INFLUENCE MAXIMIZATION IN SOCIAL NETWORKS

Kangkang Wu

A thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Computer Science

Concordia University

Montréal, Québec, Canada

November 2015

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **Kangkang Wu**

Entitled: **Influence Maximization in Social Networks**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair

Dr. C. Poulis

_____ Examiner

Dr. T. Fevens

_____ Examiner

Dr. H. Harutyunyan

_____ Supervisor

Dr. L. Narayanan

Approved _____

Chair of Department or Graduate Program Director

_____ 20 _____ _____

Dr. Amir Asif, PhD, PEng

Dean, Faculty of Engineering and Computer Science

Abstract

Influence Maximization in Social Networks

Kangkang Wu

In the social network era, every decision an individual makes, whether it is watching a movie or purchasing a book, is influenced by his or her personal network to a certain degree. This thesis investigates the power of the *word-of-mouth* effect within social networks.

Given a network $G = (V, E, t)$ where $t(v)$ denotes the threshold of node v, we model the spread of influence as follows. Influence propagates deterministically in discrete steps. An influenced node u exerts a fixed amount of influence $b_{u,w}$ on any neighbor w. For any uninfluenced node v, if the total amount of influence it receives from all its already influenced neighbors at time step $t-1$ is at least $t(v)$, node v will be influenced in step t.

Given a social network G, we study the problem of introducing an already activated external influencer v into the network, and choosing links from v to nodes in G that can maximize the spread of influence in G. We study two problems: the *Minimum Links* problem, which is to choose the minimum number of links that can activate the entire network, and the *Maximum Influence* problem, which is to choose k neighbors that will maximize the size of the influenced set. We prove that the Maximum Influence problem is NP-hard, even for bipartite graphs in which thresholds of all nodes are either one or two. We also study both problems in paths, rings, trees and cliques, and we give polynomial time algorithms that find optimal solutions to both problems for all these topologies.

Acknowledgments

First and most importantly, I would like to express my deepest gratitude for my thesis supervisor, Dr. Lata Narayanan. Not only she has provided me with very detailed instructions on how to write a thesis from scratch, but also I have learned from her the attitude of attention to details and self-discipline. I shall carry this attitude to my future endeavors.

Secondly, I want to thank my parents for their unconditional love and continuous support in so many years. It is a big decision of going back to school again after working for a few years, I really appreciate their understanding and support.

I also would like to thank my friends, Puspal Bhabak, Sunidhi Azad, Zhiyuan Li and Lei Cao for all their encouragement and help during this period.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

We live in an increasingly connected century. The average size of an individual's personal network keeps growing, and interaction and information exchange between friends and social contacts is much easier and more frequent with the advent and popularity of social network applications such as Twitter and Facebook. As a result, the study of social network structure and the dynamic process by which information propagates in a social network is a hot topic in the past decade in many areas including sociology, economics and computer science. The massive data sets which can be obtained from social networking web sites facilitate the research in this area.

In the social network era, every decision an individual makes, whether it is watching a movie or purchasing a book, is influenced by his or her personal network to a certain degree. Not surprisingly, social networks have changed traditional marketing theory. Considering only the *intrinsic value* of the customer (the expected profit from direct sales to this customer) is not enough anymore. *Viral marketing* has thus become a very important strategy in promoting new products or ideas nowadays. In viral marketing theory, once a certain fraction of the social network adopts a product, a larger cascade of further adoptions is predictable due to the *word-of-mouth* network effect [17, 31, 4].

Inspired by social networks and viral marketing, Domingos and Richardson [13, 37] were the first to study a customer's *network value*, the expected profit gained from other purchases that may follow if this customer buys the product, from the data mining perspective. They tried to identify the most influential customers in a connected network formed by customers. They raised the following important algorithmic problem in the context of social network analysis: If a company can turn a subset of customers in a given network into early adopters, and the goal is to trigger a large cascade of further adoptions, which set of customers should they target?

From then on, the influence maximization problem has attracted the attention of many researchers in the field of computer science. Kempe, Kleinberg and Tardos in their seminal paper [28], systematically studied the influence maximization problem, including models for influence diffusion process in a social network.

1.1 Modeling the spread of influence in a social network

In order to study the influence diffusion process in social network from an algorithmic perspective, we model the social network by a directed edge-weighted graph $G = (V, E)$ with $V(G)$ representing individuals in the social network and $E(G)$ denoting the social connections. If two individuals u and w are connected in the social network, then edge $\{u, w\} \in E(G)$. Finally, the weight of an edge $b_{u,v}$ denotes the amount of influence exerted by u on v.

There are two main models for the spread of influence in a social network, the *cascade model* and the *threshold model*. In [28], Kempe *et al.* formally defined these two types of models in their general form as well as special cases including *linear*

threshold model and *independent cascade model*. Most of the current influence diffusion models in use for the study of social networks are extensions of these two basic models.

We briefly introduce the so-called *linear threshold model* here, and defer a detailed discussion of influence diffusion models to the next chapter. The threshold model is based on the idea that if enough of a person's neighbors in the social network have adopted an idea or a product or a technology, then this person will also adopt it. We use the term *activated* or *influenced* to denote that a person has adopted the idea under question. Thus, each individual may have a certain *threshold* of resistance to adoption of the product, but once the collective influence exerted by activated neighbors is greater than this threshold, then the individual will be activated.

More precisely, in the linear threshold model, every node v is associated with an threshold $t(v)$. Influence propagates deterministically in discrete time steps. Let $I(G, S, t)$ be the set of influenced nodes at time t, where S is the set of nodes influenced at time 1. We term S the *seed set*. Let $N(v)$ be the set of neighbors of v. For any uninfluenced node v, if the total amount of influence it receives from all its already influenced neighbors at time step $t-1$ is at least $t(v)$, that is, if $\Sigma_{w \in I(G,t-1) \cap N(v)} b_{w,v} \geq t(v)$, then node v will be influenced in step t. A node once influenced stays influenced. That is,

$$I(G, S, t) = I(G, S, t-1) \cup \{v \notin I(G, S, t-1) \mid \Sigma_{w \in I(G,S,t-1) \cap N(v)} b_{w,v} \geq t(v)\}$$

It is easy to see that if S is non-empty, then the process terminates after at most $|V| - 1$ steps. We call the set of nodes that are activated when the process terminates as the *influenced set*, and denote it by $I(G, S)$.

1.2 Seed Set problem

In the previous section, we described a model for the spread of influence in a social network. In viral marketing, as already mentioned, the word-of-mouth effect is a key factor in the success of the promotion of new products. To take advantage of the word-of-mouth effect and maximize the number of adopters of the product, it is important to choose the right early adopters, i.e. the seed set. These early adopters can be persuaded to be activated/adopt the product by *external incentives*, and they in turn will activate other nodes via the influence diffusion process described in the previous section. The number of early adopters that we can choose is clearly limited because of budget limitations.

The problem proposed by Domingo and Richardson [13, 37] can now be formulated as the *seed set problem:* Given a social network $G = (V, E, t)$, and an integer k, find a subset $S \subseteq V$ of size k so that $I(G, S)$ is as large as possible. Notice that k corresponds to the budget, and S is a seed set that maximizes the size of the influenced set. There has been a lot of work on the seed set problem; a detailed literature survey will be given in Chapter 2.

We provide a small example in Figure 1 to illustrate the seed set problem and the propagation of influence. The influenced nodes are shaded dark in the following figures. Consider the problem of finding an optimal seed set of size 3 in the path shown in Figure 1 which has unweighted edges.

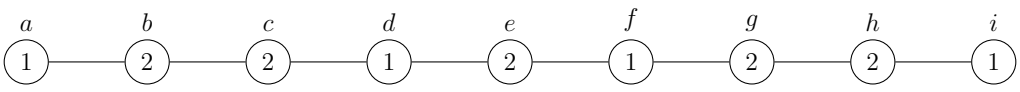

Figure 1: A social network with a path topology. Thresholds of nodes are given inside the circles.

Consider the set $\{b, e, h\}$ as the seed set. In step 1, the nodes $\{b, e, h\}$ are influenced,

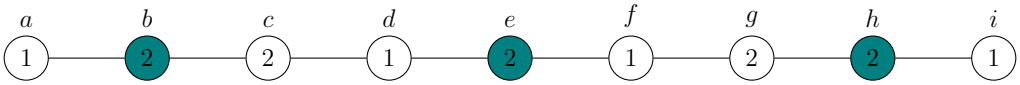

Figure 2: Influenced nodes at time step 1 with $\{b, e, h\}$ as seed set

In step 2, nodes $\{a, d, f, i\}$ are influenced.

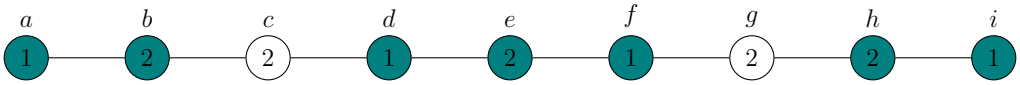

Figure 3: Influenced nodes at time step 2 with $\{b, e, h\}$ as seed set

In step 3, nodes $\{c, g\}$ are influenced.

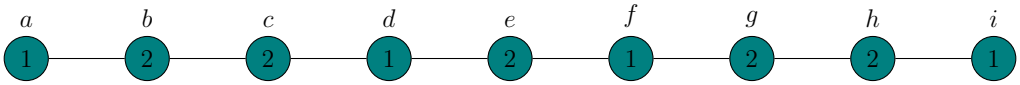

Figure 4: Influenced nodes at time step 3 with $\{b, e, h\}$ as seed set

Since all nodes are eventually influenced, clearly, the set $\{b, e, h\}$ is an optimal seed set in that it influences the maximum possible number of nodes in the network. The interested reader can verify that the seed set $\{b, e, h\}$ is the only seed set of size 3 which can influence all 9 nodes.

1.3 Critique of the seed set problem formulation

There is an interesting hidden assumption in the formulation of the seed set problem. By allowing any node in the graph to be chosen as part of the seed set, we assume that *early adopters can be chosen and activated immediately regardless of their own thresholds, and at the same cost.* This assumption has two flaws.

- **Uniform activation cost assumption**

  Highly influential nodes in the network are usually associated with high thresholds. It is not reasonable to assume that such a node can be activated at the

same cost as a node with a lower threshold. With budget constraints, it may be more cost-effective to reach such nodes through the network effect. Also, in some scenarios, there may exist high influence nodes that can *only be* reached through the word-of-mouth effect, and cannot be activated by some external process. For example, suppose we want to promote an abstract product such as a political idea. At the initial stage, even if we can identify the most influential people in the social network, they are usually the hardest to influence. In this case, the early adopters could be less influential people with lower thresholds, who subsequently influence the people with higher influence through the network effect.

- **Impossibility of giving partial incentives**

  In the specification of the seed set problem, the nodes in the seed set are completely activated immediately by external incentives, and the remaining nodes are activated solely by the network effect. However, with a given budget, rather than completely activating a large set of nodes, we may prefer to activate only a small set of nodes, and provide *partial incentives* to other nodes, which can then be activated sooner by the network effect. For example, instead of giving away free software to 1000 selected customers, a software company could offer 30% off coupons to 5000 selected customers, and free software to far fewer customers. The formulation of the seed set problem does not take into account this type of marketing strategy.

1.4 Problem statement

In this thesis, we propose a new way to model a viral marketing strategy, which addresses the above flaws in the the seed set problem formulation. There are two

main differences in our modeling of a viral marketing strategy from that assumed by the seed set problem:

1. We allow partial incentives: Nodes chosen to be early adopters may not be immediately completely activated; instead they may receive partial incentives that effectively reduce their thresholds, and in turn facilitate later activation by the network effect.

2. Each node has a maximum limit on the amount of partial incentive it may receive. If a node has a threshold higher than this maximum limit, it cannot be activated entirely by external incentives.

The problem we consider is *which nodes to incentivize and by how much* in order to influence the maximum number of nodes. We formalize the problem as follows. Let x be the maximum amount of incentive that can be given to any node, and let k be the maximum budget, that is, the maximum total incentive that can be given to all nodes. Given a social network $G = (V, E, t)$, we introduce a set V' of x additional nodes, which are called *external influencers* and are assumed to be already activated. A *link assignment* is an assignment L of k links from nodes in V' to nodes in V. Given a link assignment P for a graph G, we define $I(G, P)$ to be the set of nodes in $V(G)$ that are activated by P.

Observe that adding a link of weight 1 to a node $v \in V$ is equivalent to giving it external incentive of 1, and is also equivalent to reducing its threshold by 1. Since there are x external influencers, each node $v \in V$ can be the endpoints of at most x links from nodes in V', which corresponds to a maximum external incentive of x per node. Finally, since the link assignment contains at most k links, the constraint on the maximum total budget is also respected.

In this thesis, we consider the following two problems:

Definition 1 Minimum Links (MinLinks) problem: *Given a social network $G = (V, E, t)$, and a set of external influencers V' of size x, find a link assignment P of minimum size that activates the entire network, that is, $I(G, P) = G$.*

Definition 2 Maximum Influence (MaxIn) problem*: Given a social network $G = (V, E, t)$, a set of external influencers V' of size x, and an integer k, find a link assignment P of size k that maximizes $I(G, P)$.*

We study the complexity of the two problems above for general graphs, as well as the special cases when the graph is a path, ring, clique, or tree.

In this thesis, we generally assume that all edges have unit weight; this is generally called the *uniform weight assumption*, and models the situation when every neighbor of an individual has the same influence on him or her with respect to adoption of the product in question, and has been considered in [5, 16, 9, 10]. While our hardness results holds for multiple influencers, we generally consider the case of a single influencer v. This implies that each node in the social network can receive at most one link, that is, it receives a partial incentive of 1, or no incentive at all. This partial incentive may or may not be enough to activate the node. Since each edge in the link assignment/link set has v as one of the endpoints, it can be uniquely specified by the other endpoint. Therefore, for convenience, we usually specify the link set as a subset of V, the nodes in the social network.

Given a graph G, we denote the minimum number of links necessary to activate all nodes in G by $ML(G)$. Given a graph G and an integer k, we denote the maximum number of nodes that can be activated in G by an assignment of k links by $MI(G, k)$.

1.5 Thesis contributions

In this thesis, we give a different model for viral marketing that allows for giving partial incentives and imposes an upper bound on the amount of incentive that can be given to any single node. We introduce and study two new problems: the MinLinks and the Maximum Influence problems.

- We introduce a new to model viral marketing, and propose two related problems: MinLinks problem and Maximum Influence problem.
- We prove that Maximum Influence problem is NP-hard, even for bipartite graphs in which thresholds of all nodes are either one or two.
- We prove necessary and sufficient conditions for the MinLinks problem to have a feasible solution when the social network can be represented by a path, ring, clique, or tree.
- We give $\Theta(n)$ algorithms for the Minimum Links problem for the cases when the social network can be represented by a path, a ring, a clique, or a tree.
- We give a $\Theta(kn)$ algorithm for the Maximum Influence problem with k links for the case when the social network can be represented by a path, a $\Theta(kn^2)$ algorithm when it is a ring, a $\Theta(n)$ algorithm when it is a clique, and a $\Theta(n^2k^2)$ algorithm when it is a tree.

1.6 Thesis outline

In Chapter 2, we give a review of the literature in this area. In Chapter 3, we prove Maximum Influence problem to be NP-hard. In Chapter 4 to Chapter 7, we give polynomial time algorithms for both problems under special graph topologies. In the

last chapter, we summarize our work and conclude with some possible directions for future work.

Chapter 2

Related Work

In this chapter, we review the research work related to the influence maximization problem in social networks. There exist three parameters in the set of problems related to influence maximization in social networks.

1. The budget which restricts the size of the initial seed set
2. The coverage which is the expected size of total influenced nodes
3. The influence propagation time

In general, one or two of these parameters are restricted, and the problem is to optimize the third parameter. For example, in the seed set problem, we restrict the size of the budget (the size of the seed set), and want to maximize the total number of influenced nodes. Alternatively, in the minimum coverage problem which will be discussed in Section 2.4, we want to find a minimum-sized seed set that can influence at least a pre-specified fraction of nodes in the network. Finally, in the influence propagation with latency bound problem which will be discussed in Section 2.5, we are interested in choosing a seed set which maximizes the number of influenced nodes within a pre-specified number of rounds.

This chapter is organized as follows. In Section 2.1, we review the literature on various influence diffusion models. In Section 2.2, we review literatures on how to infer the influence probability over edges. In Section 2.3, we review the literature on Seed Set problem. We review the literature on Minimum Coverage problem and Influence Maximization Problem with Latency Bound in Section 2.4 and 2.5 respectively. In Section 2.6, we review the literature on Fractional Influence Maximization problem. In Section 2.7, we review the literature on influence maximization problem while there exist negative influences. In Section 2.8, we review the literature on Non-progressive influence diffusion models. We conclude this chapter by discussing the difference between our problem with seed set problem and fractional influence problem.

2.1 Influence diffusion model

As mentioned earlier, there are two main influence diffusion models: *threshold model* and *cascade model*.

2.1.1 Threshold model

Granovetter and Schelling [23, 39] were among the first to propose the initial version of the threshold model to capture the dynamic process of information cascading in social networks. The key concept introduced by Granovetter is the notion of *threshold*: the number of other people who must make the same decision before the given actor does so. This threshold is the critical point that the benefits begin to exceed the cost. In the threshold based model, every node v is associated with an threshold θ_v which represents the amount of influence required to influence node v.

In this section, we present some different versions of the threshold model that have been proposed in the literature.

2.1.1.1 Linear threshold model

We have already introduced this model in Chapter 1, but repeat it here for completeness. In the linear threshold model, every node v is associated with an threshold θ_v which is arbitrarily drawn from the interval $[0, 1]$, and all influences coming from different neighbors of v are aggregated linearly. A node v is influenced by its neighbor u to a degree $b_{\mathrm{u,v}}$ which satisfies:

$$\sum_{u \in N(v)} b_{\mathrm{u,v}} \leq 1$$

2.1.1.2 General threshold model

In [28], Kempe *et al.* proposed the general threshold model used for modeling the influence diffusion process. In the general threshold model, every node v is associated with an threshold θ_v which represents the hurdle that must be overcome in order to influence node v. Each node v is also associated with a monotone influence function f_v: $2^V \rightarrow [0, 1]$ that maps all subsets of V to real numbers in $[0, 1]$. $f_v(S)$ denotes the amount of influence node v receives when all nodes in set S have been activated. f_v satisfies $f_v(\emptyset) = 0$. If we assume that only direct neighbors can exert influence on node v, then $f_v(S) = f_v(S \cap N(v))$. Let S be the set of active neighbors of node v at time step t, then only if $f_v(S) \geq \theta_v$, node v will become active at time step $t + 1$.

Clearly, the linear threshold model is a special instance of the general threshold model.

In some papers, the thresholds is selected randomly due to lack of knowledge about each node's threshold. However, there also exist special cases of threshold model which have hard-wired thresholds. We will introduce some hard-wired threshold model shortly.

2.1.1.3 Majority threshold model

The *majority threshold model* is one of the most famous and well-studied threshold based models with fixed thresholds. In majority threshold model, every node has a fixed threshold of 1/2, and an influenced node u exerts a fixed amount of influence $b_{u,w} = 1/d(w)$ on any neighbor w where $d(w)$ is the in-degree of node w, such that an uninfluenced node w becomes active as soon as half of its neighbors become active.

The majority threshold model has practical applications in simulating voting system [36] and other scenarios.

2.1.1.4 Unanimous threshold model

The unanimous threshold model is another well studied threshold model with fixed thresholds. In unanimous threshold model, the threshold of each node is equal to its degree $\theta_v = d(v)$ and an influenced node u exerts a unit influence 1 on any neighbor w.

Under the unanimous threshold model, before a node gets influenced, all its neighbors must be influenced beforehand. Thus, unanimous threshold model is the most influence resistant model among all threshold based models.

The unanimous threshold model has its applications in the network security area.

2.1.2 Cascade model

Dynamic cascade model originated from the research in interacting particle systems in probability theory [14, 33].

In the cascade model, the influence propagation also progresses in discrete steps. At time step 0, only nodes in the seed set become active. At time step t, if a node u is activated in step $t-1$, it is given a chance to activate all of its inactive neighbor $v \in N(u)$. It activates each neighbor v with probability $P_{\mathrm{u,v}}$. Regardless whether

u succeeds or not, it cannot make any more attempts in the subsequent rounds. If multiple neighbors of an inactive node v become active at time step t, they would attempt to activate v in an arbitrary sequential order and the probability that v is activated does not depend on the order in which these attempts are made. If u succeeds in activating an inactive neighbor v at time step t, then at time $t+1$, node v is also active. The activation process terminates when there are no more newly activated nodes.

2.1.2.1 General cascade model

In [28], Kempe *et al.* proposed the general cascade model for modeling the influence diffusion process. Let $P_{\mathrm{u,v}}$ be the probability that node u succeeds in activating a neighbor v, $P_{\mathrm{u,v}}$ is not constant in the general cascade model, but depends on the set of v's neighbors that have already tried and failed to activate v.

At time step t, suppose u is a newly activated neighbor of an inactive node v and S is the set of active neighbors of v which have failed to activate v, then the probability that u succeeds in activating v is defined to be $P_v(u, S)$.

In the general cascade model, each node v is associated with an incremental function $p_v(u, S)$ that maps u and S to real numbers in $[0, 1]$.

2.1.2.2 Independent cascade model

The independent cascade model was proposed by Goldenberg, Libai, and Muller in [17, 18]. There are two ways in which the probability that a node u will be activated is independent.

1) The probability $P_{\mathrm{u,v}}$ by which a active node u can successfully activate an inactive neighbor v is constant, independent of the previous activation history.

2) If multiple neighbors of an inactive node u become active at time step t, the sequential order by which these nodes attempt to activate u does not affect the probability for u to be activated.

Clearly, the independent cascade model is a special case of the general cascade model.

2.1.3 Equivalence between threshold model and cascade model

In [28], Kempe, Kleinberg, and Tardos proved that the general threshold model and general cascade model are in fact equivalent models.

They provide the following equations to transform a threshold based network into cascade network.

$$P_v(u, S) = \frac{f_v(S \cup \{u\}) - f_v(S)}{1 - f_v(S)} \tag{1}$$

Conversely, consider a node v in the cascade model, and let set $S = \{u_1, ..., u_k\}$ be v's neighbors, and let $S_i = \{u_1, ..., u_i\}$. They provide the following equations to transform a cascade network into threshold based network.

$$f_v(S) = 1 - \prod_{i=1}^{k}(1 - p_v(u_i, S_{i-1})) \tag{2}$$

2.2 Learning Influence Probabilities

In this section, we review the literature on how to infer the influence probabilities.

In the Independent Cascade model, we need to know the influence probability $P_{u,v}$ on edge (u, v) to determine the chances that v will be activated once u has

been activated. However, normally we don't have much information on the influence probabilities over the edges, but instead we need to make assumptions about these probabilities. The common techniques for addressing the lacking of accurate influence probability include: (1) *Uniform model* adopting uniform weight for all the edges(such as 0.1) (2) *Trivalency model* taking influence probability over edges uniformly at random from a small set of constants such as {0.1,0.05,0.01} (3) *Weighted Cascade model* this model was first proposed in [28] and is the most widely used model. In weighted cascade model, $P_{u,v} = 1/d(v)$ where $d(v)$ is the in-degree of node v.

In the Linear Threshold model, similar to *weighted cascade model*, the most common assumption is : the threshold of node v is randomly drawn from $[0, 1]$ and an influenced node u exerts influence $b_{u,v} = 1/d(v)$ on neighbor v.

Recently, there have been some studies on learning the influence probabilities over edges from real data on past propagation traces. Saito *et al.* in [38] are among the first to investigate how to learn influence probabilities from action logs. They estimated the diffusion probabilities using the *EM algorithm* from an observed data set of information diffusion under Independent Cascade model. In [19], Goyal *et al.* studied this problem: How to build a influence model from a social graph and logs of actions by its users? They developed algorithms for learning the parameters of influence models by taking a social graph and action logs as input under the General Threshold model. In [20], instead of making assumptions on the influence probabilities over the edges, Goyal *et al.* proposed a new social influence model called *credit distribution model* which leverages available propagation traces to learn how influence flows in the network and used this model to estimate the expected influence spread.

2.3 Seed Set Problem

In this section, we review the literature on the seed set problem.

The seed set selection problem is naturally inspired by research in marketing strategies. Domingos and Richardson [13, 37] were the first to formulate the seed set problem: Given data of a social network, with budget sufficient enough to target k customers, how to choose k customers from the network in order to have the maximal further adoptions? We can come up with several natural heuristics for solving this seed set selection problem, for example in [28, 29]:

1. High-degree heuristic, choose nodes in the order of decreasing degree
2. Distance centric heuristic, choose nodes in the order of their centrality

In [28, 29], Kempe, Kleinberg, and Tardos considered the seed set problem as a discrete optimization problem. They showed that the seed set problem is NP-hard in both the linear threshold model and the independent cascade model. Besides, they proved a very important property of the seed set problem, the submodularity property which will be discussed in the next part.

2.3.1 Submodularity property for set function

Given a set N and a function f: $2^N \to R$ that maps all subsets of N to a real number, we define the following properties for set function f.

1. *Normalized:* if $f(\emptyset) = 0$, then f is normalized;
2. *Monotone:* if $f(S) \leq f(T)$ for any $S \subseteq T \subseteq N$, then f is monotone;
3. *Submodular:* if $f(S \cup \{x\}) - f(S) \geq f(T \cup \{x\}) - f(T)$ for any $S \subseteq T \subseteq N$, then f is submodular;

A submodular function exhibits a natural *diminishing returns* property which means that the marginal gain from adding an element x to a set S will not be less than adding x to a superset T of S. Submodular functions have wide applications in optimization algorithms, approximation algorithms, game theory and machine learning.

Suppose we have a monotone and submodular set function f and we want to find a set S with size k for which $f(S)$ shall be maximized. Nemhauser, Wolsey, and Fisher in [11, 35] proved that a natural and simple greedy heuristic, in which you start with an empty set, and repeatedly add an element that gives current maximum marginal gain, approximates the optimum result to within a factor of $(1 - 1/e)$.

2.3.2 Submodularity property for seed set problem

For the seed set problem, let $\sigma(\cdot)$ denote the number of nodes that can be influenced by a seed set S where $\sigma(S) = I(G, S)$.

In [28], it was proved that in both the independent cascade model and the linear threshold model, the influence capability function $\sigma(\cdot)$ is both submodular and monotone.

Theorem 1 ([28], Theorem 2.2 and Theorem 2.5) *For an arbitrary instance of the independent cascade model or linear threshold model, the resulting influence function $\sigma(\cdot)$ is submodular.*

Therefore, the seed set problem admits a greedy hill-climbing algorithm with $(1 - 1/e)$-approximation guarantee. Based on the submodularity property for seed set problem, Kempe *et al.* in [28] proposed a greedy algorithm with $(1 - 1/e)$ approximation guarantee for selecting a seed set of size k which could maximize the number of influenced nodes in the network. This greedy algorithm in shown in Algorithm 1.

Theorem 2 ([28], Theorem 2.1) *For a non-negative, monotone submodular function f, let S be a set of size k obtained by selecting elements one at a time, each time choosing an element that provides the largest marginal increase in the function value. Let S^* be a set that maximizes the value of f over all k element sets. Then $f(S) \geq (1 - 1/e) * f(S^*)$; in other words, S provides a $(1 - 1/e)$-approximation.*

Algorithm 1 Seed set selection(k, $\sigma()$) - Proposed by Kempe *et al.*

1: Input: (1) k- the size of seed set (2) influence function $\sigma()$
2: Output: $S \leftarrow$ seed set
3: $S \leftarrow \emptyset$
4: **while** $k > 0$ **do**
5: $\quad u \leftarrow Max_{u \in V \setminus S}$ ($\sigma(S \cup \{u\})$ - $\sigma(S)$)
6: $\quad S \leftarrow S \cup u$
7: **return** S

This greedy algorithm significantly out-performs the high-degree heuristic and distance centric heuristic according to simulations conducted by Kempe *et al.* One major reason is that both high-degree heuristic and distance centric heuristic ignore the network effect. However, this greedy algorithm in [28] used time consuming Monte-Carlo simulations to estimate the influence capability $\sigma(S_0)$ for a given seed set S_0. Even finding a small seed set in a moderately large network of ten thousand nodes could take days to complete on a modern computer using Algorithm 1. This gives room for future optimization and triggers a lot of interesting follow-up work, either to reduce the number of influence capability evaluations required by Algorithm 1, or propose new heuristics that can speed up the process while still achieving good performance.

In [32], Leskovec *et al.* used an optimization technique called "lazy evaluation" which could significantly reduce the number of evaluations of $\sigma(S_0)$ while producing

the same result as Algorithm 1. They named the greedy algorithm adopting lazy evaluations as *CELF* and *CELF* is 700 times faster than Algorithm 1.

In [7], Chen *et al.* first gave further improvement to Algorithm 1, proposed an algorithm the running time of which is 15% to 34% faster than the CELF algorithm for the same input. Later, they designed a new heuristic terms as *degree discount heuristic* for the seed set problem which nearly matches the performance of Algorithm 1 but orders of magnitude faster. In [8], Chen *et al.* proved that the influence capability $\sigma(S_0)$ can be computed in linear time in directed acyclic graphs (DAGs). By constructing a local DAG for every node in the network and taking advantage of the previous property of DAGs, they proposed a scalable and efficient heuristic algorithm called *LDAG* for the seed set problem under the Linear Threshold model which can scale to networks with millions of nodes and edges. In 2011, Goyal *et al.* in [22] proposed an algorithm called *CELF++* for the seed set problem and empirically showed that CELF++ is 35-55% faster than CELF. The latest work in this direction is [3] by Borgs *et al.* They introduced a pre-processing step which generates a sparse, undirected hypergraph representation of the underlying graph G and solved the maximum influence problem on the sampled hypergraph, which greatly reduces the time complexity of Algorithm 1.

After Kempe *et al.* proved that the influence function $\sigma(\cdot)$ is submodular and monotone in both Independent Cascade model and Linear Threshold model in [28], Mossel and Roch in [34] positively resolved a conjecture in [28] whether the influence function $\sigma(\cdot)$ is also submodular and monotone in general threshold model. Mossel and Roch proved that if every threshold function $f_v(S)$ for each node v is submodular, then this type of general threshold model also admits the the same $1 - 1/e$ approximation guarantee by using a similar greedy hill-climbing algorithm, which indicates that the *local* submodularity is preserved *globally* .

2.4 Minimum Coverage problem

In this section, we review the literature on the minimum coverage problem.

Minimum coverage means finding a minimum-sized seed set which is guaranteed to influence a fixed fraction of nodes in the network. Minimum coverage problem is a natural extension of the original seed set problem proposed in [13, 37]. Instead of finding an optimal seed set of fixed size, they are interested in finding the minimum-sized seed set which could influence a specified fraction of nodes. Chen formally defined this problem as *Target Set Selection* problem in [5] and he studied this problem using threshold model.

Definition 3 ***Target Set Selection problem:*** *Given a connected undirected graph $G = (V, E)$ where each node $v \in V$ is associated with a threshold $t(v)$, $1 \leq t(v) \leq d(v)$, find a minimum size seed set $S \subset V$ such that activating S would lead to the adoption of the product by all nodes (or a fraction of nodes) in the graph.*

Chen proved that the target set selection problem cannot be approximated within a ratio of $O(2^{log^{1-\epsilon}n})$ for any constant $\epsilon > 0$ for a general graph G. Chen further studied this problem under special threshold models. In the majority threshold model, target set selection problem has the same hardness of approximated ratio as general threshold model. In small threshold model in which all the thresholds are small constants and the edge has uniform weight of 1, even if all nodes has threshold ≤ 2, approximating the target selection problem is as hard as the general threshold model. However, when every node has threshold 1, this problem is trivial to solve.

Chen′s work inspires another direction for influence maximization research. He *et al.* studied a similar minimum-sized seed set selection [26]. They proved that minimum coverage problem is also NP-hard in Independent Cascade model.

2.5 Influence Maximization with Latency Bound

In this section, we review the literature on influence maximization problem with latency bound.

As we know, the influence propagation process in $G = (V, E)$ terminates in at most $n - 1$ steps where $n = |V|$. In all previous research, propagation time has not been taken into account. In reality, people are not only interested in finding an optimal seed set of fixed size, they are also concerned about propagation time. In marketing, quickly passing the product information to every desired customer is of great importance, especially when there exist similar competing products.

In [21], Goyal *et al.* proposed a problem termed as *MINTIME* problem. In the *MINTIME* problem, a fixed budget k and a coverage threshold η is given as input. The goal is to find a seed set with size no larger than k and can activate at least η nodes in the smallest number of propagation rounds.

In [9, 10], inspired by a viral campaign for an online video game, Cicalese *et al.* formulated several propagation time related problems. They are interested in finding a bounded size seed set which can influence the maximum number of nodes within bounded rounds. The first problem is termed as *Maximally influencing set problem:* Given a graph $G = (V, E, t)$ where t denotes the threshold function, together with budget k and latency bound λ, the goal is to find a seed set $S \subseteq V$, such that $|S| \leq k$ and $|I(G, S, \lambda)|$ is as large as possible. The second problem is *(λ, β, α)- Target Set Selection problem:* Given a graph $G = (V, E, t)$, together with budget β and latency bound λ plus activation requirement α. The goal is to find a seed set $S \subseteq V$, such that $|S| \leq \beta$ and $|I(G, S, \lambda)| \geq \alpha$.

It is interesting to observe that when $\lambda = 1$, $t(v) = 1$ for each node $v \in V$ and each edge has uniform weight of 1, maximally influencing set problem is reduced to domination problems in graphs. Although the maximally influencing set problem in

NP-hard in general setting, Cicalese *et al.* gave an polynomial time solution for trees.

2.6 Fractional Influence Maximization problem

In this section, we review the literature on fractional influence maximization problem.

Gunnec and Raghavan [24, 25] are the first to investigate fractional incentives in social network influence maximization area for a problem realted to product design in marketing. They formulate a problem called *Least Cost Influence* **(LCIP)** problem which can be viewed as fractional counterpart of the *Minimum Coverage problem.*

Definition 4 ***LCIP problem:*** *Given an undirected graph $G = (V, E, t)$, we define $d_{i,j}$ to be the amount of influence node i exerts on node j when i is influenced, and we define P_v to be the amount of external influence node v receives and $I(G, t)$ to be the set of influenced nodes at time step t. Node v would be influenced at time step t only if $P_v + \sum_{j \in (V \cap I(G,t-1))} d_{j,v} \geq t(v)$. The goal is to use the minimum total incentives $\sum_{v \in V} P_v$ while guaranteeing that a fixed fraction of nodes $\alpha|V|$ will be influenced.*

They proved that LCIP problem is NP-hard if the edge has non-uniform weight under the linear threshold model. They proposed a linear time algorithm for trees assuming each neighbor of node v exerts uniform weight on v.

In [12], Demaine *et al.* proposed and studied the *Fractional Influence problem* which can be viewed as the fractional counterpart of the *seed set problem.* In the fractional influence problem, for a node v with threshold $t(v)$, the designer is free to give partial external incentives ranging from 0 to $t(v)$ to node v, instead of assuming that activating different nodes has the same cost. They studied this problem under the general threshold model.

Definition 5 ***Fractional influence problem:*** *Given an undirected graph $G =$*

(V, E, t), let X be a vector of size n where X_v denotes the amount of external influence node v receives and let $f_v(S)$ denote the amount of influence node v receives when all nodes in set S have all been activated. We define $I(G, t)$ to be the set of influenced nodes at time step t. Node v would be influenced at time step t only if $X_v + f_v(I(G, t)) \geq t(v)$. The goal of the problem is to find an assignment satisfying $\sum_{v \in V} X_v \leq k$ where k represents the budget, while maximizing the total size of influenced nodes.

In the seed set problem, the designer can only give either 0 or $t(v)$ incentives to node v. The solution for the seed set problem can be seen as a vector $\mathbf{x} \in \{0, 1\}^n$ indexed by v where $\mathbf{x}_v$ is either 0 or 1. If $\mathbf{x}_v = 1$, then node v is in the seed set and be influenced in time step 1. It is easy to observe that the seed set problem can be viewed as the integral counterpart for the fractional influence problem.

For some problems such as the Knapsack problem, the fractional version is easier to solve than its integral counterpart. But Demaine *et al.* in [12] proved that the fractional influence problem is also NP-complete, by using reduction from the *Independent Set problem*. Besides, they proved that the submodularity property still holds true in the fractional influence problem. They concluded that this problem also admits a natural greedy algorithm with $(1 - 1/e)$ -approximation ratio and proposed a greedy heuristic for the fractional influence problem. With simulations on real-life social networks, they showed that the fractional influence model performs substantially better than the integral counterpart under the same budget constraint. Furthermore, they proved that it is NP-hard to approximate the factional influence problem to within any factor better than $1 - 1/e$.

2.7 Influence maximization in the presence of negative opinions

In this section, we review the literature on influence maximization problem while there exist negative opinions.

In previous discussions, we consider only positive influence propagation in the network. However, for the sane product, some customers may propagate negative influences in the network.

In [6], Chen *et al.* proposed a model called *IC-N* which is based on the independent cascade model, and this model incorporates the emergence and propagation of negative opinions. Each node in the network has three states, neutral, positive, and negative. Node v is said to be activated at time t if it becomes positive or negative at time t and was neutral at time $t-1$. *IC-N* model introduced a *quality factor* q to denote the probability that a node stays positive after it is activated by a positive neighbor. Initially only nodes in chosen seed set S are active, and for each node $v \in S$, it has probability q to stay positive and with probability $1-q$ to become negative. At time $t > 0$, every active node v (both positive and negative) has a chance to activate its neutral node with an independent probability of $p_{v,u}$. The problem is to find a seed set of size k which can activate most positive nodes. They showed that *IC-N* model maintains some nice properties such as submodularity, thus the greedy approach still applies. Besides, they give a heuristic algorithm called *MIA-N* which is much more efficient than the greedy hill-climbing approach in practical.

2.8 Non-Progressive Influence Diffusion

In this section, we review the literature on Non-Progressive influence diffusion model.

Most previous work on modeling the influence diffusion process has assumed that

once a node is influenced, it will remain active for all subsequent rounds. This type of influence diffusion can be categorized as *progressive model* [28]. However, this assumption is unrealistic in many scenarios where nodes can switch between inactive and active states and vice versa. For example, you will recommend an interesting movie to your friends in a few days after you watching it, but you will not remain active for a long time span. *Non-progressive influence diffusion* models have been widely used in epidemic and economical modeling [15, 27].

In [28], Kempe *et al.* proposed a Non-progressive threshold model. In their model, the influence diffusion process is analogous to the progressive model, except that at each step t, each node v will choose a new threshold θ_v^t. Node v will be active in step t if $f_v(S) \geq \theta_v^t$.

Recently, there has been some work dedicated to influence maximization on non-progressive models. In [16], Gargano *et al.* proposed a new non-progressive threshold model in which agents have a limit memory span λ which means at time t, only influenced received in time span $[t - \lambda, t]$ can be aggregated and influence received before $t-\lambda$ is lost. In this model, all edges have uniform weight of 1, and the threshold of node v is randomly chosen in range $[1, d(v)]$. Using this model, they studied the following problem: given a network $G = (V, E, t)$, and a time window size λ, find a small set of nodes that can influence the whole graph. They proved that this problem is hard to approximate within a poly-logarithmic factor for general graphs, and gave polynomial time algorithm for trees.

Sometimes, multiple companies will be competing to promote similar products in the network, the customer can switch between different adoptions. In this type of non-monopolistic settings, progressive influence diffusion models are not sufficient to describe how influence propagates within a social network. Instead, some non-progressive models can be used in non-monopolistic setting.

In [2], Bharathi *et al.* studied the case that multiple innovations are competing within a social network. They proposed a new model which is an extension on the Independent Cascade model for modeling the diffusion process of multiple innovations in a social network. If the last agent has know which subset of nodes its opponent has chosen, they gave a $(1 - 1/e)$ approximation algorithm for the last agent to find a subset of nodes in the network as seed set.

2.9 Discussion

Of all these problems, our thesis is most closely related to the seed set problem and the fractional influence problem. We elaborate this section by discussing the difference between the Maximum Influence problem and the seed set problem, together with the fractional influence problem.

2.9.1 Difference between Seed Set problem and Maximum Influence problem

In order to show the difference between seed set problem and the Maximum Influence problem, we provide a small example.

Now consider the problem of finding an optimal link set of size 3 for the same path as the one in Figure 1. We use node v to denote the existence of a single external influencer in this thesis.

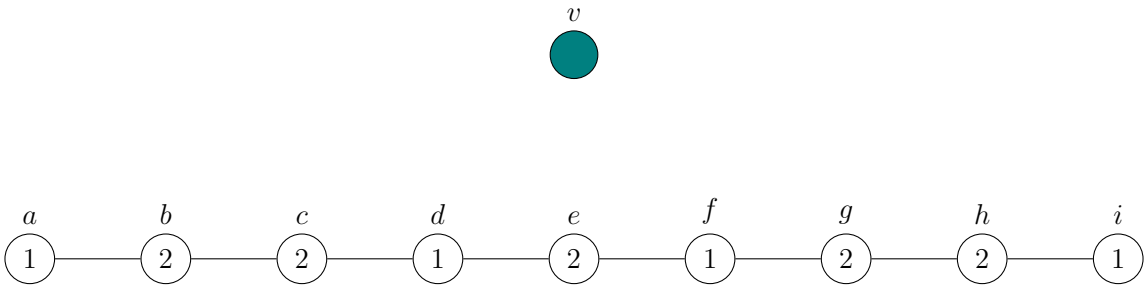

Figure 5: Maximum Influence problem for the same path as Figure 1

Observe that if the influencer links to the same set of nodes $\{b, e, h\}$ as in Figure 4, the total number of influenced nodes would be 0 instead of 9.

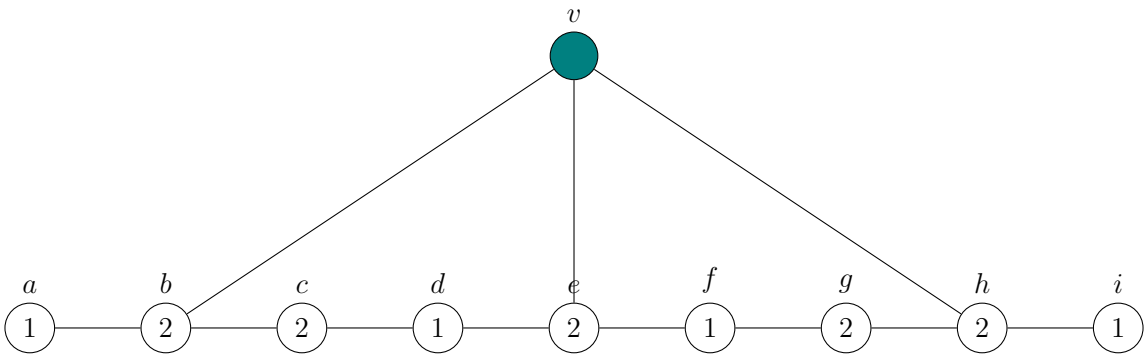

Figure 6: Influenced nodes with $\{b, e, h\}$ as link set

Instead, with 3 links, the maximum number of nodes we can activate is 4. If the influencer links to the set $\{c, d, f\}$, the influence diffusion process proceeds as follows:

In step 0, the influencer sets links to nodes $\{c, d, f\}$.

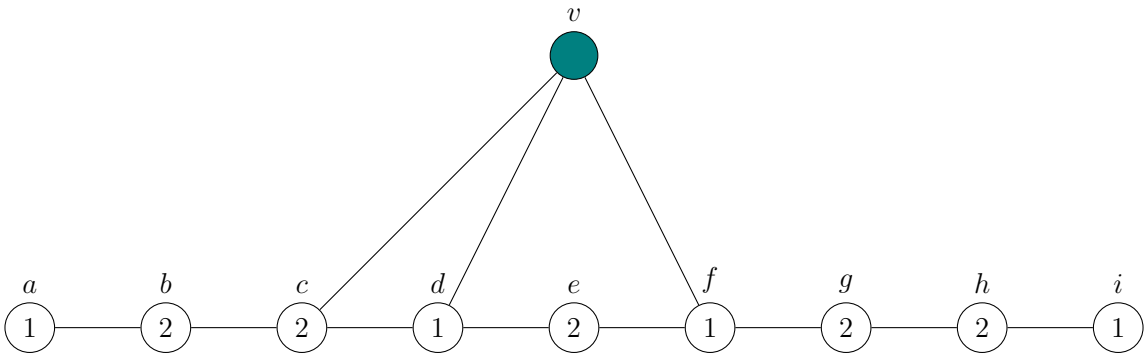

Figure 7: Influenced nodes at time step 0 with $\{c, d, f\}$ as link set

In step 1, nodes $\{d, f\}$ are influenced.

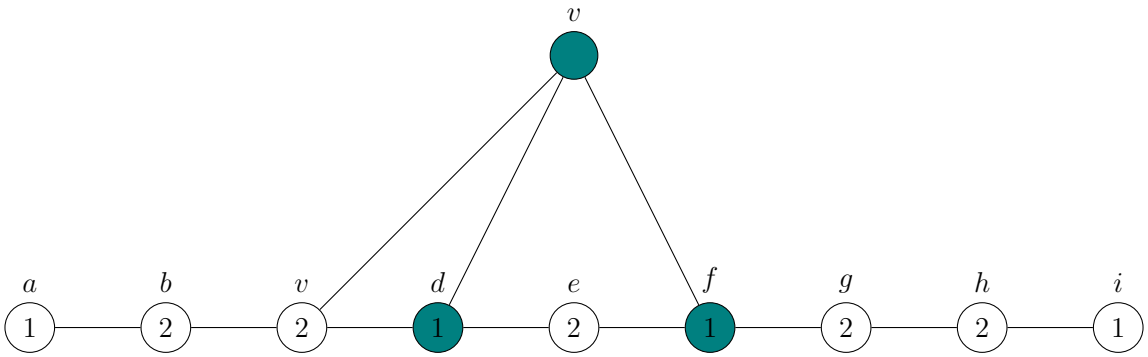

Figure 8: Influenced nodes at time step 1 with $\{c, d, f\}$ as link set

In step 2, nodes $\{c, e\}$ are influenced.

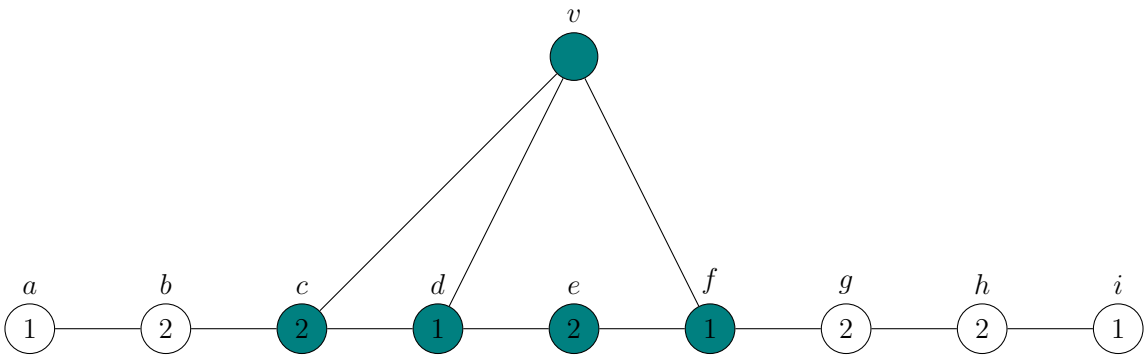

Figure 9: Influenced nodes at time step 2 with $\{c, d, f\}$ as link set

It is easy to verify that the set $\{c, d, f\}$ is an optimal link set of size 3, and it is not an optimal seed set. We conclude that an optimal link set is not an optimal seed set and vice versa. In the seed set problem, we generally prefer to choose high influential nodes (nodes with highest degree, nodes in the center, nodes with high thresholds) first. However, the same node selection strategy does not work in the Maximum Influence problem, as these nodes cannot be activated immediately.

2.9.2 Difference between Fractional Influence problem and Maximum Influence problem

Fractional influence problem can be viewed as a special case of the *Maximum influence problem*. That is if we consider a Maximum influence problem in which the number of external influencers is equal to the highest threshold in the network, the problem will be an instance of the fractional influence problem. This is because in this Maximum influence problem, there is no upper bound on the amount of influence that can be given to a single node, other than that given by the total budget.

Given a fixed budget k, the solution for the fractional influence maximization problem is a vector $\mathbf{x}$ indexed by node v satisfying $\sum_{v \in V} \mathbf{x}_v \leq k$ while maximizing the size of influenced nodes at the same time.

Given the same budget k and the number of influencers x, the solution for the Maximum Influence problem is a vector $\mathbf{x}$ indexed by node v satisfying $\sum_{v \in V} \mathbf{x}_v \leq k$ and $\mathbf{x}_v \leq x$ while maximizing the size of influenced nodes at the same time.

The authors of [12] focused on proving the submodular property of the fractional influence model and the hardness of solving fractional influence maximization problem as general. Their reduction from *Independet Set problem* does not work for the case of undirected graph while we proved that the Maximum influence problem is NP-hard in undirected graphs. Besides, they did not identify or propose any algorithm for the polynomial time solvable cases of the fractional influence maximization problem. We studied the polynomial time solvable cases for both the Minimum Links problem and Maximum Influence problem, and gave polynomial time solution for paths, rings, trees and cliques.

Chapter 3

NP-Hardness of Maximum Influence problem

In this chapter, we will prove that the Maximum Influence problem is NP-hard even for bipartite graphs in which thresholds of all nodes are either one or two. First, we give the decision versions for both the Minimum Links problem and Maximum Influence problem:

Definition 6 ***Minimum Links problem:*** *Given a social network $G = (V, E, t)$, a set of external influencers A, and an integer k, is there a link assignment S of size k that activates the entire network, that is, $I(G, S) = G$?*

Definition 7 ***Maximum Influence problem:*** *Given a social network $G = (V, E, t)$, a set of external influencers A, and integers k and p, is there a link assignment S of size k such that $|I(G, S)| = p$?*

Clearly, if there is a polynomial time algorithm for the Maximum Influence problem, the Minimum Links problem is also polynomial time solvable. We proceed to prove that the Maximum Influence problem is NP-hard. The complexity of Minimum Links problem remains open.

Theorem 3 *The Maximum Influence problem is NP-hard, even for bipartite graph with all nodes having threshold 1 or 2 and having only a single influencer.*

Proof. We show the hardness of the Maximum Influence problem by reducing from the *Max Clique problem*: Given a graph $G = (V, E)$ and an integer k, does G contain a clique of size at least k?

Given an instance of the Max Clique problem (G, k), we construct a bipartite graph $G' = (L_1 \cup L_2, E', t)$ as follows. For every node $v \in V$, we create a corresponding node v of threshold 1 in L_1. For every edge $\{u, v\} \in E$, we create a corresponding node (uv) of threshold 2 in L_2. Next, for every edge $\{u, v\} \in E$, we create the edges $(u, (uv))$ and $(v, (uv))$.

Figure 10 illustrates the reduction.

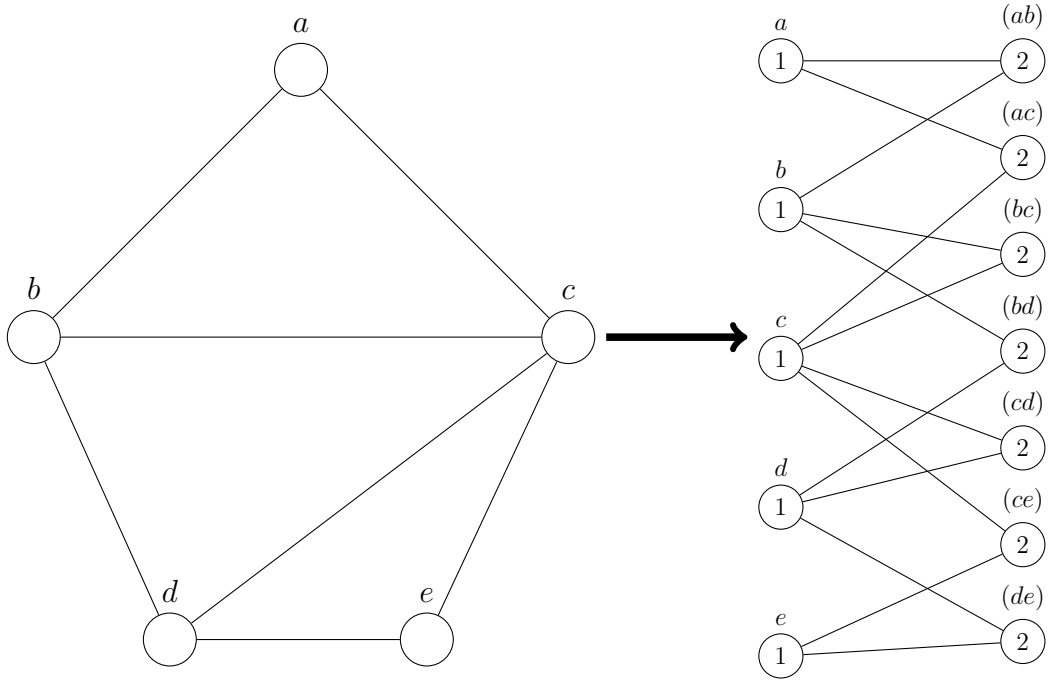

Figure 10: Reduction process from G to G'

We now show that G has a clique of size k if and only if G' has a link set of size k that can activate at least $C_k^2 + k$ nodes. To do this, we first show that it suffices to

consider link sets that contain only nodes in L_1.

Claim 1 *For any link set $T \subseteq L_1 \cup L_2$, there exists a set $S \subseteq L_1$ such that $|S| \leq |T|$ and $|I(G', S)| \geq |I(G', T)|$.*

Proof. Consider a node $v \in T \cap L_2$ that is connected to v_1 and $v_2 \in L_1$. We argue that we can either remove the link assigned to v, or assign that link to some other node in L_1 while not decreasing the size of the influenced set. The following five cases about the time of activation of v are exhaustive.

Case 1: v is not activated in the final round: We can simply remove the link assigned to v.

Case 2: v is activated before v_1 and v_2: Since v has threshold 2 while v_1 and v_2 are both of threshold 1, this is impossible with a single influencer, as the link v receives can only reduce its threshold by 1.

Case 3: v is activated after v_1 is activated and before v_2 is activated: We can move the link assigned for v to v_2, the same set of nodes will be activated eventually.

Case 4: v is activated after v_2 and before v_1: similar as the previous case, we can move the link assigned for v to v_1.

Case 5: v is activated after v_1 and v_2 are activated: The link assigned to v is unnecessary and can be removed.

■

The above claim shows that it suffices to consider only link set $S \subseteq L_1$.

Claim 2 *G' has a link set $S \subseteq L_1$ with $|S| \leq k$ and $|I(G', S)| \geq C_k^2 + k$ if and only if G has a clique of size k.*

Proof. Suppose first that G has a clique V' of size k. We claim that $V' \subseteq L_1$ is a link set such that $|I(G', V')| = C_k^2 + k$. Clearly, all nodes in V' are activated at round 1. Then since V' is a clique in G, it is easy to see that all nodes in L_2 corresponding to the C_k^2 edges between nodes in V' will be activated in round 2. This proves that $|I(G', V')| \geq C_k^2 + k$.

Suppose next that there is a link set $S \subseteq L_1$ such that $|I(G', V')| = C_k^2 + k$, then we claim that the corresponding set for S in G forms a clique of size k.

We claim that given $S \subseteq L_1$ as a link set, it is impossible to activate any new node $\in L_1$ except nodes in S and for any node $(uv) \in L_2$ that gets activated, it must be that $u \in S$ and $v \in S$.

Suppose there exists a node $d \in L_1 - S$ such that d is eventually activated. Since d has threshold 1 and d doesn't receive a link, in order for d to be activated, one of d's neighbors must be activated first. Let us say d is connected to (dx) and (dx) is activated before d. Node (dx) is of threshold 2, thus in order for (dx) to be activated, both d and x must be activated before (dx), a contradiction.

For the second part, suppose there exists a node $(uv) \in L_2$ such that $u \notin S$ and (uv) gets activated eventually. But from the previous analysis, we know that u can never be activated. Therefore (uv) cannot be activated either.

We have shown that with a set $S \subseteq L_1$ of size k, only nodes in S can be activated for the nodes in L_1; only those nodes which are connected to two nodes in S can be activated for nodes in L_2. If $|I(G', S)| \geq C_k^2 + k$, then for every pair of nodes in S, they must be connected in G, thus S must be a clique of size k in G. ■

Therefore, this completes the proof of the theorem.

■

Chapter 4

Paths

In the previous chapter, we showed that the Maximum Influence problem is NP-hard even for bipartite graphs in which thresholds of all nodes are either one or two. In this chapter, we show that both the Minimum Links problem and the Maximum Influence problem are polynomial time solvable for paths.

Let $P_n = (V, E, t)$ be a path with n nodes, $V = \{1, 2, ..., n\}$, $E = \{(i, (i+1)) \mid 1 \leq i \leq n-1\}$, and $t : t(v) \rightarrow \mathcal{Z}^+$. For $1 \leq i \leq j \leq n$, we define $P_{i,j}$ to be the sub-path of P_n consisting of all nodes in $\{i, \ldots, j\}$. We say node v *receives a link* or that we *give a link* to node v, when we add a link from the influencer to v.

4.1 Minimum Links problem

In this section, we study the Minimum Links problem in paths. Let $MinLinks(P_n)$ be the problem of finding a minimum-sized link assignment that activates the entire path, and let $ML(P_n)$ be the minimum number of links needed to activate the entire path.

First observe that if all nodes in the path have threshold one, then by giving a link to *any* of the nodes, we can activate the entire path. However, there are situations

in which it is impossible to activate the entire path, for example, if any node has threshold greater than three. We now show a necessary and sufficient condition for the Minimum Links problem to have a feasible solution:

Proposition 1 *The Minimum Links problem has a feasible solution on $P_n = (V, E, t)$ if and only if one of the following two conditions is met:*

1. *There is no node in P_n with threshold ≥ 3, and there exists a node with threshold 1.*

2. *There exists a node v with threshold 3, and both $P_{1,i-1}$ and $P_{i+1,n}$ have feasible solutions.*

Proof. First we consider the case when there is no node in P_n with threshold ≥ 3. If there is no node with threshold 1 and all nodes have threshold 2, clearly, even if all nodes in P_n receive links, no node can be activated. Conversely, suppose there does exist a node i with threshold 1. Then by giving a link to i, it is activated, and by the cascade effect, the threshold on both $i-1$ and $i+1$ (if they exist) is effectively reduced. The remaining sub-paths are shorter paths each with at least one node of threshold 1. Inductively, there exists a feasible solution.

Next we move to (2). To activate a node i with threshold 3, it is clear that i must receive a link, and both of its neighbors have to be activated before i. This is not possible if one of $P_{1,i-1}$and $P_{i+1,n}$ does not have a feasible solution. Conversely, if both these sub-paths have feasible solutions, then by using these solutions and giving a link to v, we can activate the node v as well, thereby creating a feasible solution for P_n. ■

Given a path $P_{i,j}$, we now show how to find a minimum-sized set of nodes that can activate all the nodes in the path. We define $next(i)$ to be the first node with

threshold 1 in $P_{i+1,j}$. For convenience, we define $t(j+1) = 1$.

$$next(i) = min\{k : i+1 \leq k \leq j+1 \ \& \ t(k) = 1\}$$

Clearly, the problem does not have a feasible solution if there exists a node with threshold greater than 3. We study separately the two cases that there does and does not exist a node with threshold 3 in $P_{i,j}$.

4.1.1 The path consists of only nodes with threshold 1 or 2

First, we consider the case that $P_{i,j}$ only consists of nodes with threshold 1 or 2. We solve the $MinLinks(P_{i,j})$ problem by considering the different thresholds node i may have.

Case 1: $t(i) = 2$. If $next(i) > j$, this implies that all nodes in $P_{i,j}$ have threshold two, and by Proposition 1, we know that there is no feasible solution. Otherwise, there is a feasible solution, but to activate node i, we must both give it a link, and activate its neighbor $i+1$. It follows that the optimal number of links to activate $P_{i,j}$ is one more than the optimal number of links to activate $P_{i+1,j}$.

Case 2: $t(i) = 1$. If $next(i) > j$, this implies that node i is the rightmost node with threshold one in the path $P_{i,j}$. Therefore, to activate all the nodes in $P_{i,j}$, it is necessary and sufficient to give a link to every node in $P_{i,j}$. If instead $next(i) \leq j$, then by Proposition 1, the path $P_{i+1,j}$ also has a feasible solution. In this case, we claim that any *optimal* solution for $P_{i+1,j}$ is *also* an optimal solution for $P_{i,j}$. To prove the claim, suppose that there is an optimal solution S for $P_{i+1,j}$ that has k links but S is not optimal for $P_{i,j}$. Consider an optimal solution S' for $P_{i,j}$. Since S is not optimal for $P_{i,j}$, it must be that $|S'| < k$. If S' does not have a link to node i, then clearly it is also a solution for $P_{i+1,j}$ which contradicts the optimality of S for $P_{i+1,j}$.

If S' has a link to node i, note that since S' is optimal, there must exist some node $p \in [i+1, next(i)]$ which does not receive a link (see Figure 11). Therefore, the link assigned to node i can be moved to node p, giving a valid solution S'' for $P_{i,j}$ of the same size as S'. However, S'' is also a solution for $P_{i+1,j}$ and has fewer than k links, contradicting the optimality of S for $P_{i+1,j}$.

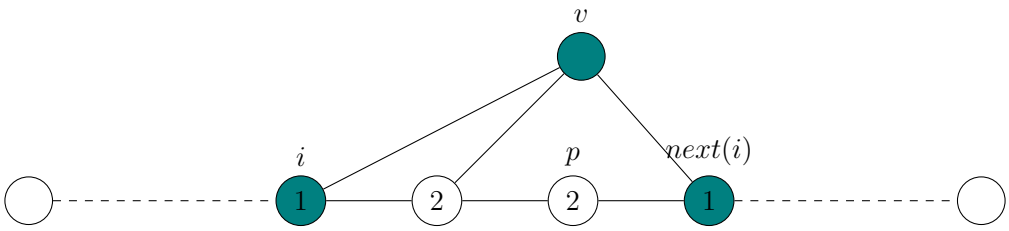

Figure 11: An optimal solution S which gives a link to node i

Let $C(i,j)$ be the minimum number of links needed to activate the sub-path $P_{i,j}$. The above discussion gives the following dynamic programming formulation to find $C(i,j)$.

$$C(i,j) = \begin{cases} 1 + C(i+1,j) & \text{if } t(i) = 2 \ \& \ next(i) \leq j \\ \infty & \text{if } t(i) = 2 \ \& \ next(i) > j \\ C(i+1,j) & \text{if } t(i) = 1 \ \& \ next(i) \leq j \\ j - i + 1 & \text{if } t(i) = 1 \ \& \ next(i) > j \end{cases}$$

If we study this dynamic programming formulation in depth, we can observe that we give a link to every node with threshold 2 and we only give a link to the last node with threshold 1. The theorem below follows immediately:

Theorem 4 *Given a path $P_{i,j}$ which admits a feasible solution and only consists of nodes with threshold 1 or 2, $C(i,j) = n_2 + 1$ where n_2 is the number of nodes with threshold 2 in $P_{i,j}$.*

4.1.2 The path contains nodes with threshold 3

Now, we consider the case that $P_{i,j}$ has nodes with threshold 3. If $t(i) = 3$ or $t(j) = 3$, clearly the problem does not have a feasible solution. If there exists an index k ($i < k < j$) with $t(k) = 3$, then it is clear that node i must receive a link, and both $P_{i,k-1}$ and $P_{k+1,j}$ must have feasible solutions.

$$ML(P_{i,j}) = ML(P_{i,k-1}) + ML(P_{k+1,j}) + 1$$

Let n_3 be the number of nodes in $P_{i,j}$ with threshold 3. Nodes with threshold 3 divide $P_{i,j}$ into n_3+1 sub-paths (Figure 12). For each sub-path, the minimum number of links required is the number of nodes with threshold 2 in this sub-path plus 1 by Theorem 4 . Therefore, the overall number of links required for all the sub-paths are $n_2 + n_3 + 1$, and we need to give a link to every node with threshold 3 in $P_{i,j}$. The theorem below follows immediately:

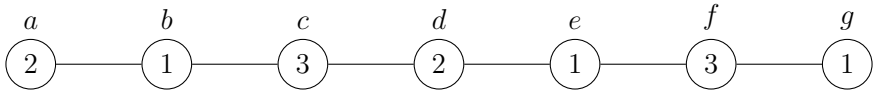

Figure 12: An path which has nodes with threshold 3

Theorem 5 *Given a path $P_{i,j}$ which admits a feasible solution, $C(i,j) = 2*n_3+n_2+1$ where n_3 and n_2 are the number of nodes with threshold 3 and 2 in $P_{i,j}$ respectively.*

Next, we give a linear time algorithm for finding a minimum-sized link set for the problem $MinLinks(P_{i,j})$.

Algorithm 2 $IsFeasible(P_{i,j})$ - Feasibility checking for paths

```
1: Input: Path P_{i,j} = (V, E, t)
2: Output: IsSolvable - P_{i,j} admits a solution or not;
3: IsSolvable ← false
4: last ← 0
5: IsLastOne[] ← {false}
6: for index p ← i, j do
7:     if t(p) == 3 then
8:         IsSolvable ← false
9:         last ← 0                                  ▷ resetting the indicator
10:        if p == i then                            ▷ special case
11:            return false
12:    if t(p) == 1 then
13:        IsSolvable = true
14:        if last == 0 then
15:            last ← p
16:            IsLastOne[last] ← true
17:        else
18:            IsLastOne[last] ← false
19:            last ← p
20:            IsLastOne[last] ← true
21: return IsSolvable
```

Algorithm 3 Minimum Links algorithm for paths

```
1: Input: Path P_{i,j} = (V, E, t)
2: Output: Minimum-sized link set S;
3: S = ∅
4: for index p ← i, j do
5:     if t(p) == 3 || t(p) == 2 || IsLastOne[p] then
6:         S ← S ∪ {p}
7: return S
```

Theorem 6 *The Minimum Links problem for a path $P_{1,n}$ can be solved in time $\theta(n)$.*

Proof. A simple linear time scan can verify if the problem instance has a feasible

solution and find all the last node with threshold 1 in every sub-path divided by nodes with threshold 3.

If the problem admits a feasible solution, as stated in Theorem 5, in an optimal solution S, every node with threshold 2 and 3 must receive a link, and every last node with threshold 1 in every sub-path divided by nodes with threshold 3 must also receive a link. The pseudocode is given in Algorithm 3. The optimal link set can be constructed by a linear scan. ■

4.2 Maximum Influence problem

In this section, we study the Maximum Influence problem in paths. Let $MaxIn(P_n, k)$ be the problem of finding an assignment of at most k links to nodes in path P_n that maximizes the number of influenced nodes, and let $MI(P_n, k)$ be the maximum number of nodes that can be influenced in the path using k links. Since not every node will be activated by an optimal assignment for the Maximum Influence problem, we develop some new notation to facilitate our discussion. Let $MIA(P_{i,n}, k)$ be the problem of finding an assignment of at most k links to nodes in the sub-path $P_{i,n}$ that maximizes the number of influenced nodes *while ensuring that node i is activated*. Similarly, let $MIB(P_{i,n}, k)$ be the problem of finding an assignment of at most k links to nodes in the sub-path $P_{i,n}$ that maximizes the number of influenced nodes *while ensuring that node i is not activated*. Let $A(i, k)$ and $B(i, k)$ be the number of nodes that can be influenced by optimal solutions to $MIA(P_{i,n}, k)$ and $MIB(P_{i,n}, k)$ respectively. Then clearly

$$MI(P_{1,n}, k) = max\{A(1, k), B(1, k)\}$$

We provide two examples to show that the optimal solution for the Maximum

Influence problem may or may not activate the first node, depending on the thresholds in the path. Therefore, it is necessary to consider both cases.

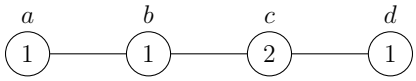

Figure 13: Maximum Influence problem in a path with $k = 1$

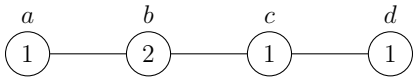

Figure 14: Maximum Influence problem in a different path with $k = 1$

In Figure 13, $A(1,1) = 2$, $\{a\}$ is an optimal link set that can activate nodes $\{a, b\}$, meanwhile $B(1,1) = 1$, $\{d\}$ is an optimal link set that does not activate node a; while in Figure 14, $A(1,1) = 1$, $\{a\}$ is an optimal link set that can activate node $\{a\}$, meanwhile $B(1,1) = 2$, $\{c\}$ is an optimal link set that does not activate node a, instead, it activates $\{c, d\}$.

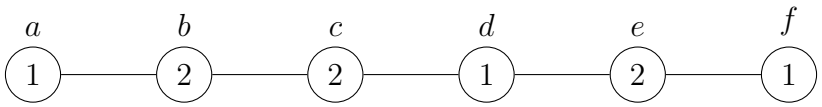

Figure 15: Maximum Influence problem in a path with $k = 2$

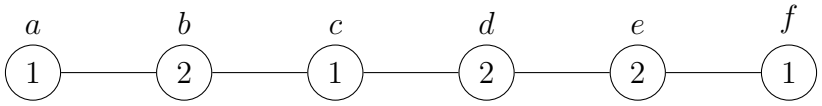

Figure 16: Maximum Influence problem in a different path with $k = 2$

In Figure 15, $A(1,2) = 2$, $\{a, b\}$ is an optimal link set that can activate nodes $\{a, b\}$; meanwhile $B(1,2) = 3$, $\{d, e\}$ is an optimal link set that does not activate node a, instead, it activates $\{d, e, f\}$. However, in Figure 16, $A(1,2) = 3$, $\{a, b\}$ is an optimal link set that can activate nodes $\{a, b, c\}$; meanwhile $B(1,2) = 2$, $\{c, d\}$ is an optimal link set that does not activate node a, instead, it activates $\{c, d\}$.

We proceed to give a solution for the problem $MI(P_n, k)$, and study separately the two cases that there does and does not exist a node with threshold > 2 in P_n.

4.2.1 The path consists of only nodes with threshold 1 or 2

Given a path P_n in which all nodes have thresholds 1 or 2 together with k links available, we give a recursive definition for $A(i,k)$ and $B(i,k)$ to solve the Maximum Influence problem. As we will see, these definitions are inter-dependent and need to be developed together. Clearly, $A(i,0) = B(i,0) = 0$ for every i. For convenience, We also define $t(n+1) = 1$ and $A(n+1,k) = B(n+1,k) = 0$ for all k.

First we consider the case when node i has threshold 2. If $next(i) > n$ (that is, no nodes with threshold 1 in $P_{i+1,n}$), then there is no way to activate any node in $P_{i,n}$, therefore $A(i,k) = B(i,k) = 0$ for every k. If instead $next(i) \leq n$, then it is possible to activate at least one node in $P_{i,n}$. Observe that in any feasible solution for $MIA(P_{i,n},k)$, not only does node i need to receive a link, but its neighbor node $i+1$ needs to be activated as well, therefore $A(i,k) = 0$ if $A(i+1,k-1) = 0$; otherwise $A(i,k) = 1 + A(i+1,k-1)$. Finally, note that any feasible solution for $MIB(P_{i,n},k)$ is a solution in which i does not receive a link, and the next node may or may not be activated, or it does receive a link, and the next node is not activated. That is $B(i,k) = max\{A(i+1,k), B(i+1,k), B(i+1,k-1)\} = max\{A(i+1,k), B(i+1,k)\}$

Putting together the recursive definitions of $A(i,k)$ and $B(i,k)$ we obtain:

$$A(i,k) = \begin{cases} 0 & \text{if } t(i) = 2 \ \&\ next(i) > n \\ 0 & \text{if } t(i) = 2 \ \&\ A(i+1,k-1) = 0 \\ 1 + A(i+1,k-1) & \text{if } t(i) = 2 \ \&\ A(i+1,k-1) > 0 \end{cases}$$

$$B(i,k) = \begin{cases} 0 & \text{if } t(i) = 2 \ \&\ next(i) > n \\ max \begin{cases} A(i+1,k) \\ B(i+1,k) \end{cases} & \text{if } t(i) = 2 \ \&\ next(i) \leq n \end{cases}$$

Next we consider the case when node i has threshold 1. If there are no other nodes with threshold 1 in the path, that is if $next(i) > n$, then clearly any activation has to progress with first node i getting activated and then consecutive nodes to its right, one in each activation round. Therefore, $A(i,k) = min(k, n-i+1)$. Otherwise, node i is not the rightmost node with threshold one. The following two technical lemmas are useful in characterizing the optimal substructure of the problem.

Lemma 1 *Given a path $P_{i,n}$ in which node i has threshold 1 and the remaining nodes have threshold 2, let S be an optimal solution to the problem $MIA(P_{i,n}, k)$. Then S can activate exactly $min\{k, n-i+1\}$ nodes, and must give links to the set $\{i, \ldots, i + min\{k, n-i+1\} - 1\}$.*

Proof. If node i does not receive a link, then it is not possible to activate any node in $P_{i+1,n}$ by Proposition 1. Then node i must receive a link and be activated first (see Figure 17). Inductively, we can show that node $i+1$ is the second node which receives a link and gets activated. Therefore, all nodes in $\{i, \ldots, i + min\{k, n-i+1\} - 1\}$ must receive a link. Any node that does not receive a link cannot get activated, the maximum number of activated nodes would be $min\{k, n-i+1\}$.

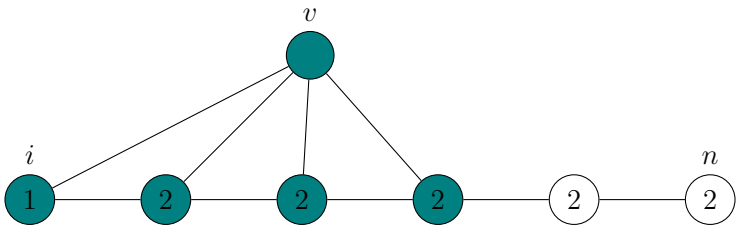

Figure 17: An optimal assignment S for $MIA(P_{i,n}, 4)$

■

Lemma 2 *Given a path $P_{i,n}$ with $i < n$ in which nodes i and n have threshold 1 and the remaining nodes have threshold 2 and k links with $k < n-i$. Let S be an optimal solution for $MIA(P_{i,n}, k)$ that activates node n as well.*

In S, the set of nodes that receive links is two sequences $[i, i+\alpha]$, and $[n-\ell, j]$ while satisfying $\alpha + \ell - 2 = k$.

Proof. Since $k < n - i$, then there are at least two nodes in the range $[i, n]$ would not receive a link, thus at least one node is the range $[i+1, n-1]$ is not activated by S.

Suppose node p is some node in range $[i+1, n-1]$ that is not activated by S (see Figure 18). Then we can break $P_{i,n}$ into two sub-paths $P_{i,p-1}$ in which node i has threshold 1 and the remaining nodes have threshold 2, and $P_{p+1,n}$ in which node n has threshold 1 and the remaining nodes have threshold 2. Then this problem can be reduced to finding an optimal assignment S of k links to $P_{i,p-1}$ and $P_{p+1,n}$ which maximizes the number of overall activated nodes while ensuring node i and node n are both activated.

According to Lemma 1, in such an optimal S for $P_{i,p-1}$ and $P_{p+1,n}$, the set of nodes that received links is two sequences $[i, i+\alpha]$, and $[n-\ell, j]$ while satisfying $\alpha + \ell - 2 = k$.

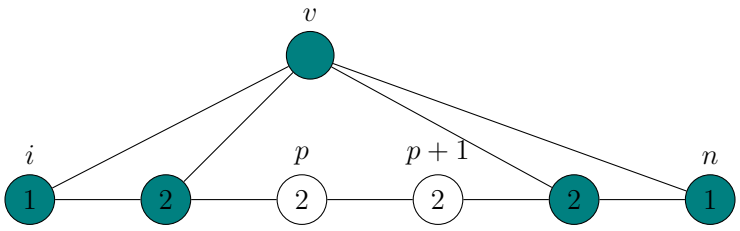

Figure 18: An optimal assignment S for $MIA(P_{i,n}, 4)$ which activates node n as well

■

We are now ready to describe the optimal substructure of the problem.

Lemma 3 *Let $P_{i,n}$ be a path with $t(i) = 1$ and $next(i) \leq n$. If in every optimal solution for $MIA(P_{i,n}, k)$, node i receives a link, then there exists an optimal solution S' for $MIA(P_{i,n}, k)$ in which neither node $i+1$ nor node $i+2$ is activated.*

Proof. Let S be an optimal solution for $MIA(P_{i,n}, k)$ where $next(i) \leq n$, which uses the fewest links possible. By assumption, node i receives a link. First observe that if $next(i) = i + 1$, then $i + 1$ cannot have a link since that would contradict the minimality of S. So we can simply move the link from node i to $i+1$ and activate the same set of nodes, but this contradicts the assumption that in every optimal solution, node i must receive a link. Next suppose $next(i) = i + 2$. Then it is not possible that both $i + 1$ and $i + 2$ receive links, as this would contradict the minimality of S. If exactly one of $i + 1$ and $i + 2$ have a link, we can move the link to i to the node among $i + 1$ and $i + 2$ that does not have a link, thus creating a solution which activates exactly the same set of nodes, a contradiction. If neither $i + 1$ nor $i + 2$ has a link, and neither is activated, then the lemma is proved. Finally, if neither has a link, but one of them is activated, it must be that $i + 2$ is activated by node $i + 3$. In this case, we can move the link from node i to node $i + 1$, getting a solution that activates the same set of nodes, a contradiction to the assumption that there is no optimal solution that does not give node i a link.

It remains to consider the case when $next(i) > i+2$, that is, both nodes $i+1$ and $i+2$ have threshold 2. We consider the set of nodes in $Z = \{i+1, i+2, \ldots, next(i)\}$ and the links assigned to the nodes in Z by S.

Claim 3 *If $next(i) \in I(P_{i,n}, S)$, there are at least two nodes in $Z - \{next(i)\}$ that do not receive links in S, and if $next(i) \notin I(P_{i,n}, S)$, there are at least two nodes in Z that do not receive a link.*

Proof. We first consider the case when $next(i) \in I(P_{i,n}, S)$. Assume for the purpose of contradiction that there are fewer than two nodes in $Z - \{next(i)\}$ that do not receive links. If all such nodes receive links, clearly S does not use the fewest links possible. So there must exist exactly one node $p \in Z - \{next(i)\}$ to which S does not give a link (see Figure 19). Observe that all nodes in $Z \cup \{i\}$ are then activated

by S. By moving the link given to node i to the node p, we get a solution which activates the same set of nodes as S, but in which node i does not receive a link, a contradiction to the assumption that there is no such optimal solution. We conclude that there are at least two nodes in $Z - \{next(i)\}$ that do not receive links in S, as needed.

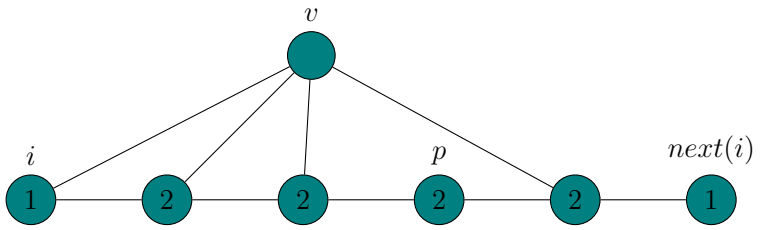

Figure 19: An optimal assignment by which $next(i) \in I(S)$

Next consider the case when $next(i) \notin I(S)$ (see Figure 20). Then clearly $next(i)$ does not receive a link in S. If every node in $Z - \{next(i)\}$ receives a link, the all these nodes will be activated, in turn activating $next(i)$, contradicting the assumption that $next(i) \notin I(S)$. Therefore, there are at least two nodes in Z without links as required. ■

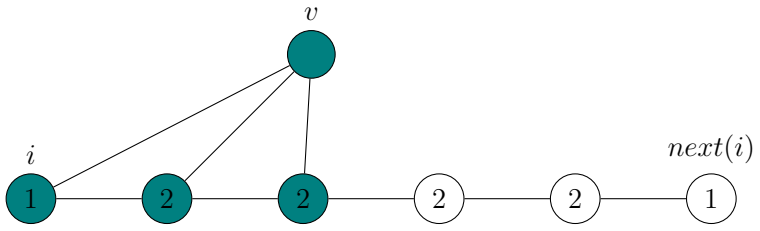

Figure 20: An optimal assignment by which $next(i) \notin I(S)$

We now show how to convert S into another optimal solution S' with the claimed properties. The following four cases are exhaustive.

$next(i)$ **is not activated by** S : By Lemma 1, the set of nodes that received links is some contiguous sequence $[i, i+j]$, giving $j+1$ nodes that are activated. We

create a new solution S' which is the same as S except that j links are removed from nodes $[i+1, i+j]$ and assigned instead to $[next(i)-j+1, next(i)]$. S' is a solution for $MIA(P_{i,n}, k)$ that activates exactly the same number of nodes as S, and is therefore optimal. By Claim 3, at least two nodes in Z did not receive links in S, and therefore, in S', nodes $i+1$ and $i+2$ are not activated.

$next(i)$ **received a link in** S**:** By Lemma 2, the set of nodes that received links is two sequences $[i, i+j]$, and $[next(i)-\ell, next(i)]$ giving $j+\ell+1$ nodes that are activated in Z. We create a new solution S' which is the same as S except that j links are removed from nodes $[i+1, i+j]$ and assigned instead to $[next(i)-\ell-j, next(i)-\ell-1]$. S' is a solution for $MIA(P_{i,n}, k)$ and activates exactly the same number of nodes as S, and is therefore optimal. Furthermore, S' does not give links to $i+1$ and $i+2$ since by Claim 3, there are at least two nodes in $Z - \{next(i)\}$ that do not receive links in S.

$next(i)$ **does not receive a link but is activated by** $next(i)-1$: In this case, it must be that all nodes in $Z - \{next(i)\}$ received links in S. But then the link to node i can be moved to $next(i)$ giving an optimal solution in which node i does not receive a link, a contradiction to the assumption that there is no such optimal solution.

$next(i)$ **does not receive a link but is activated by** $next(i)+1$: Using a proof similar to that of Lemma 2, it can be seen that the set of nodes that received links can be partitioned into two sequences $[i, i+j]$, and $[next(i)-1-\ell, next(i)-1]$ giving $j+\ell+1$ nodes that are activated in Z. We create a new solution S' which is the same as S except that j links are removed from nodes $[i+1, i+j]$ and assigned instead to $[next(i)-\ell-j, next(i)-\ell-1]$. S' is a solution for $MIA(P_{i,n}, k)$ and activates exactly the same number of nodes as S, and is

therefore optimal. Furthermore, S' does not give links to $i+1$ and $i+2$ since by Claim 3, there are at least two nodes in $Z - \{next(i)\}$ that do not receive links in S.

In all cases, S' is an optimal solution in which neither node $i+1$ nor node $i+2$ receives a link, and are therefore not activated. ■

The following lemma summarizes the optimal substructure of the problem $MIA(P_{i,n}, k)$ when $t(i) = 1$ and $next(i) \leq n$.

Lemma 4 *Suppose* $t(i) = 1$ *and* $next(i) \leq n$. *Then*

$$A(i,k) = max\{A(i+1,k) + 1, B(i+1,k-1) + 1\}$$

Proof. First observe that either there exists an optimal solution for $MIA(P_{i,n}, k)$ in which node i does not receive a link, or in every optimal solution for $MIA(P_{i,n}, k)$, node i receives a link. In the first case, let S be an optimal solution for $MIA(P_{i,n}, k)$ in which node i does not receive a link. It follows that its neighbor node $i+1$ was also activated. Clearly S must be an optimal solution for $MIA(P_{i+1,n}, k)$ (if not, and if S' is a solution for $MIA(P_{i+1,n}, k)$ that activates more nodes than S, then S' is also a better solution for $MIA(P_{i,n}, k)$, contradicting the optimality of S. Therefore, $A(i,k) = 1 + A(i+1,k)$.

In the second case, by Lemma 3, we have an optimal solution S' in which nodes $i+1$ and $i+2$ do not receive links and are therefore not activated. Furthermore, using a cut-and-paste argument, it is straightforward to see that $S'' = S' - \{i\}$ is an optimal solution for $MIB(P_{i+1,n}, k-1)$. It follows that $A(i,k) = 1 + B(i+1,k-1)$. This completes the proof. ■

Finally, any solution in which node i is not activated, we can be sure that neither node i gets a link, nor does its neighbor, node $i+1$ get activated. Therefore $B(i,k) =$

$B(i+1,k)$.

Putting together the recursive definitions of $A(i,k)$ and $B(i,k)$ we obtain:

$$A(i,k) = \begin{cases} \min\{k, n-i+1\} & \text{if } t(i)=1 \ \& \ next(i) > n \\ max \begin{cases} 1 + A(i+1,k) \\ 1 + B(i+1,k-1) \end{cases} & \text{if } t(i)=1 \ \& \ next(i) \leq n \end{cases}$$

$$B(i,k) = \begin{cases} 0 & \text{if } t(i)=1 \ \& \ next(i) > n \\ B(i+1,k) & \text{if } t(i)=1 \ \& \ next(i) \leq n \end{cases}$$

Algorithm 4 Influence maximization algorithm for paths

Input: (1) Path $P_n = (V, E, t), t(v) \in \{1, 2\}$; (2) links available k;
Output: Maximum number of nodes that can be activated;
$A(n+1, k) = B(n+1, k) = 0$, for all k
$A(i, 0) = B(i, 0) = 0$, for all i
for Index $i \leftarrow n, 1$ **do**
    if $t(i) == 2$ **then**
        for $p \leftarrow 1, k$ **do**
            $StartIndexHasThreshold2(i, p)$
    else ▷ node i is of threshold 1
        for $p \leftarrow 1, k$ **do**
            $StartIndexHasThreshold1(i, p)$
return $max(A(1, k), B(1, k))$
procedure StartIndexHasThreshold1(index i, links k)
    if $next(i) > n$ **then** ▷ no node with threshold 1 in $P_{i+1,n}$
        $A(i, k) \leftarrow \min(k, n - i + 1)$
        $B(i, k) \leftarrow 0$
    else
        $B(i, k) \leftarrow B(i + 1, k)$
        if $A(i + 1, k) > 0$ **then**
            $A(i, k) \leftarrow \max(A(i + 1, k) + 1, B(i + 1, k - 1) + 1)$
        else ▷ impossible to activate node $i + 1$
            $A(i, k) \leftarrow B(i + 1, k - 1) + 1$
procedure StartIndexHasThreshold2(index i, links k)
    if $next(i) > n$ **then** ▷ no node with threshold 1 in $P_{i+1,n}$
        $A(i, k) \leftarrow 0$
        $B(i, k) \leftarrow 0$
    else
        $B(i, k) \leftarrow \max(A(i + 1, k), B(i + 1, k))$
        if $A(i + 1, k - 1) > 0$ **then**
            $A(i, k) \leftarrow 1 + A(i + 1, k - 1)$
        else ▷ impossible to activate node $i + 1$
            $A(i, k) \leftarrow 0$

After successfully calculating $A(i,k)$ and $B(i,k)$ for all pairs of i and k, we give the following linear time algorithm (Algorithm 5) to construct an optimal link set from $A(i,k)$ and $B(i,k)$.

Algorithm 5 Finding an optimal link set of size k in paths

```
1: Input: (1) Path P_n = (V, E, t), t(v) ∈ {1, 2}; (2) links available k; (3) A(i, k) and
   B(i, k) matrix generated in Algorithm 4
2: Output: an optimal link set S
3: Start Index i ← 0
4: while k > 0 & i ≤ n do
5:     if A(i, k) ≥ B(i, k) then
6:         if t(i) == 1 then                                   ▷ node i has threshold 1
7:             if next(i) == n + 1 then                        ▷ Special case
8:                 for i ← i, min(i + k, n) do
9:                     add node i to S
10:                k ← 0
11:            else                                            ▷ next(i) ≤ n
12:                if A(i + 1, k) > 0 then
13:                    if B(i + 1, k − 1) > A(i + 1, k) then
14:                        add node i to S
15:                        i ← i + 1
16:                        k ← k − 1
17:                    else
18:                        i ← i + 1
19:                else                  ▷ A(i + 1, k) = 0 impossible to activate i + 1
20:                    add node i to S
21:                    i ← i + 1
22:                    k ← k − 1
23:        else                                                ▷ node i has threshold 2
24:            add node i to S
25:            i ← i + 1
26:            k ← k − 1
27:    else                                                    ▷ B(i, k) > A(i, k)
28:        i ← i + 1
29: return S
```

4.2.2 The path contains nodes with threshold > 2

We consider now the case that there exists a node q in the path P_n with $t(q) \geq 3$. If $q = 1$ or $q = n$ or $t(q) > 3$, it is obvious that node q cannot be activated. Similarly for any node q with $1 \leq q \leq n$, if $t(q) > 3$, it cannot be activated. Therefore, the optimal solution to the Max-Influence problem with k links does not give node q a link, and instead uses ℓ links to solve the Max-Influence problem on the path $P_{1,q-1}$ and $k - \ell$ links to solve the Max-Influence problem on the path $P_{q+1,n}$ for some value of ℓ such that $0 \leq \ell \leq k$. Thus the optimal solution can be found by checking for all possible values of ℓ between 0 and k.

If $1 < q < n$, and $t(q) = 3$, then the optimal solution may or may not activate node q. If it does not activate node q, the optimal solution can be found in the same way as the case that $t(q) > 3$. If it does activate node q, the optimal solution must not only give a link to node q, it must also activate both nodes $j-1$ and $j+1$ since q has threshold 3. Thus, it must use ℓ links to solve the Maximum Influence problem on $P_{i,q-1}$ while making sure that node $q-1$ is activated, and $k-\ell-1$ links to solve the Maximum Influence problem on $P_{q+1,j}$ while making sure that node $q+1$ is activated.

To solve the Maximum Influence problem when there are nodes with threshold 3 or greater, therefore, we proceed as follows. The values $A(i,k)$ and $B(i,k)$ are defined identically to the previous section, when $t(i) = 1$ or $t(i) = 2$.

If $t(i) \geq 3$,

$$A(i,k) = 0$$

$$B(i,k) = max \begin{cases} A(i+1,k) \\ B(i+1,k) \end{cases}$$

Now we define $C(j,k)$ and $D(j,k)$ to be the maximum number of nodes that can be influenced in the path $P_{1,j}$ while ensuring that node j is influenced, and not

influenced respectively. The values C and D are computed similarly to A and B for all threshold values, going backwards in the path rather than forwards. Finally, to derive the value of $MI(P_n, k)$, we first verify if there is a node q with threshold greater than 2. If there is no such node, then $MI(P_n, k)$ is defined as before. Suppose there is a node q with $t(q) \geq 3$.

If $t(q) > 3$,

$$MI(P_n, k) = \max_{0 \leq \ell \leq k} \{max(C(q-1, \ell), D(q-1, \ell)) + max(A(q+1, k-\ell), B(q+1, k-\ell))\}$$

If $t(q) = 3$,

$$MI(P_n, k) = max \begin{cases} \max\limits_{0 \leq \ell \leq k} \begin{cases} max(C(q-1, \ell), D(q-1, \ell)) + \\ max(A(q+1, k-\ell), B(q+1, k-\ell)) \end{cases} \\ \max\limits_{0 \leq \ell \leq k-1} 1 + C(q-1, \ell) + A(q+1, k-1-\ell) \end{cases}$$

This allows us to prove the following theorem:

Theorem 7 *The Maximum Influence problem for a path P_n using k links, can be solved in $\Theta(kn)$ time.*

Proof. The dynamic programming formulation given above can be solved using the tabular method. First we check in $\Theta(n)$ time if there is a node of threshold greater than two. If there are no such nodes, we create two-dimensional tables A and B of dimension $n \times k$. We fill the table in row major order, filling each row from right to left. Each table entry can therefore be calculated in constant time. Thus the time taken is $\Theta(kn)$. If there is a node of threshold greater than two, say node q, then we create additional tables C and D, and calculate $A(q+1, \ell)$, $B(q+1, \ell)$, $C(q-1, \ell)$, and $D(q-1, \ell)$ for all values of ℓ in $0 \leq \ell \leq k$. Now the computation of $MI(P_n, k)$ can be done in $\Theta(k)$ time. Therefore the total time taken is once again $\Theta(kn)$. The

pseudocode for finding the arrays A, B, C, and D is given in Algorithm 4. ■

Chapter 5

Rings

In this chapter, we study the Minimum Links problem and Maximum Influence problem for rings. Let $R_n = (V, E, t)$ be a ring with n nodes, $V = \{1, 2, ..., n\}$, $E = \{(i, (i+1)) \ mod \ n) \mid 1 \leq i \leq n\}$, and $t : t(v) \rightarrow \mathcal{Z}^+$. We define $P_{i,j}$ $(i \neq j)$ to be the sub-path of R_n consisting of all nodes in $\{i, \ldots, j\}$ in the clockwise direction.

5.1 Minimum Links problem

In this section, we study the Minimum Links problem in rings. Let $MinLinks(R_n)$ be the problem of finding a minimum-sized link assignment that activates the entire ring, and let $ML(P_n)$ be the minimum number of links needed to activate the entire ring.

First observe that if all nodes in the ring have threshold one, then by giving a link to *any* of the nodes, we can activate the entire ring. However, there are situations in which it is impossible to activate the entire ring, for example, if any node has threshold greater than three. We now show a necessary and sufficient condition for the Minimum Links problem to have a feasible solution:

Proposition 2 *The Minimum Links problem has a feasible solution on* $R_n = (V, E, t)$

if and only if one of the following two conditions is met:

1. *There is no node in R_n with threshold ≥ 3, and there exists a node with threshold 1.*

2. *There exists a node i with threshold 3, and $P_{i+1,i-1}$ has a feasible solution.*

Proof. First we prove that (a) is a necessary and sufficient condition for the Minimum Links problem to have a feasible solution. If there is no node with threshold 3 or 1, all nodes have threshold 2, clearly even if all nodes in R_n receive links, no node can be activated. Conversely, suppose there is no node with threshold 3, and there does exist a node i with threshold 1. Then by giving a link to i, it is activated, and by the cascade effect, the thresholds on both $i-1$ and $i+1$ are effectively reduced. The remaining would be a path $P_{i+1,i-1}$ with at least one node of threshold 1. By Proposition 1, there exists a feasible solution for $P_{i+1,i-1}$.

Next we move to (b). To activate a node i with threshold 3, it is clear that i must receive a link, and both of its neighbors have to be activated before i. This is not possible if $P_{i+1,i-1}$ does not have a feasible solution. Conversely, if $P_{i+1,i-1}$ has a feasible solution S, then $S \cup \{i\}$ will be a feasible solution for R_n. ■

We proceed to find a minimum-sized assignment of links for inputs that admit a feasible solution. Clearly, there is no node in R_n with threshold > 3. We study separately the two cases that there does and does not exist a node with threshold 3 in R_n.

5.1.1 R_n consists of only nodes with threshold 1 or 2

First, we study the case that there does not exist a node of threshold 3 in R_n. We observe that if all nodes in the ring have threshold 1 or there is only one node of threshold 2 in the ring, then by giving a link to *any* of the nodes with threshold 1,

we can activate the entire ring. Consider such a ring R_n with at least one node of threshold 1 and at least two nodes with threshold 2. We choose an arbitrary node i of threshold 1 in R_n, and we define $C(i)$ and $CC(i)$ to be the first node with threshold 2 in i's clockwise direction and counter clockwise direction respectively, $C(i) \neq CC(i)$ (see Figure 21). We also define $P_{C(i),CC(i)}$ be the path from $C(i)$ to $CC(i)$ where $t(C(i)) = t(CC(i)) = 2$; and $P'_{C(i),CC(i)}$ to be the same path as $P_{C(i),CC(i)}$ except that $t(C(i)) = t(CC(i)) = 1$.

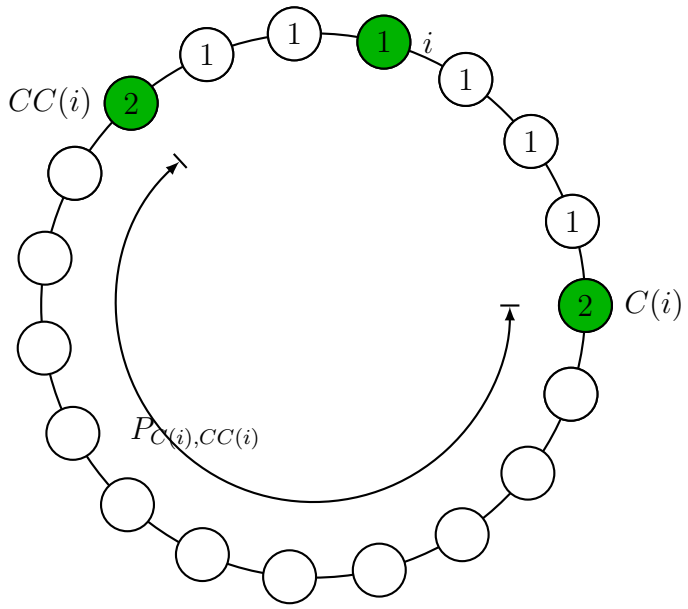

Figure 21: There does not exist a node with threshold 3 in R_n

Theorem 8 *Given a ring $R_n = (V, E, t)$ which admits a feasible solution and consists of only nodes with threshold 1 and 2, $ML(R_n) = n_2$ where n_2 is the number of nodes with threshold 2 in R_n.*

Proof. Let S be an optimal solution for $MinLinks(R_n)$. It is easy to observe that some node i with threshold 1 must receive a link in S otherwise it is impossible to start the activation process.

We claim that $S - \{i\}$ is an optimal solution to $MinLinks(P'_{C(i),CC(i)})$. Suppose not, let $|S| = k$ and let S' be an optimal solution to $MinLinks(P'_{C(i),CC(i)})$ of size

$< k-1$. Then giving a link to node i can activate all nodes in $[CC(i)+1, C(i)-1]$ and effectively reduce the thresholds of $C(i)$ and $CC(i)$. $S' \cup \{i\}$ is a solution of size $< k$ which can activate all nodes in R_n, a contradiction to the optimality of S. So we can conclude that an optimal solution S for $MinLinks(R_n)$ can be constructed by choosing any node of threshold 1 to be part of the solution, together with an optimal solution to the residual path $P'_{C(i),CC(i)}$.

Recalling that the number of nodes with threshold 2 in $P'_{C(i),CC(i)}$ is $n_2 - 2$. By Theorem 4, the minimum number of links needed to activate $P'_{C(i),CC(i)}$ is $n_2 - 2 + 1 = n_2 - 1$; so the minimum number of links needed to activate R_n is n_2. ■

5.1.2 R_n contains nodes with threshold 3

Next, we study the case that there exists a node i with threshold 3. It is clear that i must receive a link and both of its neighbors must be activated before i gets activated. Let S be an optimal solution for $MinLinks(R_n)$, we claim that $S - \{i\}$ is also an optimal solution for $MinLinks(P_{i+1,i-1})$. Suppose not, let S' be a better solution for $MinLinks(P_{i+1,i-1})$ which satisfies $|S'| < |S| - 1$, obviously $S' \cup \{i\}$ is a solution for $MinLinks(R_n)$ using fewer links, contradicting the optimality of S. Therefore, an optimal solution S for $MinLinks(R_n)$ can be constructed by choosing any node i of threshold 3 to be part of the solution, together with an optimal solution to the residual path $MinLinks(P_{i+1,i-1})$.

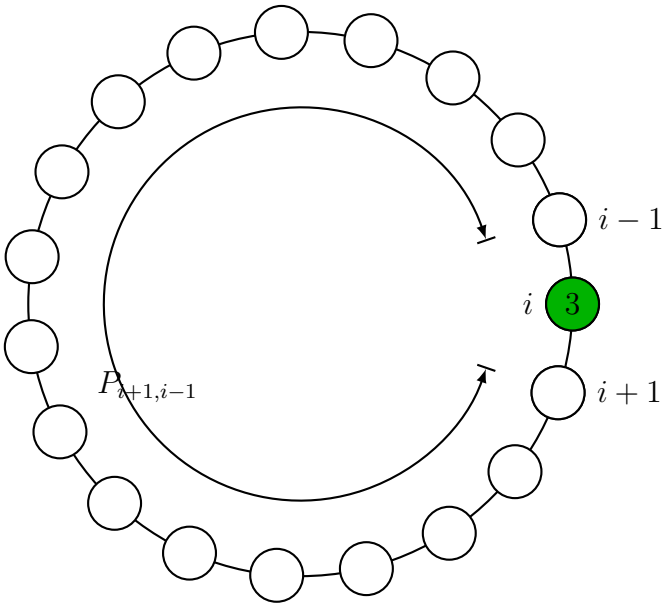

Figure 22: There exists a node with threshold 3 in R_n

We give the following theorem similar to Theorem 8.

Theorem 9 *Given a ring $R_n = (V, E, t)$ which admits a feasible solution, $ML(R_n) = 2 * n_3 + n_2$ where n_3 and n_2 are the number of nodes with threshold 3 and 2 in R_n respectively.*

Proof. Let i be an arbitrary node with threshold 3 in R_n, we have already proven that $\{i\} \cup S'$ which is an optimal solution for $MinLinks(P_{i+1,i-1})$ is an optimal solution to $MinLinks(R_n)$.

Recalling that the number of nodes with threshold 2 in $P_{i+1,i-1}$ is n_2 and the number of nodes with threshold 3 in $P_{i+1,i-1}$ is $n_3 - 1$. By Theorem 5, the minimum number of links needed to activate $P_{i+1,i-1}$ is $2 * (n_3 - 1) + n_2 + 1 = 2 * n_3 + n_2 - 1$; so the minimum number of links needed to activate R_n is $2 * n_3 + n_2$. ■

Next, we give a linear time algorithm for finding a minimum-sized link set for problem $MinLinks(R_n)$.

Algorithm 6 Minimum Links algorithm for rings

```
1: Input: Ring R_n = (V, E, t)
2: Output: Minimum-sized link set;
3: for index p ← 1, n do
4:     if t(p) > 3 then
5:         return No valid solution
6:     if t(p) == 3 then
7:         S' ← an optimal solution for MinLinks(P_{p+1,p-1})
8:         return S' ∪ {p}
9: choose an arbitrary node i of threshold 1 in R_n
10: C(i) ← the first node with threshold 2 in i's clockwise direction
11: CC(i) ← the first node with threshold 2 in i's counter clockwise direction
12: if C(i) == CC(i) then                ▷ There is only one node with threshold 2
13:     return {i}
14: else
15:     S'' ← an optimal solution for MinLinks(P'_{C(i),CC(i)})
16:     return S'' ∪ {i}
```

Theorem 10 *The Minimum Links problem for a ring R_n can be solved in time $\theta(n)$.*

Proof. If there is no node with threshold 3 in the path, as showed in the proof of Lemma 8, an optimal solution for the MinLinks problem for R_n is given by choosing any node of threshold 1 to be part of the solution, together with an optimal solution to the residual path $P'_{C(i),CC(i)}$.

If there is a node i with threshold 3 in the path, as argued before, an optimal solution to the residual path $P_{i+1,i-1}$ together with $\{i\}$ forms an optimal solution for the MinLinks problem for R_n.

The pseudocode is given in Algorithm 6. Since the MinLinks problem for a path can be solved in $\theta(n)$ according to Theorem 6, we can construct an optimal solution for a ring in $\theta(n)$ time as well. ■

5.2 Maximum Influence problem

In this section, we study the Maximum Influence problem in rings. Let $MaxIn(R_n, k)$ be the problem of finding an assignment of at most k links to nodes in the ring R_n that maximizes the number of influenced nodes, and let $MI(R_n, k)$ be the maximum number of nodes that can be influenced in R_n using k links.

Similar to the MinLinks problem, we try to find a way to reduce the problem to a Maximum Influence problem in a path. First, we check if $ML(R_n) \leq k$. If $ML(R_n) \leq k$, clearly $MI(R_n, k) = n$ since we can activate all nodes in R_n with k links. If $ML(R_n) > k$, then it is not possible to activate all the nodes in the ring, and there must exist some node p in R_n which is not activated by an optimal assignment S for the problem $MaxIn(R_n, k)$. Since S is optimal, it is clear that S cannot give a link to node p. We claim that S is also an optimal assignment for problem $MaxIn(P_{p+1,p-1}, k)$. Suppose not, let S' be a solution of size k for $MaxIn(P_{p+1,p-1}, k)$ which can activate more nodes than S. It is easy to see that S' is also a better solution for $MaxIn(R_n, k)$, contradicting the optimality of S for $MaxIn(R_n, k)$. Therefore, $MI(R_n, k) = MI(P_{p+1,p-1}, k)$. In order to find such a node p , we can try every possibility of p $(1 \leq p \leq n)$ to find the maximum $MI(P_{p+1,p-1}, k)$, this maximum value is equal to $MI(R_n, k)$.

$$MI(R_n, k) = \begin{cases} n & \text{if } ML(R_n) \leq k \\ \max\limits_{1 \leq i \leq n} \{MI(P_{i+1,i-1}, k)\} & \text{otherwise} \end{cases}$$

Theorem 11 *The Maximum Influence problem for a ring R_n using k links, can be solved in time $\theta(kn^2)$.*

Proof. From previous analysis, we can see that $MI(R_n, k)$ can computed by calling

the Maximum Influence algorithm for a path n times. The time complexity of the Maximum Influence algorithm for a path is $\Theta(kn)$ by Theorem 7. Therefore, the Maximum Influence problem for a ring R_n can be solved in time $\theta(kn^2)$. ■

5.2.1 R_n consists of only nodes with threshold 1 or 2

If there is no node of threshold > 2 in R_n, we can solve the Maximum Influence problem in rings more efficiently in $\theta(kn)$ time. Clearly, if all nodes in R_n have threshold 2, we cannot activate any node. If there is only one node with threshold 2 in R_n, we can activate the whole ring by giving a link to *any* node with threshold 1. Consider such a ring R_n with at least one node of threshold 1 and at least two nodes with threshold 2. We choose an arbitrary node p of threshold 1 in R_n. Consider an optimal assignment S for problem $MaxIn(R_n, k)$ which uses the fewest links possible. We can see that node p either belongs to $I(R_n, S)$ or not. Let $MIA(R_p, k)$ be the problem of finding an assignment of at most k links to nodes in the ring R_n that maximizes the number of influenced nodes *while ensuring that node p is activated*. Similarly, let $MIB(R_p, k)$ be the problem of finding an assignment of at most k links to nodes in the ring R_n that maximizes the number of influenced nodes *while ensuring that node p is not activated*. Let $A(p, k)$ and $B(p, k)$ be the number of nodes that can be influenced by optimal solutions to $MIA(R_p, k)$ and $MIB(R_p, k)$ respectively. Then clearly,

$$MI(R_n, k) = max\{A(p, k), B(p, k)\}$$

Let us consider the case that node p does not belong to $I(R_n, S)$ first (see Figure 23). Since node p is not activated, then nodes in the range of $[CC(p) + 1, C(p) - 1]$ cannot receive any link; besides, both $CC(p)$ and $C(p)$ cannot be activated. Since S is optimal and uses the fewest links possible, S cannot give links to $CC(p)$ and $C(p)$. We now claim that S is an optimal solution for problem $MaxIn(P_{C(p)+1,CC(p)-1}, k)$.

Suppose not, let S' be a better solution for $MaxIn(P_{C(p)+1,CC(p)-1}, k)$ which can activate more nodes than S using k links. It is easy to see that S' is also a better solution for $MIB(R_p, k)$, a contradiction.

$$B(p,k) = MI(P_{C(p)+1,CC(p)-1}, k)$$

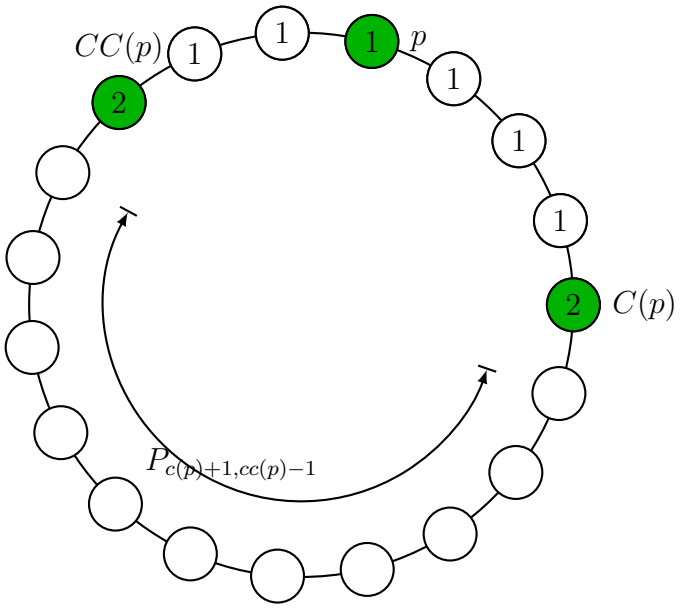

Figure 23: Node p does not belong to $I(R_n, S)$

If instead, node p belongs to $I(R_n, S)$ (see Figure 24). Since p is activated, then all nodes in the range $[CC(p)+1, C(p)-1]$ are activated too. Also, at least one node in the range of $[CC(p), C(p)]$ including $CC(p)$ and $C(p)$ must receive a link. We claim that this link can always be moved to p without reducing the size of the influenced set. If either one of $CC(p)$ and $C(p)$ receives a link, without loss of generality, we say $CC(p)$ receives a link, then we can move the link $CC(p)$ receives to node p while still effectively reducing the threshold of $CC(p)$ by one. If a node in the range of $[CC(p)+1, C(p)-1]$ receives a link, we can still move the link to node p while still be able to activate nodes in $[CC(p)+1, C(p)-1]$. This proves the claim.

Therefore, we assume that $p \in S$. We now claim that $S-\{p\}$ is an optimal solution for problem $MaxIn(P'_{C(p),CC(p)}, k-1)$. Suppose not, let S' be a better solution of $MaxIn(P'_{C(p),CC(p)}, k-1)$ which can activate more nodes than $S-\{p\}$ using $k-1$ links. It is easy to see that $S' \cup \{p\}$ is also a better solution for $MIA(R_p, k)$, a contradiction.

$$A(p,k) = C(p) - CC(p) - 1 + MI(P'_{C(p),CC(p)}, k-1)$$

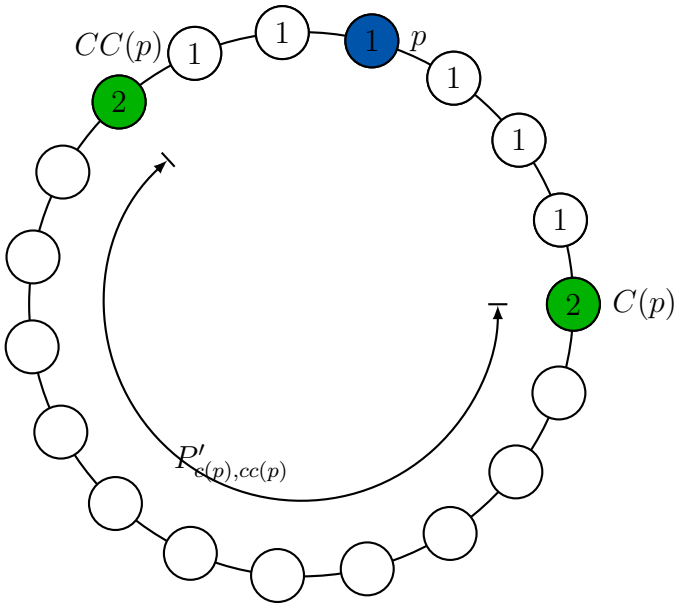

Figure 24: Node p belongs to $I(R_n, S)$

We develop the following algorithm to calculate $MI(R_n, k)$.

Algorithm 7 Influence maximization algorithm for rings

Input: (1) Ring $R_n = (V, E, t)$, $t : V \to \{1, 2\}$; (2) links available k;
Output: Maximum number of nodes that can be activated;
Select an arbitrary node p of threshold 1
$C(p) \leftarrow$ first node of threshold 2 in p's clockwise direction
$CC(p) \leftarrow$ first node of threshold 2 in p's counter clockwise direction
if p is not activated by the optimal assignment **then**
    $x \leftarrow MI(P_{C(p)+1,CC(p)-1}, k)$
if p is activated by the optimal assignment **then**
    $y \leftarrow C(p) - CC(p) - 1 + MI(P'_{C(p),CC(p)}, k-1)$
return $\max(x, y)$

Theorem 12 *The Maximum Influence problem for a ring R_n which consists of only node with threshold 1 or 2 can be solved in time $\theta(kn)$ where k is the number of links available.*

Proof. The time complexity of Algorithm 7 depends on the time complexity of Maximum Influence algorithm for a path. Since the Maximum Influence problem for a path can be solved in $O(kn)$ time by Theorem 7, so can $MaxIn(R_n, k)$. The optimal link set can also be construed using an optimal solution for the Minimum Links problem for a residual path in R_n. ■

Chapter 6

Trees

Tree model is widely used in the study of social networks such as *Facebook page tree* proposed by Sun *et al.* in [40], *retweet tree* introduced by Kwak *et al.* in [30] and *infection tree* proposed by Adar *et al.* in [1] which are all models used to track influence cascading in social networks.

Let $T_n = (V, E, t)$ be a tree with n nodes, $V = \{1, 2, ..., n\}$ and $t : t(v) \rightarrow \mathcal{Z}^+$. Fix an arbitrary root of the tree and order the children of every node in an arbitrary fashion. We define T_v and d_v to be the sub-tree rooted at node v, and the number of children of node v respectively. We define T_v^{-1} to be the same sub-tree as T_v except that the threshold of the root v is reduced by 1. We also define v_i to be i^{th} child of node v and T_{v_i} to be the sub-tree rooted at v_i.

6.1 Minimum Links problem

In this section, we study the Minimum Links problem in trees. Let $MinLinks(T_n)$ be the problem of finding a minimum-sized link assignment that activates the entire tree, and let $ML(T_n)$ be the minimum number of links needed to activate the entire tree.

We study the properties of tree based on its height. It is trivial to solve the Minimum Links problem for a tree of height 0. In order for a tree of height 0 to have a feasible solution, the threshold of the root must be 1 and it takes 1 link to activate such a tree.

6.1.1 Minimum Links problem for trees of height 1

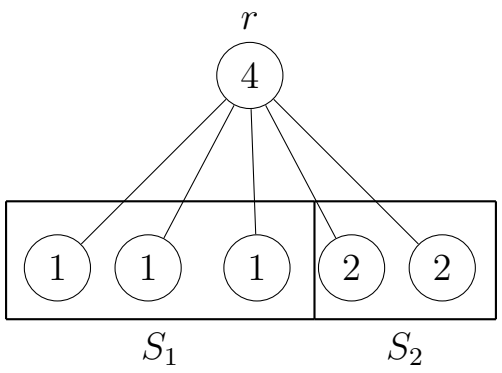

Figure 25: A star network

We call such a tree with height 1 (see Figure 25) a star network. Given a star network $T_r = (V, E, t)$ rooted at node r, we denote the set of leaf nodes in T_r by L. We define the set of nodes with threshold 1 and 2 in L to be S_1 and S_2 respectively.

First, we show a necessary and sufficient condition for the Minimum Links problem to have a feasible solution with a star network.

Proposition 3 *Given a star network $T_r = (V, E, t)$ rooted at node r, the Minimum Links problem has a feasible solution if and only if both the following conditions are met:*

1. *There is no node in L with threshold ≥ 3*
2. *$t(r) - |S_1| \leq 1$*

Proof. First we prove the necessary condition. Suppose there exists a leaf node $v \in L$ with $t(v) \geq 3$, it is easy to observe that it is impossible to activate v even if the root is activated before v. Suppose there is no node in L with threshold ≥ 3 and $t(r) - |S_1| > 1$, first we observe that nodes in S_2 can be activated only after the activation of root v, then even if all the nodes in S_1 are given a link, the effective threshold of r would remain greater than 1, so it is impossible to activate root r or nodes in S_2.

Second, we prove the sufficient condition. Suppose there exists a star network T_r which satisfies the two conditions in the statement of the lemma. Let S be a link assignment in which every node is given a link. Clearly, nodes in S_1 will be activated first, then the root r can also be activated and finally nodes in S_2 will be activated, so that S activates all the nodes in T_r. ■

Before we proceed to give a solution for the Minimum Links problem in a star network, we prove the following lemma which is applicable for multiple influencers and helpful for both MinLinks and Maximum Influence problems.

Lemma 5 *Suppose S is an assignment of k links to a tree T in which $v = root(T)$ is activated, and p links are assigned to v in S with $p < t(v)$. Then there exists an assignment S' with $t(v)$ links being assigned to v and the remaining $k - t(v)$ links to other nodes in T, such that $I(T, S') = I(T, S)$ and $|S'| = k$.*

Proof. We prove the lemma by induction on the height of the tree T. Clearly, the lemma is true for trees of height 0; the only way to activate such a tree is to give $t(v)$ links to the root v.

Now consider a tree of height h and an assignment S with p links assigned to v where $0 \leq p < t(v)$, such that v is activated. We will create a new assignment S' of the same size as S in which v is assigned $p + 1$ links and the activated set of nodes is exactly the same as in S.

Since in S, there are fewer than $t(v)$ links assigned to v, there must exist a child c of v which is activated before v, and contributes to the activation of v. By the inductive hypothesis, we can assume that $t(c)$ links have been assigned to c in S. Our new assignment S' is identical to S except that we assign $p+1$ links to v and $t(c)-1$ links to c. Any node except c that was activated by S before v, is also activated in S' before v. Therefore v will be activated by S', and will subsequently activate c. All other children of v that were activated after v in S will also be activated after v by S'. Therefore, $I(T,S) = I(T,S')$.

Clearly the above process can be repeated until $p = t(v)$. ■

The corollary below for the single influencer case follows immediately.

Corollary 1 *Suppose S is an assignment of k links to a tree T in which $v = root(T)$ is activated, and no link is assigned to v in S. Then there exists an assignment S' with a link being assigned to v and the remaining $k-1$ links to other nodes in T, such that $I(T,S') = I(T,S)$ and $|S'| = k$.*

Now, we are ready to give a solution for the Minimum Links problem in a star network.

Theorem 13 *Given a star network $T_r = (V,E,t)$ which admits a feasible solution, the minimum number of links required to activate T_r satisfies the following equation:*

$$ML(T_r) = 1 + \sum_{v \in V} (t(v) - 1)$$

Proof. By Corollary 1, there exists an optimal solution S in which the root r is given a link. Now consider such a solution S. It is easy to observe that we need to activate $t(r)-1$ leaves such that the root r can be activated, and nodes in S_2 can only be activated after the activation of r. Since $t(r) - 1 \leq |S_1|$, we give $t(r)-1$ links to nodes in S_1, then the root r as well as the remaining nodes in S_1 will be activated.

It is now necessary and sufficient to give a link to every node in S_2 to activate all the nodes in S_2.

So the minimum number of links required to activate T_r is $1 + t(r) - 1 + |S_2| = 1 + \sum_{v \in V}(t(v) - 1)$. ■

6.1.2 Minimum Links problem for trees of height > 1

Now we consider a tree with height > 1 (see Figure 26). We define $T_r - T_{r_1}$ to be the same tree as T_r after removing the sub-tree T_{r_1}, and as before, we also define $(T_r - T_{r_1})^{-1}$ to be same tree as $T_r - T_{r_1}$ except that the threshold of root r is effectively reduced by 1.

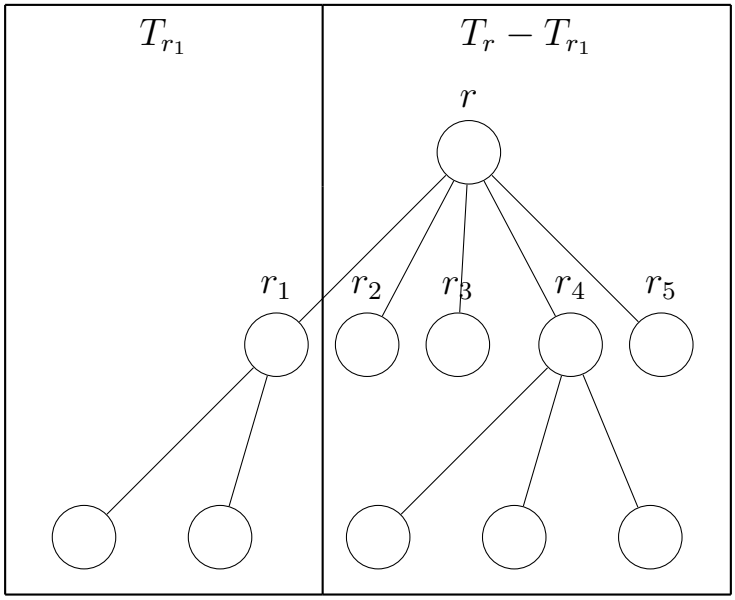

Figure 26: The definition for T_{r_1} and $(T_r - T_{r_1})$

We now give a necessary and sufficient condition for the Minimum Links problem to have a feasible solution for a tree of height greater than 1. The base case is tree of height ≤ 1.

Proposition 4 *Given a tree $T_r = (V, E, t)$ rooted at node r, the Minimum Links problem has a feasible solution if and only if one of the following conditions is met:*

1. *Both T_{r_1} and tree $(T_r - T_{r_1})^{-1}$ have feasible solutions;*

2. *Both $T_{r_1}^{-1}$ and tree $(T_r - T_{r_1})$ have feasible solutions;*

Proof. First we prove the sufficient condition for the Minimum Links problem to have a feasible solution.

- For (1), if both T_{r_1} and tree $(T_r - T_{r_1})^{-1}$ have feasible solutions, then let S be a solution for $MinLinks(T_{r_1})$ and S' be a solution for $MinLinks((T_r - T_{r_1})^{-1})$. We claim that $S \cup S'$ is a solution for $MinLinks(T_r)$. With S, every node in T_{r_1} including r_1 will be activated and with S', every node in $T_r^{-1} - T_{r_1}$ will be activated. Because of the activation of r_1, root r can be activated as well, and this completes the proof of the claim.

- For (2), if both $T_{r_1}^{-1}$ and tree $(T_r - T_{r_1})$ have feasible solutions, then let S be a solution to $MinLinks(T_{r_1}^{-1})$ and S' be a solution to $MinLinks(T_r - T_{r_1})$. We claim that $S \cup S'$ is a solution for $MinLinks(T_r)$. With S, every node in $T_{r_1^{-1}}$ will be activated and with S', every node in $T_r - T_{r_1}$ including r will be activated. Because of the activation of r, r_1 can be activated as well, and this completes the proof of the claim.

Second we prove the necessary condition for the Minimum Links problem to have a feasible solution. We define N_1 and N_2 to be the set of nodes in tree T_{r_1} and $(T_r - T_{r_1})$ respectively. Let S be a solution for $MinLinks(T_r)$. We claim that $S \cap N_1$ is either a solution for $MinLinks(T_{r_1})$ or a solution for $MinLinks(T_{r_1}^{-1})$. If not, it is easy to observe that it is impossible for S to activate all nodes in T_{r_1}. Suppose $S \cap N_1$ is a solution for $MinLinks(T_{r_1})$, then $S \cap N_2$ must be a solution for $MinLinks((T_r - T_{r_1})^{-1})$, otherwise it is impossible for S to activate all nodes in $(T_r - T_{r_1})$. Suppose next $S \cap N_1$ is a solution for $MinLinks(T_{r_1}^{-1})$, then $S \cap N_2$ must

be a solution for $MinLinks(T_r - T_{r_1})$, otherwise it is impossible for S to activate all nodes in $(T_r - T_{r_1})$.

■

We give the following algorithm to verify if T_r admits a feasible solution.

Algorithm 8 $IsFeasible(T_r)$ - Feasibility checking for trees

```
1: Input: Tree T_r = (V, E, t)
2: Output: T_r admits a feasible solution or not
3: if Height of T_r == 0 then
4:     if t(r) ≤ 1 then
5:         return true
6:     else
7:         return false
8: else
9:     if IsFeasible(T_{r_1}) & IsFeasible((T_r − T_{r_1})^{-1}) then
10:        return true
11:    else if IsFeasible(T_{r_1}^{-1}) & IsFeasible(T_r − T_{r_1}) then
12:        return true
13:    else
14:        return false
```

We proceed to give a solution for Minimum Links problem for tree with height greater than 1.

Theorem 14 *Given a tree $T_r = (V, E, t)$ which admits a feasible solution, the minimum number of links required to activate T_r satisfies the following equation:*

$$ML(T_r) = 1 + \sum_{v \in V} (t(v) - 1)$$

Proof. We give a proof by strong double induction on the height of T_r and the number of children of root r. Clearly, the statement is true for trees of height 1 where the root has any number of children by Theorem 13.

First consider a tree $T_r = (V, E, t)$ of height $h > 1$ where the root r has only one child. Suppose for any tree T_r of height $< h$ where the root has any number of children, $ML(T_r) = 1 + \sum_{v \in V}(t(v) - 1)$. We now prove the following claim.

Claim 4 *The minimum number of links required to activate a tree $T_r = (V, E, t)$ of height $h > 1$ where the root r has only one child, satisfies the following equation:*

$$ML(T_r) = 1 + \sum_{v \in V}(t(v) - 1)$$

Proof. First we claim that $t(r) \leq 2$ if root r has only one child. If $t(r) > 2$, then even if the only child r_1 is activated before r and r is given a link, it is still impossible to activate root r.

By Corollary 1, there exists an optimal solution S for problem $MinLinks(T_r)$ in which root r receives a link. Let us consider such a solution S.

If $t(r) = 1$, by the usual cut-and-past argument, we can prove $S - \{r\}$ must be an optimal solution for tree $T_{r_1}^{-1}$ which is of height $h - 1$. The minimum number of links required to activate $T_{r_1}^{-1}$ is $1 + \sum_{v \in T_{r_1}^{-1}}(t(v) - 1) = \sum_{v \in T_{r_1}}(t(v) - 1)$ by the induction hypothesis. So the overall number of links required is $1 + \sum_{v \in T_{r_1}}(t(v) - 1)$ $= 1 + \sum_{v \in V}(t(v) - 1)$.

If $t(r) = 2$, giving root r a link is not sufficient to activate it, r_1 must be activated before r. By the usual cut-and-past argument, $S - \{r\}$ must be an optimal solution for tree T_{r_1} which is of height $h-1$. The minimum number of links required to activate T_{r_1} is $1+\sum_{v \in T_{r_1}}(t(v)-1)$ by the induction hypothesis. So the overall number of links required is $1 + 1 + \sum_{v \in T_{r_1}}(t(v) - 1) = 1 + \sum_{v \in V}(t(v) - 1)$.

In both cases, $ML(T_n) = 1 + \sum_{v \in V}(t(v) - 1)$. We can conclude that for any tree with height > 1 where the root has only one child, $ML(T_n) = 1 + \sum_{v \in V}(t(v) - 1)$. ■

Now consider a tree $T_r = (V, E, t)$ of height $h > 1$ where the root r has $n > 1$

children. Suppose for any tree of height $< h$ where the root has any number of children, and for any tree of height h where the root has $n-1$ children, $ML(T_r) = 1 + \sum_{v \in V}(t(v) - 1)$. We now prove the following claim.

Claim 5 *The minimum number of links required to activate a tree $T_r = (V, E, t)$ of height $h > 1$ where the root r has $n > 1$ children, satisfies the following equation:*

$$ML(T_r) = 1 + \sum_{v \in V}(t(v) - 1)$$

Proof. By Proposition 4, T_r must satisfy at least one of these two conditions: (1) Both T_{r_1} and $(T_r - T_{r_1})^{-1}$ have feasible solutions; (2) Both $T_{r_1}^{-1}$ and $(T_r - T_{r_1})$ have feasible solutions. Observe that, T_{r_1} is a tree of height $< h$ and $(T_r - T_{r_1})$ is a tree of height $\leq h$ and the root r has $n-1$ children.

- **Case 1 : T_r only satisfies condition (1).**

  It is easy to observe that if T_r has a feasible solution, $T_{r_1}^{-1}$ must also has a feasible solution; if $(T_r - T_{r_1})$ has a feasible solution, then so will $(T_r - T_{r_1})^{-1}$. Since T_r only satisfies condition (1), then $(T_r - T_{r_1})$ has no feasible solution in this case. The only way to activate T_r is to activate T_{r_1} first, then to activate $(T_r - T_{r_1})^{-1}$. By the induction hypothesis, the minimum number of links required to activate T_{r_1} is $1 + \sum_{v \in T_{r_1}}(t(v) - 1)$ and the minimum number of links required to activate $(T_r - T_{r_1})^{-1}$ is $1 + \sum_{v \in T_r^{-1} - T_{r_1}}(t(v) - 1) = \sum_{v \in T_r - T_{r_1}}(t(v) - 1)$. So the overall number of links required is $1 + \sum_{v \in T_{r_1}}(t(v) - 1) + \sum_{v \in T_r - T_{r_1}}(t(v) - 1)$ $= 1 + \sum_{v \in V}(t(v) - 1)$.

- **Case 2 : T_r only satisfies condition (2).**

  Since T_r only satisfies condition (2), following a similar argument as in case 1, it is easy to see that T_{r_1} has no feasible solution in this case. Therefore, the

only way to activate T_r is to activate $(T_r - T_{r_1})$ first, then to activate T_r^{-1}. The minimum number of links required to activate $T_{r_1}^{-1}$ is $1 + \sum_{v \in T_{r_1}^{-1}}(t(v) - 1) = \sum_{v \in T_{r_1}}(t(v)-1)$ and the minimum number of links required to activate $(T_r - T_{r_1})$ is $1 + \sum_{v \in T_r - T_{r_1}}(t(v) - 1)$ by the induction hypothesis . So overall number of links required is $\sum_{v \in T_{r_1}}(t(v)-1) + 1 + \sum_{v \in T_r - T_{r_1}}(t(v)-1) = 1 + \sum_{v \in V}(t(v)-1)$.

- **Case 3 : T_r only satisfies both condition (1) and (2).**

  In this case, both T_{r_1} and $(T_r - T_{r_1})$ have feasible solutions. Suppose S is an optimal solution for $MinLinks(T_r)$ which gives root r a link, we claim that either r is activated before r_1 by S or vice versa. Suppose instead that r and r_1 are activated in the same time step by S. We create a new assignment S' by removing the link assigned to r. All nodes that are activated before r and r_1 in S will still be activated by S', so r_1 will still be activated, and r will be activated in the next step after the activation of r_1. The remaining nodes that will be activated after r and r_1 in S will also be activated by S'. Therefore, S' is a smaller set than S to activate T_r, which contradicts the optimality of S. This proves that r is activated before r_1 by S or vice versa.

  If r_1 is activated before r by S, an argument similar to case 1 proves the claim. If r is activated before r_1 by S, an argument similar to case 2 proves the claim. In both cases, the minimum number of links required to activate T_r is $= 1 + \sum_{v \in V}(t(v) - 1)$.

■

This completes the proof of the inductive statement. We can conclude that we need $1 + \sum_{v \in V}(t(v) - 1)$ links to activate any tree T_r with height > 1. ■

We proceed to give the algorithm for finding an optimal link set for a tree T_r which admits a feasible solution.

Algorithm 9 Minimum Links algorithm for paths - $MinLinks(T_r)$

```
1: Input: tree T_r = (V, E, t);
2: Output: an optimal link set S
3: if Height of T_r == 0 then
4:     if t(r) == 1 then
5:         return {r}
6:     if t(r) > 1 then
7:         return No feasible solution
8: else if IsFeasible(T_{r_1}) & IsFeasible((T_r − T_{r_1})^{-1}) then
9:     S_1 ← MinLinks(T_{r_1})
10:    S_2 ← MinLinks((T_r − T_{r_1})^{-1})
11:    return S_1 ∪ S_2
12: else if IsFeasible(T_{r_1}^{-1}) & IsFeasible(T_r − T_{r_1}) then
13:    S_1 ← MinLinks(T_{r_1}^{-1})
14:    S_2 ← MinLinks(T_r − T_{r_1})
15:    return S_1 ∪ S_2
16: else
17:    return No feasible solution
```

Theorem 15 *The Minimum Links problem for a tree $T_n = (V, E, t)$ can be solved in time $\Theta(n)$.*

Proof. We do a bottom-up implementation of the above algorithm. When we come to node v, we assume that we already have available solutions for T_x as well as T_x^{-1} for each child x of v. To build the solution for T_v and T_v^{-1}, in the bottom-up implementation, we would start with the subtree of T_v containing only the last child of v, which takes constant time, and then successively add in the subtrees rooted at other children of v. Adding each such subtree takes constant time. Therefore, the total amount of time needed to compute the solutions for T_v and T_v^{-1} is $O(d(v))$ where $d(v)$ is the degree of v. The total time therefore for the entire tree is $\Theta(\Sigma_{v \in T} d(v)) = \Theta(n)$.

■

6.2 Maximum Influence problem

In this section, we study the Maximum Influence problem in trees. Let $MaxIn(T_n, k)$ be the problem of finding an assignment of at most k links to nodes in tree T_n that maximizes the number of influenced nodes and let $MI(T_n, k)$ be the maximum number of nodes that can be influenced in the tree using k links.

Since some nodes might not be activated by the optimal assignment for the Maximum Influence problem, in order to facilitate our discussion, we develop some new notation. We define $MIA(T_v, k)$ to be the problem of finding an assignment of at most k links to nodes in the sub-tree T_v that maximizes the number of influenced nodes *while ensuring that root v is activated*. Similarly, let $MIB(T_v, k)$ be the problem of finding an assignment of at most k links to nodes in the sub-tree T_v that maximizes the number of influenced nodes *while ensuring that root v is not activated*. Let $A(T_v, k)$ and $B(T_v, k)$ be the number of nodes that can be influenced by optimal solutions to $MIA(T_v, k)$ and $MIB(T_v, k)$ respectively. Clearly, given a tree $T_n = (V, E, t)$ rooted at node r, by an optimal assignment S for problem $MI(T_n, k)$, root r is either activated or not. Therefore:

$$MI(T_n, k) = max\{A(T_r, k), B(T_r, k)\}$$

We provide a examples to show that the optimal solution for the Maximum Influence problem may or may not activate the root, depending on the structure of the tree and thresholds in the tree. Therefore, it is necessary to consider both cases.

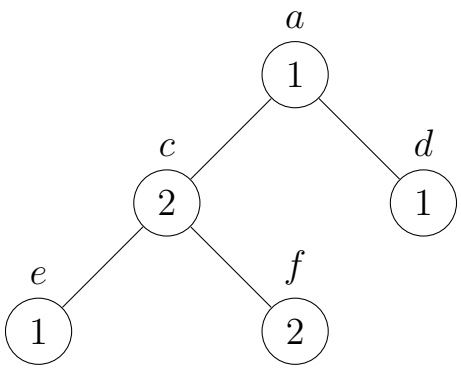

Figure 27: Maximum Influence problem in a tree with $k = 1$

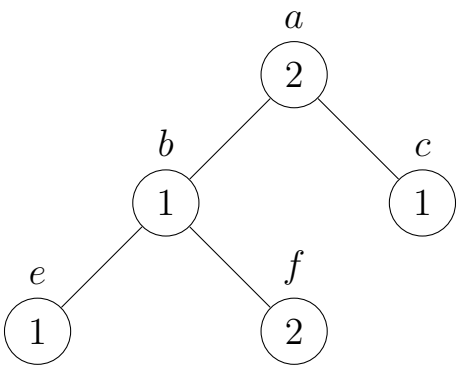

Figure 28: Maximum Influence problem in a different tree with $k = 1$

Observe that for the tree in Figure 27, $A(T_a, 1) = 2$ if we choose $\{a\}$ as the link set, and $B(T_a, 1) = 1$ if we choose $\{e\}$ as the link set, therefor $MI(T_a, 1) = A(T_a, 1) = 2$. However, for the tree in Figure 28, $A(T_a, 1) = 0$ as there does not exist a link set of size 1 which can activate the root, and $B(T_a, 1) = 2$ if we choose $\{b\}$ as the link set, therefore $MI(T_a, 1) = B(T_a, 1) = 2$.

Before we proceed to give the dynamic programming formulation for $A(T_v, k)$ and $B(T_v, k)$, we give a small example here.

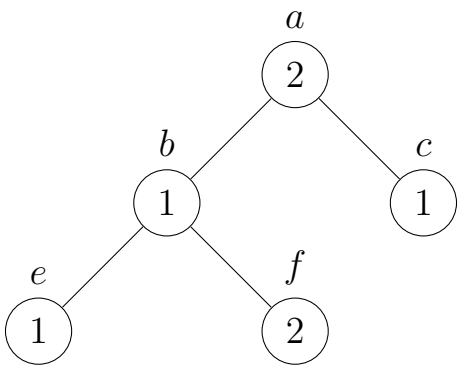

Figure 29: Maximum Influence problem in a tree with $k = 2$

Consider the tree in Figure 29. We need to compute $MI(T_a, 2)$ which is equal to $\max\{A(T_a, 2), B(T_a, 2)\}$.

- Computation of $A(T_a, 2)$

By Corollary 1, there must exist an optimal assignment in which the root receives a link, so we give a link to node a. Giving node a a link is not sufficient to activate node a, one of a's children has to be activated before a. Let $F_{a,2}$ be the forest composed of T_b and T_c, we are now facing a sub-problem of finding an optimal assignment of 1 link to $F_{a,2}$ that maximizes the number of influenced nodes in $F_{a,2}$ while ensuring that either b or c is activated before a.

Suppose c is activated before a, it is easy to observe that we need to give a link to node c. The overall activated nodes would be the activated nodes in T_c plus root v plus the nodes that would be activated by v. After the activation of c, node a will be activated, then node b and e will also be activated in the next time step.

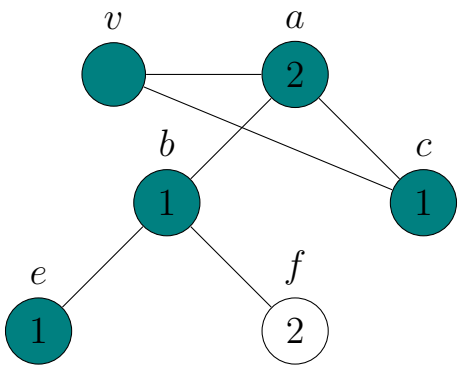

Figure 30: Node c is activated before a

Suppose b is activated before a, it is easy to observe that we need to give a link to b by Corollary 1. The overall activated nodes would be the activated nodes in T_b plus root v plus the nodes that would be activated by v. After the activation of b, node e and a will be activated in the next time step, then node c will also be activated.

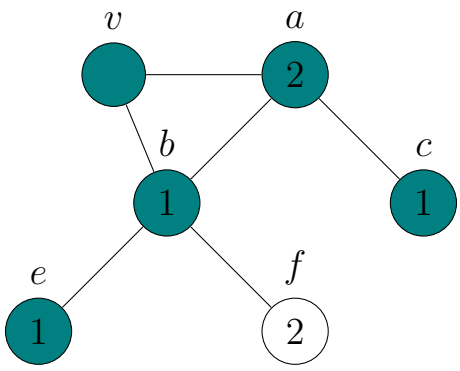

Figure 31: Node b is activated before A

In both cases, $A(T_a, 2) = 4$

- Computation of $B(T_a, 2)$

  Since the root a is not activated in $B(T_a, 2)$, we don't give any link to a. We are now facing a sub-problem of finding an optimal assignment of 2 link to $F_{a,2}$ that maximizes the number of influenced nodes while ensuring that root a is not activated which means that at most one node between b and c can be activated.

  Suppose b is the node that is allowed to be activated, we can activate nodes b, e and f in this case, therefore we have three active nodes.

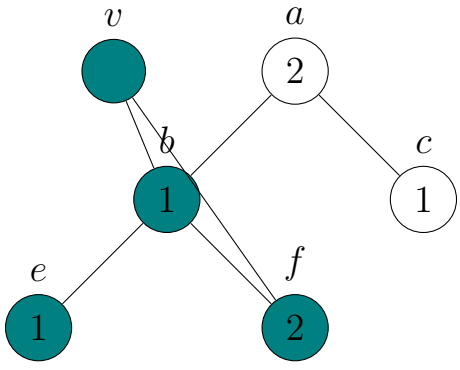

Figure 32: Node b is activated

Suppose c is the node that is allowed to be activated, we can only activate node c in this case, therefore we have only one active node.

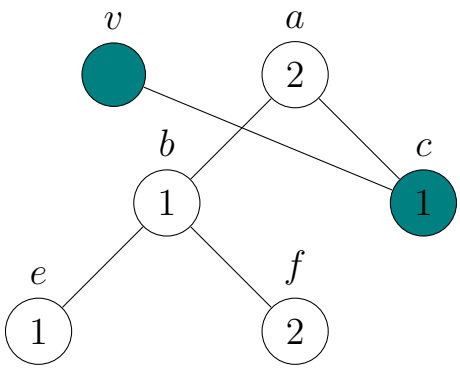

Figure 33: Node c is activated

Overall, $B(T_a, 2) = 3$

From the previous analysis, we observe that in the computation of $A(T_v, k)$ and $B(T_v, k)$, we will face the sub-problem of distributing links among the forest formed by sub-trees rooted at the children of v. We define $MIA(F_{v,d}, i, k)$ to be the problem of finding an assignment of at most k links to nodes in the forest $F_{v,d}$ consisting of sub-trees from T_{v_1} to T_{v_d} that maximizes the number of influenced nodes *while ensuring that there are i out of d children of root v are activated before v*. Similarly, we define $MIB(F_{v,d}, i, k)$ to be the problem of finding an assignment of at most k links to nodes in the forest $F_{v,d}$ that maximizes the number of influenced nodes *while ensuring that there are i out of d children of root v are activated*. Let $A(F_{v,d}, i, k)$ and $B(F_{v,d}, i, k)$ be the number of influenced nodes by optimal solutions to $MIA(F_{v,d}, i, k)$ and $MIB(F_{v,d}, i, k)$ respectively.

Now we are ready to give the recursive formulations of $A(T_v, k)$ and $B(T_v, k)$.

6.2.1 Computation of $A(T_v, k)$

In the computation of $A(T_v, k)$, we want to find an assignment of at most k links to nodes in the sub-tree T_v such that the number of influenced nodes will be maximized

while ensuring that root v is activated. Based on the relationship of $t(v)$ and k, the following four cases are exhaustive:

1) if $t(v) > k \geq 0$, it is impossible to activate the root v with only k links.

   Lemma 6 *It is impossible to activate the root of a sub-tree T_v when the number of links $k < t(v)$.*

   Proof. We prove this lemma by induction on the height of the tree T. Clearly, the lemma is true for trees of height 0.

   Now consider a tree of height h. Suppose we give all k links to root v, we cannot activate any node in the tree by such an assignment. Suppose next we give p $(0 \leq p < k)$ links to root v, and give $k - p$ links to forest F_{v,d_v}. In such an assignment, we need $t(v) - p$ children of v to be activated before v. By the inductive hypothesis, it is impossible to activate the root v with less than $t(v)$ links when the height is $h - 1$. Since $k < t(v)$, then $k - p < t(v) - p$; so we cannot activate $t(v) - p$ roots in the forest F_{v,d_v} with less than $t(v) - p$ links since the forest is consisting of trees with height less than h. This completes the proof. ■

2) if $t(v) = 1$ and $k \geq 1$, by Corollary 1, there exists an optimal assignment S in which root v receives a link. We claim that $S - \{v\}$ is also an optimal solution to problem $MIA(F_{v,d_v}, 0, k-1)$. Suppose not, let $|S| = k$ and let S' be an optimal solution to $MIA(F_{v,d_v}, 0, k-1)$ which can activate more nodes than $S - \{v\}$. Then $S' \cup \{v\}$ will be an assignment of k links which can activate more nodes than S, contradicting the optimality of S. Therefore:

$$A(T_v, k) = 1 + A(F_{v,d_v}, 0, k-1)$$

3) if $t(v) = 0$ and $k \geq 0$, we don't allow nodes with threshold 0 in the input, but in the computation process, we might come across a sub-tree T_v^{-1} in which the root

has effective threshold 0. Consider the tree in Figure 27, if we give root a a link, the threshold of d will be reduced by 1 to reach 0. When we start processing a sub-tree T_v, root v is the only node that might have threshold 0. Similar to case 2, we have

$$A(T_v, k) = 1 + A(F_{v,d_v}, 0, k)$$

4) if $1 < t(v) \leq k$, giving root v a link does not suffice to activate v, $t(v) - 1$ of v's children have to be activated before v, otherwise v cannot be activated. By Corollary 1, there exists an optimal assignment S which gives root v a link. We claim that $S - \{v\}$ is an optimal solution to problem $MIA(F_{v,d_v}, t(v)-1, k-1)$. Suppose not, let $|S| = k$ and let S' be an optimal solution to $MIA(F_{v,d_v}, t(v) - 1, k - 1)$ which can activate more nodes than $S - \{v\}$. With S', $t(v) - 1$ children of v has been activated. Adding a link to v, node v will be activated as well, so will the rest nodes that will be activated due to the activation of v. Then $S' \cup \{v\}$ will be an assignment of k links which can activate more nodes than S, contradicting the optimality of S. Therefore:

$$A(T_v, k) = 1 + A(F_{v,d_v}, t(v) - 1, k - 1)$$

Putting together all cases for $A(T_v, k)$, we obtain:

$$A(T_v, k) = \begin{cases} 1+A(F_{v,d_v}, 0, k - t(v)) & \text{if } t(v) \leq 1 \textit{ and } t(v) \leq k \\ 1+A(F_{v,d_v}, t(v) - 1, k - 1) & \text{if } 1 < t(v) \leq k \\ -\infty & \text{if } k < t(v) \end{cases}$$

6.2.2 Computation of $A(F_{v,d}, i, k)$

In the computation of $A(F_{v,d}, i, k)$, we want to find an assignment of at most k links to nodes in the forest $F_{v,d}$ that maximizes the number of influenced nodes while ensuring

that there are i out of d children of root v are activated before v.

We use the dynamic programming approach to solve this problem. Let S be an optimal solution to $MIA(F_{v,d}, i, k)$, we formulate the recursive equations based on the relationship of i and d. We give the base case for $A(F_{v,d}, i, k)$ where $d = 0$, $A(F_{v,0}, i, k) = 0$ for all i and k.

1) if $i > d$, clearly it is impossible to activate exactly i of the first d children of v when $i > d$.

$$A(F_{v,d}, i, k) = -\infty$$

2) if $i = d$, this means that all children v_j with $1 \leq j \leq d$ have to be activated before v. The optimal solution S will assign p links to $F_{v,d-1}$ and $k - p$ links to T_{v_d}, so we try all possibilities of p to find the best distribution.

$$A(F_{v,d}, i, k) = \max_{0 \leq p \leq k} \{A(F_{v,d-1}, i-1, p) + A(T_{v_d}, k-p)\}$$

3) if $i < d$, v_d may or may not be activated before v by an optimal solution S when $d > i$

   (a) v_d is not activated before v by S. Observe that v_d may or may not be activated after v. Besides, due to the fact that its parent has already been activated, the threshold of v_d is effectively reduced by one. An optimal assignment S will assign p links to $F_{v,d-1}$ and $k - p$ links to T_{v_d}, therefore we try all possibilities of p to find the best distribution. Therefore:

$$A(F_{v,d}, i, k) = \max_{0 \leq p \leq k} \{A(F_{v,d-1}, i, p) + max \begin{cases} A(T_{v_d}^{-1}, k-p) \\ B(T_{v_d}^{-1}, k-p) \end{cases} \}$$

   (b) v_d is activated before v by S. In this case, only $i - 1$ children of v in $F_{v,d-1}$ are required to be activated before v. Since v_d contributes to the activation of v, the threshold of v_d remains unchanged.

$$A(F_{v,d}, i, k) = \max_{0 \leq p \leq k} \{A(F_{v,d-1}, i-1, p) + A(T_{v_d}, k-p)\}$$

When $i = 0$, we only need to consider situation (a), instead of both (a) and (b).

Putting together all cases for $A(F_{v,d}, i, k)$ we obtain:

$$A(F_{v,d}, i, k) = \begin{cases} -\infty & \text{if } i > d \\ \max\limits_{0 \le p \le k} \{A(F_{v,d-1}, i-1, p) + A(T_{v_d}, k-p)\} & \text{if } i = d > 0 \\ max \begin{cases} \max\limits_{0 \le p \le k} A(F_{v,d-1}, i, p) + max \begin{cases} A(T_{v_d}^{-1}, k-p) \\ B(T_{v_d}^{-1}, k-p) \end{cases} \\ \max\limits_{0 \le p \le k} \{A(F_{v,d-1}, i-1, p) + A(T_{v_d}, k-p)\} \end{cases} & \text{if } 0 < i < d \\ \max\limits_{0 \le p \le k} A(F_{v,d-1}, i, p) + max \begin{cases} A(T_{v_d}^{-1}, k-p) \\ B(T_{v_d}^{-1}, k-p) \end{cases} & \text{if } i = 0 \end{cases}$$

6.2.3 Computation of $B(T_v, k)$

In the computation of $B(T_v, k)$, we want to find an assignment of at most k links to nodes in the sub-tree T_v such that the number of influenced nodes will be maximized while ensuring that root v is not activated. Since the root v is not activated in $B(T_v, k)$, we will not give any link to v. Also observe that the number of children of v which are activated cannot exceed $t(v)$, otherwise v will also get activated.

We claim that an optimal assignment S for $MIB(T_v, k)$ is also an optimal assignment for $MIB(F_{v,d}, i, k)$ where $0 \le i \le t(v) - 1$. Suppose not, let S' be an optimal solution to $MIB(F_{v,d}, i, k)$ $(0 \le i \le t(v) - 1)$ which can activate more nodes than S. With S', at most $t(v) - 1$ children of v can be activated, therefore root v will not be activated. It is easy to see that S' is a better solution for $MIB(T_v, k)$, contradicting the optimality of S. Therefore:

$$B(T_v, k) = \begin{cases} \max\limits_{0 \le i < min(t(v), d_v+1)} B(F_{v,d_v}, i, k) & \text{if } t(v) > 0 \\ -\infty & \text{if } t(v) \le 0 \end{cases}$$

6.2.4 Computation of $B(F_{v,d}, i, k)$

In the computation of $B(F_{v,d}, i, k)$, we want to find an assignment of at most k links to nodes in the forest $F_{v,d}$ that maximizes the total number of influenced nodes while ensuring that there are i out of d children of root v are activated.

We still use the dynamic programming approach to solve this problem. Let S be an optimal solution to $MIB(F_{v,d}, i, k)$, we formulate the recursive equations based on the relationship of i and d. We give the base case for $B(F_{v,d}, i, k)$ where $d = 0$, $B(F_{v,0}, i, k) = 0$ for all i and k.

1) if $i > d$, clearly it is impossible to activate exactly i of the first d children of v when $i > d$, and therefore

$$B(F_{v,d}, i, k) = -\infty$$

2) if $i = d$, same as the case for $A(F_{v,d}, i, k)$, v_d has to be activated by S when $d = i$. An optimal solution S will assign p links to $F_{v,d-1}$ and $k - p$ links to T_{v_d}, so we try all possible values of p to find the best distribution.

$$B(F_{v,d}, i, k) = \max_{0 \le p \le k} \{B(F_{v,d-1}, i-1, p) + A(T_{v_d}, k-p)\}$$

3) if $i < d$, v_d may or may not be activated by S when $d > i$.

   (a) v_d is not activated by S. In this case, S must assign p links to $F_{v,d-1}$ and $k - p$ links T_{v_d}. To find the optimal assignment, it suffices to try all possible values of $(p, k - p)$.

$$B(F_{v,d}, i, k) = \max_{0 \le p \le k} \{B(F_{v,d-1}, i, p) + B(T_{v_d}, k-p)\ \}$$

(b) v_d is activated by S. In this case, only $i-1$ children of v in $F_{v,d-1}$ should be activated. As before, we try all possible values of $(p, k-p)$ with p links assigned to $F_{v,d-1}$ and $k-p$ links assigned to T_{v_d} to find the optimal assignment.

$$B(F_{v,d}, i, k) = \max_{0 \le p \le k} \{B(F_{v,d-1}, i-1, p) + A(T_{v_d}, k-p)\}$$

When $i = 0$, we only need to consider situation (a), instead of both (a) and (b).

Putting together all cases for $B(F_{v,d}, i, k)$, we obtain:

$$B(F_{v,d}, i, k) = \begin{cases} -\infty & \text{if } i > d \\ \max\limits_{0 \le p \le k} \{B(F_{v,d-1}, i-1, p) + A(T_{v_d}, k-p)\} & \text{if } i = d > 0 \\ max \begin{cases} \max\limits_{0 \le p \le k} \{B(F_{v,d-1}, i, p) + B(T_{v_d}, k-p)\} \\ \max\limits_{0 \le p \le k} \{B(F_{v,d-1}, i-1, p) + A(T_{v_d}, k-p)\} \end{cases} & \text{if } 0 < i < d \\ \max\limits_{0 \le p \le k} \{B(F_{v,d-1}, i, p) + B(T_{v_d}, k-p)\} & \text{if } i = 0 \end{cases}$$

6.2.5 Base case for leaves

Consider a leaf node l, we give the base case for $A(T_v, k)$ and $B(T_v, k)$ here.

$$A(T_l, k) = \begin{cases} 1 & \text{if } t(v) \le 1 \,\&\, k \ge t(v) \\ -\infty & \text{otherwise} \end{cases}$$

$$B(T_l, k) = \begin{cases} 0 & \text{if } t(v) > 0 \\ -\infty & \text{if } t(v) \le 0 \end{cases}$$

Algorithm 10 Modified BFS

Input: Tree $T_n = (V, E, t)$ rooted at node r
Output: nodes in T_n ordered by levels;
$L \leftarrow List\langle List\langle Node\rangle\rangle$
Running BFS from the root r,
$H \leftarrow$ the height of T_n
$h \leftarrow 0$
while $h < H$ **do**
    add nodes in layer h to L
    $h \leftarrow h + 1$
return L;

Algorithm 11 Influence maximization algorithm for trees

Input: (1) Tree $T_n = (V, E, t)$ and level traversal of the tree L; (2) links available k;
Output: Maximum number of nodes that can be activated;
for $i \leftarrow H-1, 0$ **do**
  for each node $v \in L_i$ **do**
    if $d_v = 0$ **then** ▷ base case for leaf node
      if $t(v) == 2$ **then**
        $A(T_v, k) \leftarrow -\infty$ for all k
        $A(T_v^{-1}, k) \leftarrow 1$ for all $k \geq 1$
        $B(T_v, k) = B(T_v^{-1}, k) \leftarrow 0$ for all k
      else if $t(v) == 1$ **then**
        $A(T_v, k) \leftarrow 1$ for all $k \geq 1$
        $A(T_v^{-1}, k) \leftarrow 1$ for all k
        $B(T_v, k) \leftarrow 0$ for all k
        $B(T_v^{-1}, k) \leftarrow -\infty$ for all k
      else
        $A(T_v, k) = A(T_v^{-1}, k) \leftarrow -\infty$ for all k
        $B(T_v, k) = B(T_v^{-1}, k) \leftarrow 0$ for all k
    else ▷ Non-leaf node
      for $p \leftarrow 0, k$ **do** ▷ base case for $A(F_{v,1}, i, p)$
        $A(F_{v,1}, 0, p) \leftarrow \max\{A(T_{v_1}^{-1}, p), B(T_{v_1}^{-1}, p)\}$
        $B(F_{v,1}, 0, p) \leftarrow B(T_{v_1}^{-1}, p)$
        $A(F_{v,1}, 1, p) \leftarrow A(T_{v_1}, p)$, $B(F_{v,1}, 1, p) \leftarrow A(T_{v_1}, p)$
        for $i \leftarrow 2, d_v$ **do**
          $A(F_{v,1}, i, p) \leftarrow -\infty$, $B(F_{v,1}, i, p) \leftarrow -\infty$
      for $d \leftarrow 2, d_v$ **do** ▷ compute $A(F_{v,d}, i, p)$ and $A(F_{v,d}, i, p)$ for node v
        for $i \leftarrow 0, d_v$ **do**
          if $i > d$ **then** ▷ first case
            $A(F_{v,d}, i, p) \leftarrow -\infty$, $B(F_{v,d}, i, p) \leftarrow -\infty$
          if $i == d > 0$ **then** ▷ second case
            $A(F_{v,d}, i, p) \leftarrow \max\limits_{0 \leq p \leq k}\{A(F_{v,d-1}, i-1, p) + A(T_{v_d}, k-p)\}$
            $B(F_{v,d}, i, p) \leftarrow \max\limits_{0 \leq p \leq k}\{B(F_{v,d-1}, i-1, p) + A(T_{v_d}, k-p)\}$
          if $0 < i < d$ **then** ▷ third case
            $A(F_{v,d}, i, p) \leftarrow \max\{\max\limits_{0 \leq p \leq k}\{A(F_{v,d-1}, i-1, p) + A(T_{v_d}, k-p)\}, \max\limits_{0 \leq p \leq k}\{A(F_{v,d-1}, i, p) + max\{A(T_{v_d}^{-1}, k-p), B(T_{v_d}^{-1}, k-p)\}\}\}$
            $B(F_{v,d}, i, p) \leftarrow \max\{\max\limits_{0 \leq p \leq k}\{B(F_{v,d-1}, i-1, p) + B(T_{v_d}, k-p)\}, \max\limits_{0 \leq p \leq k}\{B(F_{v,d-1}, i-1, p) + A(T_{v_d}, k-p)\}\}$

Algorithm 12 Influence maximization algorithm for trees part2

35: **if** $i == 0$ & $d > 1$ **then** ▷ last case

36: $A(F_{v,d}, i, p) \leftarrow \max_{0 \le p \le k} \{A(F_{v,d-1}, i-1, p) + max\{A(T_{v_d}^{-1}, k-p), B(T_{v_d}^{-1}, k-p)\}\}$

37: $B(F_{v,d}, i, p) \leftarrow \max_{0 \le p \le k} \{B(F_{v,d-1}, i-1, p) + B(T_{v_d}, k-p)\}$

38: **for** $p \leftarrow 0, k$ **do** ▷ compute $A(T_v, k)$ for node v

39: **if** $t(v) \le 1$ & $t(v) \le p$ **then**

40: $A(T_v, p) = 1 + A(F_{v,d_v}, 0, p - t(v))$

41: **if** $t(v) \le 1$ *and* $t(v)$ **then**

42: $A(T_v, p) = 1 + A(F_{v,d_v}, t(v) - 1, p - t(v))$

43: **if** $p < t(v)$ **then**

44: $A(T_v, p) = -\infty$

45: **for** $p \leftarrow 0, k$ **do** ▷ compute $B(T_v, k)$ for node v

46: **if** $t(v) > 0$ **then**

47: $B(T_v, p) = \max_{0 \le i < min(t(v), d_v+1)} \{B(F_{v,d_v}, i, k)\}$

48: **if** $t(v) \le 0$ **then**

49: $B(T_v, p) = -\infty$

50: **return** $max\{A(T_r, k), B(T_r, k)\}$

Theorem 16 *The Maximum Influence problem for a tree $T_n = (V, E, t)$ using k links, can be solved in time $O(n^2k^2)$.*

Proof. In the worst case, for any node w with d children, we need to compute $A(F_{v,p}, i, j)$ and $B(F_{v,p}, i, j)$ for $1 \le p \le d$, $0 \le i \le d$, and $0 \le j \le k$. These are d^2k values, each of which can take $\Theta(k)$ time to compute. The values $A(T_v, k)$ and $B(T_v, k)$ can be computed in constant time, and there are k such values. Therefore, the total time taken is $O(n^2k^2)$. ■

Chapter 7

Cliques

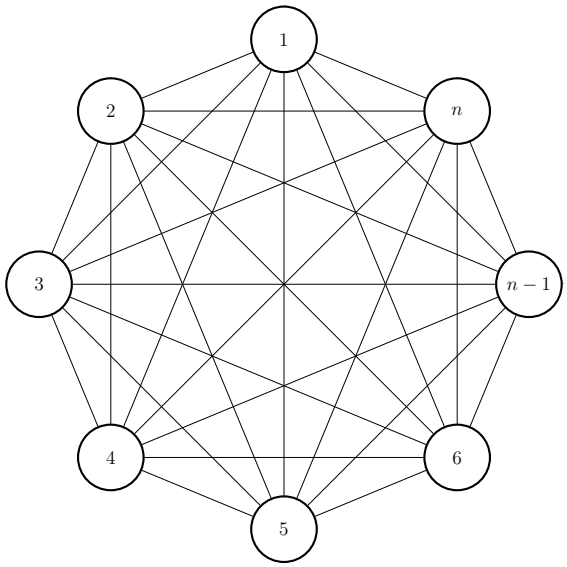

Figure 34: A clique of size N

In this chapter, we study the Minimum Links problem and Maximum Influence problem for cliques. Let $K_n = (V, E, t)$ be a clique with n nodes (see Figure 34), $V = \{1, 2, ..., n\}$ and $E = \{(i, j) : 1 \leq i < j \leq n\}$ and $t : t(v) \to \mathcal{Z}^+$. For the convenience of future analysis, we assume that the nodes in the clique are sorted according to their thresholds so that $t(i) \leq t(i + 1)$, for all $1 \leq i < n$.

7.1 Minimum Links problem

In this section, we study the Minimum Links problem in cliques. Let $MinLinks(K_n)$ be the problem of finding a minimum-sized link assignment that activates the entire clique, and let $ML(K_n)$ be the minimum number of links needed to activate the entire clique.

We observe that there are situations in which it is impossible to activate the entire clique with a single influencer. We now show a necessary and sufficient condition for the Minimum Links problem to have a feasible solution:

Proposition 5 *The Minimum Links problem has a feasible solution on $K_n = (V, E, t)$ if and only if $t(i) \leq i$ for all $1 \leq i \leq n$.*

Proof.

First, we prove the sufficient condition. If $t(i) \leq i$ for all $1 \leq i \leq n$, it is easy to see that there exists a solution S by giving a link to every node i. Since $t(1) \leq 1$, node 1 is activated in round 1. We claim that node i is activated in or before round i. Inductively, node 1 to $i-1$ are already activated in round $i-1$, the effective threshold of node i has been reduced to ≤ 1. Node i receives a link, therefore, node i must be activated in the i^{th} round, if it is not already activated.

Second, we prove the necessary condition. Suppose there exists a node p which is the first node for which $t(p) > p$. In order to activate any node q with $q \geq p$, at least $t(p) - 1 \geq p$ nodes have to be activated before q. However, there are only p-1 other nodes. It is impossible to activate such a node q, thus no node q with $q \geq p$ can be activated.

We come to the conclusion that every node in the clique should satisfy $t(i) \leq i$ for the MinLinks problem to have a solution. ■

We give the following greedy linear time algorithm (assuming that the nodes

are already sorted by threshold) for a clique which admits a feasible solution; the condition given in Proposition 5 can easily be checked in linear time. We examine the nodes in order. When we process node i, if $t(i) < i$, we simply increment i and continue; if $t(i) = i$, we give a link to node i, that is, we add i to the link set.

Algorithm 13 Minimum Links algorithm for cliques

```
1: Input: sorted list of nodes {1, 2, ... i, i + 1, ... n};
2: Output: count ← min links to activate the path; S ← an optimal link set
3: count ← 0
4: S ← ∅
5: index i ← 1
6: while i ≤ n do
7:     if t(i) > i then
8:         return no feasible solution
9:     if t(i) < i then
10:        i ← i + 1
11:    if t(i) == i then
12:        count ← count+1
13:        add node i to S
14:        i ← i + 1
15: return count and S
```

The **while** loop in line 6 repeatedly processes nodes in the order of their thresholds. This while loop maintains the following invariant: At the start of i^{th} iteration of the while loop, nodes in the range $[1, i-1]$ have been activated. We show this loop invariant as follows:

Initialization: By the time we process node 1, there are no active nodes, the loop invariant weakly holds.

Maintenance: In the i^{th} iteration of the while loop, since nodes in the range $[1, i-1]$ have been activated, thus the threshold of node i is effectively decreased by $i-1$. If $t(i) < i$, the effective threshold of i has been reduced to 0, so node i is activated automatically; if $t(i) = i$, the effective threshold of i has been reduced to

1, so by giving a link to node i, it is activated. Observe that in both cases, nodes in range $[1, i]$ have been activated after this iteration, as necessary for the maintenance of the loop invariant.

We now show that the link set produced by the greedy algorithm above is optimal. First we prove the greedy choice property by showing that there must be an optimal solution that contains the node 1. Consider an optimal solution S and let i be the smallest index of a node that receives a link in S. If $i = 1$, then we are done. If not, since there must always be a node with threshold 1 that receives a link, it must be that $t(i) = 1$. We can therefore move the link from i to 1, to create a new solution S' which will activate node i in the next step. Since $|S'| = |S|$ and $I(K_n, S) = I(K_n, S')$, S' is an optimal solution to the MinLinks problem that contains the node 1. Thus, we can assume that the optimal solution S contains the node 1. We now claim that $S - \{1\}$ is an optimal solution to the clique C' which is the induced sub-graph on the nodes $\{j, j+1, \ldots, n\}$ where $j > 1$ is the smallest index with $t(j) = j$, and with thresholds of all nodes reduced by $j - 1$. Suppose there is a smaller solution S' to C'. We claim that $S' \cup \{1\}$ activates all nodes in the clique K_n. As described above in the proof of the loop invariant, all nodes until node j are activated by the link given to node 1. Furthermore, the thresholds of all nodes in $\{j, j+1, \ldots, n\}$ are effectively reduced by $j - 1$. Thus using the links in S' suffices to activate them. Finally, since $|S'| < |S| - 1$, $S' \cup \{1\}$ is a smaller solution than S to the clique K_n, contradicting the optimality of S for K_n. This proves the optimal substructure property. We conclude that the greedy algorithm described above produces a minimum sized solution to the MinLinks problem.

We conclude with the following theorem:

Theorem 17 *The Minimum Links problem for a clique K_n can be solved in time $\theta(n)$.*

Proof. Algorithm 13 only requires a linear scan of the nodes in the clique, so it can be completed in $\theta(n)$ time. ■

7.2 Maximum Influence problem

In this section, we study the Maximum Influence problem in cliques. Let $MaxIn(K_n, k)$ be the problem of finding an assignment of at most k links to nodes in clique K_n that maximizes the number of influenced nodes, and Let $MI(K_n, k)$ be the maximum number of nodes that can be influenced in the clique using k links. We develop the following linear time greedy algorithm (assuming that the nodes are already sorted by threshold). We examine the nodes in order while we still have links to assign. When we process node i, if $t(i) > i$, we stop assigning links and break. If $t(i) < i$, similar to the MinLinks problem, we simply increment i and continue. Finally if $t(i) = i$, we give a link to node i.

Algorithm 14 Influence maximization algorithm for cliques

```
Input: (1) sorted list of nodes {1, 2, ... i, i + 1, ... n}; (2) k links available
Output: count ← maximum number of nodes that can be activated; S ← an optimal link set
count ← 0
index i ← 1
while i ≤ n do
    if t(i) > i || k == 0 then
        break
    if t(i) < i then
        count ← count+1
        i ← i + 1
    if t(i) == i then
        count ← count+1
        k ← k − 1
        i ← i + 1
        add node i to S
return count and S
```

First observe that if we reach a node i with $t(i) > i$, it is impossible to activate any node $q \geq i$, as all these nodes, even if they would be assigned a link, require at least $t(i) - 1 \geq i$ nodes to be activated before them, and there are only $i - 1$ other nodes. Therefore, the algorithm makes the correct decision to stop assigning links on reaching such a node. We now show that $MaxIn(K_n, k)$ problem exhibits the greedy-choice and optimal sub-structure properties.

The following lemma shows that the greedy-choice property holds.

Lemma 7 *There exists an optimal assignment S for problem $MaxIn(K_n, k)$ in which node 1 receives a link.*

Proof. Let S be an optimal solution for problem $MaxIn(K_n, k)$, and let node i be the node with smallest index in S. If $i = 1$, we are done, so assume $i > 1$. It must be that $t(i) = 1$, as otherwise it is impossible to activate any node, therefore, we can

create a new solution $S' = S \cup \{1\} - \{i\}$. By the activation of node 1, node i would be also activated automatically in the next round because of internal influence, thus $I(K_n, S) = I(K_n, S')$, and S' is an optimal solution that contains node 1 as needed. ■

We now prove that, the $MaxIn(K_n, k)$ problem exhibits the optimal sub-structure property. Let clique C' be the induced sub-graph on the nodes $\{j, j+1, \ldots, n\}$ where $j > 1$ is the smallest index with $t(j) = j$, and with thresholds of all nodes reduced by $j - 1$.

Lemma 8 *Suppose S is an optimal solution for problem $MaxIn(K_n, k)$ in which node 1 receives a link, then $S' = S - \{1\}$ is an optimal solution for problem $MaxIn(C', k - 1)$.*

Proof. Suppose there is a solution S'' to C' that activates more nodes than S'. Consider $S'' \cup \{1\}$ as a solution to K_n. The link to node 1 activates all nodes in $\{1, \ldots, j-1\}$, and the links in S'' now activate all the same nodes they activate in C', since these nodes now have the same effective thresholds they have in C'. Therefore, $S'' \cup \{1\}$ must activate more nodes than S in the clique K_n, contradicting the optimality of S. ■

We conclude with the following theorem:

Theorem 18 *The Maximum Influence problem for a Clique K_n can be solved in time $\theta(n)$.*

Proof. Algorithm 14 only requires a linear scan of the nodes in the clique, so it can be completed in $\theta(n)$ time. ■

Chapter 8

Conclusions and Future Work

This chapter summarizes the thesis and recommends some area for future work.

In this thesis, we proposed a new way to model the viral marketing strategy that allows for partial incentives and imposes a limit on the maximum amount of incentive that can be given to any single node. We model this by introducing external influencers to the network and allowing each external influencer to link to a subset of nodes in the network. In this thesis, we studied the single influencer case and proposed a pair of problems: the Minimum Links problem and the Maximum Influence problem. We proved that the Maximum Influence problem is NP-hard even for bipartite graphs in which thresholds of all nodes are either one or two. We proceeded to study both the Minimum Links problem and the Maximum Influence problem in paths, rings, trees and cliques, and gave polynomial time algorithms for both problems for all these graphs.

The first open problem directly related to this thesis is the complexity of the MinLinks problem for general graphs. The next area for future work is to propose an approximation algorithm with good performance for the Maximum Influence problem in general graphs.

We have systematically studied both the Minimum Links problem and the Maximum Influence problem in paths, rings, trees and cliques under the single influencer setting. We gave $\Theta(n)$ algorithms for the Minimum Links problem for paths, rings, cliques, and trees, and for the Maximum Influence problem on cliques. These algorithms are clearly optimal. However, it would be interesting to investigate if there are more efficient algorithms for the Maximum Influence problem on paths, rings and trees.

Our hardness proof implies the hardness of the Max-Influence problem for multiple influencers, but if there are m influencers and n links available, then what is the best strategy to assign links when the underlying network $G = (V, E)$ is a path, ring, tree or clique? For trees, Lemma 5 provides a starting point.

In this thesis, we assumed unweighted edges, that is, for each node, all its neighbors exert unit influence on it. It would be interesting to study the case where edges are weighted. Clearly, our hardness result for the Maximum influence problem still holds true for the more general case. But it would be interesting to investigate algorithms for the special topologies which we have studied for the case of weighted edges.

We have assumed in our thesis that once the threshold of a node i is reduced to 0, node i will be activated and remain active forever in the subsequent steps. But in reality, the activation process is more complicated. Once the threshold of a node i is reduced to 0, node i might only be active for a fixed amount of time λ and then become inactive for a period of time and might be activated again later. A new influence diffusion model is necessary if we want to study this phenomenon.

Bibliography

[1] Eytan Adar, Lada Adamic, et al. Tracking information epidemics in blogspace. In *Proceedings of IEEE/WIC/ACM international conference on Web Intelligence*, pages 207–214. IEEE, 2005.

[2] Shishir Bharathi, David Kempe, and Mahyar Salek. Competitive influence maximization in social networks. In *Proceedings of the 3rd International Conference on Internet and Network Economics*, WINE'07, pages 306–311, 2007.

[3] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. Maximizing social influence in nearly optimal time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, pages 946–957, 2014.

[4] Jacqueline Johnson Brown and Peter H Reingen. Social ties and word-of-mouth referral behavior. *Journal of Consumer research*, pages 350–362, 1987.

[5] Ning Chen. On the approximability of influence in social networks. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pages 1029–1037, 2008.

[6] Wei Chen, Alex Collins, Rachel Cummings, Te Ke, Zhenming Liu, David Rincon, Xiaorui Sun, Yajun Wang, Wei Wei, and Yifei Yuan. Influence maximization in social networks when negative opinions may emerge and propagate. In *SIAM*

International Conference on Data Mining, volume 11, pages 379–390. SIAM, 2011.

[7] Wei Chen, Yajun Wang, and Siyu Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 199–208, 2009.

[8] Wei Chen, Yifei Yuan, and Li Zhang. Scalable influence maximization in social networks under the linear threshold model. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 88–97, 2010.

[9] Ferdinando Cicalese, Gennaro Cordasco, Luisa Gargano, M. Milani, JosephG. Peters, and Ugo Vaccaro. How to go viral: Cheaply and quickly. In *Fun with Algorithms*, volume 8496 of *Lecture Notes in Computer Science*, pages 100–112. 2014.

[10] Ferdinando Cicalese, Gennaro Cordasco, Luisa Gargano, Martin Milani, and Ugo Vaccaro. Latency-bounded target set selection in social networks. In *The Nature of Computation. Logic, Algorithms, Applications*, volume 7921 of *Lecture Notes in Computer Science*, pages 65–77. 2013.

[11] Gerard Cornuejols, Marshall L Fisher, and George L Nemhauser. Location of Bank Accounts to Optimize Float: An Analytic Study of Exact and Approximate Algorithms. *Management science*, 23:789–810, 1977.

[12] Erik D. Demaine, Mohammad Taghi Hajiaghayi, Hamid Mahini, David L. Malec, S. Raghavan, Anshul Sawant, and Morteza Zadimoghadam. How to influence people with partial incentives. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 937–948, 2014.

[13] Pedro Domingos and Matt Richardson. Mining the network value of customers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 57–66, 2001.

[14] Richard Durrett. *Lecture Notes on Particle Systems and Percolation*. Wadsworth Publishing, 1988.

[15] Ayalvadi Ganesh, Laurent Massoulié, and Don Towsley. The effect of network topology on the spread of epidemics. In *24th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 1455–1466. IEEE, 2005.

[16] Luisa Gargano, Pavol Hell, Joseph Peters, and Ugo Vaccaro. Influence diffusion in social networks under time window constraints. In *Structural Information and Communication Complexity*, volume 8179 of *Lecture Notes in Computer Science*, pages 141–152. 2013.

[17] Jacob Goldenberg, Barak Libai, and Eitan Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing Letters*, 12:211–223, 2001.

[18] Jacob Goldenberg, Barak Libai, and Eitan Muller. Using complex systems analysis to advance marketing theory development: Modeling heterogeneity effects on new product growth through stochastic cellular automata. *Academy of Marketing Science Review*, 9:1–18, 2001.

[19] Amit Goyal, Francesco Bonchi, and Laks V.S. Lakshmanan. Learning influence probabilities in social networks. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 241–250. ACM, 2010.

[20] Amit Goyal, Francesco Bonchi, and Laks V.S. Lakshmanan. A data-based approach to social influence maximization. *Proceedings of the VLDB Endowment*, 5:73–84, 2011.

[21] Amit Goyal, Francesco Bonchi, Laks V.S. Lakshmanan, and Suresh Venkatasubramanian. On minimizing budget and time in influence propagation over social networks. *Social Network Analysis and Mining*, 3:179–192, 2013.

[22] Amit Goyal, Wei Lu, and Laks V.S. Lakshmanan. Celf++: Optimizing the greedy algorithm for influence maximization in social networks. In *Proceedings of the 20th International Conference Companion on World Wide Web*, WWW '11, pages 47–48, 2011.

[23] Mark Granovetter. Threshold models of collective behavior. *American journal of sociology*, pages 1420–1443, 1978.

[24] Dilek Gunnec. *Integrating social network effects in product design and diffusion*. PhD thesis, University of Maryland, College Park, 2012.

[25] Dilek Gunnec and S Raghavan. Integrating social network effects in the share-of-choice problem. Technical report, University of Maryland, College Park, 2012.

[26] Jing (Selena) He, Shouling Ji, Raheem Beyah, and Zhipeng Cai. Minimum-sized influential node set selection for social networks under the independent cascade model. In *Proceedings of the 15th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '14, pages 93–102, 2014.

[27] Matthew O Jackson and Leeat Yariv. Diffusion on social networks. *Economie publique/Public economics*, 16, 2006.

[28] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD*

International Conference on Knowledge Discovery and Data Mining, KDD '03, pages 137–146, 2003.

[29] David Kempe, Jon Kleinberg, and Éva Tardos. Influential nodes in a diffusion model for social networks. In *Proceedings of the 32rd International Conference on Automata, Languages and Programming*, ICALP'05, pages 1127–1138, 2005.

[30] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 591–600. ACM, 2010.

[31] Jure Leskovec, Lada A Adamic, and Bernardo A Huberman. The dynamics of viral marketing. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, pages 228–237. ACM, 2006.

[32] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 420–429, 2007.

[33] Thomas Liggett. *Interacting particle systems*. Springer, 1985.

[34] Elchanan Mossel and Sebastien Roch. Submodularity of influence in social networks: From local to global. *SIAM Journal on Computing*, 39:2176–2188, 2010.

[35] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14:265–294, 1978.

[36] David Peleg. Local majority voting, small coalitions and controlling monopolies in graphs: A review. In *Proceedings of 3rd Colloquium on Structural Information and Communication Complexity*, pages 152–169, 1997.

[37] Matthew Richardson and Pedro Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 61–70, 2002.

[38] Kazumi Saito, Ryohei Nakano, and Masahiro Kimura. Prediction of information diffusion probabilities for independent cascade model. In *Proceedings of the 12th International Conference on Knowledge-Based Intelligent Information and Engineering Systems, Part III*, KES '08, pages 67–75, 2008.

[39] Thomas C Schelling. *Micromotives and Macrobehavior*. New York: Norton, 1978.

[40] Eric Sun, Itamar Rosenn, Cameron Marlow, and Thomas M Lento. Gesundheit! modeling contagion through facebook news feed. In *Proceedings of International AAAI Conference on Weblogs and Social Media*, 2009.