *IEEE Trans. Knowl. Data Eng.*, 26(10):2453–2465, 2014b. doi: 10.1109/TKDE.2013.158. URL http://dx.doi.org/10.1109/TKDE.2013.158.

H. Lin and J. A. Bilmes. A class of submodular functions for document summarization. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 510–520, 2011. URL http://www.aclweb.org/anthology/P11-1052.

B. Liu, G. Cong, D. Xu, and Y. Zeng. Time constrained influence maximization in social networks. In *ICDM*, pages 439–448, 2012.

B. Liu, G. Cong, Y. Zeng, D. Xu, and Y. M. Chee. Influence spreading path and its application to the time constrained social influence maximization problem and beyond. *IEEE Trans. Knowl. Data Eng.*, 26(8):1904–1917, 2014. doi: 10.1109/TKDE.2013.106. URL http://dx.doi.org/10.1109/TKDE.2013.106.

Z. Liu, P. Sun, and Y. Chen. Structured search result differentiation. *PVLDB*, 2(1): 313–324, 2009. URL http://www.vldb.org/pvldb/2/vldb09-500.pdf.

C. Luo, K. Cui, X. Zheng, and D. Zeng. Time critical disinformation influence minimization in online social networks. *Proceedings - 2014 IEEE Joint Intelligence and Security Informatics Conference, JISIC 2014*, pages 68–74, 2014. doi: 10.1109/JISIC.2014.20.

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

M. E. Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582, 2006.

M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(026113), 2004.

C. H. Papadimitriou and K. Steiglitz. *6.1 The Max-Flow, Min-Cut Theorem*. Dover, 1998.

G. Qi, C. C. Aggarwal, and T. S. Huang. Community detection with edge content in social media networks. In *ICDE*, pages 534–545, 2012.

U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review*, 76(3), 2007.

M. Richardson and P. M. Domingos. Mining knowledge-sharing sites for viral marketing. In *SIGKDD*, pages 61–70, 2002.

M. G. Rodriguez, J. Leskovec, D. Balduzzi, and B. Schölkopf. Uncovering the structure and temporal dynamics of information propagation. *Network Science*, 2(01):26–65, 2014.

J. Ruan and W. Zhang. An efficient spectral algorithm for network community discovery and its applications to biological and social networks. In *IEEE ICDM*, pages 643–648, 2007. doi: 10.1109/ICDM.2007.72. URL http://dx.doi.org/10.1109/ICDM.2007.72.

V. Satuluri and S. Parthasarathy. Scalable graph clustering using stochastic flows: applications to community discovery. In *ACM SIGKDD*, pages 737–746, 2009. doi: 10.1145/1557019.1557101. URL http://doi.acm.org/10.1145/1557019.1557101.

C. Song, W. Hsu, and M. L. Lee. Temporal influence blocking: minimizing the effect of misinformation in social networks. In *Proceedings of the 2017 IEEE 33rd IEEE International Conference on Data Engineering*. IEEE, 2017.

F. Tang, Q. Liu, H. Zhu, E. Chen, and F. Zhu. Diversified social influence maximization. In *ASONAM*, pages 455–459, 2014a.

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

E. Tomita, A. Tanaka, and H. Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theor. Comput. Sci.*, 363(1):28–42, 2006. doi: 10.1016/j.tcs.2006.06.015. URL http://dx.doi.org/10.1016/j.tcs.2006.06.015.

V. V. Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.

M. R. Vieira, H. L. Razente, M. C. N. Barioni, M. Hadjieleftheriou, D. Srivastava, C. T. Jr., and V. J. Tsotras. On query result diversification. In *Proceedings of the 27th International Conference on Data Engineering, ICDE 2011, April 11-16, 2011, Hannover, Germany*, pages 1163–1174, 2011a. doi: 10.1109/ICDE.2011.5767846. URL http://dx.doi.org/10.1109/ICDE.2011.5767846.

M. R. Vieira, H. L. Razente, M. C. N. Barioni, M. Hadjieleftheriou, D. Srivastava, C. T. Jr., and V. J. Tsotras. Divdb: A system for diversifying query results. *PVLDB*, 4(12): 1395–1398, 2011b. URL http://www.vldb.org/pvldb/vol4/p1395-vieira.pdf.

M. Wang, C. Wang, J. X. Yu, and J. Zhang. Community detection in social networks: An in-depth benchmarking study with a procedure-oriented framework. *PVLDB*, 8(10): 998–1009, 2015. URL http://www.vldb.org/pvldb/vol8/p998-wang.pdf.

S. Wang, X. Zhao, Y. Chen, Z. Li, K. Zhang, and J. Xia. Negative Influence Minimizing by Blocking Nodes in Social Networks. *Workshops at the Twenty- . . .*, pages 134–136, 2013. URL http://www.aaai.org/ocs/index.php/WS/AAAIW13/paper/viewFile/7046/6748.

X. Wang, K. Deng, J. Li, J. X. Yu, C. S. Jensen, and X. Yang. Targeted influence minimization in social networks. In *22nd Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2018, Melbourne, Australia, Proceedings*.

X. Wang, Y. Zhang, W. Zhang, and X. Lin. Distance-aware influence maximization in geo-social network. In *32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016*, pages 1–12, 2016a. doi: 10.1109/ICDE.2016.7498224. URL http://dx.doi.org/10.1109/ICDE.2016.7498224.

Y. Wang, S. Cai, and M. Yin. Two efficient local search algorithms for maximum weight clique problem. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 805–811, 2016b. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11915.

S. White and P. Smyth. A spectral clustering approach to finding communities in graph. In *SIAM- SDM*, pages 274–285, 2005. doi: 10.1137/1.9781611972757.25. URL http://dx.doi.org/10.1137/1.9781611972757.25.

Q. Yao, R. Shi, C. Zhou, P. Wang, and L. Guo. Topic-aware Social Influence Minimization. In *Proceedings of the 24th International Conference on World Wide Web - WWW '15 Companion*, number 1, pages 139–140, New York, New York, USA, 2015. ACM Press. ISBN 9781450334730. doi: 10.1145/2740908.2742767. URL http://dl.acm.org/citation.cfm?doid=2740908.2742767.

C. Yu, L. V. S. Lakshmanan, and S. Amer-Yahia. It takes variety to make a world: diversification in recommender systems. In *EDBT 2009, 12th International Conference on Extending Database Technology, Saint Petersburg, Russia, March 24-26, 2009, Proceedings*, pages 368–378, 2009. doi: 10.1145/1516360.1516404. URL http://doi.acm.org/10.1145/1516360.1516404.

L. Yuan, L. Qin, X. Lin, L. Chang, and W. Zhang. Diversified top-k clique search. In *IEEE ICDE*, pages 387–398, 2015. doi: 10.1109/ICDE.2015.7113300. URL http://dx.doi.org/10.1109/ICDE.2015.7113300.

F. Zhao, X. Zhang, A. K. H. Tung, and G. Chen. BROAD: diversified keyword search in databases. *PVLDB*, 4(12):1355–1358, 2011. URL http://www.vldb.org/pvldb/vol4/p1355-zhao.pdf.

J. H. Zhao, Y. Habibulla, and H. Zhou. Statistical mechanics of the minimum dominating set problem. *CoRR*, abs/1410.4607, 2014. URL http://arxiv.org/abs/1410.4607.