UNIVERSITÉ CATHOLIQUE DE LOUVAIN

MASTER THESIS

Influence Spread Maximization and its Applications

Authors
Anthony Collignon
François Montoisy

Supervisor: Marco Saerens

A thesis submitted in fulfillment of the requirements for the degree of Business engineering Master

at the

Louvain School of Management

August 8, 2018

"Tim Draper persuaded Hotmail to include a promotional pitch with a clickable URL in every message sent by a Hotmail user. Therein lay one of the critical elements of viral marketing: every customer becomes an involuntary salesperson simply by using the product."

Steve Jurvetson [Jurvetson, 2000]

Université Catholique de Louvain

Abstract

Louvain School of Management

Master

Influence Spread Maximization and its Applications

by Anthony COLLIGNON & François MONTOISY

The present master thesis firstly describes the *spread maximization problem* and its computational complexity. Due to this hardness, the best seed set of nodes cannot be found directly, especially for huge networks. Hence heuristics have to be created and used. The goal of our thesis is to compare the performances (in terms of effectiveness and efficiency) of different heuristics on finding the best seed set of nodes in order to maximize the final influence spread on real social networks. Based on our findings, further applications are discussed at the end of this paper.

Acknowledgements

First and foremost, we would like to thank Professor Marco Saerens, our promotor, for giving us the opportunity to further extend the topic developed by Mr. Corentin Vande Kerkhove with his *Killed Random Walk* method to marketing applications. We are also thankful for his help in refining our thesis subject and more generally for his assistance and time throughout the whole process.

We would like to thank people from whom we received support in the writing. Thereupon, we particularly would like to thank Mr. Jean Herdies for his good advice concerning the theoretical part of our thesis and Mr. Sylvain Courtain for his insightful remarks.

Last, but definitely not least, we would like to thank every people, parent, sibling and friend, who helped us in one way or another, during the whole journey and especially in the hardest moments.

Contents

List of Figures

List of Symbols

$G(V, E)$	Social network represented by a graph
V	Set of users represented by vertices or nodes
E	Relationships represented by arcs or edges
(i, j)	Arc (edge) from node i toward j
$N^{in}(u)$	Set of u's in-neighbors
$N^{out}(u)$	Set of u's out-neighbors
S_t	Set of active nodes at time t
S_0	Seed set, the initial set of nodes selected
$\sigma(S_0)$	Influence spread/coverage (active nodes when propagation ends)
$p_{uv} \in [0, 1]$	Influence probability (IC model)
$w_{uv} \in [0, 1]$	Influence weight (LT Model)
θ_u	Threshold above which individuals becomes active (LT model)
f_0	Initial {0,1} assignment of the vertices of G
C_v	Cost of setting $f_0(v) = 1$
B	Total Budget of advertising
A	Adjacency matrix of G
T	Transition matrix of G
$k = \|S_0\|$	Size of the seed set
$n = \|V\|$	Number of nodes in graph G
$f(u\|S)$	Marginal gain of element u to set S
Q	Queue in the LazyGreedy algorithm
$u.mg$	Latest computed marginal gain of u
$u.i$	Iteration in which the marginal gain is last updated
$MIP(u, v)$	Path with maximum influence probability between nodes u and v
$MIIA(u, \theta)$	Set of $MIP's$ starting from node u, that exceeds a threshold θ
$ap(u)$	Probability that node u is activated after a diffusion process
$InfSet(u)$	Set of nodes whose in-arborescence contains u
$IncInf(u)$	Marginal influence spread of node u
$IncInf(u, v)$	Marginal contribution of node u towards the activation probability of node u.
$Pred()$	Vector of predecessors in MIA model (Dijkstra algorithm)

$D(u)$	Minimal distance vector from vertex u to all other nodes.
$\varphi(S_0)$	Profit of the seed set S_0
L_t	Sublattice of $[\emptyset, V]$ at time t
p_u	Probability that user u accepts an offer (Scheduled seeding)
p_{max}	Maximum rate of acceptance of a product
$p_{soc}(u)$	Dependency of u to his social network
$\|N_u^+\|$	# infected and infectious in-neighbors of u (Scheduled seeding).
$BC(u)$	Betweenness centrality of node u
$PR(u)$	PageRank score of node u
d	damping factor in PageRank model
$d(u, v)$	Shortest path between u and v computed by Dijkstra algorithm
ς_{uv}	Number of shortest paths between u and v
$\varsigma_{uv}(w)$	Number of shortest paths between u and v that contains w
$Entropy_u$	Entropy induced by node u in the KRW model
N_u	# crossings induced by node u in KRW
n_{uv}	# node v crossings induced by node u in KRW
T_{Kill}	Transition matrix with jump probability ≤ 1
λ	adjustment parameter of the KRW model
N	matrix of crossings in KRW
$N(i\|S_0)$	Conditional Average Crossings in IKRW model
$Entropy(i\|S_0)$	Conditional Entropy in IKRW model

Chapter 1

Introduction

1.1 Information propagation is more and more ubiquitous

The information cascades take every day more importance on the web and in particular on social networks. While surfing on the news feed of Facebook, how many times do you see a "status" first emitted by a person you do not know, which is passed on to you thanks to a friend of yours who comments to her status' update? On Twitter, how many followers are replying to pop stars "tweets" or are retweeting them?

As you can see, this information propagation is more and more ubiquitous in our today's life. As you can imagine, companies see this as an opportunity to increase in a cheap way their advertising. One of the challenges they face is "What client to tackle first (through advertising, free samples etc.) in order to reach the largest final target at the end of the propagation?". Besides these computations, marketers have to understand how to turn information cascade into commercial success.

Tackling those challenges is the objective of the present work; let us first delimit the problem in order to narrow our focus.

1.2 Introduction to graph theory

Before going into the details of the question tackled in this thesis, some terminologies taken from the *graph theory* and used throughout this paper must be highlighted (Bollobas, 1998).

- A *graph*, or network G can be represented by two main components, a set of *nodes* or *vertices*, denoted by V, and their corresponding *edges*

linking those vertices together, E. Hence, a graph is the combination of nodes and edge, i.e. $G(V, E)$. The reader can look at figure 1.1 in order to have a better idea of what a graph looks like.

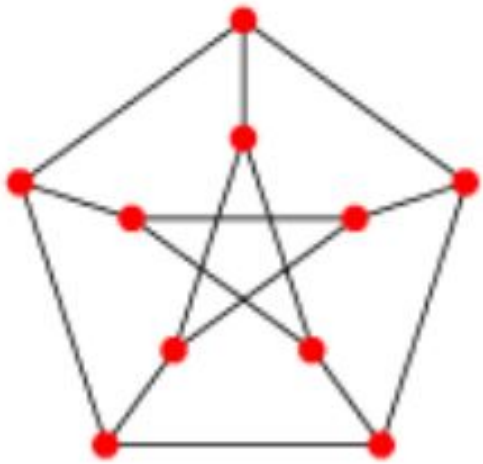


FIGURE 1.1: Example of a graph. This particular one is called *Petersen* graph (Weisstein, 2018)

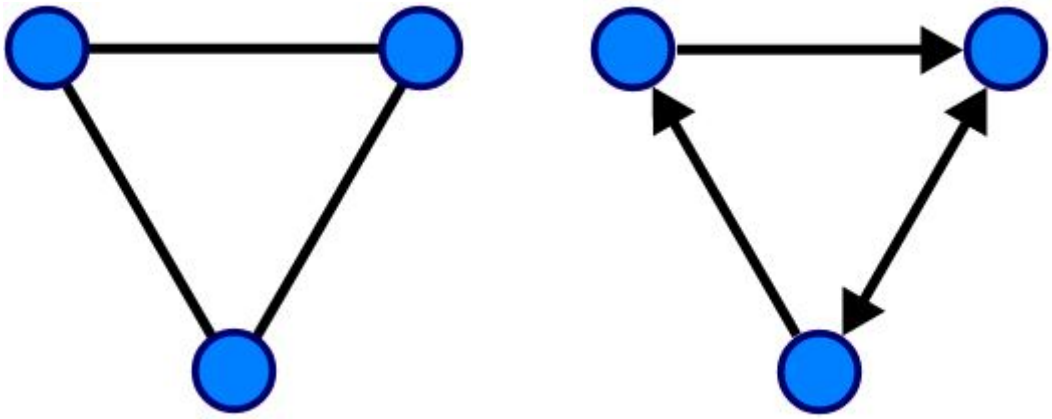
- A *vertex* or *node* is an element of a graph linked to a certain number of other vertices. Each node has two possible states, it is either *active* or *inactive*. A person is said to be active when he knows about a product or an idea, while an inactive user represents a person who does not.

- $S_t \subseteq V$ will be referred as the *active set* at time t. The set of users S_0, known as the *seed set*, represents the ones initially selected to spread the information throughout the whole network. Those targets can for example receive new samples of the product for free, or can be reached by a promotion campaign. The final active set for a certain initial seed set S_0 is denoted by $\sigma(S_0)$.

- An *edge* is a link between two nodes. Influence between vertices can only pass through those edges. Throughout this thesis, the notation used for describing an edge starting in node u and stopping in v will be denoted by (u, v).

- Graphs can be differentiated according to the type of edges existing between nodes. Facebook, for example use two-sided friendship links, that is, if user u is the friend of v, it means v is also u's friend: $(u, v) = (v, u)$. This type of graph with similar mutual influence between pairs of nodes is called *undirected*. On the contrary, social networks such as Twitter allow you to follow a user, without him to follow you. This

FIGURE 1.2: Undirected (on the left) and directed (on the right) graphs. Arrows in the directed graph are called $arcs$, while lines in the undirected graph are called $edges$.



type of graphs with different mutual influences, i.e. $(u, v) \neq (v, u)$, is known as *directed*. This difference is depicted on the figure 1.2. Links between nodes in a directed graph will further be called *arcs*, while links in undirected graphs will be named *edges*.

- When two nodes are linked together, they are said to be *neighbors*. In the context of directed graphs, we have to distinguish between in-neighbors and out-neighbors. In-neighbors of u are the nodes whose arrows are pointing towards u, while out-neighbors of u are vertices whose arrows are pointing away from u. In-and-out-neighbors of u, will be denoted by $N^{in}(u)$ and $N^{out}(u)$, respectively.

1.3 Definition of the issue

Now that the basics of graph theory have been described, we can get into the core of this thesis: the subject we will focus on throughout this thesis is called the **spread maximization problem** or **influence maximization issue**, first described by Kempe et al. (Kempe, Kleinberg, and Tardos, 2003). The problem can be described as follows:

Given the structure of a social network and how individuals influence each other (called the *diffusion model*), find a set of influential individuals S_0, that costs at most a certain budget **k**, that by introducing it with a new product/idea, will maximize the expected number of individuals eventually adopting it. (Adapted from Even-Dar and Shapira, 2011 & Domingos and

Richardson, 2001)

As we will see it later, this problem is computationally complex, especially if applied to huge networks such as Facebook or Twitter. For this reason, heuristics have to be found. The purpose of this thesis is to test some of the most used or promising heuristic algorithms on real small-sized networks (several dozen of nodes) and assess what are the possible marketing implications for companies. Also, we will discuss possible other applications of these techniques.

1.4 Outline of the thesis

Some of the most widespread *diffusion models*, which express how information is passed from node to node in networks, will be described in chapter 2. Two models will be developed: the *Independent Cascade* model (IC in short), which is largely used in the context of information spread or propagation of viruses. Next to the IC model, one can often find in the literature the *Linear Threshold* model (LT in short), which is mainly used in the context of product adoption of a new unproven technology, i.e. when a user needs advice from many friends before buying the product.

Then, we will describe the *influence maximization* problem (see chapter 3) and show that because of its complexity, one will require some heuristics in order to get as close as possible from the most influential nodes. Among all techniques currently existing in the literature, we have arbitrarily chosen to present some of the most famous ones, namely the *Monte Carlo with Greedy evaluations*, the *Lazy Greedy* and the *Maximum Influence Arborescence* method. We further study the *Killed Random Walk* algorithm developed in 2016 by M. Saerens and C. Vande Kerckhove; not yet as famous as the previous ones, nevertheless we have chosen to present it, firstly for the originality of its approach, but also for the good results obtained, knowing the computation time required. Then, we will discuss a new model developed in the present thesis, inspired by the Killed Random Walk method, called the *Iterative Killed Random Walk*.

Algorithms will be assessed according to two criteria: Firstly, the *effectiveness*, i.e. how good is the influence spread compared to the best solution, in

other words, what is the final active set of nodes at the end of the propagation. Secondly, every algorithm will be appraised by its *efficiency*, i.e. what is the *time complexity* of these algorithms.

The experimental procedure followed in this thesis, as well as the presentation of the real networks on which we have tested the algorithms, are the subjects of chapter 4.

Chapter 5 shows the results and makes analyses of algorithms on networks presented in chapter 4, in terms of *effectiveness* and *efficiency*.

In chapter 6, we will discuss some alternatives to the influence spread problem as defined above; in particular, we will focus our attention on a business-based view, which is more interesting for marketers as well as for business developers and the financial department: instead of maximizing *influence spread*, it is more interesting for companies to think about maximization of *adoption rate* of products and *profit*, or minimization of advertising *costs* for tackling a minimal target audience.

Then, in light of the results of the analyses made in chapter 5, we discuss the marketing stakes of the development of the influence maximization problem in the context of social network, as well as its applications in other fields (chapter 7).

Finally, chapter 8 concludes this thesis and emphasizes the future researches that need to be conducted.

Chapter 2

Diffusion Models

2.1 Stochastic diffusion models

Now that the problem of spread maximization has been explained in the previous chapter, it is important to know how information is passed from node to node, i.e. the **diffusion model** (Chen, Lakshmanan, and Castillo, 2014). Given a directed graph $G = (V, E)$ and the initial seed set S_0, a diffusion model "specifies the randomized process of generating active sets S_t for all $t \geq 1$". Models can be either *progressive* when nodes activated stay active forever, i.e. when $S_{t-1} \subseteq S_t$, or *non-progressive* when nodes can switch back and forth between active and inactive states. Typically, adoption of a new product or technology will be modeled by progressive models, while non-progressive models are used when describing spread of ideas, voting intentions or opinions[1]. Throughout this paper, our attention will be focused only on progressive models.

In the next sections, there are more detailed description of the two progressive models discussed in the introduction, which were described by Kempe et al. (Kempe, Kleinberg, and Tardos, 2003), namely the *Independent cascade model* and the *Linear threshold model*.

2.2 Independent cascade model

The main characteristic of the **independent cascade model (IC)** is the independence of the influence spread in each arc of the whole graph; in other words, a single person is sufficient to activate another individual. This model is appropriate for modeling the diffusion of ideas, information, viruses, etc.

[1]See for example the Voter Model presented in the appendix (Even-Dar and Shapira, 2011)

(Chen, Lakshmanan, and Castillo, 2014). Every arc $(u, v) \in E$ has an associated *influence probability* $p_{uv} \in [0, 1]$, that is, the probability that node u succeeds in its attempt to activate its neighbor node v.

The rule describing the independent cascade model can be expressed as follows: After a node *u* is newly activated at time *t-1*, it has a *single chance* to activate all of its (out)neighbors in time *t*[2]. The propagation continues as long as new nodes trigger further activations, i.e. while S_t is increasing. Once no new activations occur at a certain time *t*, S_t reaches its final state.

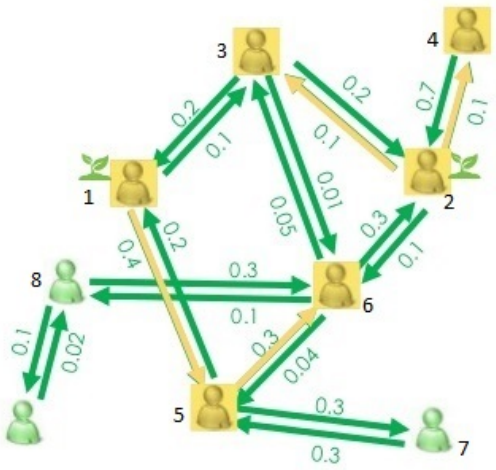


FIGURE 2.1: Example of the diffusion process of the independent cascade model with a set of two seeds (Chen, Lakshmanan, and Castillo, 2014)

You can see on figure 2.1 an example of this diffusion process in a network. The diffusion starts with a set of 2 seeds (represented by green seeds on the figure, vertices 1 and 2). The numbers next to arrows represent the *influence probability* between those users. A yellow arrow means that the propagation already appeared between users, while a green arrow is used when spread has not yet been passed. After the first iteration, node 1 activates node 5, while node 2 activates both number 3 and 4 vertices. For the second iteration, only nodes newly activated (i.e. nodes 3 to 5) can try to activate inactive nodes. Recall that they will suceed depending on $p_{uv} \in [0, 1]$, the *influence probability* between two nodes. Node 3 fails to activate node 6; node 4 has no neighbors to attempt to activate and node 5 succeeds in activating node 6, but fails to activate node 7. At iteration 3, node 6 fails to activate node 8,

[2] Kempe et al. make analogy with a contagion problem: if a user u has been activated in time $t - 1$ he is seen as $contagious$ in time t, but is $non - contagious$ in time $t + 1$. (Kempe, Kleinberg, and Tardos, 2005)

hence the propagation stops, with nodes 1-6 activated.

Formally, at every time step, we first set S_t to be S_{t-1}. Then, for every node not contained in the previous seed set ($v \notin S_{t-1}$) with neighbors which were just activated in the previous iteration ($u \in N^{in}(v) \cap (S_{t-1} \setminus S_{t-2})$), node u makes an activation trial with a probability of success p_{uv}. When the attempt is successful, we add node v to the active set S_t.

2.3 Linear threshold model

Next to the *independent cascade model*, where the diffusion can occur independently by single individuals, there exists an other model introduced by Kempe et al. (Kempe, Kleinberg, and Tardos, 2003), where individuals need reinforcement of several independent friends to change their mind. This model is called the **linear threshold model (LT)**.

The model is suitable for describing the adoption process of a new unproven product. When the sum of all their friends' recommendations reach a certain personal threshold, people are likely to adopt the new product themselves. This is mostly the case when adopting a new standard in an industry[3] (Chen, Lakshmanan, and Castillo, 2014).

While in the *IC model* every arc (u, v) was associated with an influence probability p_{uv}; the same arcs are this time assigned with *influence weight* $w_{uv} \in [0, 1]$ in the linear threshold model. This represents the degree to which user u can influence another user v. Those weights are normalized so that the sum of all incoming arcs of each node is lower or equal to 1:

$$\sum_{u \in N^{in}(v)} w_{uv} \leq 1$$

At every time step t, if the total weight of active in-neighbors of a node v is higher or equal to a specified threshold θ_v, node v is added to the set of

[3]One can think for instance of the adoption of the DVD, while most of the people were previously using VHS

active nodes S_t. Formally, it means S_t is increased by v when

$$\sum_{u \in S_{t-1} \cap N^{in}(v)} w_{uv} \geq \theta_v$$

One can see on figure 2.2 an example of the diffusion process of the LT model. Like in the IC model example, the diffusion starts with a set of 2 seeds (vertices 1 and 2). The number next to each arrow represents the *influence weights* that users have on their neighbors, while the numbers next to each node is the value of the users' threshold, picked at random on the interval [0,1]. After the first iteration, node 1 activates node 3, since $w(1,3) = 0,4 \geq 0,2 = \theta_3$. Similarly, node 2 activates vertex 4 because $w(2,4) = 0,8 \geq 0,5 = \theta_4$. Node 5 needs both 1 and 2 to be activated: $w(1,5) + w(2,5) = 0,6 = \theta_5$. Iteration 2 starts, this time vertex 6 is activated, since $+w(2,6)+(3,6)+w(5,6) = 0,7 = \theta_6$. At that moment, vertex 3 cannot activate node 8, hence the iteration ends. In iteration 3, node 6 is not able to activate node 7; the network has reached its final state.

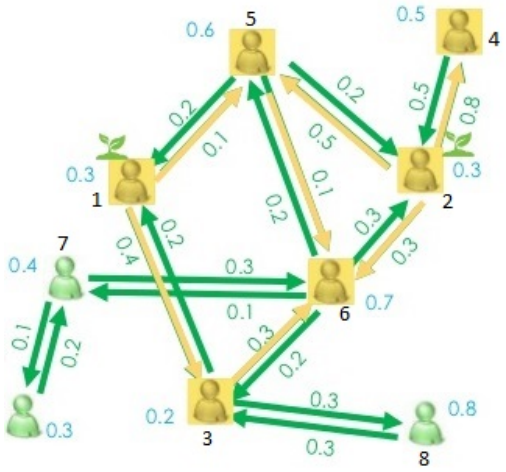


FIGURE 2.2: Example of the diffusion process of the Linear Threshold model with a set of two seeds (Chen, Lakshmanan, and Castillo, 2014)

Several specificities of this model are worth noticing. First, one can observe that the only randomness of this model is the selection of the set of thresholds θ_v's; in fact once those thresholds are chosen, the rest of the algorithm is deterministic. Second, as the sum of all in-neighbors' weights can be lower than θ_v; this means an individual can still stay inactive even if every of its neighbors are active.

Finally, we can observe that it is possible to activate a node v which possess only one active in-neighbor u, when the influence weight $w_{uv} \geq \theta$. This model is thus not specific for *complex contagions*, that is, where at least two nodes are necessary to activate an inactive node.

2.4 Monotonicity and Submodularity

In this section we will describe two mathematical features that both IC and LT model share: **Monotonicity** and **Submodularity**. Those characteristics will have a big impact in practice as we will see it later. (Kempe, Kleinberg, and Tardos, 2003)

A function is said to be monotone when "adding an element to a set cannot cause f to decrease":

$$f(S \cup v) \geq f(S)$$

for all elements v and sets S." (Kempe, Kleinberg, and Tardos, 2003).

Submodularity is a notion related to the diminishing yield: when adding an element to a set R, the marginal gain is as low as the addition of the same element to a subset of R (Dughmi, 2009). One can rewrite this property mathematically:

$$f(S \cup \{v\}) - f(S) \geq f(R \cup \{v\}) - f(R),$$

for every $S \subset R$

In their seminal article, Kempe et al. (Kempe, Kleinberg, and Tardos, 2003) proved that both IC and LT models are monotone and submodular (curious readers can refer to that paper for the proofs of these theorems).

Those two measures are worth noticing, since they will allow us to use the greedy algorithm later (see section 3.2.1), which ensures a $1 - 1/e$ approximation of the best solution (Nemhauser, Wolsey, and Fisher, 1978):

$$\frac{ValueOfGreedyApproximation}{ValueOfOptimalSolution} \geq \frac{e-1}{e}$$

where e is *Euler's number* (base of the natural logarithm).

Chapter 3

Influence Maximization

Now that we have described in the previous chapter the most famous information diffusion processes, we need to identify which individuals to target in our seed set in order to maximize *influence coverage*, i.e. the number of active vertices at the end of the diffusion. In this chapter, we will follow the approach of Kempe et al. (Kempe, Kleinberg, and Tardos, 2003), who firstly express "influence maximization as a discrete stochastic optimization problem" (Chen, Lakshmanan, and Castillo, 2014).

3.1 Definition of the problem

As stated in section 1.3, the goal of the **spread maximization problem** as defined by Kempe et al. is to find the seed set S_0 of size k that would maximize the expected final number of active users (Chen, Lakshmanan, and Castillo, 2014: $\sigma(S_0)$. This can be expressed mathematically as follows:

Given a graph $G(V, E)$, a diffusion model and the size of the seed set $k = |S_0|$, find a seed set $S_0 \subseteq V$ with $S_0 \leq k$, such that the expected influence spread $\sigma(S_0)$ is maximized. In other words, compute $S^* \subseteq V$ such that

$$S^* = \underset{S_0 \subseteq V, |S_0|=k}{\operatorname{argmax}} \sigma(S_0)$$

Notice that finding the set with broadest influence is different from identifying the k nodes which have the largest individual influence. Suppose a network where the top 2 "influencers" have impact on exactly the same people; in this case the result would be the same with only one of these individuals. Hence, integrating the second in the seed set S_0 would be pointless.

(Chen, Lakshmanan, and Castillo, 2014)

Of course, many extensions and alternatives of the problem defined above can be expressed. Two alternatives often found in the literature are described below:

- *Budget constraint*: In the classic approach, it is assumed that the cost of reaching any node in the network is the same, hence the budget constraint B is defined as a number of nodes, given beforehand. However, in some extensions[1], researchers remove the hypothesis of uniform costs for nodes. The cost of tackling node v is denoted by C_v and the budget is defined as an upper bound of the sum of nodes' costs included in the seed set: $\Sigma_{v \in S_0} C_v \leq B$

- Often, the advertising campaigns are supposed to be longlasting (in the sense that marketing teams want to increase their target incrementally and not in one shot). For that reason, other models try to maximize the active set of user at certain time steps, instead of the final active set.

3.2 Complexity and need for a greedy approach

Influence maximization problem is hard, the difficulties have two different natures (Chen, Lakshmanan, and Castillo, 2014):

- First, Wang et al. (Wang, Chen, and Wang, 2012) have proved that the computing of influence spread $\sigma(S_0)$ of any given S_0 is #P-hard for both IC and LT models, even when the seed set is composed of a single element. This is what they call the *combinatorial hardness*.

- Moreover, finding the seed set S_0 that maximizes the influence maximization problem defined in section 3.1 is also computationally hard; in fact Chen et al. proved that the influence maximization problem is NP-hard under both IC and LT models, even for only one seed node, i.e. when $k = 1$ (Chen, Lakshmanan, and Castillo, 2014).

[1] see Even-Dar and Shapira, 2011 for example

Because of the natural complexity of the problem, one would need better *central processor unit (CPU)*'s performances, larger *memory* and more time in order to reach the optimal solution of the problem. As an example, notice that the number of possibilities for a 1000 vertices network for wich you would want to find the 50 most influential members is high than the estimated number of atoms in the universe (Vande Kerckhove, 2016). Hence, evaluating every possibility would require a computation time way over a man's lifespan, thus making the result totally pointless.

Since we want to save resources (time, CPU performances and memory), we need to use some heuristics, which are supposed to get close to the optimal solution, but in a much smaller computing time. A *greedy approach* is introduced in order to tackle the problem of the *Combinatorial hardness*, while *Monte Carlo simulations* are used to overcome the hardness of computation of the influence maximization problem. Using this greedy approach is only possible since influence spread functions $\sigma()$ exhibit monotonicity and submodularity (see section 2.4).

Note: even when problems are solvable by computer means, it is worth knowing their *time complexity*, that is, the computation time required to solve them. This actual time function needed for computing algorithms, also known as *running time*, is usually extremely complex, that is why it is expressed by taking only the highest order term (and disregarding coefficient of that term) of the complete time complexity function (Sipser, 2006). Assume the function $f(n) = 2n^4 + 3n^3 + 2n^2 + 5n + 4$, its *Big-O* notation (thus representing the time complexity) will be rewritten as $f(n) = O(n^4)$.

3.2.1 Greedy algorithm

This algorithm can be understood as follows: Starting from an empty seed set S_0, in each round i (out of k rounds), one element u is added to it, such that u provides the largest marginal contribution to the monotone and submodular influence function f, with respect to S_0 [2]. This process is repeated k times, k being the size of our desired seed set S_0.

[2]Where f represents the influence function σ of both LT and IC models. As a reminder, it is computed as the sum of nodes activated at the end of each loop and ultimately at the end of the propagation.

Algorithm 1 Greedy algorithm

Inputs: number of nodes k, f: monotone and submodular function
Output: Seed set S_0

Initialize $S_0 \leftarrow \emptyset$
for $i = 1$ to k **do**
  $u \leftarrow \arg\max_{w \in V \setminus S_0}(f(S_0 \cup \{w\}) - f(S_0))$
  $S_0 \leftarrow S_0 \cup \{u\}$
end for
return S_0

As already mentioned, Nemhauser et al. (Nemhauser, Wolsey, and Fisher, 1978) proved that this greedy algorithm ensures an approximation ratio of $(1 - 1/e)$ compared to the optimal solution:

$$f(S_0) \geq (1 - 1/e) f(S^*),$$

where S_0 is the seed set computed with the greedy algorithm, S^* being the set maximizing $f(S)$ among all sets with size k.

3.2.2 Monte Carlo simulations (Greedy)

Now that the combinatorial problems are overcome, i.e. that we have found a seed set S_0 thanks to the greedy algorithm, we need to estimate the influence spread $\sigma(S_0)$ for validation. As suggested by Kempe et al. (Kempe, Kleinberg, and Tardos, 2003), we can make it with **Monte Carlo** simulations, shortly *MC*. Monte Carlo experiments have the particularity to rely on simulation of random sampling in order to evaluate the value of the exact solution. The name *Monte Carlo* was chosen by Metropolis and Ulam because of the city known for its games of chance; in fact they wanted to evaluate their chance of success on complex games, the optimal solution being hard to find at that time with the available resources (Metropolis and Ulam, 1949).

The present method can be expressed as follows: given a seed set S_g (found with the greedy algorithm) we simulate R times (R being a large number chosen in advance) a diffusion process. After each simulation, we count the number of active nodes. At the end of the R simulations, we obtain an estimation of the influence spread $\sigma(S_0)$: $\frac{SumOfActiveNodes}{R}$. Obviously, the

higher the number of simulations R, the better the accuracy.

The algorithm which combines the *Greedy* approach and *MC* algorithm for estimating $f()$, called the *Monte Carlo greedy algorithm* (in short *MC-Greedy(G,k)*), is formally defined in algorithm 2.

Algorithm 2 MC-Greedy algorithm

Inputs: Graph G, number of nodes k
Output: Influence spread $\sigma(S_0)$

Initialize $S_0 \leftarrow \emptyset$
for $i = 1$ to k **do**
  $u \leftarrow \arg\max_{w \in V \setminus S_0} \left\{ MC - Estimate(S_0 \cup \{w\}, G) \right\}$
  $S_0 \leftarrow S_0 \cup \{u\}$
end for
return S_0

function MC-ESTIMATES(S_0,G)
  count $\leftarrow 0$
  for $j = 1$ to R **do**
    Simulate the diffusion process
    $n_a \leftarrow \sharp$ activated nodes after diffusion
    $count \leftarrow count + n_a$
  end for
return count/R

end function

This approximation is easier to compute compared to the exact solution, as Chen et al. highlighted it: Let $n = |V|$, $m = |E|$, then "with probability $1 - 1/n$, algorithm MC-Greedy(G,k) achieves $(1 - 1/e - \epsilon)$ approximation ratio in time $O(\varepsilon^{-2}k^3n^2m \log n)$, for both IC and LT models". (Chen, Lakshmanan, and Castillo, 2014)

Nevertheless, the MC-greedy algorithm is still inefficient for two reasons:

- Firstly, the number of influence spread evaluations is large, in fact $O(nk)$ evaluations are necessary (Leskovec, Adamic, and Huberman, 2007).

- Secondly, while processing Monte Carlo simulations for obtaining the influence spread $\sigma(S)$, empirical results suggest that R should be large in order to keep the effectiveness of the greedy algorithm. (Chen, Lakshmanan, and Castillo, 2014)

3.3 Scalable influence maximization

Because the MC-greedy algorithm is not efficient, some improvements have been developed. In this section two of them will be described, namely *reducing the number of evaluations* and *accelerating the influence computation* (Chen, Lakshmanan, and Castillo, 2014).

3.3.1 Cost-effective lazy-forward selection (CELF)

Leskovec et al. have shown that by using submodularity property, we can use a *lazy algorithm*, or **Cost-effective lazy-forward** evaluations method (in short *CELF)*, which significantly reduces the number of evaluations. The computation time can be reduced by a factor of up to 700. The idea behind this algorithm is to avoid evaluations when they are not necessary; in fact, when running the greedy algorithm, $n.k$ evaluations are roughly required (Leskovec, Adamic, and Huberman, 2007).

Given a submodular and monotone function f (such as the influence spread for the LT and IC models), let $f(u|S) = f(S \cup \{u\}) - f(S)$ denote the marginal contribution of individual u to set S. Suppose during iteration i the algorithm has evaluated $f(w|S)$ for $w \in V \setminus S$. If we denote the seed set of a previous iteration by $S' \subset S$ and presume algorithm has evaluated $f(x|S')$ for $x \in V \setminus S$ and $f(x|S') \leq f(w|S)$, we can conclude thanks to submodularity that $f(x|S) \leq f(x|S') \leq f(w|S)$. Thus, there is no need to select x in the *i*-th iteration in order to evaluate $f(x|S)$. In practice, this is implemented by creating a *priority queue* Q as shown in algorithm 3, the so called *LazyGreedy algorithm* or *CELF algorithm*.

In this algorithm, $u.mg$ denotes the latest computed marginal gain of node u to its corresponding set, while $u.i$ highlights the iteration in which $u.mg$ is

last updated.

Let us now describe how this algorithm works: First, the marginal gain of every node is computed and set in the priority queue Q, classified by decreasing order of $u.mg$. Then the top node v is taken out of the queue and if $v.i$ is the current iteration, v is added to the seed set; otherwise (i.e. if $v.mg$ has not been computed in iteration "i") $v.mg$ is recomputed and added at the right place in the queue. (Chen, Lakshmanan, and Castillo, 2014)

Algorithm 3 The CELF algorithm

Inputs: number of seeds k, f: monotone and submodular function
Output: Seed set S_0
Initialize $S_0 \leftarrow \emptyset$; priority queue $Q \leftarrow \emptyset$; $iteration \leftarrow 1$; Stack's top element $u \leftarrow \emptyset$
for $v = 1$ to n **do**
    $v.mg \leftarrow f(v|\emptyset)$; $v.i \leftarrow 1$
    insert v to Q with $v.mg$ as the key
end for
while $iteration \leq k$ **do**
    extract top element of Q
    if $u.i = iteration$ **then**
        $S_0 \leftarrow S_0 \cup \{u\}$
        $iteration \leftarrow iteration + 1$
    else
        $u.mg \leftarrow f(u|S_0)$
        $u.i \leftarrow iteration$
        reinsert u into Q
    end if
end while
return S_0

3.3.2 Maximum Influence Arborescence algorithm for the IC Model (MIA)

The second source of problem in the MC-Greedy algorithm is the slowness of the Monte Carlo simulations in finding an estimation of the influence spread. For this reason, other approaches based on the specificities of graph structures and the diffusion models used to describe the way information is propagated. Hence, for the *Independent cascade (IC)* model, Wang et al. (Wang,

Chen, and Wang, 2012) suggested the **Maximum In-Arborescence (MIA)** algorithm.[3] .

Wang et al. have shown, through their **Maximum Influence Arborescence (MIA)** algorithm, that the structure of trees can be exploited in order to compute efficiently the influence spread. The efficiency of their heuristics comes from the fact that they are "restricting computations on the *local influence regions* surrounding each node v" (Wang, Chen, and Wang, 2012).

Arborescence is a "tree" in a directed graph where all arcs are either pointing toward a certain node (in-arborescence), or away from the node (out-arborescence). In their algorithm, Wang et al. firstly compute the distance between nodes D as the logarithm of the inverse of the influence probability P_{uv}: $D(u,v) = log(\frac{1}{P_{uv}})$. Then, one can compute the shortest paths between every pair of nodes thanks to the famous *Dijkstra's shortest-path* algorithm shown below (algorithm 4). Each of those shortest paths are called *maximum influence paths*, in short $MIP(u,v)$. The probability of influence of a node u on another node v located on the same MIP can simply be computed as the product of the intermediary influence probabilities of all arcs between them along that path.

Then, they fix an influence threshold $\theta \in [0,1]$ and ignore MIP's with probabilities smaller than θ. This threshold allows them to define the concept of *maximum influence in-arborescence (MIIA)* for a certain node u given a threshold θ; the arborescence obtained by gathering all MIP's from node u, which exceeds θ. Formally, we have

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

Let us now denote by $ap(u)$ the *activation probability* of node u, the probability that node u is activated at the end of the diffusion process, knowing the seed set S_0 and the arborescence $MIIA$. Its recursive computation can be done thanks to algorithm 5.

In this algorithm, one can notice that the probability that w activates node u is given by $ap(w)p_{wu}$. In this expression, $ap(w)$ means that node w has

[3] In order to overcome problems related to *LT* diffusion models, Goyal et al. (Goyal, Lu, and Lakshmanan, 2011) invented a method based on the marginal contributions of simple paths. This algorithm is called the *SimPath* and is described in the appendix (section 9.1.2)

Algorithm 4 Dijkstra algorithm for computing shortest paths

Inputs: Graph G, initial node s
Outputs: Minimal distance vector from vertex s: $D(s)$, vector of Predecessors $Pred()$

Initialize $N' \leftarrow \{s\}$; Pred $\leftarrow \emptyset \; \forall v \neq s$; $D \leftarrow \infty \; \forall v \neq s$; $D(s) \leftarrow 0$
for nodes v in the network **do**
  if v neighbor of u **then**
    $D(v) \leftarrow (s, v)$
  end if
end for
while not all v's in N' **do**
  find w not in N', such that $D(w)$ is minimum
  add w to N'
  for all nodes v neighbors of w **do**
    $D(v) \leftarrow min(D(v), D(w) + c(w, v))$
    $Pred(v) \leftarrow w$
  end for
end while
return $D(s)$ and $Pred()$

Algorithm 5 Activation probabilities computation

Inputs: node u, seed set S_0
Output: activation probability $ap(u, S_0, MIIA(w, \theta))$

if $u \in S_0$ **then**
  $ap(u) \leftarrow 1$
else
  $ap(u) \leftarrow 1 - \Pi_{w \in N^{in}(u)}(1 - ap(w).p_{wu})$
end if
return $ap(u)$

to be active in order to try to activate any node, while p_{wu} is the probability that the trial is successful. With this in mind, we can understand that $\Pi_{w \in N^{in}(u)}(1 - ap(w).p_{wu})$ is the probability that none of u's in-neighbors activate node u.

This algorithm assumes influence only diffuses through tree branches of $MIIA(v, \theta)$ to reach node v. Though this heuristics cannot give any theoretical quality guarantee, empirical results have shown good level of effectiveness.

Let $InfSet(u)$ be the set of nodes whose in-arborescence contains u; $InfSet(u) = \{v | u \in MIIA(v, \theta)\}$ and $IncInf(w)$ denote the incremental influence (or marginal influence) of w when the seed set is empty, while $IncInf(w, v)$ is the marginal contribution of w towards activation probability of node v.

Suppose now that the marginal influence spread $\sigma(w|S_0)$ has been computed for all $w \in V \setminus S_0$. When the next seed u is added to the seed set, only nodes sharing in-arborescence with u, i.e. $InfSet(u)$, are updating their activation probabilities and marginal influence spread. This greatly enhances the computation time, thanks to the lower number of computations in each iteration. Formally, the algorithm 6 shows how to update the marginal influence when a new node is added to the seed set.

We now have all the elements of the MIA method shown in algorithm 7 (Chen, Lakshmanan, and Castillo, 2014)

In fact, algorithm 7 allowed us to find our seed set S_0. Next, thanks to the approximation of the *activation probabilities*, we can now estimate the value of the influence spread $\sigma(S_0)$ by $\sum_{v \in V} ap(v, S_0, MIIA(v, \theta))$.

Algorithm 6 Update Algorithm - Computes the updated marginal influence spread of u

Inputs: Seed set S_0, node u added to the seed set
Outputs: Updated values for *IncInf(w)* and *IncInf(w,v)*, $\forall v \in InfSet(u) \setminus S_0$ and $w \in MIIA(v,\theta) \setminus (S_0 \cup \{u\})$
for $v \in InfSet(u) \setminus S_0, w \in MIIA(v,\theta) \setminus S_0$ **do**
    $IncInf(w) \leftarrow IncInf(w) - IncInf(w,v)$
end for
for $v \in InfSet(u) \setminus (S_0 \cup \{u\})$ and $w \in MIIA(v,\theta) \setminus (S_0 \cup \{u\})$ **do**
    $IncInf(w,v) \leftarrow ap(v, S_0 \cup \{u,w\}, MIIA(v,\theta)) - ap(v, S_0 \cup \{u\}, MIIA(v,\theta))$
    $IncInf(w) \leftarrow IncInf(w) + IncInf(w,v)$
end for

Algorithm 7 MIA algorithm

Inputs: Graph G with influence probabilities; number of seeds k
Outputs: Seed set S_0

initialize $S_0 \leftarrow \emptyset$; $IncInf(w) \leftarrow 0, \forall w \in V$
for $v \in V$ **do**
    compute $MIIA(v,\theta)$ and $InfSet(v)$ via Dijkstra's algorithm
    for $w \in MIIA(v,\theta)$ **do**
        $IncInf(w,v) \leftarrow p(MIP(w,v))$
        $IncInf(w) \leftarrow IncInf(w) + IncInf(w,v)$
    end for
end for
for $i = 1$ to k **do**
    $u \leftarrow arg\max_{v \in V \setminus S} IncInf(v)$
    update(S_0, u) with *Update* algorithm (algorithm 6)
    $S_0 \leftarrow S_0 \cup \{u\}$
end for
return S_0

3.4 Killed Random Walk Algorithm (KRW)

3.4.1 Random walker

A new method for computing the spread of influence, based on random walks on a graph was proposed by Corentin Vande Kerckhove and Marco Saerens (Vande Kerckhove, 2016). The problem of the random walker on graph is well known and was among others developed by Page and Brin in their seminal paper when they introduced the **PageRank** algorithm (see section 9.1.1). The idea behind is the following: a random walker on a graph G is a user starting from a random node $v_0 \in V$, and jumping randomly from neighbor to neighbor through arcs. This randomness is function of a *transition matrix* T, which is defined as the normalized version of the *adjacency matrix* A. The latter is a matrix where element a_{ij} is the influence weight that i is exercing on j.

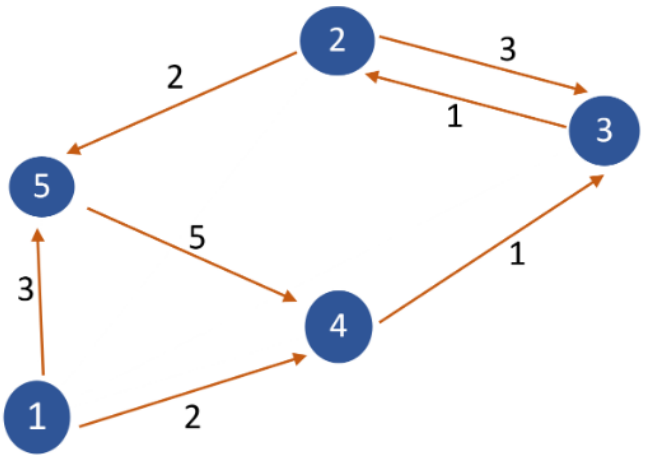


FIGURE 3.1: Example of a directed graph

As an example, the weighted and directed graph presented on figure 3.1 , can be depicted by the adjacency matrix A_1 and its corresponding transition matrix T_1:

$$A_1 = \begin{pmatrix} 0 & 0 & 0 & 2 & 3 \\ 0 & 0 & 3 & 0 & 2 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 \end{pmatrix} \quad T_1 = \begin{pmatrix} 0 & 0 & 0 & 2/5 & 3/5 \\ 0 & 0 & 3/5 & 0 & 2/5 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Informally, the T matrix represents the probability that a *random walker* jumps from a node to another node: $T_{u,v}$ = Probability that the walker is in position v at a time t, knowing that he was in position u in time $t-1$. One can observe that the lines of this matrix sum to one and the diagonal elements are zeros. Hence, the probability that a random walker lies in u at time t, knowing that he was in u at time $t-1$ is equal to zero; in other words, the random walker is forced to jump at each time step.

Whatever the initial graph and the initial allocation of random users on a network, when the number of iterations tends to infinite, the distribution on a network reaches a steady state, called the *stationary distribution* of the walkers.

3.4.2 Entropy

In the special case of the **Killed Random Walk** (KRW), a special heuristics is used to compute the diffusion throughout the network: *entropy*. This value can be calculated from the rate of crossings induced by a walker to each node of the graph. Formally, starting from node u, we can count the number of crossings to node v during a defined random walk (when number of iterations tends to infinite), n_{uv}. The set of crossings induced by node u (starting point) is denoted by $N_u = \sum_v n_{uv}$. Entropy of the stationary distribution can now be defined by

$$Entropy_u = -\sum_v \frac{n_{uv}}{N_u} log(\frac{n_{uv}}{N_u}).$$

One can see that if the diffusion ends with a uniform distribution of walkers in the network, entropy will end up with a large value. In contrast, if u generates crossings to only a few number of nodes, this will result to a low entropy $S_u \simeq 0$.

3.4.3 Killings

In order to ease the computations (for now the number of iterations tends to infinite), Vande Kerkhove and Saerens (Vande Kerckhove, 2016) introduced the idea of killing the walkers, so that the number of iterations decreases. In practice, this is done by reducing the arcs' weights, so that walkers' number

decrease with time. Let T_{kill} represent this modified matrix T and be computed thanks to the following elementwise product

$$T_{kill} = T.W,$$

where

$$W = exp(-\lambda C),$$

with λ an adjustment parameter and C a matrix with positive costs associated to transitions between nodes. In this paper, it has been chosen to define the cost matrix C in two different ways that we will compare in the analyses of chapter 5.

First of all, we describe this cost as inversely proportional to confidences (weights of arcs): $C_{uv} = \frac{1}{t_{uv}}$, where t_{uv} represents the transition probability from u to v. We will later refer to this method by *KRW-P*.

The second computation is done by replacing the transition matrix by the adjacency matrix: $C_{uv} = \frac{1}{a_{uv}}$, where a_{uv} depicts the element (u, v) of the adjacency matrix as defined in subsection 3.4.1. In chapter 5, this will be denoted by *KRW-adj*.

Regarding the adjustment parameter and following the results of Vande Kerkhove (Vande Kerckhove, 2016), we have set λ to be equal to 0.35.

3.4.4 Fundamental matrix

In his thesis, Vande Kerkhove also proved that the matrix containing the number of crossings can be deduced from the T_{Kill} matrix, thanks to the relation [4]

$$N = ((I - T_{Kill})^{-1})^t.$$

Knowing that $(I - T_{Kill})^{-1}$ contains all the information required, it is called the fundamental matrix.

[4] curious reader can refer to this master thesis for a proof of this relation (see also Fouss, Saerens, and Shimbo, 2016).

3.4.5 Killed Random Walk algorithm

The KRW algorithm is the combination of two intermediary algorithms: firstly, the *GetEntropy* will compute the entropy for all nodes of the network thanks to the formula expressed hereunder (see algorithm 8). Secondly, it will sort the nodes by decreasing order of *entropy* and select the k first vertices in the stack, k being the the budgeted number of nodes that can be selected. The function used to take element by element the nodes on top of the stack is called POP and is presented formally in the algorithm 9. The whole KRW can thus be seen in a formal way in algorithm 10.

Algorithm 8 GetEntropy algorithm

Inputs: Transition matrix T, cost coefficient λ
Output: Entropy vector $Entropy$
Initialize $Entropy \leftarrow [0]^n$
$W \leftarrow exp(-\lambda C)$
$T_{kill} \leftarrow T.W$
$N = ((I - T_{Kill})^{-1})^t$
$N_{col} \leftarrow ColSums(N)$
for $i = 1$ to n **do**
    $Entropy[i] \leftarrow -\Sigma_j \frac{|N|_{ij}}{N_{col}(i)} log \frac{|N|_{ij}}{N_{col}(i)}$
end for
return Entropy

Algorithm 9 POP algorithm

Input: Stack
while $Top \geq 0$ **do**
return Stack [Top]
    $Top \leftarrow Top - 1$
end while

Algorithm 10 Killed Random Walk algorithm

Inputs: Graph G with Transition matrix T, cost coefficient λ
Output: Seed set S_0
initialize $S_0 \leftarrow \emptyset$; $entropy \leftarrow [0]^n$

$Entropy\_List \leftarrow GetEntropy(T, \lambda)$
$Ordered\_IDs \leftarrow ArgSort(Entropy\_List)$
for $i = 1$ to k **do**
  $i \leftarrow ordered\_IDs.Pop()$
  $S_0 \leftarrow S_0 \cup \{v_i\}$
end for
return S_0

3.5 Iterated Killed Random Walk (IKRW)

In order to further improve the *effectiveness* of the **Killed Random Walk** defined in the previous section, it has been chosen to reassess the entropy's marginal contribution of all nodes (except those already being seeds), every time a new node is added to the seed set. This way, we could avoid selecting two nodes sharing a lot of (or even all) common edges. Hence, the random walks and killings of the random walkers do take into account the seeds already selected.

Formally, this is done by creating two conditional variables, called **Conditional Average crossings** (CAC) and **Conditional Entropy** (CE), denoted by $N(i|S_0)$ and $Entropy(i|S_0)$ respectively, where i is the node being tested and S_0 being the set of all seeds selected so far.

The *Conditional Average Crossings* matrix is computed as follows:

$$N(i|S_0) = \frac{e_i + \sum_{u \in S_0} e_u}{|S_0| + 1} N,$$

Where e_i is a unit vector and $|S_0|$ is the number of elements in the seed set.

For its part, the *Conditional Entropy* is computed as before, by replacing the former $N(i)$ variable by $N(i|S_0)$.

The **Iterative Killed Random Walk** algorithm is described in pseudo-code hereafter.

As for the *KRW* model, the costs in the case of *IKRW* are computed according to two different matrices, i.e. *Transition* as well as *Adjacency* (see subsection 3.4.3 for a reminder of those computations). Those models based on *transition* and *adjacency* matrices will be further abbreviated by *IKRW-P* and *IKRW-adj*, respectively.

Algorithm 11 Iterative Killed Random Walk algorithm

Inputs: Graph G with Transition matrix T, cost coefficient λ
Output: Seed set S_0
initialize $S_0 \leftarrow \emptyset$; $CE \leftarrow [0]^n$; $CAC \leftarrow N$

for $iteration = 1$ to k **do**
  for $i \notin S_0$ **do**
    Compute $CAC(i|S_0)$
    Compute $CE(i|S_0)$
  end for

  $u \leftarrow \arg\max CE(i|S_0)$
  $S_0 \leftarrow S_0 \cup u$
  $iteration \leftarrow iteration + 1$
end for
return S_0

3.6 Betweenness Centrality (Between)

In order to analyze the performances of every algorithm presented so far, it is convenient to check whether they offer better results than algorithms which can be easily computed. Among all *centrality* methods taken from the book of Wasserman, we have arbitrarily chosen to select the algorithm called the **betweenness** centrality.

It uses the fact that several users could be dependent on a particular user because he is located in a good position in the network, in the sense that this vertex is in the middle of many shortest paths. Assume the shortest path between u and w is uvw. In this case v possess some control on the interactions between u and w. (Wasserman and Faust, 1994)

The formal computation of the betweenness centrality has been modeled by Freeman in 1977, the reader can find this computation below (Freeman, 1977):

$$BC(v) = \sum_{u \neq v \neq w \in V} \frac{\varsigma_{uw}(v)}{\varsigma_{uw}},$$

where ς_{uw} denotes the number of shortest paths between u and w, while $\varsigma_{uw}(v)$ represents the number of shortest paths between u and w that contain v.

Finally, the k nodes having the best BC score are selected in the seed set.

3.7 Selected list of algorithms

To sum up, the retained algorithms of the present chapter (chapter 3) further developed and tested in chapter 5 are namely the *Betweenness centrality (Between)*, the *Monte Carlo Greedy (MC-Greedy)*, the *Killed Random Walk (KRW)* method developed by Vande Kerkhove and Saerens (Vande Kerckhove, 2016), the *iterative Killed Random Walk (IKRW)*, the *Cost-Effective Lazy Forward (CELF)* algorithm and eventually the *Maximum Influence in-Arborescence (MIA)* algorithm.

Chapter 4

Experimental methodology and experiments

The reader can find in this chapter the experimental procedure followed throughout this thesis so as to perform experiments. This method can be summarized in 6 distinct steps:

1. *Dataset selection* on which algorithms will be tested.
2. *"Relationships" extraction*, i.e. existing links between nodes.
3. *Weighting of edges* between nodes
4. Assignation of *theta* values for each node.
5. Best *seeds extraction* for each algorithm.
6. *Computation of the spread* of influence based on the best seed extraction.

Then are discussed the several networks on which tests of influence spread are conducted. Those networks further developed are **real networks** only, since they better represents how relationships occur in the real world; hence the results obtained can be used directly by marketers and advertisers.

Eventually, the last subject of this chapter is a description of how the parameters of the algorithms, i.e. θ and α, are determined.

4.1 Experimental procedure

4.1.1 Dataset selection

Dataset selection is an important step, as our experiments will only be as insightful as the type of dataset we are testing. Therefore, all of the selected

datasets derive from real-life situations based on observation of human behaviors. Thus, the conclusions of our experiments approach what we would see in real life and present an added value for marketers, among others.

More formally the dataset selected must be composed of nodes and edges. This is done by taking a square matrix, where every element (i, j) represents the relationship between users i and j. The matrices used in the present thesis is the subject of the next section (section 4.2).

4.1.2 Relationships extraction

In this step, the elements (i, j)'s of the square matrix are set either to 1 or 0, depending if nodes i and j share some relationships with each other or not, respectively. This leaves us with a binary matrix mapping every link between all pairs of nodes.

4.1.3 Relationship's weights

The above-mentioned matrix will then be weighted by multiplying the 1's by more realistic numbers, in order to represent real-life relationships more accurately. This allows us to make some differentiations among relationships; i.e. differentiating between your best friends and acquaintances on Facebook. The higher the weight, the bigger the influence between two nodes.

Since in real life only a few number of people have a real influence on our behavior, weights need to be attributed with caution and high weights should be rather rare. In practice, weights are attributed following a power law of parameter *k* equals two.

$$f(x) = ax^{-k}$$

More in detail, we used the *Inverse transform sampling* technique to compute the weights. This technique is well known, and we will not go into details regarding it as plenty of documentation already exists on the subject. Hereunder, a graph of the *Cumulative Distribution Function* of the used power

law.

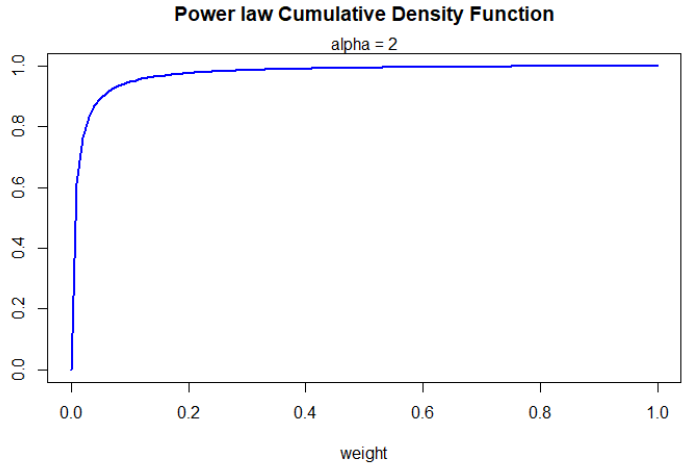


FIGURE 4.1: Cumulative density function of a Power Law

One can indeed see that a vast part of the density function lies around a $[0, 0.1]$ interval and thus weights tend to be small.

4.1.4 Theta assignment

In order to exploit the *Linear Threshold* model, one must assign θ values to each of the nodes of the graph. It has been chosen to allocate those values according to a uniform law between 0 and 1. θ represents the tenacity to get influenced for a particular vertex. Thus, the higher the value of θ is, the more difficult it will be to "convince" that individual (i.e. the total influence on this node will have to be higher). More details on this can be found in chapter 2.

4.1.5 Seed set extraction

All the parameters of the network have now been set. We thus run every algorithm developed in chapter 3 through the network and extract the seed set S_0. Each algorithm delivers its own seed set S_0.

4.1.6 Spread computation

Eventually, the seed sets S_0 are run through the network on each of the two *diffusion models*, so as to compare the different algorithms. For each S_0, two different measures are being taken into consideration: the total number of

activated nodes and the computing time of the seed set S_0.

Since the *Independent Cascade model* is probabilistic, we require a *Monte Carlo* simulation. This process is described in section 3.2.2, and large documentation exists for the curious reader. As a rule of thumb, we used 10.000 simulations per S_0.

Also, regarding the *Linear Threshold model*, it could be that some settings of θ's could favor one algorithm over the others. In order to prevent that, we also used a *Monte Carlo* with 10.000 simulations, with a new computation of θ's for each simulation. Notice that since some algorithms use θ as a parameter, the seed set S_0 had to be computed for each simulation. Thus, the computing time in that case becomes the average computing time through the 10.000 simulations (see hereunder).

4.1.7 Results interpretation

For each S_0, two different measures are being taken into consideration. First, we record the total number of activated nodes in the network at the end of the diffusion, i.e. the influence coverage $\sigma(S_0)$. Of course, we want this number to be as high as possible. We will refer to this number as the *effectiveness* of one algorithm. The higher the number of activated nodes, the more effective.

Secondly, we also track *computing times* for each algorithm. The question here is "How long does it take for the algorithm to compute the seed set S_0?". Computing time should be as low as possible. We will refer to the computing time as the *efficiency* of one algorithm. The lower the computing time, the more efficient.

Effectiveness and *efficiency* are the two measures we use to compare algorithms. *Ceteris paribus* , we will always favor the most effective and efficient algorithm. However, the choice of one algorithm over the other could be driven by a particular business case. Say, one company has limited financial means but has time before going on the market. Then, the focus would rather be on effectiveness rather than efficiency. On the other hand, a major company has witnessed a bold move from a competitor and needs to act quickly.

In that case, focus will more likely be on efficiency.

Thus, the two concepts are useful and should drive decisions of which algorithm to use depending on the situation.

4.2 Datasets

The tests of influence spread conducted in this paper are run on several real networks. For the sake of computing time savings, it has been chosen to exclusively use networks of several dozen of nodes, but not more.

4.2.1 Fraternity network

The "Fraternity" dataset (Bernard, Killworth, and Sailer, 1981) describes interactions among students living in a fraternity at a West Virginia college. The data collected is the number of time students were talking to each other in public areas of the building (new record every 15 minutes) during 5 days, 21 hours per day. The number of students in this fraternity was 58; hence the the matrix' size representing those relations is (58×58) and is of course symmetric and weighted.

4.2.2 Office network

The "Office" network (Killworth and Bernard, 1976) is quite similar to the "Fraternity" network in the sense that it represents the frequency of interactions between people (40 employees in this context), recorded by an inside observer. In this case, new records were taken every 15 minutes for 8 working days, during office hours. Hence, the matrix used for this network is (40×40), symmetric and weighted.

4.2.3 HAM Radio network

The "HAM radio" network (Killworth and Bernard, 1976) describes how frequently do radio operators of "HAM radio" talk to each others. In particular the interactions among the 44 operators were recorded for a period of one month. The matrix used for this network is thus (44×44) and symmetric.

4.2.4 Technical research group

This dataset (Killworth and Bernard, 1976) presents the interactions' frequency among a technical research group at Western Virginia University. One record is taken every 30 minutes during a working week (5 days). The team in question is composed of 34 researchers, hence the matrix is (34×34), weighted and symmetric.

4.2.5 Prison network

The "Prison" network (Zachary, 1977) is a matrix representing the friendship between prisoners. In fact 67 prisoners were asked "What fellows on the tier are you closest friends with?". The responses were thus binary (1 if a prisoner considers another as close, 0 otherwise), the matrix being unweighted. As "closeness" can be non-reciprocal, the matrix is non-symmetric and (67×67).

4.2.6 Tailor shop network

The "Tailor shop" network (MacRae, 1960) pictures the interactions between employees of a tailor shop in Zambia. This dataset is composed of two types of interactions depicted by two different matrices. Firstly, we can find an "instrumental" interactions (assistance related) matrix exclusively composed of binary values. Secondly, we can find a "friendship" interactions matrix, which is binary as well. Moreover, those matrices were recorded twice in the same company in a 7 months interval; Hence, 4 matrices can be found in the "Tailor shop" dataset. Since 39 employees were working in this tailor shop, the relationship matrices are square and (39×39).

4.2.7 Karate club network

This dataset (Zachary, 1977) depicts the extra-sportive members' interactions of a karate club, composed of 34 affiliates. The elements of the matrix corresponds to the number of interactions occurring outside the club during a certain period of time. For those reason, the matrix is symmetric and (34×34).

4.3 Parameters

The two parameters used throughout the experiments are the following: used in litterature

1. Regarding the *KRW* and *IKRW* the value λ tuning the killing of random walkers was set to 0.35, following results driven from Van de Kerkhove.
2. Regarding the *MIA* algorithm, threshold θ for the selection of *MIP*'s was set to 0.1, following the common value used in literature.

Now that the networks are set and defined and that the algorithms are presented (chapter 3); the goal of the next chapter is to present and discuss the results obtained, in terms of both **effectiveness** and **efficiency**.

Chapter 5

Results and Analysis

As stated previously, the different algorithms have been tested on several real networks, ranging from 34 to 67 nodes. You may refer to chapter 4 to get all the specifications of those networks. Tests have been performed on a 2.0 GHZ Intel Core i7-4750HQ processor, with the statistical computer language R, from which two specific packages have been used: *R.matlab* and *igraph*.

All further developed datasets are compared according to two complementary criteria: the *effectiveness* and the *efficiency*.

The further analyses are carried out dataset by dataset according to the pre-cited specifications; each dataset outcome (one for the *IC* and one for the *LT*) will be analyzed thanks to three complementary elements:

First of all, a graph depicting the final diffusions of coverage (influence spread) with different seed sets sizes, will be produced. The vertical axis represents the *coverage*, while the horizontal axis depicts the size of the seed set selected. One can notice that the maximum value that the vertical axis can reach is the size of the initial matrix. Achieving that value would mean a 100% *diffusion*.

Then, a table that shows exact coverage, i.e. *effectiveness*, for specific seed set sizes will be shown. This is simply a numerical representation of the previous graph (yet containing less information).

Eventually, the *efficiency* will be highlighted in a table that contains the time required to select the seeds for each of the algorithms, for a seed set size specified in the previous table.

List of abbreviations used in this chapter

Between Betweenness Centrality algorithm. viii, 29

CELF Cost-effective lazy forward algorithm. vii, 18

Greedy Monte-Carlo Greedy algorithm. vii, 16

IKRW Iterative Killed-Random Walk method. viii, 28

IKRW-adj IKRW with cost matrix based on adjacency matrix A. 29

IKRW-P IKRW with cost matrix based on transition matrix T. 29

KRW Killed-Random Walk method. vii, 24, 25, 27

KRW-adj KRW with cost matrix based on adjacency matrix A. 26

KRW-P KRW with cost matrix based on transition matrix T. 26

MIA Maximum Influence In-Arborescence algorithm. vii, 19

5.1 Karate Dataset (34×34 matrix)

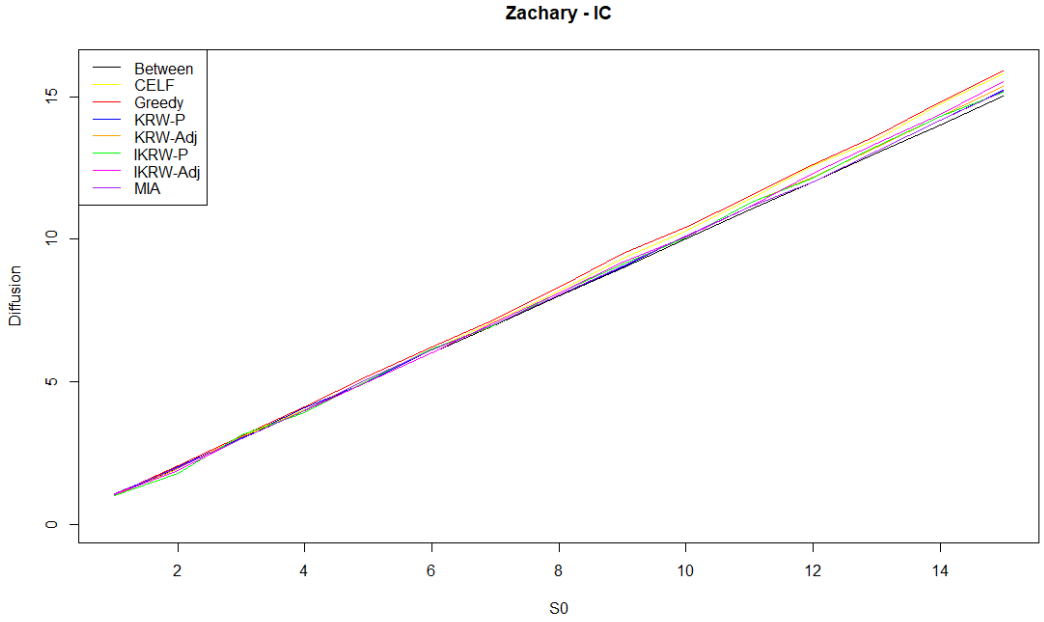


FIGURE 5.1: Karate: Diffusion of the information, with respect to the size of the seed set for the IC model

Between	CELF	Greedy	KRW-P	IKRW-P	MIA	KRW-adj	IKRW-adj
15.0168	15.7985	15.9120	15.2358	15.1671	15.1896	15.3597	15.5168

TABLE 5.1: Karate: Diffusion of information for a seed set of size 15, for the IC model

Between	CELF	Greedy	KRW-P	IKRW-P	MIA	KRW-adj	IKRW-adj
0.692s	66.068s	123.378s	0.087s	0.247s	1.379s	0.037s	0.168s

TABLE 5.2: Karate: Computation time required to run the algorithms, for the IC model

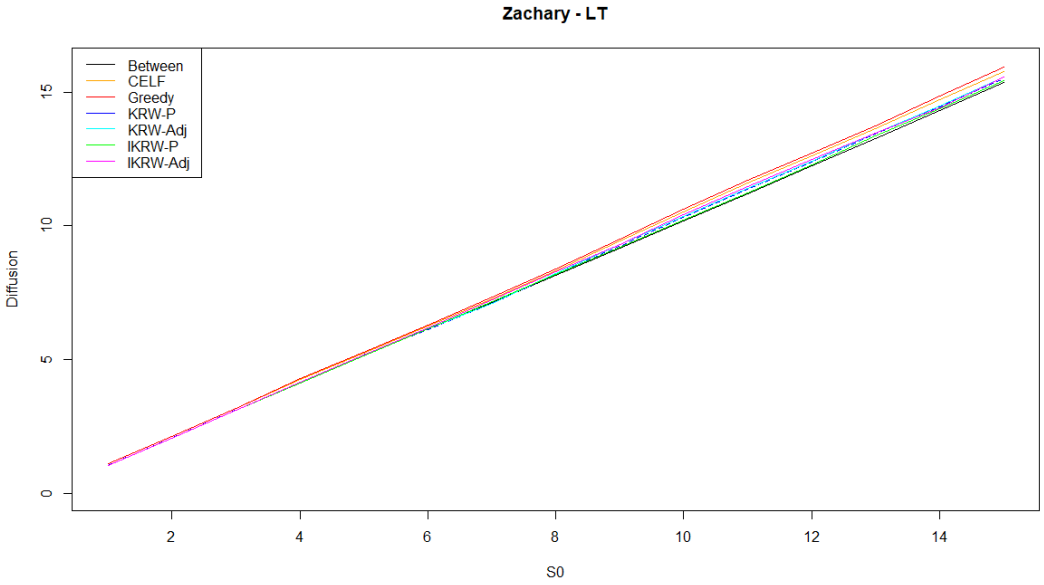


FIGURE 5.2: Karate: Diffusion of the information, with respect to the size of the seed set for the LT model

Between	CELF	Greedy	KRW-P	IKRW-P	KRW-adj	IKRW-adj
15.3514	15.7658	15.9335	15.4845	15.4358	15.5636	15.5601

TABLE 5.3: Karate: Diffusion of information for a seed set of size 15, for the LT model

Between	CELF	Greedy	KRW-P	IKRW-P	KRW-adj	IKRW-adj
0.692s	59.143s	115.678s	0.047s	0.349s	0.037s	0.168s

TABLE 5.4: Karate: Computation time required to run the algorithms, for the LT model

5.2 Fraternity Dataset (58X58 matrix)

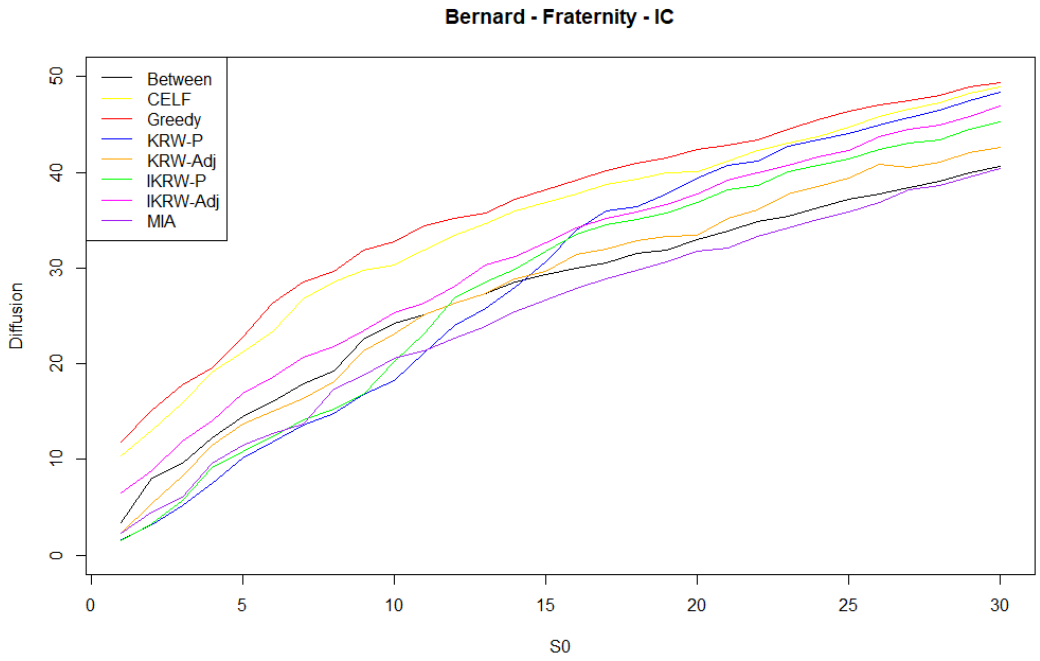


FIGURE 5.3: Fraternity: Diffusion of the information, with respect to the size of the seed set for the IC model

Between	CELF	Greedy	KRW-P	IKRW-P	MIA	KRW-adj	IKRW-adj
40.5656	48.4200	49.3248	48.3342	45.3026	40.4212	42.6476	46.8752

TABLE 5.5: Fraternity: Diffusion of information for a seed set of size 30, for the IC model

Between	CELF	Greedy	KRW-P	IKRW-P	MIA	KRW-adj	IKRW-adj
5.295s	600.648s	1555.998s	0.146s	1.793s	23.782s	0.114s	1.146s

TABLE 5.6: Fraternity: Computation time required to run the algorithms, for the IC model

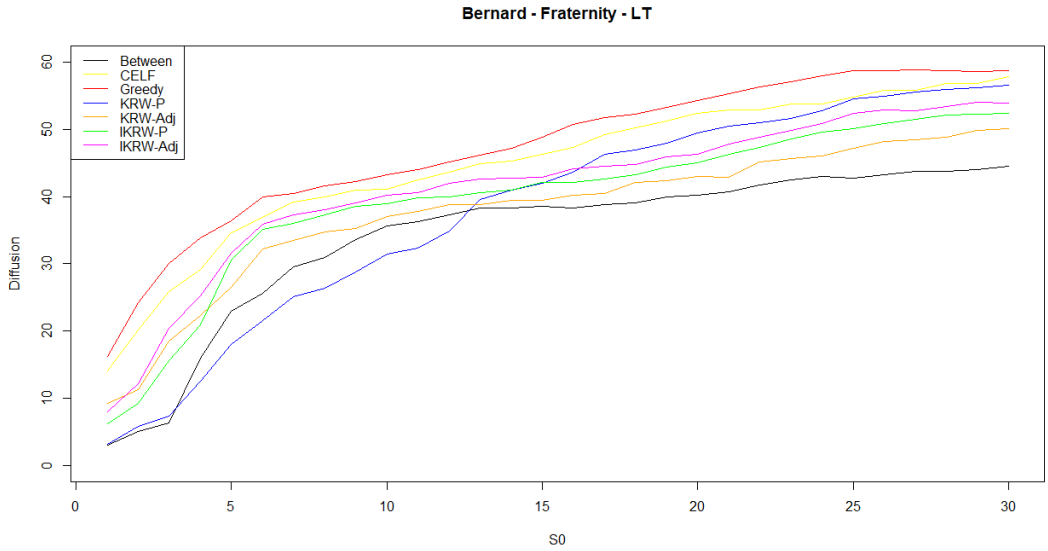


FIGURE 5.4: Fraternity: Diffusion of the information, with respect to the size of the seed set for the LT model

Between	CELF	Greedy	KRW-P	IKRW-P	KRW-adj	IKRW-adj
44.5028	57.8540	58.7804	56.6052	52.6310	50.1676	53.8694

TABLE 5.7: Fraternity: Diffusion of information for a seed set of size 30, for the LT model

Between	CELF	Greedy	KRW-P	IKRW-P	KRW-adj	IKRW-adj
1.168s	1.234s	1.779s	0.146s	1.793s	0.114s	1.146s

TABLE 5.8: Fraternity: Computation time required to run the algorithms, for the LT model

5.3 Office Dataset (40X40 matrix)

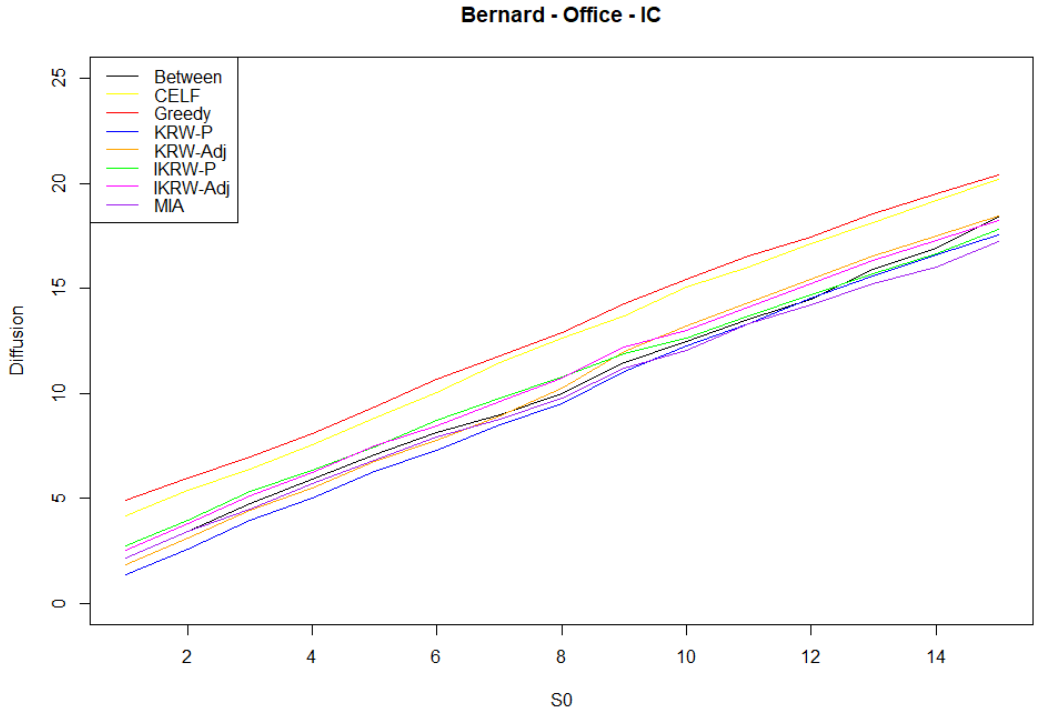


FIGURE 5.5: Office: Diffusion of the information, with respect to the size of the seed set for the IC model

Between	CELF	Greedy	KRW-P	IKRW-P	MIA	KRW-adj	IKRW-adj
18.3878	20.1812	20.3948	17.5606	17.8022	17.2078	18.4494	18.2134

TABLE 5.9: Office: Diffusion of information for a seed set of size 15, for the IC model

Between	CELF	Greedy	KRW-P	IKRW-P	MIA	KRW-adj	IKRW-adj
2.018s	125.765s	355.146s	0.069s	0.693s	3.619s	0.047s	0.540s

TABLE 5.10: Office: Computation time required to run the algorithms, for the IC model

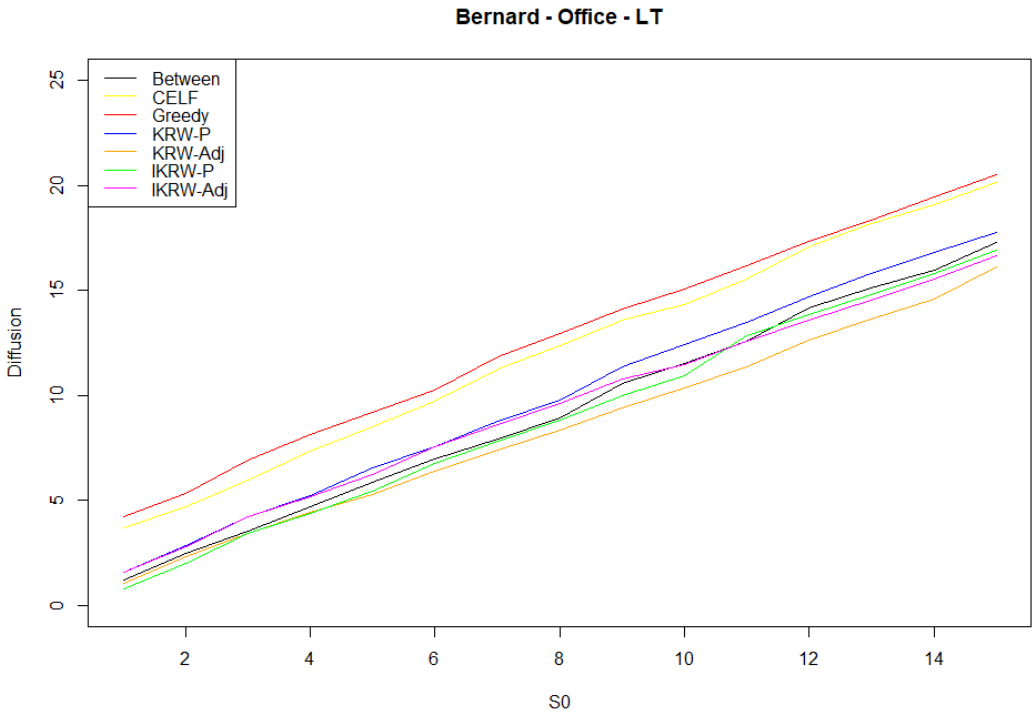


FIGURE 5.6: Office: Diffusion of the information, with respect to the size of the seed set for the LT model

Between	CELF	Greedy	KRW-P	IKRW-P	KRW-adj	IKRW-adj
17.2646	20.1309	20.5183	17.7571	16.8985	16.1208	16.6304

TABLE 5.11: Office: Diffusion of information for a seed set of size 15, for the LT model

Between	CELF	Greedy	KRW-P	IKRW-P	KRW-adj	IKRW-adj
0.971s	71.277s	128.185s	0.069s	0.693s	0.047s	0.540s

TABLE 5.12: Office: Computation time required to run the algorithms, for the LT model

5.4 Radio Dataset (44X44 matrix)

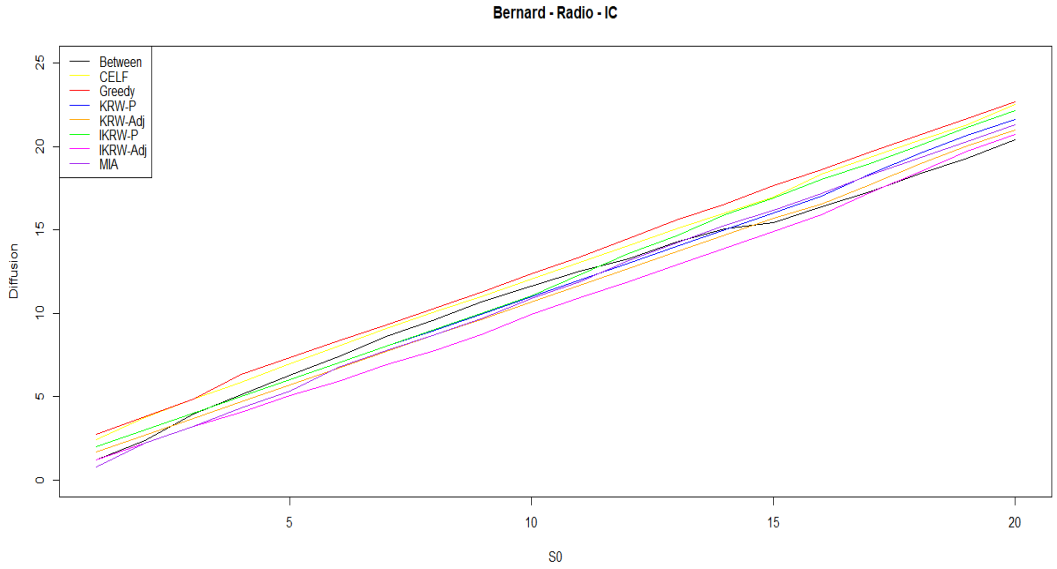


FIGURE 5.7: Radio: Diffusion of the information, with respect to the size of the seed set for the IC model

Between	CELF	Greedy	KRW-P	IKRW-P	MIA	KRW-adj	IKRW-adj
20.3786	22.5112	22.6942	21.6278	22.140	21.3118	21.0102	20.6996

TABLE 5.13: Radio: Diffusion of information for a seed set of size 20, for the IC model

Between	CELF	Greedy	KRW-P	IKRW-P	MIA	KRW-adj	IKRW-adj
1.128s	167.694s	22.498s	0.051s	0.471s	1.053s	0.052s	0.465s

TABLE 5.14: Radio: Computation time required to run the algorithms, for the IC model

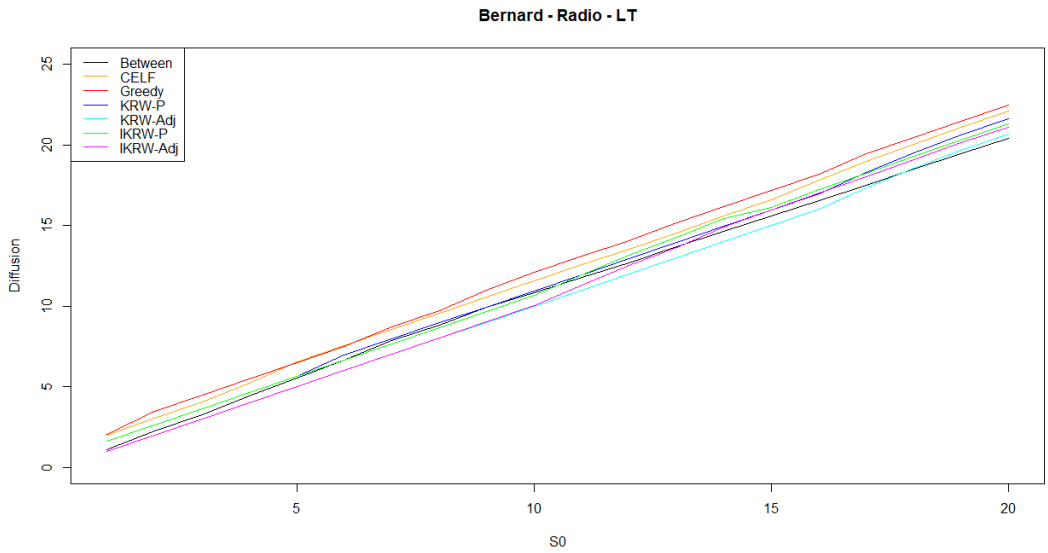


FIGURE 5.8: Radio: Diffusion of the information, with respect to the size of the seed set for the LT model

Between	CELF	Greedy	KRW-P	IKRW-P	KRW-adj	IKRW-adj
20.3915	22.0806	22.4795	21.6008	21.3084	20.6459	21.1132

TABLE 5.15: Radio: Diffusion of information for a seed set of size 20, for the LT model

Between	CELF	Greedy	KRW-P	IKRW-P	KRW-adj	IKRW-adj
2.606s	136.965s	198.499s	0.051s	0.471s	0.052s	0.465s

TABLE 5.16: Office: Computation time required to run the algorithms, for the LT model

5.5 Prison Dataset (67X67 matrix)

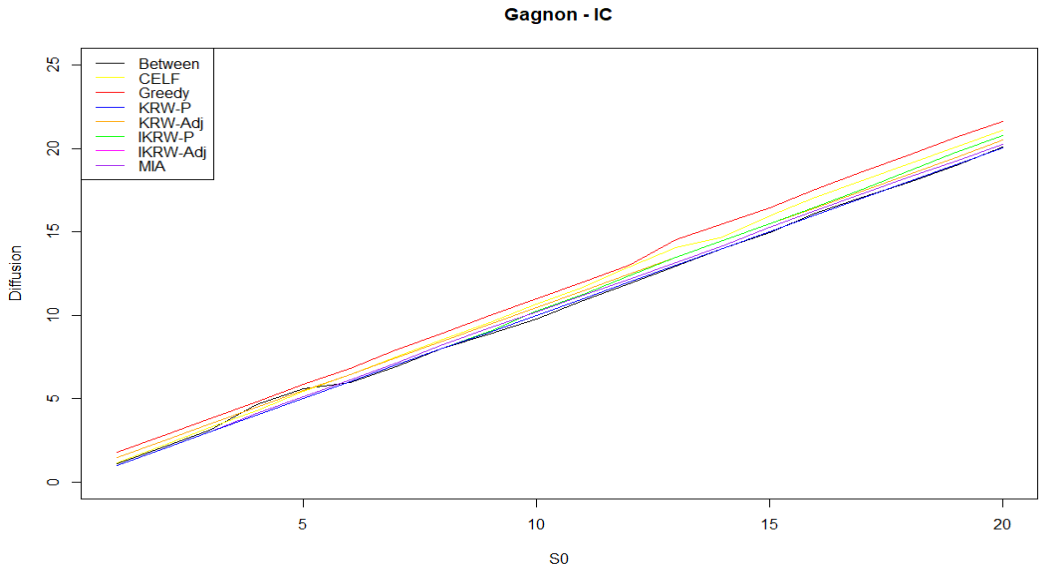


FIGURE 5.9: Prison: Diffusion of the information, with respect to the size of the seed set for the IC model

Between	CELF	Greedy	KRW-P	IKRW-P	MIA	KRW-adj	IKRW-adj
20.0829	21.0768	21.6426	20.0192	20.7892	20.2628	20.4832	20.0180

TABLE 5.17: Prison: Diffusion of information for a seed set of size 20, for the IC model

Between	CELF	Greedy	KRW-P	IKRW-P	MIA	KRW-adj	IKRW-adj
0.953s	260.206s	512.519s	1.132s	1.184s	1.952s	0.771s	0.962s

TABLE 5.18: Prison: Computation time required to run the algorithms, for the IC model

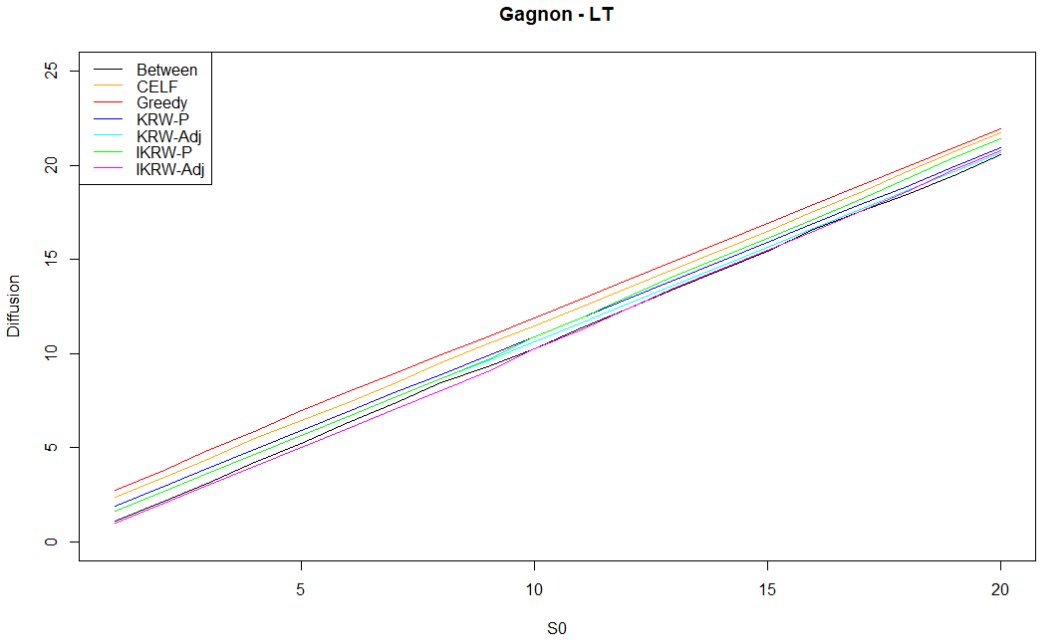


FIGURE 5.10: Prison: Diffusion of the information, with respect to the size of the seed set for the LT model

Between	CELF	Greedy	KRW-P	IKRW-P	KRW-adj	IKRW-adj
20.5408	21.7076	21.9431	20.9122	21.4039	20.6586	20.7664

TABLE 5.19: Prison: Diffusion of information for a seed set of size 20, for the LT model

Between	CELF	Greedy	KRW-P	IKRW-P	KRW-adj	IKRW-adj
0.953s	51.636s	68.490s	1.132s	1.184s	0.771s	0.962s

TABLE 5.20: Prison: Computation time required to run the algorithms, for the LT model

5.6 Tailor Dataset (39X39 matrix)

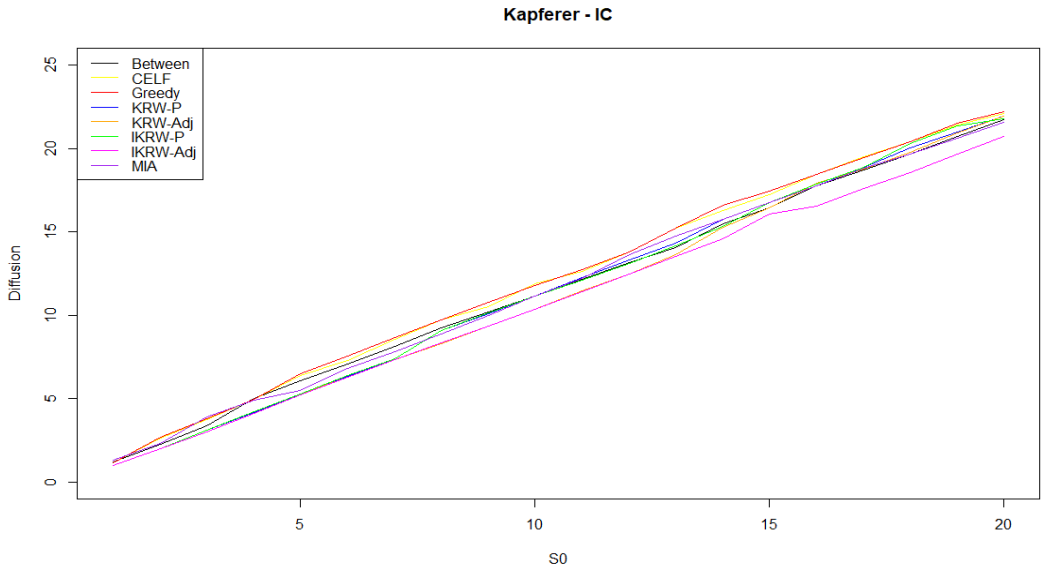


FIGURE 5.11: Tailor: Diffusion of the information, with respect to the size of the seed set for the IC model

Between	CELF	Greedy	KRW-P	IKRW-P	MIA	KRW-adj	IKRW-adj
21.7146	22.1224	22.1796	21.9372	21.7808	21.5720	21.9248	20.7476

TABLE 5.21: Tailor: Diffusion of information for a seed set of size 20, for the IC model

Between	CELF	Greedy	KRW-P	IKRW-P	MIA	KRW-adj	IKRW-adj
1.772s	198.570s	470.016s	0.469s	0.5618s	3.250s	0.861s	0.779s

TABLE 5.22: Tailor: Computation time required to run the algorithms, for the IC model

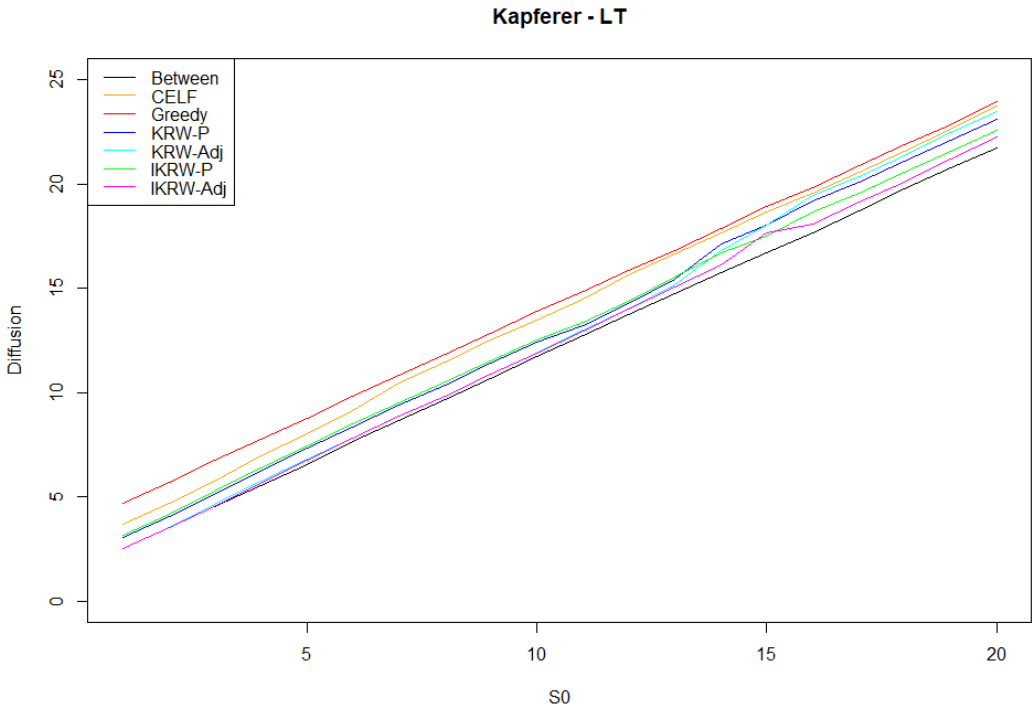


FIGURE 5.12: Tailor: Diffusion of the information, with respect to the size of the seed set for the LT model

Between	CELF	Greedy	KRW-P	IKRW-P	KRW-adj	IKRW-adj
21.7483	23.7524	23.9325	23.0915	22.5815	23.4729	22.2583

TABLE 5.23: Tailor: Diffusion of information for a seed set of size 20, for the LT model

Between	CELF	Greedy	KRW-P	IKRW-P	KRW-adj	IKRW-adj
0.903s	0.322s	1.736s	0.469s	0.562s	0.861s	0.779s

TABLE 5.24: Tailor: Computation time required to run the algorithms, for the LT model

5.7 Analysis of the results

From the results obtained in the last sections, we can observe that the differences in *efficiency* and *effectiveness* greatly differ from one network to another. As a reminder, on the first hand, *effectiveness* corresponds to the total number of nodes activated at the end of the process; on the second hand, *efficiency* expresses the time (in seconds) required to find the set of seeds for each method.

5.7.1 Effectiveness

For some specific networks, *effectiveness* varies a lot depending on the algorithm used, while for some other networks we barely see any difference. In particular, the ***Office*** and especially ***Fraternity*** networks highlight huge differences; while Karate, Radio, Prison and Tailor do not. Therefore, we conclude that the typology of the network itself is a determinant factor regarding diffusion analysis and spread maximization. In fact, when we compare the intrinsic networks, one can notice that most of them are rather *sparse*, while Fraternity and Office are *denser*, meaning that the two latest networks have a large number of arcs between nodes. This can explain most of the differences between networks in terms of *effectiveness*.

Based on the analyses carried out on the two datasets highlighting larger differences among algorithms and as a general conclusion regarding the *effectiveness*, the *Greedy* and the *CELF* have, as expected, provided the best results. Those algorithms are followed by the *KRW* methods (*KRW-P* and *KRW-adj*) aswell as the *IKRW* methods . One can also observe that *MIA* and *Between* methods are worse off the *KRW*'s for both *LT* and *IC* diffusion models on the relevant datasets.

5.7.2 Efficiency

In terms of *efficiency*, algorithms can be more easily compared, since computation time greatly differs in a similar way for all datasets; same conclusions can be taken out of each test.

First of all, one can notice that the *CELF* and especially the *Greedy* algorithms take much more time to compute than *KRW*'s methods. As an example, the computation time required to find a seed set thanks to the

Greedy method in the Fraternity dataset with diffusion model *IC* is 10650 times greater than the computation time required to get the seed set from *KRW-P*. In the case of the present thesis, the time was still "humanly relevant", but as *complexity* (hence time) increases greatly with the size of the network, the slowness of the *CELF* and *Greedy* will become a serious issue of feasibility.

5.7.3 Conclusion of the results

In light of the previous comments, we suggest to investigate on the *scalability* of different algorithms presented using *larger networks*.

As *Greedy* and *CELF* does not seem to be useful on large datasets (since they require too much time to be computed on real large-scale networks), we would suggest not to waste resources on those.

As previously said, we would suggest to test the algorithms on *denser datasets* in order to obtain more differentiated results in terms of *effectiveness*. We would indeed need more data to confirm the insight deduced from the Office and Fraternity datasets: *KRW*-like algorithms (KRW and IKRW) seem to work pretty well in terms of *effectiveness*, while being way better than their "competitors" (Greedy and CELF) in terms of *efficiency*. If our hypotheses are right, we are convinced that those algorithms could be useful on real and large-scale datasets in the near future, e.g. for marketing purposes.

Chapter 6

Influence versus Adoption, Profit and Costs

In the previous chapters, we have tried to approach the influence spread maximum; however marketers and sales departments of companies are rather more interested in product adoption rate, profit and cost minimizing, which are not especially equated with influence maximization. Some recent studies have covered these subjects, we will first describe in this chapter the maximization of adoption, then we will focus on studies related to maximizing the profit in the context of viral marketing; and eventually, we will cover the subject of advertising's costs minimization.

6.1 Adoption

6.1.1 Linear Threshold with Colors (LT-C)

So far, we have assumed that when a user in a social network is influenced by another user about a special product or innovation, he will adopt it and even encourage his own friends and acquaintances to further adopt it too.

In reality however, when a user is influenced by a friend about a special item he still can choose not to buy it. If he still buys it, he can decide not to share an opinion with his friends, that is, stopping the influence spread. In some cases, a user can receive endorsement from a friend and decide not to buy it (because he already has a similar product, or does not want to spend money on that at the moment), but still propagates the influence of that innovation to his own network (Chen, Lakshmanan, and Castillo, 2014).

All these adjustments are developed by *Bhagat et al.* in their **Linear Threshold model with Colors (LT-C)** (Bhagat, Goyal, and Lakshmanan, 2012). It is an extension of the *LT-model* (see section 2.3), where there are more states than the dual choice active-inactive; in this model, 6 possible states are foreseen: *inactive, active, adopt, tattle, promote* and *inhibit*. While a user has not been influenced about a product he is in the *inactive* state. Once influenced by his neighbors, a node v becomes *active* if the net influence of node $v's$ in-neighbors are above its own threshold, i.e. if $f_v(S_t) \geq \theta_v$. In the case of *recommendations*, we can compute $f_v(S_t)$ by

$$f_v(S_t) = \frac{\sum_{u \in S_t} w_{u,v}(r_{u,i} - r_{min})}{r_{max} - r_{min}},$$

where r_{max} and r_{min} represent the best and worst ratings, respectively.

The influenced users have enough information to have their own opinion on the product and can choose either to *adopt* it, or to *tattle* on it. Tattling can be positive (*Promote* state) or negative (*inhibit* state). All those states can be seen below (figure 6.1)

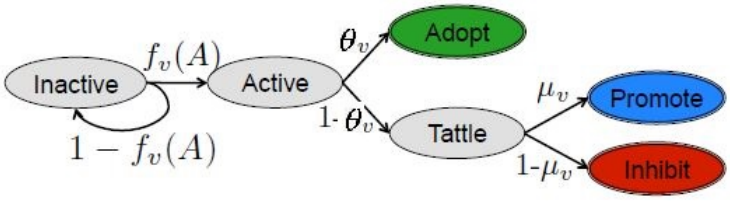


FIGURE 6.1: Linear Threshold with colors model (taken from Bhagat, Goyal, and Lakshmanan, 2012)

Unlike the standard LT model, $\sigma(S_0)$ does not represent the influence spread anymore, but rather denotes the *coverage*, the expected number of nodes in the "adopt" state. The goal of this problem is thus finding the k nodes that maximize $\sigma(S_0)$ under the LT-C model. Bhagat et al. (Bhagat, Goyal, and Lakshmanan, 2012) have proved that $\sigma()$ is still submodular and monotone in the LT-C model. Hence, the greedy algorithm of section 3.2.1 can be used to find an approximation of the optimal solution $(\sigma(S^*))$, with the same optimality guarantees.

6.1.2 Scheduled seeding

Sela et al. (Sela et al., 2016) have shown the importance of *timing* in order to increase the adoption's rate of products. In fact, when it comes to product adoption, the peer influence is a latent process, and is most of the time insufficient to trigger the buying of the product. Thus, additional sales effort are required to sell the products. For this reason, they created a method for seeding the best node at the best point in time in order to use the fixed budget B in the best way possible: the **scheduled seeding**. The advertising is working as follows: At each time step, one user u is chosen by the company, which suggests him to buy their product. Of course, not all (potential) customers can be selected; the next paragraph clarifies which users can be targeted at a particular moment.

Four different stages are modeled by Sela et al.: first a user is *non-infected* and stands as such as long as he does not buy the product. Once he decides to buy the product, he becomes *infected and infectious* for the next t^{inf} timesteps. That is, he is now able to influence its neighbors to buy the product, but only for a certain period of time. After that period, he remains *infected*, but is now *non-infectious*. If a customer refuses an offer, he will not accept further ones for a certain period of time, since this will annoy him. This period is called a *cooling time*.

The probability that user u adopts the suggested product, denoted by p_u depends on two factors: the inherent features of the product and the adoption rate of u's neighbors. Let p_{max} be the inner characteristics of a product (price, quality, features etc.) and p_{soc} be the dependency of u to his neighbors. Formally, p_u can be computed by

$$p_u = p_{max} \times p_{soc}(u),$$

where $p_{soc}(u)$ is defined by $\frac{min(\theta_u, |N_u^+|)}{\theta_u}$. $|N_u^+|$ represents the number of infected and infectious neighbors of u, while θ_u represents the threshold (defined as a number of infectious neighbors) above which user u adopts the product. As the reader can observe, the definition of θ_u differs from the one of the LT model in section 2.3.

Finally, we have now the whole bunch of elements for choosing our seed set; at each time step, one will add the uninfected node u with the highest SC

score. Those scores are computed as follows:

$$SC_u = p_u \times [1 + \Sigma_{v \in N_u^-} p_v * |N_v^-| + (1 - p_v) * |N_v^-| * ln(t^{ct})]$$

6.2 Profit maximization

In this section, we can find a method described by Buchbinder et al., which does not require the presetting of a budget for the number of seeds selected (Buchbinder et al., 2012). Instead, it creates a trade-off between the rewards of influence spread and the costs of seed selection.

Even though the *submodularity* property remains true, this method is thoroughly different from the ones that maximize the spread of influence. Unlike the influence maximization function, the profit maximization function is not *monotone*. For this reason, the computation of profit maximization is much more challenging than that of influence maximization.

Buchbinder et al. suggested a method for easing the approximation of these functions, the *double greedy* algorithm (Buchbinder et al., 2012). In addition to this method, in order to prune the size of the set to explore for finding the seeds, Tang et al. presented the *IterativePrune* algorithm (Tang, Tang, and Yuan, 2016). The idea behind is to make an interval with an upper bound and a lower bound, which contains the nodes of the optimal solution. By progressively reducing the size of those intervals, they finally have less vertices to test when looking for the best set of seed nodes.

Let us define the problem in more details:

- On the one hand, a cost C_v is associated to each node $v \in V$, this represents the cost that a company has to pay for selecting that node in the seed set.
- On the other hand, the benefits of a seed set are calculated by the number of activated nodes $\sigma(S_0)$, assuming that each node offers the same profit.

The main goal of this method is to maximize the profit of a seed set, $\varphi(S_0)$, which can be computed as the difference between the influence spread and

the costs associated to the seed set selected:

$$\varphi(S_0) = \sigma(S_0) - C_{S_0},$$

where C_{S_0} is the sum of costs of all seeds selected.

Let $\sigma(v|S_0) = \sigma(S_0 \cup \{v\}) - \sigma(S_0)$ be the marginal influence gain of adding node v to seed set S_0 and let $\varphi(v|S_0) = \varphi(S_0 \cup \{v\}) - \varphi(S_0)$ be the marginal profit gain of adding node v to seed set S_0. Tang et al. have proved that when the influence function is submodular, then the profit function is also submodular. However, since the marginal profit gain of adding a new node can be negative, $\varphi(S_0)$ is not monotone. Hence, the present case is an unconstrained submodular maximization problem. (Tang, Tang, and Yuan, 2016)

Buchbinder et al. developed an algorithm in order to cope with unconstrained submodular maximization problems: the **double greedy** algorithm. In their model, they make the strong assumption that the profit of selecting all nodes is non-negative[1].

First we initiate S_0 to the empty set $\emptyset$ and T to the entire node set V. Iteratively and for each node $u \in V$ we have to decide whether or not adding it to the seed set S_0, and whether or not removing a node to the set T. The decision of adding u to S_0 is taken only if its addition to S_0 generates higher marginal profit gain, than the marginal profit loss to T when quitting that set and vice versa. Algorithm 12 describes these explanations more rigorously.

Under the strong assumption that selecting all nodes of the network as seed set is profitable, i.e. $\varphi(V) \geq 0$ (which is of course not plausible in the context of viral marketing), Buchbinder et al. have proved that the profit of the seed set satisfies some approximation guarantees:

$$\varphi(S_0) \geq (1/3) \max_{s \subseteq V} \varphi(S)$$

The hard condition above should be subverted by pruning the search space from V to smaller lattices. First, let us define two node sets A_1 and

[1] In practice, it is not the case, but we will see later on how to counter this assumption

Algorithm 12 Deterministic Double Greedy algorithm

Inputs: set of nodes V, Profit function φ
Output: Seed set $S_0 = T$

initialize $S_0 \leftarrow \emptyset$; $T \leftarrow V$
for each node $u \in V$ **do**
  $r^+ \leftarrow \varphi(u|S_0)$
  $r^- \leftarrow -\varphi(u|T \setminus \{u\})$
  if $r^+ \geq r^-$ **then**
    $S_0 \leftarrow S_0 \cup \{u\}$
  else
    $T \leftarrow T \setminus \{u\}$
  end if
end for
return S_0

B_1:

$$A_1 = \{v : \varphi(v|V\setminus\{v\}) > 0\}$$

$$B_1 = \{v : \varphi(v|\emptyset) \geq 0\}$$

Thanks to the submodularity property of the profit function, one can find that $A_1 \subset B_1$. The first lattice, defined by $L_1 = [A_1, B_1]$ "contains all global maximizers S^* for the profit function" (Tang, Tang, and Yuan, 2016). One can prune the current lattice even further using some recursions by restricting B_1 to $B_2 = \{v : \varphi(v|A_1) \geq 0\}$ and A_1 to $A_2 = \{v : \varphi(v|B_1\setminus\{v\}) > 0\}$. The result is thus the new lattice $L_2 = [A_2, B_2] \subseteq L_1$. Those restricting operations can be repeated until both A and B remain unchanged, i.e. when no further pruning is possible. The last lattice obtained $L^* = [A^*, B^*]$ still holds the global maximum. Mathematically, we have

$$A_t \subseteq A_{t+1} \subseteq A^* \subseteq S^* \subseteq B^* \subseteq B_{t+1} \subseteq B_t.$$

Furthermore, $\varphi(A_t)$ and $\varphi(B_t)$ are non-decreasing in t; hence one can use the double greedy algorithm starting with the lattice L^*. Though the conditions are much weaker than the former ones ($\varphi(A^*) + \varphi(B^*) > 0$ instead of $\varphi(V) > 0$), the guarantee $\varphi(S_0) \geq (1/3)\max_{s\subseteq V}\varphi(S)$ stated before remains consistent.

Algorithm 13 IterativePrune algorithm

Inputs: set of nodes V, profit function φ
Output: Reduced set of nodes $[A_t; B_t]$

while $A_t \neq A_{t-1}$ and $B_t \neq B_{t-1}$ **do**
$A_{t+1} \leftarrow \{v : \varphi(v|B_t \setminus \{v\}) > 0\}$
$B_{t+1} \leftarrow \{v : \varphi(v|A_t) \geq 0\}$
$t \leftarrow t + 1$
end while
return A_t and B_t

6.3 Minimizing the budget with guaranteed spread

The main goal of this method is to select the smallest seed set in order to reach a pre-defined *coverage* (spread of influence), with guaranteed probability. The idea behind this methodology is that any *topic* needs to be discussed by a minimal number of people before becoming a *hot topic*, which in turn can spread virally on social or regular media. In practice, this can be highly useful for marketers, since they could know how much they have to invest (e.g. how many free samples to offer) in order for their product to be used/known by the targeted number of final users, with a certain probability guarantee. This model is called by Zhang et al. (Zhang et al., 2014) the *seed minimization with probabilistic coverage guarantee (SM-PCG)*.

Please note that in their paper, although they presented it along with the *IC* model, it is applicable with any monotone and submodular diffusion function.

Zhang et al. have defined their model as follows: For a given social graph $G(V, E)$ with associated p_{uv} on edges, a minimal target set V, a *coverage threshold* $\eta < |V|$ and a *probability threshold* $P \in [0, 1]$; the aim is to find the seed set of minimal size S^* in such a way that S^* activates at least η nodes in V with probability P. Formally, we have

$$S^* = \underset{S_0 : Pr(\sigma(s_0) \geq \eta) \geq P}{\operatorname{argmin}} |S_0|$$

The specificity of this model lies in the fact that it considers influence spread with *probabilistic* guarantees instead of guarantees of *expected* influence spread (Zhang et al., 2014).

As for the *Independant Cascade model*, one needs to assess the *influence coverage* via *Monte-Carlo* simulations. But in addition, one needs here to estimate the the probability guarantees of coverage, i.e. $Pr(\sigma(S) \geq \eta)$, where η is the critical threshold that the marketer wants to reach: the *tipping point* (Zhang et al., 2014). Starting from a given seed set S_O, Zhang et al. simulates the diffusion process R times and and estimate the probability by the fraction of runs in which the coverage is at least equal to the threshold η. This paragraph is summed up formally in pseudo-algorithm 14.

Algorithm 14 MC-CompProb[R] algorithm

Inputs: $G(V, E)$, Influence Probability P_{uv}, V, S_0, η
Output: Coverage probability $Pr(\sigma(S_0) \geq \eta)$

initialize $t \leftarrow 0$
for $i = 1$ to R **do**
  Simulation of diffusion with seed set S_0
  $N_i \leftarrow$ number of active nodes in V
  if $N_i \geq \eta$ **then**
    $t \leftarrow t + 1$
  end if
end for
return $\hat{P} = t/R$

Let $\hat{P}$ be the estimate of the true value $P = Pr(\sigma(S_0) \geq \eta)$, found thanks to algorithm 14. The authors have proved that it is sufficient to run $R \geq ln(2n^{\delta})/2\epsilon^2$ iterations of the Monte-Carlo in order to guarantee an error of at most ϵ: $|\hat{P} - P| \leq \epsilon$

Unlike the classical *IC model*, influence function here is *not submodular*. For that reason, one has to find the best set of seeds S_0 in another way as previously. Algorithm 15 highlights the different stages required to approximate the optimal set S^*. The first step of the loop is the selection of the seed u having the highest marginal increase in *expected* influence spread, w.r.t. S_{i-1}. This node u is added to the existing seed set S_{i-1} in order to create S_i. Then, we compute the *coverage probability*, thanks to algorithm 14. If this value is

higher than the sum of the fixed probability threshold plus a certain parameter $\varepsilon \in [0, (1-P)/2]$, the loops stops since we have found a seed set of minimal size.

Algorithm 15 MinSeed-PCG[ε] algorithm

Inputs: $G(V, E)$, Influence Probability P_{uv}, V, η, P
Output: Seed set S_0

Initialize $S_0 \leftarrow \emptyset$
for $i = 1$ to n **do**
  $u \leftarrow \text{argmax}_v \left\{ \hat{E}\big[\sigma(S_{i-1} \cup v)\big] - \hat{E}\big[\sigma(S_{i-1})\big] \right\}$
  $S_i \leftarrow S_{i-1} \cup \{u\}$
  $Prob \leftarrow CompProb()$
  if $prob \geq P + \varepsilon$ **then**
    return S_i
  end if
end for

Chapter 7

Marketing stakes of social media and their influence spread

7.1 What is a social media

Throughout this chapter the definition of *social media* will be taken in its broad sense. Hence a social media can be defined as "any website which allows user to share their content, opinions, views and encourages interaction and community building" (Neti, 2011). According to this definition, Facebook, Twitter, Instagram, Youtube, LinkedIn or Pinterest are typical social media.

7.2 Growing importance of advertising on social networks

The explosion of social media during the last decade has given marketers the opportunity to greatly develop their advertising through this channel. Between 2014 and 2019, the budget allocated for digital marketing on social media in the US is supposed to skyrocket; it moved from 7,52 billion USD in 2014 to an expected 17,34 billion in 2019 (Statista, 2017a).

The huge budget that firms are prone to spend on social media has to be allocated in a thoroughly thought manner; in fact, the several platforms offer different types of messages, which are related to different costs: On *Twitter* for example, companies can only make promotion of their products in short messages of maximum 140 characters (called Tweets) which appear on the timelines of brands' followers. On *Facebook* however, companies' messages can be larger, be displayed as videos such as testimonials. Followers can choose to make comments on the products, so that their "*friends*" can see the advertisements. When it comes to advertising, this platform is supposed

to be the most profitable for companies. Facebook is indeed today (statistics from January 2017) the most used social media platform in the world, with more than 94% of marketers worldwide using it to market their business (Statista, 2017b).

7.3 Reasons for promoting on social media

The reasons for companies to spend time and money on advertising on social media can be split in two categories, depending on the nature of the public targeted: The B2C and B2B markets.

7.3.1 Adverting in B2C

Three of the main reasons for companies working in B2C to spend a certain part of the marketing budget in social media are summed up below:

- Firstly, social media allow firms and not-for-profit organizations to get *in touch* with existing customers and enhance their products so that they better fit their personal needs. Some companies even involve final users in the product development and in the decision making process.
- Secondly, advertising can be really personalized through *targeted promotions* depending on users' interests. Those common interests can be spotted thanks to community detection on social networks (see for example the article of Blondel (Blondel et al., 2008) or the thesis of Vanderlinden (Vanderlinden, 2015)). The personalization can also appear when marketers tackle people following specific events related to the core business of their company. Several tools can be used in order to measure the importance of marketing events (advertisement, promotion video etc.). On Twitter for example, one can use *Twitter Analytics* to know the *engagement* of brand's followers. Engagement is a measure counting each day, the number of *replies*, *retweets*, *mentions* and *favorite* of a Twitter account (SimplyMeasured, 2014). Figure 7.1 shows an example of these statistics available for marketers. By exploiting those data, companies can see "peaks and throughs" and measure the impact of their messages (for instance, which users retweet certain type of messages).

- Finally, B2C companies try to *become visible* and known for people who do not know their brands or their products, that is, improve their *brand awareness*. The present thesis explained formally how companies can become visible to the largest public as possible through information cascades. Some tools, such as *Twitter Analytics* presented just above, can of course provide good measure of the brand awareness and the spread of influence on social networks.

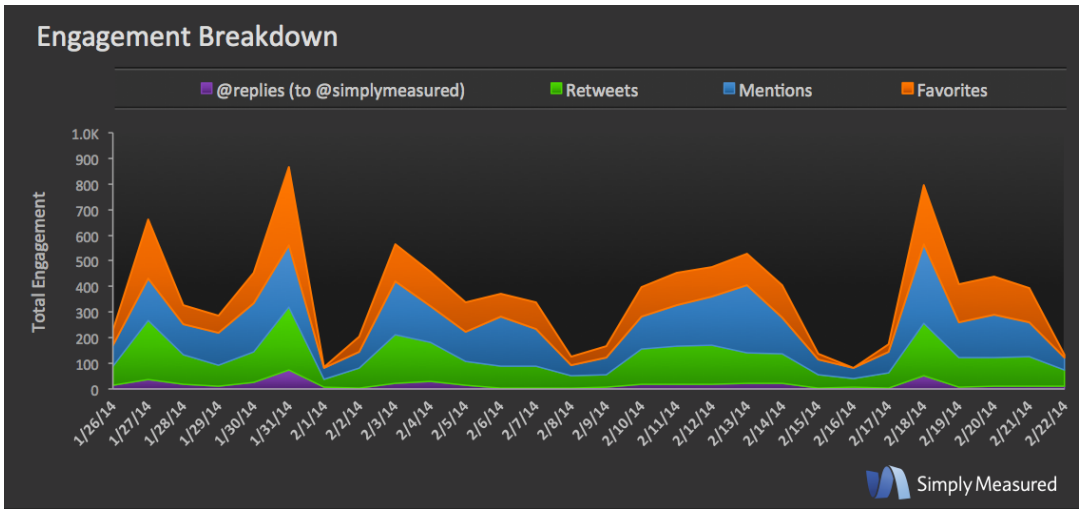


FIGURE 7.1: Engagement evolution of a Twitter page.
source: Simply Measured

7.4 Promotion on the B2B market

On the B2B market, there is also a growing interest from companies to use social media as Cawsey and Rowley highlighted it. According to them, the main goals for B2B companies to engage in social media are improving their *brand image*, extending their *brand awareness* and increase the *customer engagement* (Cawsey and Rowley, 2016).

Moreover, Agnihotri et al. emphasized that *customer satisfaction* in the B2B market increases thanks to the use of social media (Agnihotri et al., 2015): In fact, they found that a salesperson communicating directly with his clients through this channel, positively impacts its own responsiveness to their needs, which in turn improves the customer's satisfaction.

7.5 Applications of influence spread on social networks

In the previous chapter, *Results and Analysis* (chapter 5), we have stated the relative efficiency of different algorithms and showed that new techniques become more and more scalable and precise. However, their usefulness can only be assessed by evaluating their possible real life applications. Indeed, without any application in the harsh world of business, the algorithms are doomed to remain theory.

The present section tries to imagine a bunch of possible applications that could be used in today's business life. As these are just ideas, we hope to give some inspiration to our readers so as to check the validity of our propositions.

The standard application covered in many articles regarding the subject is about direct marketing, meaning that the algorithms are providing us the list of people to advertise to. As this is a quite straightforward and covered application and considering the fact that we are trying to push the border further, we will not discuss the subject.

Let us begin with a summary of the most important results from the previous chapter:

We have concluded that with some proper equipment and settings, tests could run properly on networks in a timespan one could consider as humanly feasible. We therefore take as hypothesis that tests are runnable and efficient, provided that the networks they are based on are accurate. Thus, we consider that the results driven from the presented algorithms can be useful in business. Also, we have seen that the presented algorithms can be tuned in order to take into account other factors such as profitability. In the same way, the network itself can be tuned to correspond to the specificities of the application it will be used for.

With those hypotheses, we show in the underneath subsections that there are numerous applications in several fields.

7.5.1 Events

Consider the following case: You are asked to host an event so as to promote either your company or a specific product. However, out of the 300 people in your network, your budget only allows you to invite 50 of them. Who will be on the guest list of your upcoming event, so as to maximize the visibility of your event?

Behind this rather simple and "school made" problem, some companies already rely on this question. For instance, video game developers have nowadays embraced the benefits of marketing through social networks such as YouTube. The context is the following: video game developers are facing a rise of the costs of developing games, mainly due to the fact that games have to be graphically enhanced through time; hence more and more lines of code are required to fully exploit the hardware capabilities. Therefore, the development teams have to be enlarged with valuable staff (Loftus, 2013). The price of a single game staying relatively stable, the amount of sold games has to rise in order to follow the rise of costs and keep the same level of profitability. This is the reason why video games developers are investing more and more in marketing, and are exploring new marketing techniques.

Lately, one of this new techniques was to invite YouTubers (owner of a famous YouTube channel) to a special event prior to the release date of the game. Each YouTuber is then allowed to record some exclusive footage of the game he or she will be able to upload on his channel, generating views and ultimately some income via the YouTube monetization policy. These videos participate largely in the advertising of the video game and are really cheap compared to some other advertising vectors, such as TV commercials. If one could be able to extract the network generated by the different YouTubers, the usefulness of our "seeds selection" algorithms would become clearer. All the considered YouTubers represent the possible set of seeds, and the algorithm returns the chosen number of seeds so as to maximize influence through the network, ultimately returning the list of YouTubers to invite, maximizing the marketing coverage.

Even if this example is precise, one could imagine to extend this technique to other products and services. YouTubers gather millions of subscribers and could leverage an interesting income over cost ratio. Eventually, we believe that those techniques could become part of a possible mean of advertising

among the offer of marketing companies.

We can also extend the use of influence maximization regarding events. For instance, one could host an event for a specific product and the only way of participating to this event is through an application system linked to a Facebook account. At this point, we make sure that each applicant is sharing its useful information in order to build an efficient network. After the application period is over, running the influence maximization algorithms trough the network composed of all applicants would return the guest list maximizing marketing coverage.

In a complete opposite way, a company could also base the decision to participate to a specific event based on its potential marketing coverage. Consider for example large forums, job fairs, etc. It could also be useful for a famous musician to know which music festivals to attend in order to maximize his coverage.

Those examples are just a small glance of what could be a whole new marketing and coverage technique. What is important to remember is: for a specific occasion for which the number of applicants is higher than the total amount of guests and for which a specific network can be computed, influence maximization techniques will attempt to select the best possible set of guests. With such a broad definition, we hope to be able to trigger the imagination of the reader.

7.5.2 Recruitment

One possible application that is not related to marketing concerns recruitment and head-hunting, especially for high profiles and experts. With the spread of Internet through the world, experts can very easily get in contact with each others and publish results on specific platforms. For instance, *R-bloggers* is a notorious forum regarding the programming language R. Analyzing this forum and extracting a network out of it could lead to the possibility of using the techniques previously described in order to detect the most influential members of the forum. If we take the hypothesis that one person's influence throughout a community of experts is directly linked to her skills, influence maximization techniques could point out the most competent people.

This idea can also be extended to any group of people forming a community, since there is a high chance that one can derive a network from it. We directly see that regardless of the field of expertise, influence maximization can under certain circumstances be used in a recruitment goal.

7.5.3 Social Media Population Screening

In light of recent events, we have seen that a coordinated campaign on social media might be used to influence large groups of people. These techniques have been used to influence, namely, the midterm elections in the U.S.A. taking place this November 2018[1].

In response to those, social medias have decided to delete accounts whose purpose is solely to conduct mass influence. However, it is nearly impossible to detect those fake accounts without an initial use of screening algorithms who will point out possible fake accounts. On the other hand, it is probably unwise to let the decision to delete an account or not to a computer. Therefore, human intervention is still needed at some point in the process so as to decide whether to delete an account or not.

Say, you have a list of X possible fake accounts pointed out by the previous algorithm, and a limited number of *FTEs* to answer that demand (*Full-Time Equivalent*). In order to limit the influence of the fake accounts, one might want to begin with the ones that are the most influential. This is where influence spread algorithms can be useful, since you can consider the accounts listed in X as potentiel seeds for your seed set. Then, by extracting and ordering the marginal influence of each seed, you can easily derive the "potential fake accounts with the highest influence".

Of course, as fake account screening techniques are unknown and unique for each company, the latest idea could be incorporated in some other way who would fit the actual technique. Therefore, we believe this idea is worth considering.

[1]See for example Solon, 2018.

7.5.4 Others

If we broaden our definition of network, there are still some uncovered fields. Consider for example a problem posed by the US Environmental Protection Agency: given a water distribution network and data on how contaminants are spreading through it, what is the best way to allocate sensors so as to detect all possible contaminations?

It has been proven (Leskovec, Adamic, and Huberman, 2007) that influence maximization techniques are more efficient than classical techniques used to address this kind of problems. Similarly, we can extend this result to road traffic. Roads being the edges and crossroads being the vertices, influence maximization could enhance large scale road blocks and could help in finding a fugitive for instance.

Furthermore, the reasoning could be extended for the dispatching of security agents in crowded areas, such as concert halls or department stores.

7.6 Conclusion

We see that there are numerous cases where influence maximization could be useful. Moreover, the number of applications is likely to grow as the popularity of those techniques get larger. However, one strong hypothesis was that we could identify a relevant network to the concerned problem. Where some people might argue that this is an issue, we prefer to say that it represents an opportunity.

As getting access to the network could represent a difficult task for a neophyte, an entity able to leverage some kind of expertise in the field could quickly become leader in a whole new sector. We highly believe in the possibility of making business out of those techniques, and have a strong feeling that they will become popular in a near future and be part of the set of tools used in marketing and other fields.

Chapter 8

Further work and conclusions

In this thesis we have focused our attention on the problem of *influence spread maximization* on social networks, as defined by Kempe et al. (Kempe, Kleinberg, and Tardos, 2003). The idea is the following: knowing the structure of a network and how information is passed inside of it, what are the best k seeds to select in order to maximize the final number of active users?

Then we described two diffusion models, which characterized how information can spread across a network. Two models were developed, namely the *Independent Cascade* model, and the *Linear Threshold* model.

In chapter 3, we have firstly showed that the exact values for the influence spread maximization problem could not be computed. In fact hardness comes from two sources: finding the spread $\sigma(S)$ knowing a seed set S, and finding the seed set S which maximizes $\sigma(S)$. Then we presented some heuristics for approximating the real value of $\sigma(S)$, while easing the computations, i.e. reducing the time complexity. The algorithms presented in this chapter were namely the *Monte-Carlo Greedy*, the *CELF*, the *MIA* algorithm (for the IC model), the *KRW*, the *IKRW* and finally the *Betweenness centrality* method.

Chapter 4 depicts the procedure followed for setting up our networks and to compute the *spreads of influence* of every algorithm. The same chapter covers the different networks chosen for running the algorithms on, are described. In practice, those are small and real networks ranging from 34 to 67 nodes.

Chapter 5 shows the results of our algorithms in terms of both *effectiveness* and *efficiency*, algorithms are compared with each other. Then, we made analyses of those results and discussed some conclusions.

This thesis allowed us to highlight huge differences in terms of results (effectiveness and efficiency) between the different algorithms. The *Killed Random Walk* method achieves good effectiveness compared to the presented competitors and requires less time. For this reason, we strongly believe that one should test it on larger real networks in order to assess its *scalability*.

Chapter 6 highlights the fact that modeling the problem differently would probably give insights on what should be done in the field of online marketing in the future in order to be more profitable for companies. In particular, instead of focusing our attention on influence spread maximization, given a certain number of seeds (i.e. a budget). The authors suggested to maximize the products' adoption or even the profit of the promotion campaign on social networks.

Chapter 7 describes the importance and the reasons of marketing on social media, and their implications in the particular context of influence on social networks. Discussions on what the future of online marketing will/should look like and the probable further developments in the field of spread maximization concludes this chapter (see section 7.5).

8.1 Further work

- First of all, although we have tested 6 algorithms, described in chapter 3, we have not tested every influence maximization algorithm that does exist. In a further work, we really think it would be worth replacing the algorithm CELF by the so-called **CELF++** developed by Goyal et al. (Goyal, Lu, and Lakshmanan, 2011). In fact the authors have shown that empirically, the computation time gain is between 35 an 55%, while keeping the same level of effectiveness as the CELF algorithm(Tang, Shi, and Xiao, 2015).

  Secondly, we believe that testing the *PageRank* algorithm (see appendix, section 9.1.1) would help analyze the intrinsic value of the *KRW* and *IKRW*, since they both use the concept of random walker on graphs.

Finally, it would also be worth comparing all the algorithms presented in this thesis with a method which is said to be the state-of-the-art technique based on statistical concept of *martingales*: the **Influence maximization via Martingales** algorithm, in short IMM.

- *T*hroughout this paper, the analyses made were mostly *descriptive*, we described how information is transfered from node to node (chapter 2) and how to select the best set of nodes in order to get the larger final influence spread (chapter 3). However, nothing was done at the *prescriptive* level, although it would be worth knowing "how to engineer products that spread virally"? What does explain the virality of the Volkswagen piano stairs for example (more than 22 million views on Youtube) (Rolighetsteorin, 2009), are there special features for viral marketing, is it a full design process or something else?

- As we have seen in chapter 6, maximizing the added value of a promotion campaign instead of allocating a budget to it is from far more interesting in terms of final profit and products' adoption rate for the company. This lack in the thesis should be made up for in next researches and is, according to us, one of the most substantial development that has to be made in the future.

- Also, when testing our algorithms, tuning of the networks' parameters was done randomly following some hypothesis. However, parametrization has a major influence on the selected seed set and the influence diffusion throught the network. Thus, for the algorithms to be used in a real-life situation, one might want to develop techniques so as to extract relevant parameters from a given social network as accurately as possible. This way, it would ensure consistency between calculated influence spread and real-life situation, giving the techniques some rightfulness.

- Eventually, as widely discussed throughout this paper, all those algorithms should be tested on larger and denser real networks in order to highlight even further the differences among algorithms. This way, one could also know which algorithms are worth being further deployed and enhanced in order to tackle the prominent issues of the Big Data and particularly problems related to social networks.

Chapter 9

Appendices

9.1 Other widespread algorithms

9.1.1 PageRank

The **PageRank (PR)** algorithm is a model initially created by Page et al. (Page et al., 1999) in order to rank all the web pages. This ranking will be used within a search engine in order to retrieve the best pages containing the bag of words of a user's query. This algorithm eventually allocates a value to each node (website) based on the probability for a *random walker* to end-up on that page when jumping from link to link[1](Page et al., 1999).

The PageRank value of a node u (representing its importance), denoted by $PR(u)$ can be iteratively computed thanks to the following formula:

$$PR(u) = (1-d) + d\left[\sum_{v \in N^{in}(u)} \frac{PR(v)}{|N^{out}(v)|}\right],$$

where d is a *damping factor* in the interval $[0, 1]$ and represents the probability that the user keeps on clicking on another link.

Assume a network of only 3 nodes A, B and C with directed links between them: From A to B and C, from B to C and from C to A, as depicted in figure 9.1.

If we initiate each $PR()$ to 1 with a damping factor $d = 0,5$, the first iteration is computed as follows (Gohil, 2012):

- $PR(A) = 0,5 + 0,5[PR(w)] = 1$
- $PR(B) = 0,5 + 0,5[\frac{PR(u)}{2}] = 0,75$

[1]for more information on random walks, please refer to the section 3.4.1

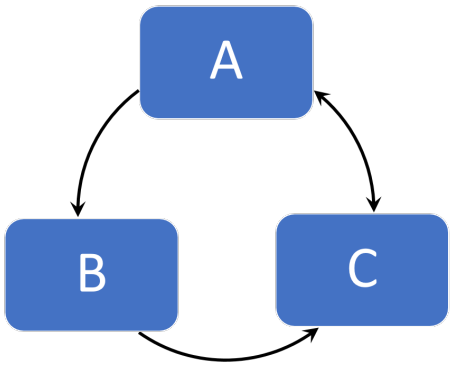


FIGURE 9.1: Network with 3 vertices and directed arcs (taken from Gohil, 2012)

- $PR(C) = 0,5 + 0,5 \times [\frac{PR(U)}{2} + PR(v)] = 1,125$

After 12 iterations, it reaches a steady state with PR values for A, B and C of 1,07692308, 0,76923077 and 1,15384615, respectively. (Gohil, 2012). The websites will then be classified according to a decreasing order of PR values; in this case C first, A second and B in the last position.

We will then select the k nodes having the highest PR score as seed set.

9.1.2 SimPath algorithm for the LT model

Like the MIA algorithm for the *IC model*, there exists a scalable algorithm for influence maximization under the $LTmodel$ introduced by Goyal et al. (Goyal, Lu, and Lakshmanan, 2011)

The idea behind this algorithm is to replace the Monte Carlo simulations for estimating the influence spread $\sigma(S)$ (which takes long time to compute) by the sum of the marginal contributions of the **simple paths** within a small neighborhood of the seed nodes. The authors have decided to restrict the paths to the neighborhood, since probabilities of paths rapidly decrease with path length.

Let us now define some variables and notations used specifically in this model. For a graph $G = (V, E)$ and a subset $U \subseteq V$, we define the *induced subgraph* of G w.r.t. U by $G'(U, E')$, where $E' = \{(u, v) \in E | u, v \in U\}$.

For a given set of seed nodes $S \in U$, we let $\sigma^U(S)$ denote the influence spread of S in the subgraph G'. In order to ease the notation, we will denote set differences $V \setminus S$ by $V - S$ and the union of set $S \cup \{u\}$ by $S + u$. Hence, $\sigma^{V-S+u}(S)$ denotes the influence spread of S in the subgraph induced by $(V \setminus S) \cup \{u\}$.

As for the MIIA algorithm for the IC diffusion model, we will use the *activation probabilities* of nodes v to compute the influence spread $\sigma(S)$ with the SimPath algorithm. In fact those probabilities will be defined in a different way, since they come from different diffusion models, with different parameters. We will denote them by $\Upsilon_{S,v}$ the node v's activation probability given a seed set S. The influence spread of a given seed set S can be computed by adding up all the nodes' activation probabilities for this seed set:

$$\sigma(S) = \sum_{v \in V} \Upsilon_{S,v}$$

In the specific case of a set composed of a singleton u, the activation probabilities of v, given the seed set u can be expressed as $\Upsilon_{u,v}$. We can compute this by using the set of all simple paths between them, i.e. paths which do not contain more than once the same vertices: $\wp(u, v)$. The probability of a path $P \in \wp$, denoted by $Pr(P)$ can be found by multiplying the weights of the arcs along that path P. This can be understood as the probability that the destination node is active, given the source of the path being the seed.$\Upsilon_{u,v}$ is the sum of all probabilities of paths from u to v:

$$\Upsilon_{u,v} = \sum_{P \in \wp(u,v)} Pr(P)$$

Now we can calculate the value of $\Upsilon_{S,v}$ by considering the contribution of each $\Upsilon_{u,v}$. Of course we can only consider seed nodes that do not pass through another seed node, otherwise the contributions would be counted more than once. We thus have

$$\sigma(S) = \sum_{u \in S} \sigma^{V-S+u}(u)$$

The influence of node u in the previous equation can be computed thanks

to *Backtracking*. The *BackTrack* algorithm computes the sum of influence spreads over simple paths starting at node u, that exceeds a threshold δ (see alorithm 17). As you can observe, this algorithm calls the *Forward* method. The latter returns the next simple path exceeding the threshold δ used in *Backtrack* (algorithm 16). The *SimPath* algorithm is shown with more details in algorithm 18.

Algorithm 16 Forward algorithm

Inputs: Queue Q; out-neighbors D; ongoing influence spread spd; path probability pp; threshold δ; node sets W, U
Output: Next simple path to visit, with $probability \geq \delta$
initialize $x \leftarrow Q.last()$
while $\exists y \in N^{out}(x) : y \notin Q, y \notin D[x], y \in W$ **do**
  if $pp.b_{x,y} \leq \delta$ **then**
    $D[x].insert(y)$
  else
    $Q.add(y)$
    $pp \leftarrow pp.b_{x,y}$
    $spd \leftarrow spd + pp$
    $D[x].insert(y)$
    $x \leftarrow Q.last()$
  end if
end while
return $[Q, D, spd, pp]$

Algorithm 17 BackTracking algorithm

Inputs: node u; threshold δ, set of nodes W
Output: sum of influence spreads over simple paths starting at u with probability above δ

initialize $Q \leftarrow \{u\}$; $spd \leftarrow 1$; $D \leftarrow \emptyset$
while $Q \neq \emptyset$ **do**
  $[Q, D, spd, pp] \leftarrow Forward(Q, D, spd, pp, \delta, W)$ (see algorithm 16)
  $u \leftarrow Q.last()$
  $Q \leftarrow Q - u$
  delete D[u]
  $v \leftarrow Q.last()$
  $pp \leftarrow pp/b_{v,u}$
end while
return spd

Algorithm 18 SimPath algorithm

Inputs: Seed Set S; Threshold δ
Output: Estimation of Influence spread $\sigma(S)$

initiate $\sigma(S) \leftarrow 0$
for each $u \in S$ **do**
$\sigma(S) \leftarrow \sigma(S) + BackTrack(u, \delta, V - S + u)$ (see algorithm 17)
end for
return $\sigma(S)$

9.1.3 Degree centrality

The definition of **degree centrality (DC)** stays that the most central actors are the ones with the highest numbers of neighbors(Wasserman and Faust, 1994). Formally the degree centrality index of user u, $DC(u)$ is the sum of all edges between u and its neighbors:

$$DC(u) = \sum_{v}(A_{uv}) = \Sigma_v(A_{vu}),$$

where $\boldsymbol{A}$ is the adjacency matrix representing the existing edges between nodes (a value of 1 is assigned to (u, v) if there is an edge between u and v, 0 otherwise). If the graph is undirected, the elements A_{uv} and A_{vu} are equal, for every $u, v \in N$; in other words, the matrix A is symmetric. Once all nodes indices are computed, the choice between nodes is done by selecting the k nodes with the highest *DC* scores.

9.1.4 Closeness centrality

The idea of **closeness centrality (CC)** is to know whether a user can quickly interact with others, i.e. the most central nodes in this view are the ones with the *minimum distance* with every users (Wasserman and Faust, 1994).

Suppose we have computed the shortest-paths distances between all pair of nodes {u,v} thanks to *Dijkstra algorithm* (see section 3.3.2); those distances being denoted by d_{uv}. We can now compute the closeness centrality index of node v, $CC(v)$ thanks to the formula

$$CC(v) = \frac{n-1}{\sum_{u \in V} d_{vu}}$$

The reader can observe that the $CC(v)$ score is simply the inverse of the average shortest-path distance from node v to any other vertex (Okamoto, Chen, and Li, 2008)[2]. Eventually, the algorithm chooses the k vertices with the highest *CC* scores as seed set.

9.2 Another Diffusion Model: the voter model

In addition to the two presented *diffusion models*, (*IC* and *LT*, other related models have been developed; we can for instance think of the **voter model** developed by Even-Dar and Shapira, which works like the *LT model* in the sense that "a person is more likely to change his opinion to the one held by most of it neighbors" (Even-Dar and Shapira, 2011). However, it is drastically different because it allows to *deactivate* nodes which were already activated, thus to switch back and forth from one state to another. A key property of the voters' model is the fact that eventually a *consensus* is always found. This model is really suitable for describing the process of adoption of new technologies. In fact most technologies reach a consensus: Google as search engine, Windows as OS, YouTube as a platform for sharing videos, to mention but a few.

The propagation of information in the voter model is defined as follows: Given an undirected graph $G(V, E)$ with self loops (an individual can choose to stay with its own choice from the previous time step), the set of user u's neighbors $N(u)$ and a random 0/1 assignment to the vertices of G, then at each time step, each node picks and adopts at random the choice of one of its neighbors (or his own one). Explicitly, starting from any assignment $f_0 : V \to \{0, 1\}$, we can find by induction

$$f_{t+1} = \begin{cases} 1 & \text{with probability } = \frac{|u \in N(v) : f_t(u) = 1|}{|N(v)|}, \\ 0 & \text{with probability } = \frac{|u \in N(v) : f_t(u) = 0|}{|N(v)|}. \end{cases}$$

[2]Notice that other definitions of closeness centrality do exist, but this version is presented, since it allows comparisons of nodes for networks of different size

The goal is to find the initial assignment $f_0 : V \rightarrow \{0, 1\}$ that will maximize the expectation E:

$$E[\Sigma_{v \in V} f_t(v)]$$

subject to the budget constraint B. In this model, the authors allowed the costs of reaching different nodes not to be identical in the network. If we denote the cost of sending information to node v by C_v, the sum of the individual selected nodes' cost has to be lower or equal to the budget constraint. Formally, we thus have

$$\Sigma_{\{v:f_0(v)=1\}} C_v \leq B$$

Even-Dar and Shapira (Even-Dar and Shapira, 2011) have demonstrated that for any graph G with transition matrix T [3] (the normalized version of the adjacency matrix A, where a_{uv} denotes the influence probability of u towards v), the set S which maximizes the spread can be computed by $T^t \times 1_S$, subject to $\Sigma_{v \in S} C_v \leq B$, where 1_S is the 0/1 vector, whose ith entry is 1 if $i \in S$, otherwise it is replaced by zero.

When the costs of all nodes are identical, the optimal solution can easily be found; one can pick the (B/C) highest degree vertices of G (see section 9.1.3).

Also, they proved that a consensus (everything is set either as 0 or 1) is found after $O(n^3 logn)$ steps with probability $1 - o(1)$ (for more information on *time complexity*, the reader can refer to the box of section 3.2).

Curious readers seeking for other models of spread can consult for instance Chen et al. (Chen, Lakshmanan, and Castillo, 2014).

[3] The reader may refer to section 3.4.1 for more details on transition matrices.

Bibliography

Agnihotri, R. et al. (2015). "Social media: Influencing customer satisfaction in B2B sales". In: *Industrial Marketing Management*.

Bernard, H., P. Killworth, and L. Sailer (1981). "Informant accuracy in social network data IV". In: *Social Networks*.

Bhagat, S., A. Goyal, and L.V.S. Lakshmanan (2012). "Maximizing Product Adoption in Social Networks". In: *Proceedings of the 5th ACM International Conference on Web Search and Data Mining*, pp. 603–612.

Blondel, V. et al. (2008). "Fast unfolding of communities in large networks". In: *Journal of Statistical Mechanics: Theory and Experiment* 10.

Bollobas, B. (1998). *Modern Graph Theory*. New York: Springer-Verlag.

Buchbinder, N. et al. (2012). "A tight linear time (1/2)-approximation for unconstrained submodular maximization". In: *Proceedings of the IEEE FOCS*.

Cawsey, T. and J. Rowley (2016). "Social media brand building strategies in B2B companies". In: *Marketing Intelligence & Planning* 34.

Chen, W., L.V.S. Lakshmanan, and C. Castillo (2014). *Information and Influence Propagation in Social Networks*. Morgan & Claypool.

Domingos, P. and M. Richardson (2001). "Mining the Network Value of Customers". In: *Proceedings of the 7th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining*.

Dughmi, Shaddin (2009). "Submodular Functions: Extensions, Distributions, and Algorithms A Survey". PhD thesis. Department of Computer Science, Stanford University.

Even-Dar, E. and A. Shapira (2011). "A note on maximizing the spread of influence is social networks". In: *Information Processing Letters* 111, pp. 184–187.

Fouss, F., M. Saerens, and M. Shimbo (2016). *Algorithms and Models for Network Data and Link Analysis*. Cambridge University Press.

Freeman, L.C. (1977). "A set of Measures of Centrality Based on Betweenness". In: *Sociometry* 40-1, pp. 35–41.

Gohil, D. (2012). *Search Optimization Technique: PageRank*. https://www.slideshare.net/jdhaar/pagerank-algorithm-explaned. Accessed: 2017-06-16.

Goyal, A., W. Lu, and L.V.S. Lakshmanan (2011). "SIMPATH: An Efficient Algorithm for Influence Maximization under the Linear Threshold Model". In: *Proceedings of the 2011 IEEE International Conference on Data Mining*, pp. 211–220.

Jurvetson, S. (2000). "What exactly is viral marketing?" In: *Red Herring* 78, pp. 110–112.

Kempe, D., J. Kleinberg, and E. Tardos (2003). "Maximizing the Spread of Influence through a Social Network". In: *Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining*.

— (2005). "Influential Nodes in a Diffusion Model for Social Networks". In: *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming*.

Killworth, P. and H. Bernard (1976). "Informant accuracy in social network data". In: *Human Organization*.

Leskovec, J., L.A. Adamic, and B.A. Huberman (2007). "The Dynamics of Viral Marketing". In: *ACM Transactions on the Web* 1.

Loftus, T. (2013). *Top video games may soon cost more*. https://www.nbcnews.com/id/3078404/ns/technology_and_science-games/t/top-video-games-may-soon-cost-more/#.WYqg04jyjIU. Accessed: 2017-07-27.

MacRae, J. (1960). "Direct factor analysis of sociometric data". In: *Sociometry*.

Metropolis, N. and S. Ulam (1949). "The Monte Carlo Method". In: *Journal of the American Statistical Association* 44.

Nemhauser, G.L., L.A. Wolsey, and M.L. Fisher (1978). "AN ANALYSIS OF APPROXIMATIONS FOR MAXIMIZING SUBMODULAR SET FUNCTIONS-I". In: *Mathematical Programming* 14, pp. 265–294.

Neti, S. (2011). "SOCIAL MEDIA AND ITS ROLE IN MARKETING". In: *International Journal of Enterprise Computing and Business Systems*.

Okamoto, K., W. Chen, and X.Y. Li (2008). "Ranking of Closeness Centrality for Large-Scale Social Networks". In: *Frontiers in Algorithmics: Second Annual International Workshop, FAW 2008, Changsha, China, June 19-21, 2008, Proceeedings*. Springer Berlin Heidelberg, pp. 186–195.

Page, L. et al. (1999). *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report 1999-66. Stanford InfoLab.

Rolighetsteorin (2009). *Piano stairs - TheFunTheory.com - Rolighetsteorin.se*. https://www.youtube.com/watch?v=2lXh2n0aPyw. Accessed: 2017-08-01.

Sela, A. et al. (2016). "Scheduled Seeding for Latent Viral Marketing". In: *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 642–643.

SimplyMeasured (2014). *Twitter Engagement*. https://simplymeasured.com/definition/twitter-engagement/#sm.00001kz1xmwbyye9cx26cv3ihy85q. Accessed: 2017-07-19.

Sipser, M. (2006). *Introduction to the Theory of Computation, Second Edition*. Thomson Course Tecnology.

Solon, O. (2018). *Facebook deletes accounts over signs of Russian meddling in US midterms*. https://www.theguardian.com/technology/2018/jul/31/facebook-russia-election-midterms-meddling. Accessed: 2018-08-02.

Statista (2017a). *Digital marketing spending in the United States from 2014 to 2019, by segment (in billion U.S. dollars)*. https://www.statista.com/statistics/275230/us-interactive-marketing-spending-growth-from-2011-to-2016-by-segment/. Accessed: 2017-07-26.

— (2017b). *Which social media platform(s) do you use to market your business?* https://www.statista.com/statistics/259379/social-media-platforms-used-by-marketers-worldwide/. Accessed: 2017-07-26.

Tang, J., X. Tang, and J. Yuan (2016). "Profit Maximization for Viral Marketing in Online Social Networks". In: *IEEE 24th International Conference on Network Protocols*, pp. 1–10.

Tang, Y., Y. Shi, and X. Xiao (2015). "Influence Maximization in Near-Linear Time: A Martingale Approach." In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*.

Vande Kerckhove, C. (2016). "Optimisation de la couverture de marché pas l'identification d'acteurs influents au sein des réseaux sociaux". MA thesis. Belgium: Université Catholique de Louvain.

Vanderlinden, Q. (2015). "Community Detection in Complex Networks". MA thesis. Belgium: Université Catholique de Louvain.

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

Wasserman, Stanley and Katherine Faust (1994). *Social Network Analysis - Methods and Applications*. Cambridge University Press.

Weisstein, E. (2018). *Petersen Graph*. http://mathworld.wolfram.com/PetersenGraph.html.

Zachary, W. (1977). "An information flow model for conflict and fission in small groups". In: *Journal of Anthropological Research*.

Zhang, P. et al. (2014). "Minimizing Seed Set Selection with Probabilistic Coverage Guarantee in a Social Network". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*.