Near Optimal Strategies under Knapsack Constraint for Targeted Marketing in Social Networks

Matthieu Cordier
Victor Catteau

Course : Networks (IEOR8100-004)

05/06/2019

Abstract

Abstract : We address the problem of targeted influence maximization in a social network. It is a common business problem of trying to determine the best influencers to advertise for a product on a social network, under a constrained budget.

Towards this end, the project will focus on proposing a model for the influence inside the network, and methods to solve the maximization. This study emphasize particularly on submodular optimization and heuristics. For experiments, we will study undirected social network such as Facebook.

1 Introduction

Social media marketing on networks such as Facebook or Instagram has become a privileged channel of advertising for a great number of brands. One of the reasons this channel is appreciated is its potential for launching cascade effects of "word-of-mouth" and "viral marketing". Viral marketing can be understood as a spontaneous phenomenon of diffusion of the information throughout the network, just as an epidemic. It has the potential to create cascade effects where the information spontaneously diffuses from a single individual to his neighbors and so on recursively, reaching out a great number of individuals. It therefore allows to reach a great number of individuals with minimal cost involved, since it was only necessary to initially "buy" the influence a few key individuals. We design those key individuals as "influencers". The question that arises is how to chose those influencers. From the point of view of the brand, an essential metric of success of a campaign is the "reach", which is the total number of individuals affected by the campaign.

More formally, a viral marketing campaign can be modelled as an optimization problem: maximizing the campaign reach under the constraint of a fixed budget, by choosing a set of initial influencers. This influence maximization problem is well studied under the framework established by David Kempe, Jon Kleinberg and Eva Tardos in their 2003 paper [1]. A fundamental result established in this paper is that the influence maximization problem is NP-hard. The authors of this paper then simplify the problem to finding the k most influential nodes in the network for k fixed, and use a simple linear threshold model to describe the spread of influence. They then propose a submodular optimization heuristic which has then been improved in several posterior papers [2,3,4].

The goal of our project is extending this work on viral marketing by proposing a more realistic framework. We depart from the Kempe and Kleinberg model in two important ways. First, a brand needs to find a best set of influencers, it often doesn't work with a fixed number of influencers, and influencer costs have high variance. A common trade off which occurs is deciding whether to concentrate the budget on a few expensive "macro-influencer" with millions of followers each, or to spread the budget across several less costly "micro-influencers", with only thousands of followers. This is why we chose to introduce the knapsack constraint to the spread maximization problem. Solving the problem would allow us to understand which of those two advertising strategies is optimal for the brand. Second, the discrete time linear threshold model doesn't seem realistic to us because it doesn't represent the fact that a single individual can influence thousands of others on a social network. This is why we chose the model the spread of influence on the social network as a continuous time Susceptible-Infected-Removed (SIR) stochastic model.

Towards this end, we propose two very different heuristic methods to find near-optimal solutions to the NP-hard influence maximization problem. We

then compared the two approaches on real-world social networks. We first study submodular optimization, using greedy algorithm, on the expected number of contaminated nodes. Then, we define a new submodular function, inspired from the summarization problem [3], and much easier to compute. The second method is an heuristic and is inspired from network topological analysis, using centrality measures. Its advantage is that it is simple and interpretable. Its main drawback is its computational cost.

This report is organised as follow : in **section 2**, we present the contagion model we studied during this project. In **section 3**, we present topological heuristics to find a near-optimal strategy. In **section 4**, we review submodularity and prove the submodularity for the expected number of infected nodes. In **section 5**, we design a new submodular function for the influence maximization problem. Finally, in **section 6**, we present the results of the different methods studied on a real social network.

2 Modelisation

2.1 General stochastic model : contagion

We consider the **undirected graph** $G(V, E)$ which represents our network. The influence spreads in G according to the SIR model, each node being in either one of the following three states :

- susceptible (s): the person is a potential buyer of our product but hasn't heard about it.
- infected (i): the person adopts the product and advertises for it through "Word-of-Mouth"
- removed (r): the person has adopted or heard about the product, but stops talking about it.

We call $I_0 \subset V$ the subset of the initial infected nodes, corresponding to our initial targets to spread the infection.

We represent the **dynamic of the network as a Continuous Markov Chain** with three states (s, i or r). For each nodes i, the transition time to the next state are exponential with following rates :

- λ_i is the **contagion rate** from a susceptible state to a infected state.
- γ_i is the **immunization rate**, from a infected state to a immune state (the node is not susceptible to spread influence anymore).

We can derive from those rate the probability β_i that an infectious node i will infect a susceptible neighbor. $\beta_i = \frac{\lambda_i}{\gamma_i + \lambda_i}$

We call $I_t \subset V$ **and** $R_t \subset V$ **the subsets of infected node at time t, and the subsets of immunized node at time t**. R_0 are the initial immunized node. In the following, we will consider $R_0 = 0$.

2.2 Influence Function

The objective of the problem is to maximize the influence function $\sigma(I_0, T)$, **i.e. the expected number of infected node during T**, with $T \in \mathbf{R}$.

$$\sigma(I_0, T) = \mathbf{E}[\sum_{v \in V} \mathbf{1}(t_v < T)] = \mathbf{E}[I_T + R_T] \tag{1}$$

with t_v is the time a node $v \in V$ is infected (potential infinite if a node is never infected).

2.3 Problem Formulation

We define the cost of choosing I_0 by the function $C(S) = \sum_{s \in S} c_s(s)$ with $S \subset V$. We define $b \in \mathbf{R}$ our budget. As we will see in some of the following sections, $\sigma(S, T)$ follow some very interesting properties (monotone and submodular)

We choose the following formulation for the problem :

$$\underset{S}{\operatorname{argmax}}\, \sigma(S, T) \text{ s.t } C(S) < b \tag{2}$$

The main challenge in this influence maximization problem arise from the following proposition, which was initially demonstrated by Kemp and al. [1]:

Property 2.1. *Maximizing the influence function $\sigma(I_0, T)$ is a NP-Hard problem*

Therefore, we will need heuristics to provide scalable and operational near optimal-solutions.

3 Topological heuristic

The goal of this section is to describe how to empirically build a heuristic based on network topology to solve this problem.

3.1 Predicting the Influence Function

A first intuition to simplify this problem is approximating $\sigma(I_0, T)$ by a simpler function. For that, we first restrict ourselves to the case of a unique initially infected node: $I_0 = v$. Here we argue that for large enough T, $\sigma(v, T)$ is a function of the topology of the network, using the following property:

Property 3.1. *$\sigma(v, \infty)$ is equal to the number of nodes connected to v in a random graph created by removing each edge (i, j) independently with probability β_i*

While this property is difficult to use in practice (computing $\sigma(v, T)$ this way would still involve costly Monte-Carlo simulations), it supports the intuition that we can efficiently predict $\sigma(v, T)$ using topological measures from the network analysis literature such as degree centrality, eigenvector centrality, betweeness centrality... The interest of doing so is that it replaces the expectation of a stochastic function by a deterministic function. It is much more efficient to compute those centrality measures than doing costly Monte-Carlo simulations for each nodes. It also allows to provide an interpretable optimization algorithm, which chooses nodes based on their topologic properties.

Remark 3.1. *$\sigma(v, T)$ can be approximated by a multivariate function of topological network measures, which we note $\tilde{\sigma}(v, T)$. This function can efficiently be estimated with Machine Learning techniques. This function depends essentially on the topology of the network.*

This remark has been subject of specific work, mainly in the context of epidemiological literature. [2] [5]

Remark 3.2. *Social networks share common topological properties: sparsity, small-world connectivity...*

This remark allows us to argue that the approximate influence spreading function $\tilde{\sigma}(v, T)$ will likely vary little from one social network to the other.

3.2 Empirical influence predictor function with k-core decompostion

Lets now describe how we built empirically this influence prediction function for the Facebook social network described in part 6. We first built a training set by sampling 300 nodes from this network and doing a Monte-Carlo simulation for each of those node to estimate its spread of influence. We then tried to predict this spread with four common network topological measures:

- degree centrality
- eigenvector centrality
- k-core decomposition index
- betweeness centrality

The summary of this regression is presented in Figure 1.The main takeaway is that in the studied social network, k-core decomposition index is the best predictor of spreading influence. This can also be visualized in Figure 2. This result is conform with the results in the epidemiological literature: Katsek and al. having made the same conclusion studying contact network of impatients in hospitals.[2]

OLS Regression Results

Dep. Variable:	Spread	**R-squared:**	0.848
Model:	OLS	**Adj. R-squared:**	0.846
Method:	Least Squares	**F-statistic:**	411.6
Date:	Tue, 07 May 2019	**Prob (F-statistic):**	2.55e-119
Time:	18:16:58	**Log-Likelihood:**	-142.56
No. Observations:	300	**AIC:**	295.1
Df Residuals:	295	**BIC:**	313.6
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.2037	0.075	-16.138	0.000	-1.351	-1.057
Degree_centrality	0.0802	0.075	1.063	0.288	-0.068	0.229
Eigenvector_centrality	-0.6227	0.035	-17.759	0.000	-0.692	-0.554
Core	0.0419	0.002	16.938	0.000	0.037	0.047
Betweeness_centrality	-0.0559	0.045	-1.246	0.214	-0.144	0.032

Figure 1: Results of Regression on node influence in a Facebook Network

At this point, we remind the definition of k-core decomposition:

Definition 3.1. *The k-core of a graph G is the largest induced subgraph of G in which every vertex has degree at least k.*

The definition of k-core decomposition index of a node both integrates both local (degree) and global properties (relative location: core or periphery of the network), as we can see in Figure 3. This explains why it is a good predictor of node spreading influence.

An interesting property of k-core decomposition is that it can be computed efficiently in linear time $O(|E|)$ by recursively pruning subgraphs.

We then refined our regression model and removed unsignificant features to create our final $\tilde{\sigma}(c_v, T)$ function, from which we derived the two following heuristics.

3.3 Deriving Heuristics for the Maximization Problem

3.3.1 Naive Knapsack Heuristic

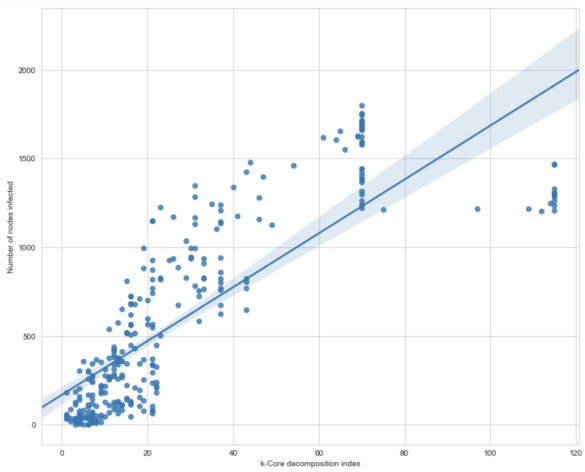

Figure 2: Regression on spreading influence against k-core index

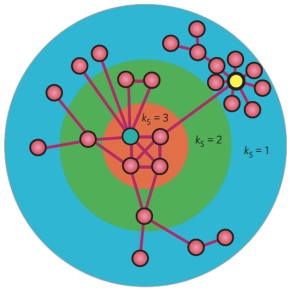

Figure 3: k-core shell decomposition

A first way of using this function to solve the problem would be to neglect interaction terms between influencer nodes:

$$\sigma_{approx}(S,T) = \sum_{v \in S} \tilde{\sigma}(c_v, T) \tag{3}$$

While this approximation might seem naive in the initial problem formulation, it could model a situation where targeting repeatedly a node would increase its likeliness to buy the product.
The maximization problem then becomes similar to the classic knapsack problem:

$$\underset{S}{\text{argmax}} \sum_{v \in S} \tilde{\sigma}(v,T) \text{ s.t } \sum_{v \in S} c(v) < b \tag{4}$$

Property 3.2. *The runtime of the Naive Knapsack Heuristic is linear:* $O(|V|+|E|)$

Proof. The $|V|$ term comes from the classic dynamic programming algorithm w-for the knapsack problem. The $|E|$ comes from the recursive pruning algorithm used to compute k-core decomposition of the graph, which is used to compute $\tilde{\sigma}(v, T)$ for each node v. □

Therefore, this Naive Knapsack heuristic is very quick. However, its performance tends to suffer from the fact chosen nodes are often close, with overlapping infection areas. This brings us to our second heuristic.

3.3.2 Penalized Knapsack Heuristic

We now take into account the fact that choosing close initial nodes will probably have overlapping infection areas. Therefore, we penalize the fact of choosing close influencers. We define a parameter λ controlling penalization:

$$\sigma_{approx}(S, T) = \sum_{v \in S} \tilde{\sigma}(c_v, T) + \lambda \sum_{v_1, v_2 \in S} dist(v_1, v_2)$$

(5)

This parameter λ will be optimized a posteriori using grid search. The new maximization problem becomes:

$$\underset{S}{\text{argmax}} \sum_{v \in S} \tilde{\sigma}(v, T) + \lambda \sum_{v_1, v_2 \in S} dist(v_1, v_2) \text{ s.t } \sum_{v \in S} c(v) < b \tag{6}$$

Which also be written as an Integer Linear Programming Problem (7), and solved with Simplex Algorithm:

$$\text{argmax} \sum_{v \in V} \tilde{\sigma}(v, T) x_v + \lambda \sum_{v_1, v_2 \in V} dist(v_1, v_2) y_{v_1, v_2} \tag{7}$$

$$\text{s.t } \begin{cases} \sum_{v \in V} c_v x_v < b \\ \forall v_1, v_2 \in V, \quad y_{v_1, v_2} \leq x_{v_1} \\ \forall v_1, v_2 \in V, \quad y_{v_1, v_2} \leq x_{v_2} \\ \forall v, t \in V x_v, \quad y_{v,t} \in \{0, 1\} \end{cases} \tag{8}$$

The results of both heuristics are presented in part 6.

We now take a totally different view of the initial problem and to build alternative heuristics using an interesting property of the objective function.

4 Submodular optimization

4.1 Definition

There exists two equivalent definitions of submodularity. The first one is the following :

Definition 4.1. *Let V be a finite set, and denote by 2^V the power set of V, i.e., the family of all subsets of V. A function $f : 2^V \to \mathbf{R}$ is called submodular if, for each $A, B \in 2^V$, we have:*

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B). \tag{9}$$

Another one, more intuitive, is the following :

Theorem 4.1 (Law of diminishing returns)**.** *Let $f : 2^V \to \mathbf{R}$ be a function.f is submodular if and only if it satisfies the law of diminishing returns, i.e.*

$$f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B) \text{ for all } A \subseteq B \in 2^V \text{ and } e \in V \backslash B \tag{10}$$

Intuition of the diminishing return property : The following figure presents an example which illustrate the diminishing return property.

- Given a set A of colored balls
- $f(A)$: the number of distinct colors contained in the urn
- The incremental value of an object only **diminishes** in a **larger** context (diminishing returns).

Figure 4: Diminishing return example from [3]

4.2 Properties

4.2.1 First Properties

Submodular functions share a number of properties in common with convex and concav function, including their wide applicability, their generality, and their closure under a number of common of operator. We first want to formulate properties that will be quit useful when studying submodular functions in this report.

Property 4.1 (Combination of submodular functions)**.** *If a collection of functions* $\{f_i\}_i$ *is submodular, then any non negative weighted sum* $g = \sum_i \alpha_i f_i$ *is submodular, where* $\forall i, \alpha_i \geq 0$.

Property 4.2 (Composition)**.** *Given functions* $\mathcal{F} : 2^V \to \mathbb{R}$ *and* $f : \mathbb{R} \to \mathbb{R}$*, the composition* $\mathcal{G} = f \circ \mathcal{F} : 2^V \to \mathbb{R}$ *is non decreasing submodular if* f *is non-decreasing concave and* $\mathcal{F}$ *is nondecreasing submodular.*

This properties will be useful later.

4.2.2 Optimization properties

In discrete optimization, submodularity is a very interesting function property since minimization can be done in polynomial time.

But our initial problem is a maximization problem. For a submodular function, the maximization problem is NP-hard ([1]). However, some greedy algorithms guaranties a near-optimal solution for any submodular function. In this study, we will work with the Algorithm 1 (Figure 5), a modified greedy algorithm [4].

Theorem 4.2 (Submodular maximization with cost function and budget)**.** *f a normalized monotone submodular function. Let $\hat{S}$ be the set output by Algorithm 1 with $r = 1$ (enhanced greedy algorithm, Lin and Bilmes [4]) and let S^* be the optimal solution. Then $f(\hat{S}) \geq (1 - \frac{1}{\sqrt{e}}) f(S^*)$*

Algorithm 1 Modified greedy algorithm

$G \leftarrow \emptyset$
$U \leftarrow V$
while $U \neq \emptyset$ **do**
  $k \leftarrow \arg\max_{\ell \in U} \frac{f(G \cup \{\ell\}) - f(G)}{(c_\ell)^r}$
  $G \leftarrow G \cup \{k\}$ **if** $\sum_{i \in G} c_i + c_k \leq B$ **and** $f(G \cup \{k\}) - f(G) \geq 0$
  $U \leftarrow U \setminus \{k\}$
end while
$v^* \leftarrow \arg\max_{v \in V, c_v \leq \mathcal{B}} f(\{v\})$
return $G_f = \arg\max_{S \in \{\{v^*\}, G\}} f(S)$

Figure 5: Modified greedy algorithm for submodular function (Bilmes and Line, 2010) [4]

In practice, the near-optimality is even better, which makes submodularity a very interesting property.

Finally, we can a little bit speed up the algorithm with the following idea. One of the problem of the greedy algorithm is that we need to evaluate the function many times during the algorithm, and this may computationally expensive (like in the case of σ which is expensive to compute) The **accelerated greedy algorithm** [6] (also called "lazy") use a trick to diminish the number of evaluations of f for the "$argmax$" (cf figure 5). At the first round of greedy, compute $f(e) \forall e \in V$ and sort them. Let's take now the example in the figure 6.

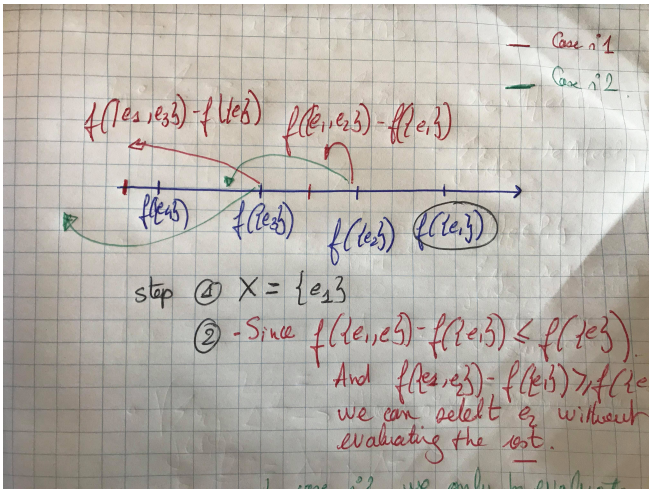

Figure 6: Simple example of the speed up in accelerated greedy

In the example (case 1), since $(f(\{e_1, e\}) - f(\{e_1\}))/c_e \leq f(\{e\}$ and $(f(\{e_1, e_2\}) - f(\{e_1\}))/c_{e_2} > f(\{e_3\})/c_{e_3}$, we can select e_2 without evaluating the rest. Thus, the main idea is that the marginal gain of a node in the current iteration cannot exceed its marginal gain in previous iterations. With this idea, optimization significantly reduces the number of calls made to the spread estimation procedure.

4.3 Submodularity in the context of Targeted Marketing

Is the expected number of contaminated nodes submodular ?

Property 4.3 (Submodularity of σ [1])**.** $\forall T \in \mathbf{R}$*, the expected number of infected node during* T*,* $\sigma(\cdot, T)$*, is submodular.*

Proof. 1. Let's consider the stochastic model described in the *section 2.1.* The main idea in this proof is to show that the stochastic model can be viewed as an independent cascade model. Then, we will write the proof for the submodularity of σ in independent cascade model.

We want to compute for any neighbors i and j, $P_{i,j} = \mathbb{P}$ (the edge (i,j) is added to the infection directed random graph before $T|t_i < T$)

, which means we want to compute the probability that the node j is infected during time T by node i when node i is infected before T.

$$P_{i,j} = \frac{\lambda_i}{\rho_i + \lambda_i} \times \mathbb{P}(t_i \leq T - t_j | t_j < T)$$

. Thus, we can define an Independent Cascade model from the initial stochastic model, and will prove now the submodularity of σ for Independent Cascades (proof from [1]).

2. The Independent Cascade process is underspecified, since we have not prescribed the order in which newly activated nodes. Our proof deals with these difficulties by formulating an equivalent view of the process. We can view the outcome of this random event as being determined by flipping a coin of bias $P_{i,j}$. From the point of view of the process, it clearly does not matter whether the coin was flipped at the moment that v became active, or whether it was flipped at the very beginning of the whole process and is only being revealed now. Thus, we can assume that for each pair of neighbors (v,w) in the graph, a coin of bias $P_{i,j}$. is flipped at the very beginning of the process (independently of the coins for all other pairs of neighbors), and the result is stored. All those coin processes can be summed in a new directed graph (green graph in figure **??**). The edges in the new graph G' for which the coin flip indicated an activation will be successful are declared to be live; the remaining edges are declared to be blocked.

3. Now, let's fix G'. A node is active nodes if it is reachable via paths in the new graph G' from initially targeted nodes S. We define $\sigma'_G(S,T)$ the expected number of infected nodes (active) knowing G'.

4. $A \subseteq B \subseteq V$ and v a node not in A (and B). Let's prove that $\sigma_{G'}(\cdot,T)$ is submodular, ie. $\sigma_{G'}(A+v,T)\sigma_{G'}(A,T) \geq \sigma_{G'}(B+v,T) - \sigma_{G'}(B,T)$. Let $R(v,X)$ denote the set of all nodes that can be reached from v on a path consisting entirely of live edges. $\sigma_{G'}(A+v,T)\sigma_{G'}(A,T)$ corresponds to the number of elements in $R(v,X)$ that are not already in the union $Cup_{u\in A}R(u,X)$; it is at least as large as the number of elements in $R(v,X)$ that are not in the (bigger) union $\uplus_{u\in T}R(u,X)$. Thus $\sigma_{G'}(\cdot,T)$ is submodular.

5. $\sigma(\cdot,T) = \sum_{allG'possible} \mathbb{P}(G') \times \sigma_{G'}(\cdot,T)$. Thus, $\sigma(\cdot,T)$ is a linear sum of submodular function with positive weights, which concludes the proof (Property 3.1).

□

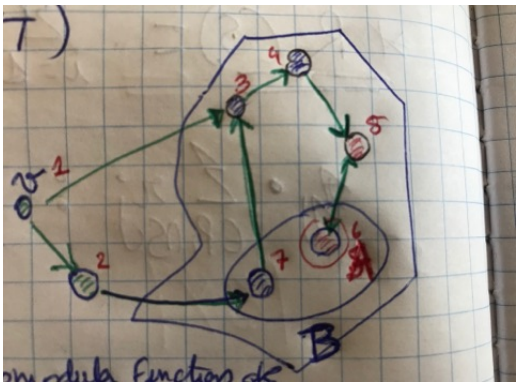

Figure 7: Example for the proof : fixed graph G' and illustration of σ submodularity

Thus, we could use the greedy algorithm to get a near-optimal solution by optimising directly σ with the accelerated enhanced greedy algorithm. But is σ easy to compute ?

5 Monotone submodular objective functions

In this section, two submodular functions we will optimise using the accelerated modified greedy algorithm are studied.

5.1 Expected number of contaminated nodes (Monte Carlo Simulation)

σ is hard to compute in general, even for the SIR model. Thus, we need to simulate this expectation σ to compute a near-optimal solution for the influence maximization problem.

When the cost is the number of items, the complexity of the classic (not accelerated) greedy algorithm for submodular functions is $O(|V|^2).O(\sigma)$, where $O(\sigma)$ is the time complexity of the function σ ([6]). Since the number of simulation needed to simulate is big, the computation time or the maximization problem becomes easily long for big networks with a standard greedy algorithm. We will see later that the computation is still long with the accelerated greedy algorithm. We will call this method **greedy** σ.

So, it appears that it can be difficult to optimize or even compute σ for very large networks. Thus, we need to find a submodular function cheaper to compute, which once maximized, give a good subset for the influence maximization problem.

5.2 Coverage-Diversity function

A first idea is to use submodular functions encountered in other submodular maximization problems. For instance, in [3], the document summarization problem can be formulated as the maximization of the sum of a reward function and a diversity function.

Definition 5.1 (Coverage-Diversity submodular function (Lin and Bilmes, 2010 [3]))**.** *We say that $\mathcal{F} : 2^V \to \mathbf{R}$ is coverage-diversity submodular function if has the following form :*

$$\forall S \in 2^V, \mathcal{F}(S) = \mathcal{L}(S) + \lambda \mathcal{R}(S) \tag{11}$$

$$= \sum_{i \in V} \min\{\mathcal{C}_i(S), \alpha \mathcal{C}_i(V)\} + \lambda \sum_{i}^{K} \sqrt{\sum_{j \in P_i \cap S} r_j} \tag{12}$$

Where $\lambda \geq 0$, $\mathcal{C}_i(S) : 2^V \to \mathbf{R}$ is a monotone submodular function, and r_i indicates a singleton reward, which represents the importance of i in the summary. $(P_i)_{i=1...K}$ is partition of the set V. Thus, $\mathcal{L}$ measures the coverage ("fidelity") and $\mathcal{R}$ rewards diversity.

How can we reuse this idea ? In our problem, we would like to get a subset S which represents well the graph, in the sense that the nodes in S should be highly connected to the rest of the graph (better contagion), and each $i \in S$ should be enough distant of each others(we do not want 2 contagions from 2 different nodes to overlap). In that sense, we want to get a subset S which summarize the better the set V.

In [3], $\forall i, C_i(S) = \sum_{j \in S} w_{i,j}$ and $\forall j, r_j = \frac{1}{N} \sum_{i \in V} w_{i,j}$ where $w_{i,j}$ is the cosinus similarity between i and j.

Thus, **we reuse this formulation and call $\mathcal{F}_{net}$ the following function** :

$$\mathcal{F}_{net}(S) = \sum_{i \in V} \min\{\sum_{j \in S} w_{i,j}, \alpha \sum_{j \in V} w_{i,j}\} + \lambda \sum_{i}^{K} \sqrt{\sum_{j \in P_i \cap S} \frac{1}{N} \sum_{i \in V} w_{i,j}} \qquad (13)$$

where $w_{i,j}$ **is the cosinus similarity between A_i, A_j (A is the adjacency matrix)**. Thus, we are summarizing, as in the *Document Summarization Problem*, but the data is the adjency matrix (neighbors of each nodes),and it forces the algorithm to select diverse and covering nodes regarding neighbors, and thus should maximize the contagion. To our knowledge, this approach has never been used in the context of influence maximization in networks. We will call this method in the experiments section "**greedy $\mathcal{F}_{net}$**".

To define and optimise a coverage-diversity function, we need to compute a clustering to create a partition. We will use a Spectral Clustering algorithm in the experiments.

Let's now focus a little bit on each term of $\mathcal{F}_{net}$ and prove that it is submodular.

5.2.1 Coverage function

Property 5.1. *The function $\mathcal{L} : 2^V \to \mathbb{R}$ defined by*

$$\forall S \subseteq V, \mathcal{L}(S) = \sum_{i \in V} \min\{\sum_{j \in S} w_{i,j}, \alpha \sum_{j \in V} w_{i,j}\}$$

is non-decreasing submodular.

Proof. The non-decreasing property is trivial.

- (Sub)modularity of $\sum_{j \in S} w_{i,j}$: Let's take $A \subseteq B \subset V$ and an element v not in A. $\sum_{j \in (A+v)} w_{i,j} - \sum_{j \in (A)} w_{i,j} = \sum_{j \in (B+v)} w_{i,j} - \sum_{j \in (B)} w_{i,j}$, so this is "modular" (equality).
- The min operator is concave. Thus, the *property 3.2* shows that $\min\{\sum_{j \in S} w_{i,j}, \alpha \sum_{j \in V}\}$ is submodular.

- Since $\mathcal{L}$ is a sum of positive weighted submodular function, $\mathcal{L}$ is submodular.

□

5.2.2 Diversity function

Property 5.2. *For a partition $\{P_k\}_k$, the function $\mathcal{R} : 2^V \to \mathbb{R}$ defined by*

$$\forall S \subseteq V, \mathcal{L}(S) = \sum_{i}^{K} \sqrt{\sum_{j \in P_i \cap S} \frac{1}{N} \sum_{i \in V} w_{i,j}}\}$$

is non-decreasing submodular.

Proof. Again, the non-decreasing property is trivial.

- It is easy to see that $\sum_{j \in P_i \cap S} \frac{1}{N} \sum_{i \in V} w_{i,j}$ is modular
- Since the square function is concave, from the *property 3.2*,

  $$\sqrt{\sum_{j \in P_i \cap S} \frac{1}{N} \sum_{i \in V} w_{i,j}}$$

  is submodular.
- Thus, since this a sum of submodular function, positively weighted. This concludes the proof (*property 3.1*)

□

6 Experiment and results

Now, the model we will consider is a simple continuous SIR model ($\forall i, \lambda_i = \lambda$, $\gamma_i = \gamma$). We first define a number of parameters before presenting the results.

c **(Cost function):** The cost function $c : 2^V \to \mathbf{R}$ is define as the number of neighbors of each selected node : $\forall S \subseteq V, c(S) = \sum_{i \in S} deg(i)$

Budget B **:** We take the budget as the maximum of neighbors in the graph. Indeed, we want to modelize a situation where the choice is whether to concentrate our budget on one "macro-influencer" or to spread it across several less costly "micro-influencers".

Stochastic model : We choose to model contagions with a simple SIR model. This is a strong hypothesis, but for the moment, we prefer to focus more on methods than stochastic models. We will choose $\lambda = \mathbf{0.01}$, $\rho = \mathbf{0.1}$. For those parameters, the infection ends up dying out quickly. Also, it is important to have some parameters which discriminate the different methods. For instance, sometimes, if T is too big, $\sigma(S, T) = |V|$ for any $S \subseteq V$ (saturation of contagion). This is something we want to avoid, to be able to compare the different methods.

N_0 **(Number of iteration for simulation):** We take 100 simulations to get a sufficiently good approximation of σ and to limit time computation.

In this section, we will first optimize parameters for the submodular optimization, and compare the time computation. Finally, we will compare all the method studied :

1. Naive Knapsack Heuristic
2. Penalized Knapsack Heuristic
3. Greedy submodular optimization of $\sigma(\cdot, T)$
4. Greedy submodular optimization of $\mathcal{F}_{net}(\cdot)$
5. Random selection (choose nodes randomly until the budget is reached)

6.1 Submodular functions and greedy algorithm : optimizing parameters

We first optimize λ for the function $\mathcal{F}_{net}$ and find an optimal $\lambda = 0.1$ (Figure 8 a)).

Secondly, we optimize its saturation parameter α and find $\alpha = 0.7$.

Next, we optimise the parameter r for the modified greedy algorithm for the function $\mathcal{F}_{net}$ and find $r = 5$, which is large. (Figure 8 c)). We interpret this as the following : at each steps, it is better to not take the element with the better return $\mathcal{F}_{net}(G+l) - \mathcal{F}_{net}(G)$, but a good one with a little cost. It enables to select more items less costly in the final subset S, which is suitable to the problem. Indeed, the budget is the maximum of neighbors in the considered graph, so selecting the best connected node is probably not the best solution.

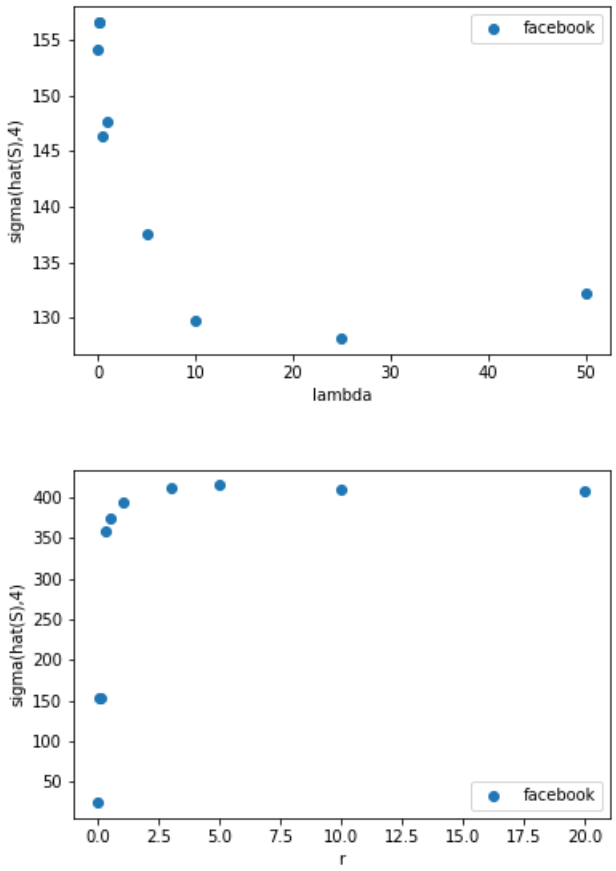

Figure 8: a) Influence value of the submodular $\mathcal{F}_{net}$ near optimal $\hat{S}$ in function of the parameter λ (threshold between coverage and diversity) b) Influence value of the submodular $\mathcal{F}_{net}$ near optimal $\hat{S}$ in function of the parameter r of the modified greedy algorithm

Finally, for the number of clusters K (clustering needed for $\mathcal{F}_{net}$), we take $K = 0.05|V|$ (aribitrary) and apply a spectral clustering with the graph's ad-

jency matrix.

6.2 Submodular optimization : time computation comparison

We present here a comparison of time computation in function of the size of the network. We use here a real social network from SNAP : "com-youtube.ungraph". We select randomly some edges and plot the time computation of the **accelerated greedy algorithm** for σ and $\mathcal{F}_{net}$. Thus, we confirm that the time

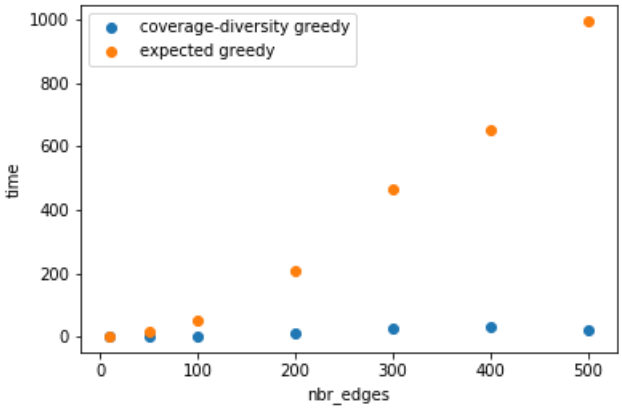

Figure 9: Time computation of the greedy algorithm for σ and $\mathcal{F}_{net}$ in function of the number of edges selected in the graph "com-youtube.ungraph"

computation is long for σ, even for small and medium graphs with an accelerated greedy algorithm. It confirms that the $\mathcal{F}_{net}$ is a better monotone submodular function to optimize. We will next compare their performance.

6.3 Experiment on a real social network

For this experiment, we will study the "Facebook Combined" graph from SNAP. This is a large scale social network graph ($|V| = 4039$ and the number of edges is $|E| = 88234$).

6.3.1 Experiment 1 : T=4

It seems that $\mathcal{F}_{net}$ outperforms a direct submodular optimization of σ (and is very easy to compute) which seems strange. We explain this result : the construction of the $\mathcal{F}_{net}$ into two distinct function (coverage/diversity) makes it maybe easier to select nodes far from each others, and highly connected. Another explanation is that r has been optimized for $\mathcal{F}_{net}$, but not for σ, (the

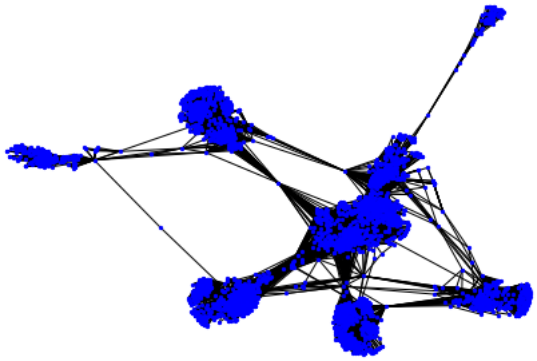

Figure 10: SNAP's facebook undirected graph representation

	Random	Greedy σ	Greedy $\mathcal{F}_{net}$	Naive Knapsack	Penalized Knapsack
$\sigma(\hat{S}, T)$	88	264	455	237	331
I_0	24		372	174	253
Runtime	-	11 h	5 min	1s	4h

Table 1: Results on the SNAP facebook dataset (T=4)

grid-search is too long to compute for σ greedy). Thus, the cost scale r is probably not adapted to the function σ.

Moreover, the greedy $\mathcal{F}_{net}$ leads to a high number of selected nodes (361), but less connected, which let us think that diversifying is a better solution than selecting the most connected node. Empirically, we find this approach in many Instagram marketing strategies where a firm prefers to sponsor a lot of small influencers than just an important one.

7 Conclusion and discussion

The strategies selecting the most nodes seem to perform best. This is evidence that the increasing interest of brands for micro-influencers is justified.

Moreover, during this study, we designed a new submodular function inspired from the document summarization task able to outperform all the others strategies, for a large scale irregular graph. It would be interesting to quantify this gain in function of the parameters of the stochastic model. Indeed, we found out that for very regular graphs such as a Erdős–Rényi random graph, all the methods give similar result.

Potential next steps would include improving the predictive algorithm by including better spreading measures [5], using distributed computing to increase power, which could imply a formal parameter optimizatin for "greedy σ".

Another limit of our study is the $\lambda_i = cst$ hypothesis. Indeed, in reality, the more famous an influencer is (high node degree), the bigger λ_i should be. It would probably change a little bit our conclusion concerning the threshold between a lot of small of influencers, or a few very important.

References

[1] David Kempe, Jon Kleinberg, and Éva Tardos. "Maximizing the Spread of Influence Through a Social Network". In: *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '03. Washington, D.C.: ACM, 2003, pp. 137–146. ISBN: 1-58113-737-0. DOI: 10.1145/956750.956769. URL: http://doi.acm.org/10.1145/956750.956769.

[2] Maksim Kitsak et al. "Identification of influential spreaders in complex networks". In: *Nature physics* 6.11 (2010), p. 888.

[3] Hui Lin and Jeff Bilmes. "A Class of Submodular Functions for Document Summarization". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. HLT '11. Portland, Oregon: Association for Computational Linguistics, 2011, pp. 510–520. ISBN: 978-1-932432-87-9. URL: http://dl.acm.org/citation.cfm?id=2002472.2002537.

[4] Hui Lin and Jeff Bilmes. "Multi-document Summarization via Budgeted Maximization of Submodular Functions". In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. HLT '10. Los Angeles, California: Association for Computational Linguistics, 2010, pp. 912–920. ISBN: 1-932432-65-5. URL: http://dl.acm.org/citation.cfm?id=1857999.1858133.

[5] Ying Liu et al. "Accurate ranking of influential spreaders in networks based on dynamically asymmetric link weights". In: *Phys. Rev. E* 96 (2 2017), p. 022323. DOI: 10.1103/PhysRevE.96.022323. URL: https://link.aps.org/doi/10.1103/PhysRevE.96.022323.

[6] Zengfu Wang et al. "An accelerated continuous greedy algorithm for maximizing strong submodular functions". In: *Journal of Combinatorial Optimization* 30 (Jan. 2013). DOI: 10.1007/s10878-013-9685-x.