Chapter 4
The Independent Cascade and Linear Threshold Models

4.1 Introduction

In Chaps. 2 and 3, we presented the SIR model and the tipping model respectively. In the former any node infected in last time step has a single chance (with probability β, a parameter of the model) to infect any of its neighbors which are not in a susceptible state. In the latter, an individual adopts a behavior if it has certain number of adopted incoming neighbors.

In this chapter, we focus on independent cascade (IC) model which is a generalized of SIR model, and two other models known as linear threshold (LT) and generalized threshold (GT) models, which are probabilistic extensions of the tipping model. These models are similar to the tipping dynamics of Chap. 3, except that the tipping threshold for each node is drawn at random.

In this chapter, we describe properties of these models and study problems of influence maximization and spread in this context. Finally, we present approaches to address the influence maximization problem to find the seed sets that maximize the number of adopters in expectation.

4.2 Model Definitions

We assume a social network $G = (V, E)$, where V is a set of vertices and E is a set of directed edges. For a given node $v \in V$, the set of incoming neighbors and outgoing neighbors are considered as $\eta^{in}(v)$ and $\eta^{out}(v)$ respectively. We will use the notation $|\cdot|$ for the cardinality of the sets.

The diffusion process occurs in discrete time steps t. If a node adopts a new behaviour or idea, it becomes active, otherwise it is inactive. An inactive node has the ability to become active. The set of active nodes at time t is considered as $\mathscr{X}_t$. The tendency of an inactive node v to become active is positively correlated with the number of active incoming neighbors v. Also, we assume that each node can only switch from inactive state to active state, and an active node will remain active for the rest of the diffusion process—hence these models are often referred to as "progressive" or "montonic." On the other hand, in non-progressive models active nodes can also switch back and become inactive—we will cover these in Chap. 6 when we describe evolutionary graph theory. In general, we start with an initial seed set $\mathscr{X}_0$ (when it is clear from context, we shall often drop the subscript and use $\mathscr{X}$ to denote the seed set), and through the diffusion process, for a given inactive node v, its active neighbors attempt to activate it. The process runs until no more activations occur.

4.2.1 Independent Cascade Model

Independent cascade (IC) model generalizes the SIR model described in Chap. 2. Instead of a single probability infection, there is a probability of infection associated with each edge. The probability $P_{u,v}$ is the probability of u infecting v. This probability can be assigned based on frequency of interactions, geographic proximity, or historical infection traces. Each node, once infected, has the ability to infect its neighbor in the next time step based on the probability associated with that edge.

Definition 4.1 (Independent Cascade Model (IC)). Under the Independent Cascade model dynamics, at each time step t where $\mathscr{X}_{t-1}^{new}$ is the set of *newly* activated nodes at time $t-1$, each $v \in \mathscr{X}_{t-1}^{new}$ infects the inactive neighbors $u \in \eta^{out}(v)$ with a probability $P_{u,v}$.

An example of this model is shown in Fig. 4.1. Active nodes are shown in yellow dotted line. At initial time, two nodes C and D are activated. At the next time step, node C and D has a chance to activate their three neighbors (A, G, and H) and (B, E, and F) respectively. According to Fig. 4.1b, only three nodes A, H, and E are successfully activated and the initial active nodes change to gray (denoting it stays active but no chance to activate others). In the next time step, two nodes G and F become active, and the previous active nodes A, E, and H change to gray. At time $t = 2$, two nodes F and G become active. Node G's neighbors are active, so it does not have a chance to activate any nodes. Node F has an option to activate node I; however it fails as shown in our given example in Fig. 4.1d. Since there is no more new active node, the diffusion process stopped.

4.2.2 Linear Threshold Model

The linear threshold model extends tipping model to its natural, weighted variant where each directed edge $(u,v) \in E$ has a non-negative weight $b(u,v)$. For any node $v \in V$, the total incoming edge weights sum to less than or equal to one, i.e. $\sum_{u \in \eta^{in}(v)} b(u,v) \leq 1$. The dynamics of the model are specified below.

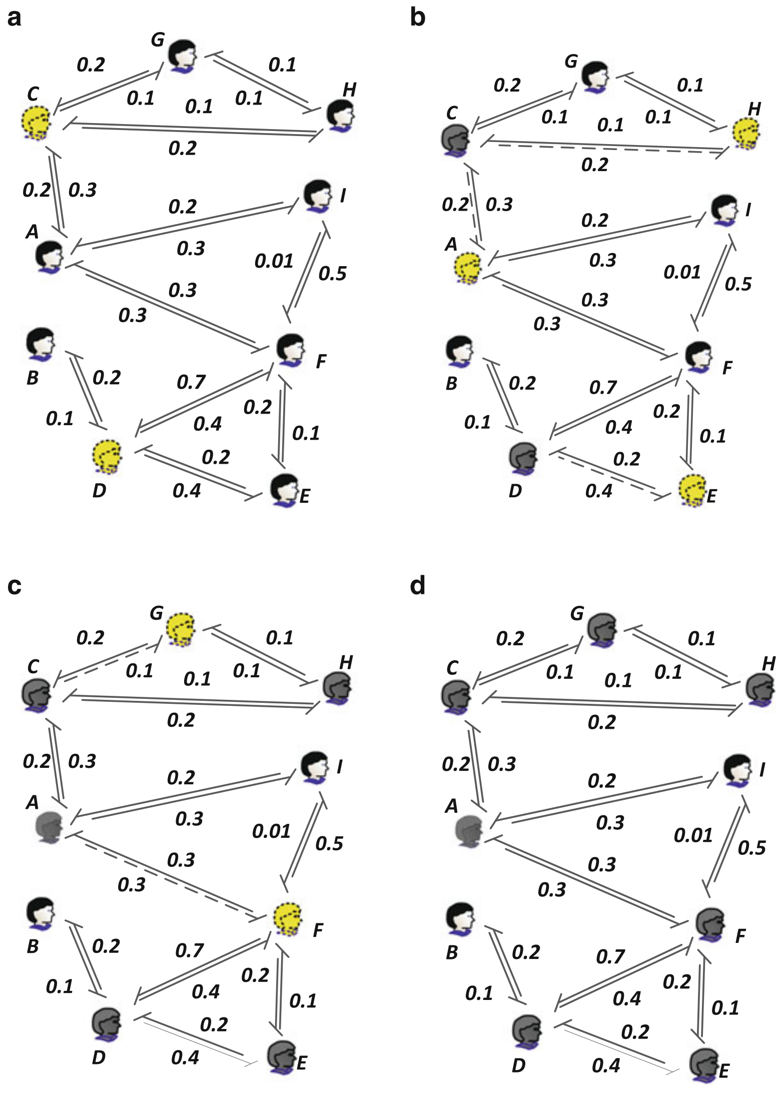

Fig. 4.1 Independent cascade model. Probability of transitions from each state to its successor state(in alphabetic order) are 7.0^{-3}, 6.0^{-3}, 5.0^{-1}, and 1.0, respectively. The probability of occurring for this sequence is 2.0^{-5}. (**a**) $t = 0$. (**b**) $t = 1$. (**c**) $t = 2$. (**d**) $t = 3$

Definition 4.2 (Linear Threshold Model (LT)). Under the linear threshold model dynamics, each node v selects a threshold θ_v in the interval [0,1] uniformly at random. Then, at each time step t where H_{t-1} is the set of nodes activated at time $t-1$ or earlier, each inactive node becomes active if $\sum_{u \in \eta^{in}(v) \cap H_{t-1}} b(u,v) \geq \theta_v$.

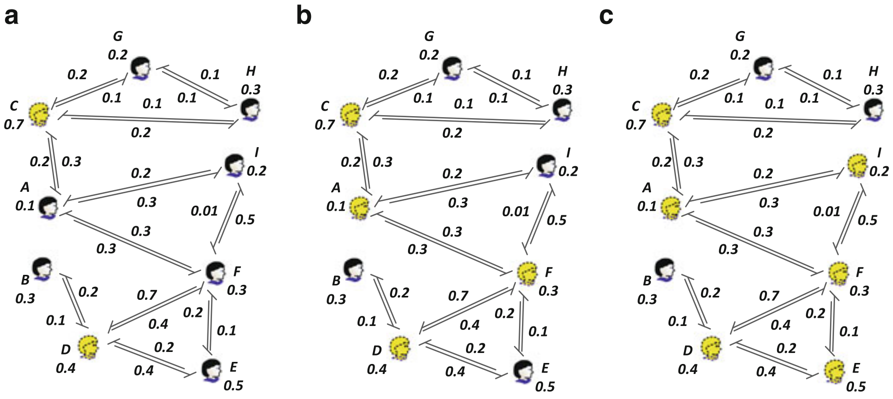

Fig. 4.2 Linear threshold model. The probability is determined based on the thresholds drawn at the first step, then the model proceeds deterministically.The probability of this sequence is almost 7.0^{-3}. (**a**) $t = 0$. (**b**) $t = 1$. (**c**) $t = 2$

Once the thresholds are drawn, these dynamics are equivalent to the tipping model of Chap. 3 where the activation function is re-written as follows:

$$A_\theta(V') = V' \cup \{v \in V s.t. \sum_{u \in \eta^{in}(v) \cap V'} b(u,v) \geq \theta_v\} \tag{4.1}$$

An inactive node is influenced by all of its active neighbors at each time step. An active node influences its inactive neighbors according to the weights. At each step, an inactive node becomes active if the total weight of its incoming neighbors is at least θ_v. Thresholds θ_v are selected randomly due to lack of knowledge of the tendency of nodes, and express the different levels of tendency of nodes to adopt an idea or innovation.

An example of this model is shown in Fig. 4.2. Each node is assigned a random threshold in [0,1] and two yellow dotted nodes C and D are initially activated. Node C is unable to activate two nodes G and H as its influence weight is not large enough, but it is able to activate node A ($0.2 \geq 0.1$). Node D also activates node F ($0.4 \geq 0.3$), but not B and E. In the next step (Fig. 4.2b), there are four active nodes and they are able to activate node I ($0.3 + 0.5 \geq 0.2$) and E ($0.4 + 0.2 \geq 0.5$). In the next time step, no new active node exists; so, the diffusion process terminates.

4.2.3 Generalized Threshold Model

Generalized threshold model is a broader framework of which the linear threshold and independent cascade models are special cases. Kempe et al. [1] also presented a generalized cascade model, which is equivalent to generalized threshold model. Therefore, we only provide the generalized threshold model definition here.

Definition 4.3 (Generalized Threshold Model (GT)). Given node v, a monotone threshold function $f_v : 2^{\eta^{in}(v)} \rightarrow [0,1]$, a threshold value $\theta_v \in [0,1]$, and an active set $\mathscr{X}_t$ at time t, node v is infected at time step t if $f_v(\mathscr{X}_t) \geq \theta_v$.

Again, the threshold value θ_v is uniformly randomly chosen for each node v. Linear threshold model is a special case of generalized threshold model, where the threshold function is in the form of summation over all active neighbors of node v, denoted $\mathscr{X}^v$, i.e. $f_v(\mathscr{X}^v) = \sum_{u \in \mathscr{X}^v} b(u,v)$ where $\sum_{u \in \eta_v^{in}} b(u,v) \leq 1$.

4.3 Influence Maximization Problem

In this section, we examine the influence maximization problem for the previously-mentioned models. First, we introduce some common terminology and concepts among all three models. The diffusion models involve an initial set of nodes to start. The *influence* of an initial set is defined as the number of active nodes at the end of the diffusion process. This is often referred to as *influence spread*.

Definition 4.4 (Influence Spread). Given an initial seed set $\mathscr{X}$, influence spread is the expected number of infectees $\sigma(\mathscr{X})$.

A natural optimization problem is to find the set of maximum influence nodes with a specific size k. That is, the initial k-node set $\mathscr{X}$ has been targeted to become active. We can formally define this problem as:

Definition 4.5 (Influence Maximization Problem). Given a natural number k, find an initial seed set $\mathscr{X}$, where $|\mathscr{X}| \leq k$, such that $\sigma(\mathscr{X})$ is maximized.

Unfortunately, this problem is NP-hard. This reduction is shown by an embedding of the max-k-cover problem.

In 1978, Nemhauser et al. [2] showed that (under the assumption that the influence spread can be efficiently calculated) a greedy algorithm provides an $(1-1/e)$ approximation if f meets normalization, monotonicity and submodularity conditions.

Definition 4.6 (Normalization). If there is no initial infectees then there is no spread, i.e. $f(\emptyset) = 0$.

Definition 4.7 (Montonicity). For $S \subseteq S'$, $f(S) \leq f(S')$.

Definition 4.8 (Submodularity). An arbitrary set function $f : S \rightarrow \mathbb{R}$ is submodular if and only if for all $S', S'' \subseteq S$, it is the case that if $S' \subseteq S''$, then $f(S' \cup \{s\}) - f(S') \geq f(S'' \cup \{s\}) - f(S'')$. Intuitively, a submodular function has diminishing returns property.

The intuition behind submodularity can be explained with the following example. Suppose you have a poor man with very few possessions (H_1) and a rich man with many more possessions (H_2). Suppose neither possesses a Ferrari car (h). Giving the poor man the Ferrari would make a greater difference to his net worth (computed via f as a function of the person's possessions) than giving it to the rich man.

4.3.1 Influence Maximization Under the IC Model

Kempe et al. [1] show that influence maximization problem can be viewed as a general case of an NP-complete Set Cover problem under the IC model.

Theorem 4.1 (Complexity of Influence Maximization in the IC Model). *The influence maximization in the independent cascade model is NP-hard within a factor of* $1-1/e+\varepsilon$ *for any* $\varepsilon > 0$.

They [1] also show that σ meets the requirements (mentioned under Definition 4.6, 4.7 and 4.8) under the IC model.

Theorem 4.2. *In the independent cascade model, the influence function* $\sigma_{IC}(\cdot)$ *is normalized, monotone, and submodular.*

We will give an intuition for this finding under IC dynamics. The influence function in IC is normalized, since if there is no initial infectees then there is no spread, i.e. $\sigma_{IC}(\emptyset) = 0$. It is also monotone, because each element of $\mathscr{X}$ contributes at least one expected infectee. However, this is not true if we do not count nodes in set $\mathscr{X}$ toward the expected infectees. In this case, we can view the oracle function as $\sigma_{IC}(\mathscr{X}) - |\mathscr{X}|$, which is not monotonic. This is a special case of *profit maximization* [6, 7].

We can prove submodularity of the influence function in IC model using live-edge model. The proof of submodularity relies on a manipulation of the "live edge" model—a mathematically equivalent representation of the IC model (and several others as well). This technique is commonly used in much of the literature relating to the models described in this chapter. We outline the technique used to obtain this result next.

First, we define an *outcome* under the IC models as a subgraph of G. A given outcome intuitively is one possible way the IC process can occur. Using this idea, we can assign a probability to an outcome as follows. For outcome $G' = (V, E')$ the probability of the outcome is: $\Pi_{(u,v)\in E'} P_{u,v} \times \Pi_{(u,v)\in E\setminus E'}(1-Pu,v)$. For a given outcome $\mathscr{W}$, we denote this probability $Pr(\mathscr{W})$.

Definition 4.9 (Live-Edge Path). Given graph G, and an initial active set $\mathscr{X}$, the path from $\mathscr{X}$ to other nodes is called a live-edge path.

According to the definition, given graph G with edge probabilities, we can view some subgraph G' as a potential set of available edges that diffusion can deterministically occur.

Definition 4.10 (Live-Edge Model). Given graph $G = (V, E)$, seed set $\mathscr{X}$, and probability distribution Pr over subgraphs of G, the probability a node $v \in V \setminus \mathscr{X}$ is defined as the sum of the probabilities of subgraphs of G (based on Pr) where exists a path from a node in $\mathscr{X}$ to v is active.

As, for a given subgraph G', the infection is deterministic, a node x is infected under the live-edge model if there is simply a path from a node in $\mathscr{X}$ to x. The edges successfully activate are declared as *live*, and the rest of the edges are declared as *blocked*. Kempe et al. show that live-edge model is equivalent to IC model.

Algorithm 2 Greedy Algorithm

```
1: procedure GreedyApproximation($V$, $k$)
2:     Set $\mathscr{X} = \emptyset$
3:     while $|\mathscr{X}| \leq k$ do
4:         Pick $s \in V$ where $\sigma(\mathscr{X} \cup s) - \sigma(\mathscr{X})$ is the greatest
5:         Add $s$ to $\mathscr{X}$
6:     end while
7:     return $\mathscr{X}$
8: end procedure
```

Consider two sets S and S' such that $S \subseteq S'$ and an element s which is in neither. For random weight of edges $\mathscr{W}$, Let $R_{\mathscr{W}}(S)$, $R_{\mathscr{W}}(S')$, and $R_{\mathscr{W}}(\{s\})$ be the sets of nodes reachable by S, S', and s by weights $\mathscr{W}$, respectively. Since $|R_{\mathscr{W}}(S') \cap R_{\mathscr{W}}(\{s\})| \geq |R_{\mathscr{W}}(S) \cap R_{\mathscr{W}}(\{s\})|$; thus, $\sigma_{\mathscr{W}}(S \cup \{s\}) - \sigma_{\mathscr{W}}(S) \geq \sigma_{\mathscr{W}}(S' \cup \{s\}) - \sigma_{\mathscr{W}}(S')$. For any $\mathscr{W}$, we have the following relationship: $\sigma_{IC}(S) = \sum_{\mathscr{W}} Pr(\mathscr{W}) \times \sigma_{\mathscr{W}}(S)$ As positive linear combinations of submodular functions are also submodular, we have completed the proof.

Greedy Approximation Algorithm To find the initial seed set we can use *Greedy Algorithm* as presented in Algorithm 2. In each iteration, the element with maximum marginal influence is added to the seed set. Let k be the size of the seed set, there are k and $|V|$ iterations of the outer loop and inner loop respectively, where each of these $k \times |V|$ iterations require many evaluations of σ. Thus, this becomes very expensive. And note, this assumes that the computation of σ can be done efficiently. However, the straight-forward method is to rely on simulation, which is also expensive.

Let us consider the issue of the calculation of σ. It turns out, that by leveraging the live-edge model, that calculation influence is #P-hard by a reduction from the counting version of s-t connectivity (this result was originally proven by Chen et al. in [5]). The problem of s-t connectivity deals with determining if their exist a path between two nodes (denoted s and t) in a graph. Chen's reduction works by creating an instance of the live-edge model where the probability of t adopting given a seed set consisting of node s is proportional to the number of paths between these two nodes.

Theorem 4.3 (Complexity of Influence Spread in the IC Model). *In the independent cascade model, influence spread $\sigma_{IC}(\cdot)$ is #P-hard.*

Practically, we can obtain arbitrarily close approximation using simulation. However, this is expensive and at the time of this writing, does not provide any formal guarantee.

However, the expected number of infectees is solvable in polynomial time for directed acyclic graphs [5]. We can compute the activation probability, the probability of a node u is infected given seed set $\mathscr{X}$, of each node using Algorithm 3. Though this seems to be a restrictive case, this intuition is useful in building a heuristic approach to this problem—as we shall describe later in the chapter.

Algorithm 3 Computing Activation Probability in DAG

```
1: procedure ActivationProbabilty($D$, $\mathscr{X}$)
2:     $\forall u \in$ DAG $D, ap(u) = 0$
3:     $\forall u \in$ seed set $\mathscr{X}, ap(u) = 1$
4:     Topologically sort all nodes reachable from $\mathscr{X}$ in $D$ into a sequence $\rho$, with in-degree zero
       nodes sorted first.
5:     for $u \in \{\rho \setminus \mathscr{X}\}$ according to the order $\rho$ do
6:         $ap(u) = \sum_{x \in \eta^{in}(u) \cap \rho} ap(x) \times b(x,u)$
7:     end for
8:     return $S$
9: end procedure
```

4.3.2 Influence Maximization Under the LT Model

We now turn our attention to influence maximization and influence spread under the LT model. It turns out that influence spread calculation is also #*P*-hard [4]. This time the proof is shown by a reduction from the problem of counting the number of simple paths between nodes—again using the live-edge model.

Theorem 4.4 (Complexity of Influence Spread in the LT Model). *In the linear threshold model, influence spread $\sigma_{LT}(\cdot)$ is #P-hard.*

Further, even if influence spread can be computed efficiently, solving the influence maximization problem under this model is NP-hard by an embedding of the vertex cover problem [1].

Theorem 4.5 (Complexity of Influence Maximization in the LT Model). *The influence maximization in the linear threshold model is NP-hard.*

However, with respect to the optimization problem associated with influence maximization under LT, the same properties hold as with the IC model. Intuitively, the LT model is normalized (no seed, no diffusion). Monotonicity of the LT model follows the same argument showed for the IC model. Kempe et al. leverage the live-edge model and show that the model is submodular.

Theorem 4.6. *In the linear threshold model, the influence function $\sigma_{LT}(\cdot)$ is normalized, monotone, and submodular.*

4.3.3 Influence Maximization Under the GT Model

In [1] the generalized threshold model is shown to capture both the IC and LT models, hence the hardness results for both influence maximization and influence spread still hold.

Theorem 4.7 (Complexity of Influence Maximization in the GT Model). *The influence maximization in the generalized threshold model is NP-hard.*

Theorem 4.8 (Complexity of Influence Spread in the GT Model). *In the generalized threshold model, computing influence function $\sigma_{GT}(\cdot)$ is #P-hard.*

The influence maximization problem in GT model can be reduced to max-k-cover problem. It is NP-hard within a factor of $|V|^{1-\varepsilon}$ [1]. However, the result of Mossel and Roch [3] relate the local activation functions to influence spread, providing for an elegant result that allows the greedy algorithm to obtain an approximation guarantee in this large special case.

Theorem 4.9. *If all the activation functions are normalized, monotonic, and submodular, then the expected number of infectees under the generalized threshold model is also normalized, monotonic, and submodular [3].*

4.4 Scaling Influence Maximization

Due to the demands of potential real-world applications, Influence maximization problem should be scalable for the real world—as these are often of size 10^5 nodes or greater (i.e. see the datasets described in Chaps. 2 and 3). As we said in the previous section (Algorithm 2), Greedy algorithm is computationally expensive since in each iteration, it iterates through all the nodes in the given network, and we run the simulation multiple times to get a closer approximation of the diffusion outcome. In this section, we present two algorithms and one model to find the seed set with maximum influence for the LT and IC models.

4.4.1 Lazy Greedy Approximation

We can accelerate greedy algorithm and reduce its computational complexity under the certain assumption. In 1978, Minoux [8] showed that if a given function f is submodular, we can optimally accelerate greedy algorithms which is confirmed theoretically.

The key intuition behind "Lazy Greedy" algorithm is that by the definition of submodularity, the incremental increase to σ afforded by a node is always bounded above by its incremental increase on previous iterations. By checking this, we can avoid unnecessary calculation.

Consider the ith iteration of the Greedy Algorithm 2. Let $\mathscr{X}_i$ be the set of elements picked at the end of iteration $i-1$. For node s, the algorithm evaluates the following quantity:

$$\lambda(i,s) = \sigma(\mathscr{X}_i \cup s) - \sigma(\mathscr{X}_i) \tag{4.2}$$

Let us call $\lambda(i,s)$ the incremental increase afforded to s at iteration i. Now, there are many nodes evaluated at a given iteration. Let us assume that s does not do very well—in fact it is somewhere in the middle of the pack. Now consider the $(i+1)^{th}$ iteration of the Greedy Algorithm. Let $\mathscr{X}_{i+1}$ be the set of elements picked at the end of iteration i. For node s, the algorithm evaluates the following quantity:

$$\lambda(i+1,s) = \sigma(\mathscr{X}_{i+1} \cup s) - \sigma(\mathscr{X}_{i+1}) \tag{4.3}$$

This value is incremental increase afforded to s at iteration $i+1$. Let us assume that s again does not do very well. We note that $\mathscr{X}_{i+1}$ is a superset of S_i. Hence, as $\sigma(\cdot)$ is submodular, we get the following:

$$\sigma(\mathscr{X}_{i+1} \cup s) - \sigma(\mathscr{X}_{i+1}) \leq \sigma(\mathscr{X}_i \cup s) - \sigma(\mathscr{X}_i) \tag{4.4}$$

$$\lambda(i+1,s) \leq \lambda(i,s) \tag{4.5}$$

So, suppose at iteration i we saved $\lambda(i,s)$ in some data structure. Suppose we start evaluating $\lambda(i+1,s)$ at the start of iteration $i+1$, and let s' be the node where currently $\lambda(i+1,s')$ is the greatest. So, now instead of evaluating $\lambda(i+1,s)$ directly, we perform the following steps:

1. If $\lambda(i+1,s') \leq \lambda(i,s)$:
   a. Evaluate $\lambda(i+1,s)$
   b. If $\lambda(i+1,s') \leq \lambda(i+1,s)$, then s is the node that affords the greatest incremental increase
2. Otherwise:
   a. Go to the next node

This can produce significant speedup in practice. Performance may vary depending on the ordering of the nodes and ordering the nodes may increase algorithm runtime. So, worst-case time complexity does not change. This algorithm also needs to iterate through all nodes during the first iteration of the outer loop.

To avoid costly simulation runs, we turn to the issue of scaling the computation of σ. In general, the approaches presented in the literature for scaling σ is tied to the underlying model, i.e. IC vs. LT. Hence, we shall describe a method for IC and a method for LT.

4.4.2 Maximum Influence Arborescence (MIA) Model

Computing the influence spread in the IC model is # P-hard, yet computable in polynomial time for directed trees. Chen et al. [5] introduce the maximum influence arborescence (MIA) model where the probability that node v' infects node v is based on the probability of v' infecting v only by the most influential path, called the *maximum influence path*. An arborescence is a directed tree with a root node v and for any other node v' there is exactly one directed path from v' to v.

For a given pair of nodes, $(u,v) \in E$, the MIA model is defined as the path between two nodes whose probability is greatest, denoted $MIP(u,v)$. If there is no path, then $MIP(u,v) = \emptyset$. This is uniquely and consistently determined for each node pair—hence ties are assumed to be broken in a consistent manner. If we create an alternative graph where each edge (u,v) is weighted by $\log(P(u,v)^{-1})$ then we can easily find the MIP's using Dijkstra's algorithm.

For a given node v, and a threshold $\theta \in [0,1]$, we define its *maximum influence in-arborescence* (MIIA)—the arborescence created by the union of all maximum influence paths starting from each other node to v whose probability exceeds threshold θ as follows:

$$MIIA(v,\theta) = \cup_{u \in V, Pr(MIP(u,v)) \geq \theta} MIP(u,v) \tag{4.6}$$

This is the graph created by the union of all MIP's from other nodes to v whose probability is at least θ. By keeping the MIP unique and consistently defined, we know the resulting graph is an arborescence. Note that this can be computed before any algorithmic attempt to solve the maximum influence problem.

Given a seed set $\mathscr{X}$, node u and an arborescence A, the activation probability $ap(u,\mathscr{X},A)$ is the probability u is infected given seed set $\mathscr{X}$ in graph A under the IC model. There are three cases: 1) If u is in $\mathscr{X}$, then $ap(u,\mathscr{X},A) = 1$, 2) If u is not in $\mathscr{X}$ and has no incoming neighbors, then $ap(u,\mathscr{X},A) = 0$, 3) If the first two cases do not hold, then:

$$ap(u,\mathscr{X},A) = 1 - \mathop{\Pi}_{w \in \eta^{in}(u)} 1 - (ap(w,\mathscr{X},A)) \times P(w,v)) \tag{4.7}$$

This can be computed in polynomial time by a single traversal of A by considering nodes from the leaves to the root.

Let the activation of node v given seed set $\mathscr{X}$ and its MIIA $ap(u,\mathscr{X},MIIA(u,\theta))$, the expected number of infectees in the MIA model, denoted σ_M, given seed set $\mathscr{X}$ is computed as follows:

$$\sigma_M(\mathscr{X}) = \sum_{v \in V} ap(u,\mathscr{X},MIIA(u,\theta)) \tag{4.8}$$

Note that the graph used to calculate the activation probability for each node can be different. As the activation probabilities can be computed in polynomial time, so can the expected number of infectees.

As we said, the goal is to replace the IC model with the MIA model in the algorithm in the hope that the seed set returned provides a large number of infectees (in expectation) under the IC model. Influence maximization problem under the MIA model is NP-hard. However, this problem becomes easier than influence maximization under the IC model, which, due to the difficulty of the influence spread problem, is actually harder [9] showed that this problem is actually #P hard. This NP-completeness result is due to the fact that influence spread under this model is computable in PTIME (see Algorithm 2). This also allows us to avoid costly simulation runs. Further, the MIA model is normalized, monotonic and submodular. So, the greedy algorithm can be applied and achieve the $1 - 1/e$ approximation ratio with respect to the expected number of infectees under the MIA model. The greedy algorithm for the MIA model possess no (known) theoretical properties with respect to the IC model though a close relationship has been shown experimentally. However, we note that the results with respect to the MIA model hold if we use lazy evaluation.

Chen et al. [5] demonstrate speedups to improve incremental increase specific to the MIA model. Suppose we are considering node w on the ith iteration of the greedy algorithm. When we compute the activation probability of some node v, the shortest path from w to v has a node previously picked. This causes an incremental increase of node w to be zero with regards to v. So, the intuition is at each iteration, re-compute the in-arborescence for each node such that no paths from other nodes outside of $\mathscr{X}$ contain a node in $\mathscr{X}$.

Chen et al. create a prefix-excluding variant of the MIA model, called PMIA. They defined a special version of MIIA, called PMIIA, that takes the current seed set into account. Then they identify extensions to monotonicity and submodularity and show the PMIA model has these properties. The greedy algorithm also achieves the $1 - 1/e$ approximation under a new propriety identified by Chen et al. called "sequence submodularity". The runtime has several order-of-magnitude improvement as well. It also outperforms centrality-based heuristics on large datasets (millions of edges) [5].

4.4.3 SIMPATH Algorithm

According to the hardness of influence spread under the LT model, Goyal et al. [4] leverage heuristic algorithm to tackle this problem. The key intuition behind SIMPATH algorithm is to enumerate the simple paths from the seed set instead of running costly simulations. A *simple path* is a directed path in a graph where no nodes are repeated. Let $\Upsilon_{u,v}$ be the probability that v is infected by u. If P_{uv} be the set of simple paths between u and v, then we have the following relationship:

$$\Upsilon_{u,v} = \sum_{P \in P_{uv}} Pr(P) \tag{4.9}$$

Where Pr is computed as: for some subset S, $\mathscr{X}$ of V, let $\sigma^S(\mathscr{X})$ be the expected number of infectees in the LT model given seed set $\mathscr{X}$ on the subgraph induced by the set of nodes in S. So, the expected number of infectees is:

$$\sigma(\mathscr{X}) = \sum_{u \in \mathscr{X}} \sigma^{V-\mathscr{X}+\{u\}}(u) \tag{4.10}$$

Algorithm 4 SIMPATH Spread Algorithm

```
1: procedure SIMPATHSPREAD($\mathscr{X}$)
2:     $\sigma = 0$
3:     for $u \in \mathscr{X}$ do
4:         Compute all simple paths from $u$ to all other nodes in the graph $G(V - \mathscr{X} + u)$
5:         Compute the probability for all the paths, let this equal $T$
6:         $\sigma = \sigma + T$
7:     end for
8:     return $\sigma$
9: end procedure
```

According to the Eq. (4.9), the expected spread for a singleton is the sum of the probability of each node being influenced as follows:

$$\sigma(\{u\}) = \sum_{v \in V} \Upsilon_{u,v} \tag{4.11}$$

So, we can compute the expected number of infectees as follows:

$$\sigma(\mathscr{X}) = \sum_{u \in \mathscr{X}} \sum_{v \in V - \mathscr{X} + \{u\}} \sum_{P \in P_{uv} \ on \ G(V - \mathscr{X} + u)} Pr(P) \tag{4.12}$$

Let us suppose we have an oracle that enumerates all simple paths from some node u to all other nodes within a graph. Goyal et al. use the backtrack algorithm to achieve this. We can compute σ exactly as Algorithm 4.

As we said, computing the expected number of infectees in the LT model is #P hard because just counting the number of simple paths between nodes is #P hard. So, Algorithm 4 is not efficient—as it computes σ exactly. As the longer paths will occur with a much lower probability (the influence events are independent), a heuristic is only enumerating the paths that have a probability greater than a certain threshold. Hence, SIMPATH computes a lower bound on the expected number of infectees—the hope is that the paths it does not consider, do not add up to a whole lot which, would make the lower bound tight. This is not an approximation guarantee—there are currently no known results regarding how close this bound is to the actual expected value. However, the threshold can be used to trade runtime for accuracy.

The overall algorithm for scalable influence maximization under the LT model is called SIMPATH and calls SIMPATH-Spread to approximate σ. Goyel et al. leverage lazy submodular evaluation as well as some graph-theoretic techniques to limit the calls to SIMPATH-Spread. However, one major issue is to limit the number of calls to SIMPATH-Spread on the first iteration of the greedy algorithm. SIMPATH shortens the runtime of the first iteration by computing SIMPATH Spread for nodes within a vertex cover.

SIMPATH performed comparable to the greedy algorithm and outperform degree centrality, PageRank, and LDAG [5] in terms of expected number of infectees and provides $3 - 4x$ order of magnitude improvement over the greedy in terms of runtime.

4.5 Conclusion

In this chapter we reviewed the popular independent cascade and linear threshold models as well as their associated influence maximization problems. We also described various algorithmic approaches to these problems and the current state-of-the-art techniques for achieving scalability. However, as with the tipping model and SIR model of the previous chapters, this framework does not take the attributes of the nodes and edges into account during the diffusion process. In the next chapter, we describe a logic-programming based framework that includes this dimension.

References

1. Kempe, David and Kleinberg, Jon and Tardos. Maximizing the spread of influence through a social network. (2003) ACM 137–146.
2. Nemhauser, George L and Wolsey, Laurence A and Fisher, Marshall L. An analysis of approximations for maximizing submodular set functions. (1978) 14(1) 265–294.
3. Mossel, Elchanan and Roch, Sebastien. On the submodularity of influence in social networks. (2007) ACM 128–134.
4. Goyal, Amit and Lu, Wei and Lakshmanan, Laks VS. Simpath: An efficient algorithm for influence maximization under the linear threshold model.(2011) Data Mining (ICDM), 2011 IEEE 11th International Conference, 211–220.
5. Chen, Wei and Wang, Chi and Wang, Yajun. Scalable influence maximization for prevalent viral marketing in large-scale social networks. (2010) Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, 1029–1038.
6. Shakarian, Paulo and Salmento, Joseph and Pulleyblank, William and Bertetto, John. Reducing gang violence through network influence based targeting of social programs. (2014) 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 1829–1836
7. Lu, Wei and Lakshmanan, Laks VS. Profit maximization over social networks. (2012) arXiv preprint arXiv:1210.4211
8. Minoux, Michel. Accelerated greedy algorithms for maximizing submodular set functions. (1978) , Optimization Techniques 234–243.
9. Wang, Chi and Chen, Wei and Wang, Yajun. Scalable influence maximization for independent cascade model in large-scale social networks (2012) Data Mining and Knowledge Discovery. 545–576.