ARIEL UNIVERSITY

Information Diffusion and Collective Decision Making in Social Networks

A thesis submitted in partial fulfillment of the requirements for the degree Doctor of Philosophy

By

Yael Sabato

This work was prepared under the supervision of

Prof. Amos Azaria
Dr. Noam Hazon

Submitted to the Senate of Ariel University
June 2023

Contents

Abstract

This PhD thesis investigates three distinct aspects of social networks: Influence Maximization, Source Detection, and prediction of public goods games (PGN) outcomes. In the first chapter, we approach the problem of influence maximization from a utilitarian perspective, where messages with different propagation probabilities and utilities are considered. Through the analysis of bisubmodular functions, we propose a greedy algorithm with a tight approximation ratio of 1/2. Furthermore, we present a dynamic programming-based algorithm that surpasses the performance of the greedy approach in extensive simulations.

Moving on to the second chapter, we focus on the source detection problem in the Independent Cascade (IC) model, which models the diffusion of information or infection through social networks. We introduce an efficient method based on computing the stationary distribution of a Markov chain to tackle this problem under the Maximum Likelihood Estimation (MLE) approach. Through simulations, we demonstrate the superior effectiveness of our proposed method compared to existing state-of-the-art techniques.

In the third chapter, we delve into the ability of human subjects to predict the outcomes of public goods games on networks without prior knowledge of game theory and graph theory concepts. Through a survey involving 96 participants, we observe intriguing trends where a majority of participants provide predictions aligning with Pure Strategy Nash Equilibrium (PSNE). Our findings indicate that human intuition and reasoning, even in the absence of explicit knowledge, tend to converge towards Nash equilibrium in certain scenarios.

This thesis contributes to a comprehensive understanding of social networks by addressing computational challenges in influence maximization and source detection, as well as shedding light on human decision-making in predicting game outcomes. The results presented herein provide valuable insights for optimizing influence campaigns, improving source identification techniques, and understanding human behavior in networked environments.

1 Viral vs. Effective: Utility Based Influence Maximization

1.1 Abstract

The computational problem of Influence maximization concerns the selection of an initial set of nodes in a social network such that, by sending this set a certain message, its exposure through the network will be the highest. We propose to study this problem from a utilitarian point of view. That is, we study a model where there are two types of messages; one that is more likely to be propagated but gives a lower utility per user obtaining this message, and another that is less likely to be propagated but gives a higher utility. In our model the utility from a user that receives both messages is not necessarily the sum of the two utilities. The goal is to maximize the overall utility.

Using an analysis based on bisubmodular functions, we show a greedy algorithm with a tight approximation ratio of $\frac{1}{2}$. We develop a dynamic programming based algorithm that is more suitable to our setting and show through extensive simulations that it outperforms the greedy algorithm.

1.2 Introduction

In recent years, social networks have surged in popularity, enabling people to share information easily and interact with each other. One of the main properties of social networks is the fast spread of messages; due to the multiple links of the networks, when a user receives a message, she may transfer it to a subset of her neighbors in the network, which may, in turn, transfer the message to their neighbors, and so on. This phenomenon was used by several stakeholders to promote their goods, agendas or ideas. For example, marketing companies advertise by social networks [52], social movements reach the public in order to get their support [11], and politicians run several social network pages [50, 47]. Naturally, the research in this field is thriving.

One natural problem that arises is the Influence Maximization (IM) problem. In this problem one needs to select an initial set of users (of a given size), to receive a message, such that the message will reach the largest number of users in the network. Most of the research on the IM problem concentrates on measuring a spread of a messages by counting the number of users that received that message, and this measure is used even where there are multiple types of messages. However, in many setting the effect of a message depends on its type, and thus counting the number of users that received any message is not the appropriate objective. For example, suppose there is an association that is promoting anti-smoking by running a campaign on a social network. Assume that there are two potential types of anti-smoking advertisements. One advertisement consists of

humorous content[1], and thus it has a high potential to become viral and heavily spread throughout the network. However, this message has a low potential of affecting users to quit smoking. The second type of advertisement consists of harsh content[2], and it is thus more effective but less likely to spread. As another example, consider two advertisements for vaccine promotion. One advertisement includes graphic pictures of people who refused to vaccinate, and the other advertisement includes fun facts regarding the necessity of vaccination.

In this thesis we propose to study the IM problem from a utilitarian perspective. That is, we study a model where there are two types of messages; one that is more likely to be propagated but gives a lower utility per user obtaining this message, and another that is less likely to be propagated but gives a higher utility. Clearly, whenever a user receives the same message multiple times it does not affect the resulting utility. However, when a user receives messages from different types, it is natural to assume that the resulting utility should be at least as high as the utility from each one of the messages. On the other hand, the utility in this case should not be higher than the sum of utilities. Overall, the Utility Based Influence Maximization (UBIM) problem is to select two initial sets of users (with a given total size), one for each type of message that is sent, such that the sum of the resulting utilities will be maximized.

We first show that a greedy method for UBIM is guaranteed to reach at least 50% of the optimal solution. The analysis is based on the maximization of monotone and bisubmodular functions [39], which is an extension of the traditional approach that maximizes a submodular function [21]. We also prove that the approximation ratio is tight. We note that the greedy algorithm has a drawback in our setting. Specifically, since there are two types of messages, if the greedy algorithm decides to send one type of message to a specific user, this commitment may harm its performance later on, as it may turn out that the other type of message is more valuable at later stages. We thus introduce our Efficient Table-based Algorithm for Bisubmodular functions (ETAB), which does not commit to a specific type of message at early stages.

For the evaluation of our algorithms we use two graphs, a random graph and a graph based on a known social network (Digg). We compared the performance of ETAB with the greedy algorithm, its more efficient variant (Celf), and with additional baseline heuristics, in terms of achieved utility. Our results demonstrate that ETAB outperforms the other alternatives.

The contribution of this paper is threefold:

- We study the influence maximization problem from a utilitarian perspective. This is a natural extension, which allows to study the trade-off between sending viral and effective messages.

[1]See for example `https://www.youtube.com/watch?v=IKbxMIWCto0`

[2]See for example `https://www.youtube.com/watch?v=AIyqcST29wQ`

To the best of our knowledge, this model has not been investigated yet.

- We prove that a greedy algorithm achieves a tight approximation ratio of $\frac{1}{2}$, since the function in our model is bisubmodular.
- We introduce an Efficient Table-based Algorithm for Bisubmodular functions (ETAB) and show through simulation that it outperforms the greedy algorithm, as well as additional baselines, on two different graphs.

1.3 Background and Related Work

The problem of Influence Maximization was introduced by Domingos and Richardson [12]. Motivated by application of marketing, they point out that a satisfied customer is likely to recommend the product to her friends, which raises the probability for them to buy the product as well. Naturally, a good marketing strategy will be to filter the users of a certain network in order to find the most influential users. Domingos and Richardson model the problem as a Markov random field, and provide heuristics for choosing costumers with a large overall effect.

The seminal work of Kempe, Kleinberg, and Tardos [21] is the first to analyze the Influence Maximization as a discrete optimization problem. They show that the IM problem is NP-hard, but it can be analyzed as a maximization of a *monotone* and *submodular* set function. This problem was studied by [38], and shown to admit an approximation of $1 - 1/e \approx 63\%$. Thus, the IM problem has the same approximation ratio. However, unlike the problem studied by [38], in the IM problem the underlying set function cannot be evaluated exactly (in polynomial time) but it can be approximated by sampling, and thus the approximation ratio is slightly lower. Since the work of Kempe, Kleinberg, and Tardos, the IM problem received a high amount of attention, see the recent surveys [30],[2].

An important generalization of the IM problem, derives from the observation that many products are being promoted through a social network simultaneously. Some of them are complementary to each other and others are competitive. For example, buying a cellphone increases the probability of buying a cellphone case, but decreases the probability of buying a different type of cellphone. Therefore, there are several works that generalizes the IM problem such that there are multiple messages.

Specifically, Borodin, Filmus, and Oren [5] discuss competitive IM. That is, they analyze several models where the goal is to maximize the spread of technology A while there is a different competitive technology B that is also spread in the network. They show that the greedy approach of [21] is applicable only for a subset of their models.

Datta, Majumder, and Shrivastava model viral marketing of multiple products [10]. They assume that the products are independent of each other. That is, a user's decision to buy a product is independent of her decision to buy other products. They also assume that multiple messages can be initially sent to each user, but the number of such messages is limited. The objective of this work is to maximize the *number* of products that the users buy, since all products are equally counted. Datta, Majumder, and Shrivastava show a greedy algorithm for their problem with a $1/3-$approximation ratio.

Narayanam and Nanavati study a model where cross-sell among products is possible [36]. That is, in their model when a user buys the first product it increases the probability that she will buy a second product. While they define a utility for the objective, it is a simple sum over the utilities from buying the products separately. In addition, they study the IM problem according to the linear threshold model while we study the IM problem according to the independent cascade model (see Section 1.4.1 for details).

In our work we show that a greedy algorithm obtains an approximation ration of $\frac{1}{2}$. This result is obtained by proving that the utility function in UBIM is monotone and bisubmodular. Ohsaka and Yoshida [39] were the first to show that a greedy algorithm on a $k-$submodular and monotone set function achieves an approximation ratio of $\frac{1}{2}$. They further show that their solution can be applied to the IM problem with multiple messages. Their objective function is to maximize the sum of users receiving these messages. Overall, none of these works extend the IM problem to the setting in which every message has its own (different) utility while a user receiving multiple messages might yield a value that is different from the sum of utilities.

Another related extension of the IM problem is the topic-aware Influence Maximization problem, which was introduced by Barbieri, Bonchi, and Manco [3] and Chen, Lin, and Yang [9]. In this model the influence between a pair of users may differ depending on the topic. Barbieri, Bonchi, and Manco introduce the model, and provide methods for learning the diffusion probabilities from data of past diffusion. Chen, Lin, and Yang study efficient algorithms for the IM problem in this setting. Note that the objective function is identical to that of the standard IM problem, and the existence of multiple topics affects only the probabilities of influences among the users.

A different perspective is studied by Li et al. [29]. They consider multiple types of messages, and use an agent-based modelling to study the diffusion process of the different influences.

1.4 Preliminaries

1.4.1 The independent cascade model

The research on the IM problem has considered two main models of diffusion: the linear threshold model, and the independent cascade model. We focus on the independent cascade model and generalize it to our setting. The independent cascade model works as follows. The social network is represented by a directed graph $G = (V, E)$, where each user of the network is represented by a node and every connection between two users is an edge. The process of diffusion concerns a message that is propagated thorough the network. During this process each node can either become active or inactive, where an active node indicates that the associated user is influenced by the message. For every edge $u \to v \in E$, let $p^{uv} \in [0, 1]$ be the probability of the influence of u on v. That is, p^{uv} is the probability for v to become active after she received the message from u. The process starts with an initial seed set $S \subseteq V$, such that all the nodes of S are initialized with the message, and thus they become active. The process then unfolds in discrete steps according to the following rule. Every node $v \in V$ that becomes active, activates each currently inactive neighbour w with probability p^{vw}. Moreover, if multiple neighbors of a vertex w try to activate it at the same time, their attempts are considered in an arbitrary order. v does not attempt to activate its neighbours again. The process runs until no more activations occur.

As shown by Kempe, Kleinberg, and Tardos [21], the diffusion process is equivalent to first selecting the participating edges according to their probabilities, (by a series of coin flips with the corresponding probability) obtaining a graph of connections $G' = (V, E')$. Then, every node $v \in V$ of the graph that has a directed path starting from one of the nodes of the seed and ending at v is assumed to be active. We note that in G' all edges, E', have a fixed probability of 1.0.

1.4.2 Monotone and Bisubmodular set functions

We now provide the basic definitions of a bisubmodular function. This function is later used to model the utility in our setting.

Let U be a set of n nodes. 3^U is the set of all possible tuples of two disjoint subsets of U. That is, $X = (X_1, X_2) \in 3^U$ if $X_1, X_2 \subseteq U$ and $X_1 \cap X_2 = \emptyset$. Note that every such tuple can be viewed as a vector of size n over $\{0, 1, 2\}$. Let $f : 3^U \to \mathbb{R}^+$ be a function that takes two disjoint subsets of U and outputs a non negative real number.

For every tuple $X = (X_1, X_2) \in 3^U$ and $e \in U \setminus (X_1 \cup X_2)$, let $\Delta_{e,1}(X) = f(X_1 \cup \{e\}, X_2) - f(X_1, X_2)$. That is, $\Delta_{e,1}(X)$ is the contribution of adding e to X_1, as measured by f. Similarly, let $\Delta_{e,2}(X) = f(X_1, X_2 \cup \{e\}) - f(X_1, X_2)$.

Definition 1 (Monotonicity)**.** *A function $f : 3^U \to \mathbb{R}^+$ is monotone if for every tuple $X = (X_1, X_2) \in 3^U$ and every $e \in U \setminus (X_1 \cup X_2)$, it holds that $\Delta_{e,1}(X) \geq 0$, and $\Delta_{e,2}(X) \geq 0$. That is, adding nodes does not reduce the value of f.*

For completeness, we first provide the definition of a submodular function. Note that submodular functions receive a single set as their input.

Definition 2 (Submodularity)**.** *A function $f : 2^U \to \mathbb{R}^+$ is submodular if for every two subsets $X, Y \subseteq U$ it holds that:*

$$f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y)$$

Definition 3 (Bisubmodularity)**.** *A function $f : 3^U \to \mathbb{R}^+$ is bisubmodular if for every two tuples $X = (X_1, X_2) \in 3^U$ and $Y = (Y_1, Y_2) \in 3^U$ it holds that:*

$$f(X) + f(Y) \geq f(X \sqcup Y) + f(X \sqcap Y)$$

where,

- $f(X \sqcup Y) = f(X_1 \cup Y_1 \setminus (X_2 \cup Y_2), X_2 \cup Y_2 \setminus (X_1 \cup Y_1))$.
- $f(X \sqcap Y) = f(X_1 \cap Y_1, X_2 \cap Y_2)$.

An equivalent definition for bisubmodularity (see [49]), is when both properties of *orthant submodularity* and *pairwise monotonicity* hold.

Definition 4 (Orthant Submodularity)**.** *A function $f : 3^U \to \mathbb{R}$ is orthant submodular if for every tuples $X = (X_1, X_2)$ and $Y = (Y_1, Y_2)$ such that $X_1 \subseteq Y_1$ and $X_2 \subseteq Y_2$, and every item $e \in U \setminus (Y_1 \cup Y_2)$, it holds that $\Delta_{e,1}(X) \geq \Delta_{e,1}(Y)$ and similarly $\Delta_{e,2}(X) \geq \Delta_{e,2}(Y)$.*

Definition 5 (Pairwise Monotonicity)**.** *A function $f : 3^U \to \mathbb{R}$ is pairwise monotone if for every tuple $X = (X_1, X_2)$ and every $e \in U \setminus (X_1 \cup X_2)$, $\Delta_{e,1}(X) + \Delta_{e,2}(X) \geq 0$.*

Note that monotonicity implies pairwise monotonicity.

1.5 Our model

In our work we consider the diffusion process of two types of messages M_1 and M_2 in a social network. The social network is represented by a directed graph $G = (V, E)$, where each $v \in V$ represents a user in the social network, and each edge $u \to v \in E$ represents the influence of

u on v. Each user can be activated by each of the two messages. Thus, there are four possible combinations for activeness; a user can be active with regard to only M_1, only M_2, active with both the messages or inactive. For every edge $u \to v$ we define $p_1^{uv}, p_2^{uv} \in [0, 1]$ to be the influence probabilities. p_1^{uv} is the probability that user u activates user v with regard to M_1. Similarly, p_2^{uv} is the influence probability of u on v with regard to M_2.

For the diffusion process we consider a generalization of the independent cascade model (see Section 1.4.1). Our process starts with two initial seed sets $S_1, S_2 \subseteq V$, $S_1 \cap S_2 = \phi$, such that all the nodes of S_1 are initialized with message M_1 and all the nodes of S_2 are initialized with message M_2. The nodes of the seed sets S_1, S_2 are assumed to be activated by the initial corresponding messages. The process then unfolds in discrete steps according to the following rule. Let $i, j \in \{1, 2\}$, $i \neq j$. Every node v that became active with message M_i activates each neighbour w not activated yet with message M_i, with probability p_i^{vw}. In addition, if multiple neighbors of a vertex w try to activate it at the same time, their attempts are considered in an arbitrary order. v does not attempt to activate its neighbours with message M_i again. The process runs until no more activations are possible.

As mentioned in Section 1.4.1, the diffusion process of the IC model is equivalent to first selecting the participating edges according to their probabilities, then, every node of the graph that can be reached by a directed path from a node in the seed is considered active. For our model, we need to bring to consideration the fact that each edge has two different diffusion parameters. For the selection of the participating edges we use a series of $2|E|$ coin flips π, (two for each edge), obtaining a multi-graph $G_\pi = (V, E_1 \cup E_2)$ such that the edges of E_1 and E_2 represents the connections of the nodes regarding the messages M_1 and M_2 respectively. Every node $v \in V$ that can be reached by a directed path from a node in S_1 is assumed to be active with regard to M_1. (Note that all the edges of this directed path should be from E_1). Similarly, every node $v \in V$ that can be reached by a directed path from a node in S_2 is assumed to be active with regard to M_2. (Again, all the edges of this directed path are from E_2). We note that in G_π all edges $e \in E_1$, have a fixed probability $p_1^e = 1.0$ and all edges $e \in E_2$, have a fixed probability $p_2^e = 1.0$.

Given a siries of coinflips π and a graph G_π, Let $A_1(S_1)$ be the set of all nodes that are active with M_1 at the end of the process, when S_1 is the seed set for M_1. Similarly, Let $A_2(S_2)$ be the set of all nodes that are active with M_2 at the end of the process, when S_2 is the seed set for M_2. Note that $A_1(S_1)$ depends only on the set S_1 and $A_2(S_2)$ depends only on the set S_2. Further note that $A_1(S_1) \cap A_2(S_2)$ consist of nodes that are active with both messages, $A_1(S_1) \setminus A_2(S_2)$ consist of nodes that are active with M_1 but are inactive with M_2, and $A_2(S_2) \setminus A_1(S_1)$ consist of nodes that are active with M_2 but are inactive with M_1.

We assume that each active node is associated with some utility. Let $u_1 \geq 0$ be the utility for nodes that are active with M_1 but are inactive with M_2. Similarly, let $u_2 \geq 0$ be the utility for each node that is active only with M_2, and let $u_{1,2}$ be the utility for nodes that are active with both messages.

We define the utility function $\sigma_\pi(S_1, S_2)$ to be the sum of every active node at the end of the process multiplied with the corresponding utility, i.e,

$$\sigma_\pi(S_1, S_2) = u_1 \cdot |A_1 \setminus A_2| + u_2 \cdot |A_2 \setminus A_1| + u_{1,2} \cdot |A_1 \cap A_2|$$

Where we abbreviate $A_1(S_1)$ and $A_2(S_2)$ to A_1 and A_2 respectively. Finally, the overall utility function is the weighted average of each $\sigma_\pi(S_1, S_2)$ multiplied by the probability for π to occur, $p(\pi)$:

$$\sigma(S_1, S_2) = \sum_\pi \sigma_\pi(S_1, S_2) \cdot p(\pi)$$

Definition 6 (UBIM)**.** *Given an integer B (the budget), the Utility Based Influence Maximization problem is to find seed sets $X = (S_1, S_2)$ such that $|S_1| + |S_2| \leq B$, that maximize the utility $\sigma(S_1, S_2)$.*

1.6 Algorithms

1.6.1 The greedy approach

We extend the greedy algorithm that was presented by Kempe, Kleinberg, and Tardos to the UBIM problem. It runs as follows. First two sets S_1 and S_2 are initiated with empty sets. At any step $1 \leq i \leq B$ the algorithm greedily chooses a node $e \in V \setminus (S_1 \cup S_2)$ and a message type $i \in \{1, 2\}$ such that adding e to S_i gives the largest marginal gain to $\sigma(S_1, S_2)$ (see Algorithm 1). In this section, we show that if $\max(u_1, u_2) \leq u_{1,2} \leq u_1 + u_2$, then our utility function $\sigma(\cdot, \cdot)$, is monotone and bisubmodular, and therefore the greedy algorithm guarantees at least 50% of the optimal solution.

Algorithm 1 Greedy

Input: A directed graph $G = (V, E)$ and an integer B.

Output: Two sets $S_1, S_2 \in V$, with $|S_1| + |S_2| = B$.

Initiate $S_1 \leftarrow \emptyset$, $S_2 \leftarrow \emptyset$

for $i = 1$ *to* B **do**

  $u' \leftarrow \arg\max_{u \in V \setminus (S_1 \cup S_2)} \{\Delta_{u,1}(S_1, S_2)\}$

  $u'' \leftarrow \arg\max_{u \in V \setminus (S_1 \cup S_2)} \{\Delta_{u,2}(S_1, S_2)\}$

  if $u' \geq u''$ **then**

    $S_1 \leftarrow S_1 \cup \{u'\}$

  else

    $S_2 \leftarrow S_2 \cup \{u''\}$

  end

end

Return: (S_1, S_2);

Theorem 1. *If* $\max(u_1, u_2) \leq u_{1,2} \leq u_1 + u_2$ *then the function* $\sigma(\cdot, \cdot)$ *is monotone and bisubmodular.*

Proof. We start with monotonicity, we need to show that for every tuple $X = (X_1, X_2) \in 3^U$, and every node $e \in U \setminus (X_1 \cup X_2)$,

$$\sigma(X_1 \cup \{e\}, X_2) \geq \sigma(X_1, X_2). \tag{1}$$

$$\sigma(X_1, X_2 \cup \{e\}) \geq \sigma(X_1, X_2). \tag{2}$$

We will show (1) by:

$$\Delta_{e,1}(X) = \sigma(X_1 \cup \{e\}, X_2) - \sigma(X_1, X_2) \geq 0. \tag{3}$$

For (2) the analyze is symmetric.

For given coin flip π and a corresponding graph G_π, Fix a tuple $X = (X_1, X_2)$ and a vertex $e \in U \setminus (X_1 \cup X_2)$, Let $A_1 = A_1(X_1)$, $A_2 = A_2(X_2)$ and let $B_1 = A_1(X_1 \cup \{e\})$. Note that $A_1 \subseteq B_1$ (see Figure 1). We get:

$$\sigma_\pi(X_1, X_2) = u_1 \cdot |A_1 \setminus A_2| + u_2 \cdot |A_2 \setminus A_1| + u_{1,2} \cdot |A_1 \cap A_2|$$

After adding e to S_1, we have:

$$\sigma_\pi(X_1 \cup \{e\}, X_2) = u_1 \cdot |B_1 \setminus A_2| + u_2 \cdot |A_2 \setminus B_1| + u_{1,2} \cdot |B_1 \cap A_2|$$

Let $\Delta_{\pi,e,1}(X) = \sigma_\pi(X_1 \cup \{e\}, X_2) - \sigma_\pi(X_1, X_2)$. Observe that in $B_1 \setminus (A_1 \cup A_2)$ there are nodes that where inactive for both messages and became active with M_1. Furthermore, in $(B_1 \cap A_2) \setminus A_1$ there are nodes that where active only with M_2 and became active with both messages. Therefore:

$$\Delta_{\pi,e,1}(X) = u_1 \cdot |B_1 \setminus (A_1 \cup A_2)| + (u_{1,2} - u_2) \cdot |(B_1 \cap A_2) \setminus A_1| \geq 0$$

where the inequality is due to the assumption that $u_{1,2} \geq \max(u_1, u_2)$, (therefore $u_{1,2} - u_2 \geq 0$), and that the sizes of the mentioned sets are non negative. We showed that the for every π, the function $\sigma_\pi(\cdot, \cdot)$ is monotone. Now, since $\sigma(\cdot, \cdot)$ is a convex combination of monotone functions, it is monotone as well.

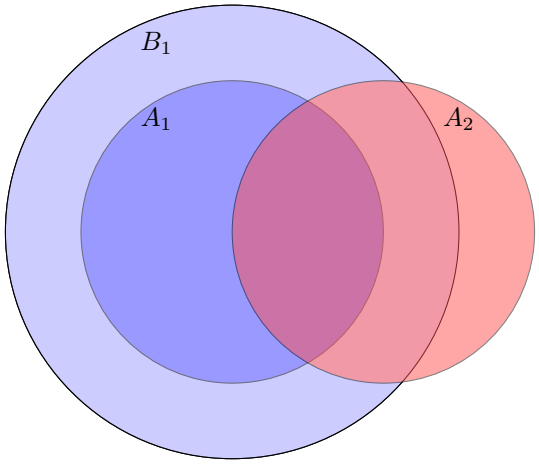

Figure 1: Venn's diagram with 3 sets, such that $A_1 \subseteq B_1$

For the bisubmodularity; we need to show that the properties of *orthant submodularity* and *pairwise monotonicity* hold (See Definitions 4 and 5 in Section 1.4.2). Since we showed monotonicity, pairwise monotonicity stems. For the orthant submodularity, Let π be a set of $2|E|$ coin flips, and let $G_\pi = (V, E_1 \cup E_2)$ be the corresponding multigraph. Let $X = (X_1, X_2) \in 3^U$ and $Y = (Y_1, Y_2) \in 3^U$ such that $X_1 \subseteq Y_1$ and $X_2 \subseteq Y_2$. Let $A_1 = A_1(X_1)$, $A_2 = A_2(X_2)$, $B_1 = A_1(Y_1)$ and $B_2 = A_2(Y_2)$. note that $A_1 \subseteq B_1$ and $A_2 \subseteq B_2$. In addition, Let $e \in U \setminus (Y_1 \cup Y_2)$. We will prove the claim for the case of adding e to X_1. The corresponding case (of adding e to X_2) is symmetric.

We label the nine[3] different areas of those four sets as follows (see Figure 2 for a Venn diagram):

[3]Note that a four set Venn diagram has $2^4 = 16$ different areas, here since $A_1 \subseteq B_1$ and $A_2 \subseteq B_2$ the following seven areas are empty; $A_1 \setminus (A_2 \cup B_1 \cup B_2)$, $A_2 \setminus (A_1 \cup B_1 \cup B_2)$, $A_1 \cap A_2 \setminus (B_1 \cup B_2)$, $(A_1 \cap A_2 \cap B_1) \setminus B_2$, $(A_1 \cap A_2 \cap B_2) \setminus B_1$, $A_1 \cap B_2 \setminus (B_1 \cup A_2)$, $A_2 \cap B_1 \setminus (B_2 \cup A_1)$.

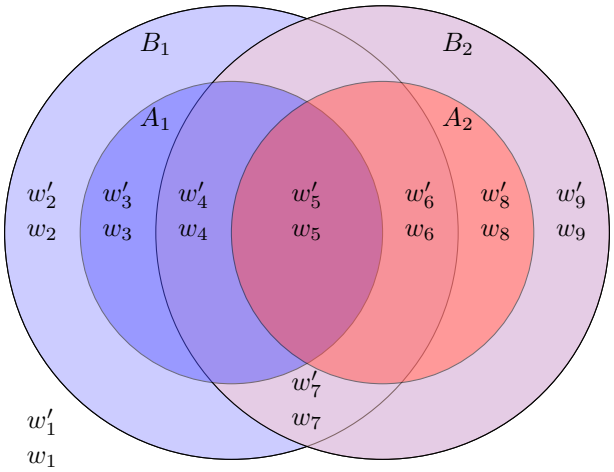

Figure 2: Venn's diagram with 4 sets such that $A_1 \subseteq B_1$ and $A_2 \subseteq B_2$

- $C_1 = U \setminus (B_1 \cup B_2)$
- $C_2 = B_1 \setminus (B_2 \cup A_1)$
- $C_3 = A_1 \setminus B_2$
- $C_4 = (B_2 \cap A_1) \setminus A_2$
- $C_5 = A_1 \cap A_2$
- $C_6 = (B_1 \cap A_2) \setminus A_1$
- $C_7 = (B_1 \cap B_2) \setminus (A_1 \cup A_2)$
- $C_8 = A_2 \setminus B_1$
- $C_9 = B_2 \setminus (A_2 \cup B_1)$

For every $1 \leq k \leq 9$, let W'_k be the set of nodes in C_k that are affected by e with regard to M_1. In other words; for every node $w \in W'_k$, G_π contains a directed path from e to w. (Note that all the path edges are from E_1). Moreover, let W_k be the set of nodes in C_k that are not affected by e. We denote be w'_k and w_k the number of nodes in W'_k and W_k respectively.

We need to show that $\Delta_{e,1}\sigma(X) - \Delta_{e,1}\sigma(Y) \geq 0$. Since we focus on the graph G_π, let $\Delta_{\pi,e,1}(X) = \sigma_\pi(X_1 \cup \{e\}, X_2) - \sigma_\pi(X_1, X_2)$ and Let $\Delta_{\pi,e,1}(Y) = \sigma_\pi(Y_1 \cup \{e\}, Y_2) - \sigma_\pi(Y_1, Y_2)$. We first note that:

$$\sigma_\pi(X_1, X_2) = u_1 \cdot (w_3 + w_3' + w_4 + w_4') + u_2 \cdot (w_6 + w_6' + w_8 + w_8') + u_{1,2} \cdot (w_5 + w_5') \quad (4)$$

After adding e to X_1, the nodes of W_1', W_2', W_7' and W_9', which where inactive, became active with M_1, the nodes of W_3', W_4' and W_5' did not change, and the nodes of W_6' and W_8', which where active only with M_2, became active with both messages. Thus:

$$\sigma_\pi(X_1 \cup \{e\}, X_2) = u_1 \cdot (w_3 + w_3' + w_4 + w_4' + w_1' + w_2' + w_7' + w_9') + u_2 \cdot (w_6 + w_8) + u_{1,2} \cdot (w_5 + w_5' + w_8' + w_6') \quad (5)$$

We get,

$$\Delta_{\pi,e,1}\sigma(X) = u_1 \cdot (w_1' + w_2' + w_7' + w_9') - u_2 \cdot (w_6' + w_8') + u_{1,2} \cdot (w_6' + w_8') \quad (6)$$

Similarly, the analysis for $\Delta_{\pi,e,1}\sigma(Y)$ gives:

$$\sigma_\pi(Y_1, Y_2) = u_1 \cdot (w_2 + w_2' + w_3 + w_3') + u_2 \cdot (w_8 + w_8' + w_9 + w_9') + u_{1,2} \cdot (w_4 + w_4' + w_5 + w_5' + w_6 + w_6' + w_7 + w_7') \quad (7)$$

After adding e to Y_1, all nodes in W_1', which were inactive, became active with M_1. the nodes in W_8' and W_6', which were active only with M_2, became active with both messages. This gives us:

$$\sigma_\pi(Y_1 \cup \{e\}, Y_2) = u_1 \cdot (w_3 + w_3' + w_2 + w_2' + w_1') + u_2 \cdot (w_8 + w_9) + u_{1,2} \cdot (w_4 + w_4' + w_5 + w_5' + w_6 + w_6' + w_7 + w_7' + w_8' + w_9') \quad (8)$$

Therefore:

$$\Delta_{\pi,e,1}\sigma(Y) = u_1 \cdot (w_1') - u_2 \cdot (w_8' + w_9') + u_{1,2} \cdot (w_8' + w_9') \quad (9)$$

Finally, let us now show that (6)$-$(9)≥ 0:

$$\begin{aligned}\Delta_{\pi,e,1}\sigma(X) - \Delta_{\pi,e,1}\sigma(Y) &= u_1 \cdot (w_2' + w_7' + w_9') - u_2 \cdot (w_6') + u_2(w_9') + u_{1,2} \cdot (w_6') - u_{1,2} \cdot (w_9') \\ &= u_1 \cdot (w_2' + w_7') + (-u_2 + u_{1,2}) \cdot (w_6') + (u_1 + u_2 - u_{1,2}) \cdot (w_9') \geq 0 \quad (10)\end{aligned}$$

where the inequality is due to assumption that $\max(u_1, u_2) \leq u_{1,2} \leq u_1 + u_2$, and that all the w's are non negative integers. We get that $\sigma_\pi(\cdot, \cdot)$ is bisubmodular. Finally, $\sigma(\cdot, \cdot)$ is a convex combination of bisubmodular functions, therefore it is also a bisubmodular function. □

An important improvement to the time complexity of the greedy algorithm is the Celf algorithm, which is an extension of the algorithm introduced by [33] and [27] to UBIM. The algorithms starts with initializing two empty sets S_1 and S_2. For each node the algorithm calculates a

tuple $T = (n, a_1, a_2, valid)$, such that $T(n)$ is the node name, $T(a_1), T(a_2) \in \mathbb{R}$ are the expected addition to the overall utility when adding n to S_1 or S_2 respectively, and $T(valid) \in \mathbb{N}$ is the index of the iteration of the last evaluation of $T(a_1)$ and $T(a_2)$. The algorithm uses a queue that sorts the tuples by the value $\max(a_1, a_2)$ of each tuple. At every iteration, a tuple T is pulled from the queue; if its *valid* value is equal to the current iteration, then $T(n)$ is added to S_1 or S_2 (according to $\max(a_1, a_2)$). Otherwise, $T(n)$'s utility is sampled again, $T(a_1)$, $T(a_2)$ and $T(valid)$ are updated and T is inserted back into the queue. Finaly, at the end of B iterations, the algorithm outputs S_1 and S_2. See Algorithm 2.

Algorithm 2 Celf (cost effective lazy forward selection)

$Q \leftarrow$ a priority order queue, sorted by $\max(a_1, a_2)$
for $i = 1$ *to* n **do**
  $a_1 \leftarrow \sigma(\{i\}, \emptyset)$
  $a_2 \leftarrow \sigma(\emptyset, \{i\})$
  $Q.add((i, a_1, a_2, 1))$
end
$S_1 \leftarrow \emptyset$
$S_2 \leftarrow \emptyset$
for $i = 1$ *to* B **do**
  $T \leftarrow Q.poll$
  if *T(valid) = i* **then**
    if $T(a_1) = \max(T(a_1), T(a_2))$ **then**
      $S_1 \leftarrow S_1 \cup \{T(n)\}$
    else
      $S_2 \leftarrow S_2 \cup \{T(n)\}$
    end
  else
    reevaluate $T(a_1)$ and $T(a_2)$
    $T(valid) = i$
    $Q.add(T)$
  end
end
return (S_1, S_2).

1.6.2 The dynamic programming approach

One major drawback of the greedy algorithm (and Celf) is that, since there are two types of messages, if it decides to send one type of message to a specific user, this commitment may harm its performance later on, as it may turn out that the other type of message is more valuable at later stages. For example, consider Figure 3. The red arrows correspond to message 1, and the blue arrows correspond to message 2. Note that the influence probability is 1.0. For this example assume that $u_1 = 1$, $u_2 = 1.5$, and $u_{1,2} = 1.5$, and that $B = 2$. Clearly, at the first stage the greedy algorithm will select node A and add it to S_1; this will result in a utility of 5. In the second step, the greedy algorithm will select node B and add it to S_2; this will result in an additional utility of 1.5, thus resulting in a total utility of 6.5. However, an algorithm that would not commit to using the message 1 in the first step, could add nodes B and C to S_2 and yield a utility of 7.5.

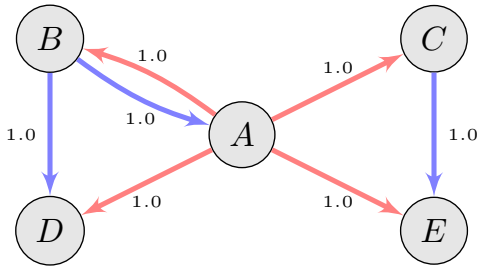

Figure 3: A graph demonstrating the problem with the greedy algorithm committing to a specific message type at early stages.

To overcome this problem, we introduce our Efficient Table-based Algorithm for Bisubmodular functions (ETAB), which does not commit to a specific type of message at early stages. We first describe a dynamic programming based algorithm (TAB), and then describe a method for using insights from Celf to create the Efficient version of TAB (i.e. ETAB). In order not to commit to a specific type of message TAB builds its solution in two dimensions.

TAB uses the upper triangle of a table Mat of size $(B+1) \times (B+1)$. Each cell of Mat consists of a tuple $T = (S_1, S_2)$, such that $T(S_1), T(S_2) \subseteq V$ are disjoint seed sets. In each (i, j) cell of the table, it holds that $|S_1| = i$ and $|S_2| = j$. TAB starts with the tuple $T = (\emptyset, \emptyset)$ in cell $(0, 0)$. The algorithm then fills in the first row; for every $1 \leq i \leq B$, let $R_{i-1} = (R_1, \emptyset)$ be the tuple in cell $(i-1, 0)$. The node $n \in V \setminus R_1$, that maximizes the value of the marginal gain to $\sigma(R_1, \emptyset)$, (i.e. the output of $\max_{n \in V \setminus R_1}(\Delta_{n,1}(R_1, \emptyset))$) is selected. Next the tuple $R_i = (R_1 \cup \{n\}, 0)$ is stored in cell $(i, 0)$. Similarly, TAB fills in the first column; for every $1 \leq i \leq B$, let $C_{i-1} = (\emptyset, C_2)$ be the tuple in cell $(0, i-1)$. The node $n \in V \setminus C_2$, that maximizes the value of the marginal gain to

$\lvert S_1\rvert$ \ $\lvert S_2\rvert$	0	1	2
0	0 $(\emptyset, \emptyset)$	5 $(\{A\}, \emptyset)$	5 $(\{A, C\}, \emptyset)$
1	4.5 $(\emptyset, \{B\})$	6.5 $(\{A\}, \{B\})$	-
2	7.5 $(\emptyset, \{B, C\})$	-	-

Table 1: The table that TAB builds when running on the settings of Figure 3. TAB's output is $(\emptyset, \{B, C\})$, and the corresponding utility value is 7.5.

$\sigma(\emptyset, C_2)$ is selected, and the tuple $C_i = (0, C_2 \cup \{n\})$ is stored in cell $(0, i)$. The next phase is to fill in the cells of the middle; For each cell (k, l) TAB builds two optional tuples; T' is created by greedily adding an unselected node to seed 1 of the tuple in cell $(k-1, l)$. Similarly, T'' is ceated by greedily adding an unselected node to seed 2 of the tuple in cell $(k, l-1)$. TAB Then picks the tuple with the higher expected utility and store it in the cell (k, l).

After filling up all the cells of upper triangel of the table (All the cells with indexes (i, j) such that $i + j \leq B$), TAB outputs the tuple with the maximal expected utility over the diagonal cells of the table (where $i + j = B$). (see Algorithm 3).

Reconsidering the example of the graph in Figure 3. We show in Table 1 the outcome of applying TAB on those settings. TAB will output the maximum utility of the tuples in the diagonal (light blue) cells. Thus, the output utility of TAB is 7.5.

We believe that the approximation ratio of the TAB algorithm is very close to 50%. Moreover, Lemma 2 shows that TAB cannot guarantee an approximation ratio that is higher than 50%.

Lemma 2. *The approximation ratio of the TAB algorithm is at most $50\% + \varepsilon$. In other words; let S_{TAB} be the output of TAB and let O be the optimal solution, then there is a case in which $f(S_{TAB}) \leq \frac{1}{2} f(O) + \varepsilon$.*

Proof. We present an example where TAB outputs a solution that is just slightly higher then 50% of the optimal value. Let f be a monotone and bisubmodular set function. Let $U = \{1, 2\}$, the function values are represented in Table 2. Since we assume that f is monotone and bisubmodular we get the following inequalities:

$$\max(x_1, x_2) \leq z_1 \leq x_1 + x_2, \quad \max(x_1, y_2) \leq z_2 \leq x_1 + y_2,$$
$$\max(x_2, y_1) \leq z_3 \leq x_2 + y_1, \quad \max(y_1, y_2) \leq z_4 \leq y_1 + y_2.$$

Algorithm 3 TAB (Table-based Algorithm for Bisubmodular functions)

$Mat \leftarrow$ a $(B+1) \times (B+1)$ matrix of tuples.
$Mat[0][0] \leftarrow (\emptyset, \emptyset)$
for $i = 1$ *to* B **do**
  $S_1 \leftarrow Mat[i-1][0](S_1)$
  $n \leftarrow \arg\max_{n \in V \setminus S_1}(\Delta_{n,1}(S_1, \emptyset))$
  $Mat[i][0] \leftarrow (S_1 \cup \{n\}, \emptyset)$
end
for $i = 1$ *to* B **do**
  $S_2 \leftarrow Mat[0][i-1](S_2)$
  $n \leftarrow \arg\max_{n \in V \setminus S_2}(\Delta_{n,2}(\emptyset, S_2))$
  $Mat[0][i] \leftarrow (\emptyset, S_2 \cup \{n\})$
end
for $i = 1$ *to* B **do**
  for $j = 1$ *to* $(B - i)$ **do**
    $L_1 \leftarrow Mat[i-1][j](S_1)$
    $L_2 \leftarrow Mat[i-1][j](S_2)$
    $l \leftarrow \arg\max_{n \in V \setminus (L_1 \cup L_2)}(\Delta_{n,1}(L_1, L_2))$
    $U_1 \leftarrow Mat[i][j-1](S_1)$
    $U_2 \leftarrow Mat[i][j-1](S_2)$
    $u \leftarrow \arg\max_{n \in V \setminus (U_1 \cup U_2)}(\Delta_{n,2}(U_1, U_2))$
    if $\sigma(L_1 \cup \{l\}, L_2) > \sigma(U_1, U_2 \cup \{u\})$ **then**
      $Mat[i][j] \leftarrow (L_1 \cup \{l\}, L_2)$
    else
      $Mat[i][j] \leftarrow (U_1, U_2 \cup \{u\})$
    end
  end
end
$(i, j) \leftarrow \arg\max_{i+j=B}(\sigma(Mat[i][j]))$
return $Mat[i][j]$

$B=0$	$f(\emptyset,\emptyset)=0$	
$B=1$	$f(\{1\},\emptyset)=x_1$	$f(\emptyset,\{1\})=y_1$
	$f(\{2\},\emptyset)=x_2$	$f(\emptyset,\{2\})=y_2$
$B=2$	$f(\{1,2\},\emptyset)=z_1$	$f(\{2\},\{1\})=z_3$
	$f(\{1\},\{2\})=z_2$	$f(\emptyset,\{1,2\})=z_4$

Table 2: A monotone and bisubmodular function for the proof of Lemma 2.

$\lvert S_1\rvert$ \ $\lvert S_2\rvert$	0	1	2
0	0 $(\emptyset,\emptyset)$	$x_1=p+\varepsilon$ $(\{1\},\emptyset)$	$z_1=p+\varepsilon$ $(\{1,2\},\emptyset)$
1	$y_2=p+\varepsilon/3$ $(\emptyset,\{2\})$	$z_2=p+\varepsilon$ $(\{1\},\{2\})$	-
2	$z_4=p+\varepsilon/3$ $(\emptyset,\{1,2\})$	-	-

Table 3: The table that TAB builds when running on the settings of the function given in Table 2.

Where for every $1 \leq i \leq 4$, the lower bound of z_i is due to the monotonicity of f and the upper bound on z_i is due to the bisubmodularity of f.

Let $p \in \mathbb{R}^+$ and let $\varepsilon > 0$. We further assume that $x_1 = p+\varepsilon$, $x_2 = p+2\varepsilon/3$, $y_1 = p$ and $y_2 = p-\varepsilon/3$. In addition, $z_3 = 2p+2\varepsilon/3$, $z_1 = z_2 = p+\varepsilon$ and $z_4 = p+\varepsilon/3$. Note that $z_3 \geq max(z_1, z_2, z_4)$, i.e. z_3 is the optimal solution when $B=2$. Table 3 shows the outcome of running TAB on the settings of f with $B=2$:

As we described the algorithm, TAB will output the tuple that corresponds to $\max(z_1, z_2, z_4)$, Thus the output will be $z_1 = p+\varepsilon$. On the other hand, 50% of the optimal solution is $\frac{1}{2}z_3 = \frac{1}{2}(2p+2\varepsilon/3) = p+\varepsilon/3$. Indeed, if we add ε to half the optimal solution we get $p+4\varepsilon/3$, which is greater than z_1. □

Corollary 3. *The approximation ratio of the greedy algorithm is at most* $50\%+\varepsilon$. *I.e. the approximation ratio of the greedy algorithm is tight.*

Proof. Observe that if the greedy algorithm is applied to the settings of the function in the proof of Lemma 2, the greedy algorithm will output the value of z_1 or z_2 which is $p+\varepsilon$, while the optimum value is $2p+2\varepsilon/3$. □

Unfortunately, TAB is computationally expensive; However, as in the greedy algorithm, we may use the queue method to order the nodes and reduce the amount of evaluations. This brings us to our final algorithm, the Efficient version of TAB (i.e. ETAB). The general idea is similar to that of TAB except that every tuple contains additional two queues Q_1 and Q_2. Each queue take the unselected nodes of each cell and sorts them in decreasing order of the expected contribution they might add to the overall utility (see Algorithm 4).

Like in TAB, ETAB uses the upper triangle of a table Mat of size $(B+1) \times (B+1)$. Each cell of Mat consists of a tuple $T = (S_1, S_2, u, Q_1, Q_2)$, such that $T(S_1), T(S_2) \subseteq V$ are disjoint seed sets. $T(u) \in \mathbb{R}^+$ is the expected utility $\sigma(T(S_1), T(S_2))$. Moreover, $T(Q_1)$ and $T(Q_2)$ are two priority queues that sort all the nodes in $V \setminus (T(S_1) \cup T(S_2))$ in decreasing order of their expected contributions $\Delta_{T(n),1}(T(S_2), T(S_2))$ and $\Delta_{T(n),2}(T(S_2), T(S_2))$, respectively. In addition, for each (i, j) cell of the table, it holds that $|S_1| = i$ and $|S_2| = j$. ETAB runs similarly to TAB, except that when ever it needs to select a node in order to add it to a certain seed of a certain cell, it uses the cell's priority queue. Thus reducing the amount of the function evaluations (see Algorithm 4).

We note that the space complexity of ETAB is very high since we store many queues (each queue has space complexity of $O(n)$). Each cell has two queues and we use $(B+1)^2/2 = O(B^2)$ cells. To slightly reduce space complexity, after calculating the tuples T' and T'' of the last loop, we can free the queues of the *above* cell, as we do not use them again. Thus, the number of queues we store during the running of the algorithm is at most $2(2B+1) = O(B)$, therefore the overall space complexity of ETAB is $O(Bn)$.

1.7 Experiments

For evaluating the performance of ETAB and comparing it to the greedy algorithm, Celf and additional baselines, we built two graphs that include diffusion parameters on their edges. Our first graph is a randomly induced graph, with $1,000$ nodes and a density of 0.001. That is, each directed edge was inserted to the graph with a probability of 0.001. Then, for every edge $u \to v$, we assigned $p_1^{uv} = 0.5/degree_{in}(v)$ and $p_2^{uv} = 1/degree_{in}(v)$. The average values of p_1 and p_2 were 0.3184 and 0.6367, respectively.

Our second graph is based on the Digg2009 data-set[4], a publicly available data-set that was obtained from the Digg website. Digg is a social news website, that allows people to vote on web content. The data-set contains friendship connections between the users, and a list of votes for various stories; each vote contains a story id, a vote date and user id. In order to extract the

[4] https://www.isi.edu/~lerman/downloads/digg2009.html

Algorithm 4 ETAB (Efficient Table-based Algorithm for Bisubmodular functions)

$Mat \leftarrow$ a $(B+1) \times (B+1)$ matrix of queue tuples.
$Q_1 \leftarrow$ a priority order queue of nodes, sorted by $\Delta_{n,1}(\emptyset, \emptyset)$
$Q_2 \leftarrow$ a priority order queue of nodes, sorted by $\Delta_{n,2}(\emptyset, \emptyset)$
$Mat[0][0] \leftarrow (\emptyset, \emptyset, 0, Q_1, Q_2)$
for $i = 1$ *to* B **do**
  $Mat[i][0] \leftarrow$ copy $Mat[i-1][0]$
  Add a node to $Mat[i][0](S_1)$ by using the queue $Mat[i][0](Q_1)$
  Update $Mat[i][0](Q_1)$, $Mat[i][0](Q_2)$ and $Mat[i][0](u)$
end
for $i = 1$ *to* B **do**
  $Mat[0][i] \leftarrow$ copy $Mat[0][i-1]$
  Add a node to $Mat[0][i](S_2)$ by using the queue $Mat[0][i](Q_2)$
  Update $Mat[0][i](Q_1)$, $Mat[0][i](Q_2)$ and $Mat[0][i](u)$
end
for $i = 1$ *to* B **do**
  for $j = 1$ *to* $B - i$ **do**
    $T' \leftarrow$ copy $Mat[i-1][j]$
    Add a node to $T'(S_1)$ by using the queue $T'(Q_1)$
    Update $T'(Q_1)$, $T'(Q_2)$ and $T'(u)$
    $T'' \leftarrow$ copy $Mat[i][j-1]$
    Add a node to $T''(S_2)$ by using the queue $T''(Q_2)$
    Update $T''(Q_1)$, $T''(Q_2)$ and $T''(u)$
    if $T'(u) > T''(u)$ **then**
      $Mat[i][j] \leftarrow T'$
    else
      $Mat[i][j] \leftarrow T''$
    end
  end
end
$(i, j) \leftarrow \arg\max_{i+j=B}(Mat[i][j](u))$
return $(Mat[i][j](S_1), Mat[i][j](S_2))$

diffusion parameters we considered that a vote of user v is influenced by her friend u, if v voted for a story after u has voted for it.

For every edge $u \rightarrow v$, we calculated a parameter x, which is the number of stories that v voted for as an influence of u, divided by the overall number of stories that u voted for. We then assigned $p_2^{uv} = x$, and $p_1^{uv} = 0.5 \cdot x$. The average values of p_1 and p_2 were 0.05832 and 0.11664, respectively. We deleted edges with both $p_1 = 0$ and $p_2 = 0$ and removed all isolated nodes. At the end of this process we randomly selected $1,000$ nodes, resulting with $51,398$ edges.

In addition to ETAB, TAB, Greedy and Celf, we also ran the following baseline heuristics:

- **Degree**$_{count}$: The nodes are selected in descending order according to their degrees, and each selected node is randomly assigned either to S_1 or to S_2.

- **Degree**$_{expected}$: The nodes and the message types are selected in descending order according to the sums of their diffusion parameters. That is, for each node x, and for each type i, let $d_i^x = \sum_{xu \in E(g)} p_i^{xu}$; the nodes are selected according to $\max(d_1^x, d_2^x)$ and are added to the seed corresponding to the message type with the higher d_i^x.

- **Degree**$_{sampled}$ Every node n is assigned two values $SD_1(n) := \sigma(\{n\}, \emptyset)$ and $SD_2(n) := \sigma(\emptyset, \{n\})$. The nodes are selected in descending order of $\max(SD_1, SD_2)$, and each selected node is assign to S_1 if $SD_1 \geq SD_2$, or to S_2 if $SD_1 < SD_2$.

- **Random**: A random selection of nodes, which are randomly assigned to one of the seeds.

We assigned the following utilities $u_1 = 2$, $u_2 = 1$ and $u_{1,2} = 2.5$. The evaluations of the utility function was sampled 100 times during execution of each of the algorithms, and was sampled $10,000$ times for final evaluation of the algorithm output. In order for our results to be less biased, we ran separate simulations for each Budget level; therefore, there may be a slight drop in the utility achieved by an algorithm when running it with some budget compared to the same algorithm when using a lower budget.

The budget was set to multiplies of 10, up-to a budget of 200 initial nodes. Our results are presented in Figures 4 and 5. Note that some values are missing from the plot due to the extensive time required for computing them.

As can be seen in Figures 4 and 5, ETAB achieves the highest utility (after TAB), outperforming all other methods at all budget levels.

An interesting aspect, that has not been fully studied before, is that the greedy algorithm gives a better solution than Celf. We believe that this is because the stochastic nature of Celf, as it relies on samples to estimate the addition of each node to the overall utility. Since those samples are

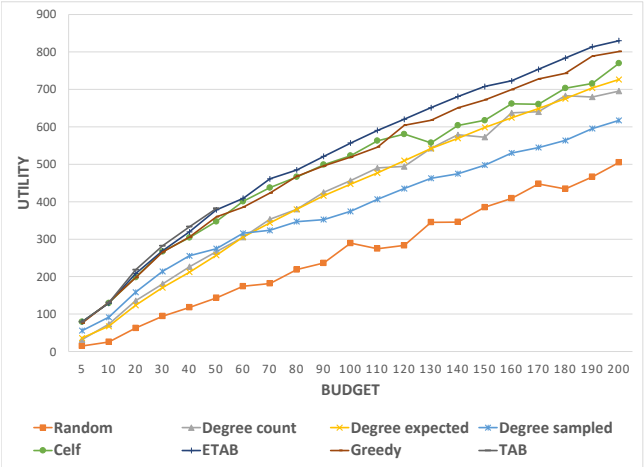

Figure 4: The utility of all the evaluated methods on the random graph.

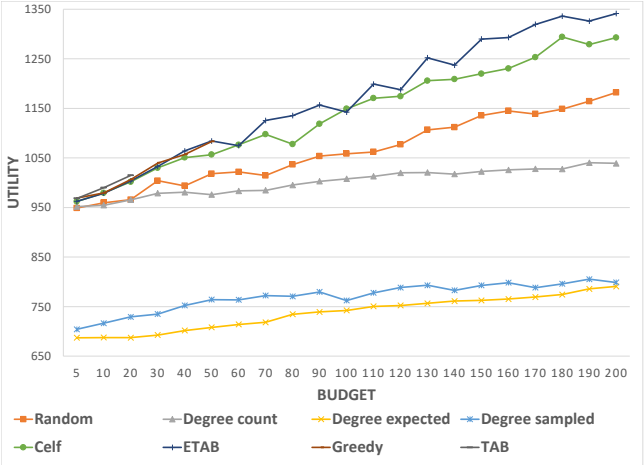

Figure 5: The utility of all the evaluated methods on the Digg based graph.

stochastic, there is an error rate that is accumulated during Celf's execution. Similar behavior is observed when comparing TAB with ETAB.

1.8 Conclusion

In this thesis proposal we introduce the Utility Based Influence Maximization problem, which is a natural extension of the common Influence Maximization problem. We defined a model that has two types of messages. One message that has a high propagation rate and a rather low utility per active user, and another message, that has a low propagation rate but gives higher utility per active user, We described a utility function that calculates the number of active nodes multiplied by the corresponding utility. We show that under some (natural) conditions, our utility function is monotone and bisubmodular and thus we provide a greedy algorithm that achieves an approximation ratio of $\frac{1}{2}$. We develop ETAB, which is a dynamic programming based algorithm suitable to our setting, and show that it outperforms the greedy algorithm.

We note that ETAB is a general solution, as it is applicable for any problem of maximizing a monotone and bisubmodular function. For example, in the antennas placement problem one needs to decide where to place antennas such that the overall reception will be maximized. Similar to our setting, it is possible that there are two types of antennas: one that has a strong signal but a small bandwidth, and the other has a weaker signal but a larger bandwidth. In this setting the reception where there is an overlap between the antennas is assumed to be at least as high as the reception from each type of antenna, but no more than their sum. Therefore, ETAB can be used to efficiently place the two types of antennas.

Nowadays, the diffusion of information through social networks is a powerful phenomenon. One common way to model diffusions in social networks is the Independent Cascade (IC) model. Given a set of infected nodes according to the IC model, a natural problem is the source detection problem, in which the goal is to identify the unique node that has started the diffusion. Maximum Likelihood Estimation (MLE) is a common approach for tackling the source detection problem, but it is computationally hard.

In this work, we propose an efficient method for the source detection problem under the MLE approach, which is based on computing the stationary distribution of a Markov chain. Using simulations, we demonstrate the effectiveness of our method compared to other state-of-the-art methods from the literature.

2 Source Detection in Networks using the Stationary Distribution of a Markov Chain

2.1 Abstract

Nowadays, the diffusion of information through social networks is a powerful phenomenon. One common way to model diffusions in social networks is the Independent Cascade (IC) model. Given a set of infected nodes according to the IC model, a natural problem is the source detection problem, in which the goal is to identify the unique node that has started the diffusion. Maximum Likelihood Estimation (MLE) is a common approach for tackling the source detection problem, but it is computationally hard.

In this work, we propose an efficient method for the source detection problem under the MLE approach, which is based on computing the stationary distribution of a Markov chain. Using simulations, we demonstrate the effectiveness of our method compared to other state-of-the-art methods from the literature.

2.2 Introduction

In the age of social media, the spread of information and infection through networks is a significant phenomenon. Understanding the dynamic of information spread and its origin are important for a wide range of applications, including marketing, public health, and identification of fake news. The Independent Cascade (IC) is a common model of the spread of information in a social network [16]. In the IC model, the process of diffusion concerns a message that is propagated through the network. Every connection between two friends is associated with a probability; this value determines the probability that if the first user shares the message, the second user will share the message with her friends as well. The diffusion process starts with an initial source, and a natural goal is that given a set of users who shared a specific message, to seek the unique source that started the diffusion.

There are many approaches to finding the source of a diffusion in the literature, each assuming different spreading models and various amounts of knowledge of the network parameters (for example, [25, 17, 57, 24]). When the probabilities associated with the connections are known or can easily be estimated, a natural mathematical approach for finding the source of the diffusion is the Maximum Likelihood Estimation (MLE) principle. According to the MLE principle, one should compute the likelihood of each user being a source, and output the user with the maximum likelihood.

The first to formalize the computational problem of finding the source of a diffusion in a network in the IC model are [25]. They show that for arbitrary graphs, the source detection problem is not only NP-hard to find but also NP-hard to approximate. Therefore, they propose an efficient heuristic, but it does not utilize the MLE principle. [56] present a heuristic that utilizes the MLE principle, and they further show that their heuristic outperforms the heuristic of [25]. However, their heuristic requires extensive computation and does not perform very well. In addition, they note that "although the IC model is popular in social network research, finding source in the IC model is rarely studied".

In this paper, we propose an efficient method that uses the MLE principle for source detection in the IC model. Our method is based on computing the stationary distribution of a Markov chain and is inspired by [24]. Specifically, we recognize that if we represent the social network as a weighted directed graph, the diffusion in the IC model induces a tree that spans the set of users who shared the message, and the root of the tree is the user that initiated the diffusion. In addition, the tree is associated with a weight, which equals to the product of the weights of its edges. In order to estimate the probability of a specific user to be the source, we would like to sum the weights of all spanning tree rooted at this user. However, directly considering all trees is computationally expensive. Therefore, we propose converting the social network into a Markov chain, and the Markov chain tree theorem [26], allows us to compute the sum of the weights of all spanning trees rooted at each user in polynomial time. We consider two approaches for converting the social network to a Markov chain—the *self-loops* and the *no-loops* methods. We show that when using a direct calculation of the stationary distribution, both methods compute the exact value of the sum of the weights of all spanning trees rooted at each user, but this is not guaranteed when using a random walk to estimate the stationary distribution. For evaluating the effectiveness of our approach, we use 14 types of graphs, and sample 1000 graphs from each type. We show that our methods outperform several baseline methods, including the method proposed by [56]. We further show that the *no-loops* method outperforms the *self-loops* method when using a random walk to estimate the stationary distribution. That is, the no-loops method requires fewer random walk steps to approach the performance of our methods when using a direct calculation of the stationary distribution.

2.3 Related Work

Viral spread of information trough social networks inspired various researchers. The most studied problem is the Influence Maximization (IM) problem, in which the goal is to find the most influ-

ential node (or set of nodes) in a social network [21]. The source detection problem, which we focus on, is (in a way) the inverse of the IM problem: instead of finding the node that will result in the maximal diffusion (i.e., the IM problem), we are given the outcome of a diffusion that has occurred, and we would like to find the source node that initiated it.

There have been multiple approaches for attempting to solve the source detection problem, with different models and with different settings, as can be seen in the following reviews [19, 45, 20]. Indeed, most of the prior work consider epidemic models such as the susceptible–infected (SI) model, and the susceptible–infected–recovery (SIR) model. For example, [42] and [43] consider the SIR and SI models, respectively, and present algorithms that are based on rumor centrality. [1] also consider the SI model, and propose an algorithm that is based on geodesic distances on a randomly-weighted version of the network. [24] consider the SI model, but with an assumption that some information regarding the edges that participate in the diffusion is provided. Since they consider the SI model, they assume that the network is undirected and the edges are unweighted. That is, each user is equally likely to get infected (i.e., receive a message) by each of her neighbors.

The IC model is the classic information-propagation model, and is widely used in social network research. However, only few papers study the source detection problem with the IC model. [25] are the first to formulate the computational problem of finding the source of a diffusion in the IC model. They propose an efficient heuristic, which is based on dynamic programming, but they do not utilize the MLE principle. [56] also consider the IC model, and show that finding the source of a diffusion is $\#P$-complete. Therefore, they develop a heuristic that utilizes the MLE principle, and is based on a Markov chain. However, their use of the Markov chain is completely different from ours. Specifically, they define the states of the Markov chain as different samples of the network. Their algorithm then perform random walks on the Markov chain, and for each sample it computes the set of all possible source nodes. Although their algorithm outperforms the heuristic of [25], it requires extensive computation to do so, and it does not perform well when compared to our method. Similar to the work of [56], [57] use a Markov chain in which the states are different samples of the network. However, their model of diffusion is the linear threshold model, instead of IC. [46] consider the IC model and utilize the MLE approach, but they study a slightly different problem, the effector detection problem. That is, instead of trying to find the source of a diffusion, their goal is to find the node that can best explain the current state of the other nodes. Therefore, given the same input, the solution to the effector detection problem may be different from the solution to the source detection problem.

2.4 Preliminaries

2.4.1 Directed Rooted Trees

A directed rooted tree is a directed acyclic graph (DAG) whose underlying undirected graph is a tree, and one of its vertices has been designated the root. An out-tree is a directed rooted tree, in which all the edges point away from the root. Similarly, an in-tree is a directed rooted tree, in which all the edges point toward the root. Clearly, we can convert an out-tree into an in-tree by reversing the directions of the edges. A spanning in-tree/out-tree is an in-tree/out-tree that spans all vertices of the underlying graph.

2.4.2 The Diffusion Model

The research on the diffusion of information on social networks has considered several models. In this paper we focus on the independent cascade model (IC), which is the following. There is a social network that is represented by a weighted directed graph $G_N = (V_N, E_N)$ with no self loops, where each user of the social network is represented by a node, every connection between two users is an edge, and the weights represent the influence probabilities. The process of diffusion concerns a message that is propagated thorough the social network. During this process, each node can either be inactive or become active. The diffusion process starts with an initial source $v \in V_N$, which is the first active node. The process then unfolds in discrete steps according to the following rule. Every node $v_i \in V_N$ that becomes active in step $t-1$, attempts to activate each currently inactive neighbor v_j in step t, and only in step t. The probability that v_i succeeds in activating an inactive neighbor v_j is p^{ij}, which is the weight of the edge $(v_i, v_j) \in E_N$. We denote by $w_{in}(v_i)$ the weighted in-degree of a node v_i, $w_{in}(v_i) = \sum_j p^{ji}$. If multiple neighbors of a vertex v try to activate it at the same time, their attempts are considered in an arbitrary order. The process runs until no more activations occur.

Note that since each active node is activated by a single parent, the active nodes and activating edges form an out-tree, which spans the set of active nodes, and the source node is the root of the tree.

2.4.3 Markov Chain

A (discrete-time) Markov chain is an infinite sequence of discrete random variables $(X_i)_{i=0}^{\infty}$. All variables have the same finite set of possible values, $S = \{s_1, s_2, \ldots, s_k\}$, which is called the state set of the Markov chain. The variables of the sequence $(X_i)_{i=0}^{\infty}$ have the Markov property

of *forgetfulness*, that is, each variable X_t is dependent only on the previous variable X_{t-1}, and is independent of all other previous variables. Moreover, all the variables have the same probability distribution. Namely, for every $t > 0$, $P(X_t = s_{i_t}|X_1 = s_{i_1}, X_2 = s_{i_2}, ..., X_{t-1} = s_{i_{t-1}}) = P(X_t = s_{i_t}|X_{t-1} = s_{i_{t-1}}) = P(X_{t+1} = s_{i_t}|X_t = s_{i_{t-1}})$, where $s_{i_j} \in S$. For a pair of states s_i and s_j, let $q^{ij} = P(X_t = s_j|X_{t-1} = s_i)$. We call q^{ij} the *transition probability*. Note that $\sum_{j=1}^{k} q^{ij} = 1$.

A Markov chain can be represented as a weighted directed graph $G_M = (S_M, E_M)$, where each state is represented by a node. For clarity reasons, we refer to the nodes of S_M as states. There is an edge $(s_i, s_j) \in E_M$ if the transition probability $q^{ij} > 0$, with a weight of q^{ij}. A random walk on the graph G_M is called a Markov process.

Irreducibility A Markov chain is *irreducible* if for each pair of states s_i, s_j, there are two directed paths $s_i \rightsquigarrow s_j$ and $s_j \rightsquigarrow s_i$ in G_M. That is, a Markov chain is irreducible if its graph, G_M, is strongly connected.

Stationary Distribution If the Markov chain is irreducible then the long run average number of visits of the Markov process to any state s_i, converges to a number Π_i, regardless of the initial state. That is, for all $s_i \in S$ it holds that $\Pi_i = \lim_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n-1} \mathbb{I}(X_t = s_i)$, where $\mathbb{I}(\cdot)$ is an indicator function. Moreover, $\Pi = (\Pi_1, \Pi_2, \ldots, \Pi_n)$ is a probability distribution over the set of states S, which is called the *Stationary Distribution*[5]. The stationary distribution can be computed in polynomial time [28]. However, if G_M is very large the exact computation of the stationary distribution may become impractical; in this case, it is common to estimate the stationary distribution by sampling, i.e., using random walks on the graph.

Vector Notations

Let V be a vector, $V = (V_1, V_2, \ldots, V_n)$. We denote by $\hat{V}$ the normalized vector, $\hat{V} = (\frac{V_1}{\sum_{i=1}^{n} V_i}, \frac{V_2}{\sum_{i=1}^{n} V_i}, \ldots, \frac{V_n}{\sum_{i=1}^{n} V_i})$.

2.5 Problem Statement

In this section, we define the source detection problem and present the MLE principle, which provides a framework for addressing it.

Definition 7 (Source Detection)**.** *Given a social network, $G_N = (V_N, E_N)$, and a set of active nodes $A \subseteq V_N$ at the end of a propagation process that is compatible to the IC diffusion model, find the source node.*

[5]Some works provide a slightly different definition for the stationary distribution, which requires that the Markov chain will also be ergodic [4].

We first note that a source node v must be in A. Moreover, there must be a directed path from v to each node in A. Let $A' \subseteq A$ be the set of all nodes that have directed paths to all nodes in A. Clearly, A' is strongly connected. In addition, if A' is the singleton $\{v\}$ then v is the source node.

Our approach for identifying the source node is to follow the maximum likelihood principle [35]. That is, each node is associated with the probability that it is the source, and we select a node with the maximal probability. Formally, let R_i be the event that v_i is the source node, and let $\mathcal{A}$ be the event that the set A is the set of active nodes; $\mathcal{A}'$ is defined similarly for A'. We would like to solve the following problem:

Definition 8 (ML-Source)**.** *Given a social network, $G_N = (V_N, E_N)$, and a set of active nodes $A \subseteq V_N$ at the end of a propagation process according to the IC diffusion model, find the most likely source node v^*, i.e.,*
$v^* = \arg\max_{v_i \in V_N} P(R_i|\mathcal{A})$.

Note that $P(R_i|\mathcal{A}) = \frac{P(R_i,\mathcal{A})}{P(\mathcal{A})}$, and thus

$$\arg\max_{v_i \in V_N} P(R_i|\mathcal{A}) = \arg\max_{v_i \in V_N} P(R_i, \mathcal{A}).$$

Unfortunately, the ML-Source problem was shown to be computationally hard [56]. Indeed, the following brute-force procedure computes the exact value of $P(R_i, \mathcal{A})$, in exponential time. Let $G_N[A] = (A, E(A))$ be the subgraph of G_N induced by the set A. That is, $E(A)$ is the set of edges of G_N that have both nodes in A. We consider every subset $X \subseteq E(A)$, and each such X is associated with a probability for its occurrence:

$$p(X) = \prod_{e \in X} p^e \cdot \prod_{e \in E(A) \setminus X} (1 - p^e).$$

Let G_X be a graph, $G_X = (A, X)$. If there exists an out-tree in G_X that spans A and with the node v_i as the root, then the probability $p(X)$ should be added to $P(R_i, \mathcal{A})$. Namely:

$$P(R_i, \mathcal{A}) = \sum_{X \subseteq E(A)} I(X, v_i) \cdot p(X)$$

where $I(X, v_i)$ is an indicator function that returns 1 if the graph G_X has a spanning out-tree with v_i as the root, and 0 otherwise. In order to return the most likely source node, one should compute the above expression for every $v_i \in A$. Moreover, each $p(X)$ can be used more than once, in the case where G_X has multiple spanning out-trees with several roots.

2.6 The Markov Chain Approach

Kumar, Borkar, and Karamchandani [24] suggested the Markov chain approach for estimating the probability of a node to be the source, in their setting. Their main idea is to count for each node v every possible spanning out-tree, in which v is the root, and for each such out-tree to calculate its occurrence probability. We adapt the Markov chain approach to our setting, as follows.

For a diffusion that started with a source node v_i, and resulted with a set A of active nodes, let $T_{i,A}$ be the corresponding spanning out-tree, and let $\mathcal{T}_{i,A}$ be the associated event. We denote by $w(T)$ the weight of a directed rooted tree, $w(T) = \prod_{e \in T} p^e$. A good estimation of the probability of $\mathcal{T}_{i,A}$ is:

$$P(\mathcal{T}_{i,A}) \approx P(R_i) \cdot w(T_{i,A}).$$

Note that this is an estimation, since we ignore the edges that are not part of the spanning out-tree. In order to calculate the probability of a single node v_i to be the source, we go over every spanning out-tree rooted at v_i and sum the weights of those out-trees:

$$P(R_i, \mathcal{A}) \approx \sum_{T_{i,A} \in OT_{i,A}} P(\mathcal{T}_{i,A}).$$

where $OT_{i,A}$ is a set of all the out-trees of $G_N[A]$ that are rooted at v_i and span A. Note that this summation is also an estimation, since the events $\mathcal{T}_{i,A}$ for every spanning out-tree are not independent (which can be fixed with an inclusion-exclusion calculation).

For $v_i \in A'$, let $\Gamma_i = \sum_{T_{i,A'} \in OT_{i,A'}} w(T_{i,A'})$, let $\Gamma = (\Gamma_1, \Gamma_2, \ldots, \Gamma_{|A'|})$, We assume that the *prior* probability, $P(R_i)$, is equal for every $v_i \in V_N$. It is also enough to consider A' instead of A since $P(R_i, \mathcal{A}) \propto P(R_i, \mathcal{A}')$ (according to [24]). We get that

$$v^* \approx \arg\max_{v_i \in A'} \Gamma_i. \tag{11}$$

Based on this formulation, a naive approach is to compute for each $v_i \in A'$ the set $OT_{i,A'}$, (using an algorithm for finding all spanning out-trees, e.g. [13]), and to return the node v_i that maximizes Γ_i. We refer to this approach as the out-tree counting method. Clearly, the out-tree counting method is (also) computationally expensive, as the size of $OT_{i,A'}$ is most likely exponential in $|A'|$. We thus propose to use the following theorem [26]:

Given a finite state irreducible Markov chain as a directed graph $G_M = (S_M, E_M)$, Let $w(T) = \prod_{e \in T} p^e$ be the weight of a spanning in-tree $T \subseteq E_M$. Let IT_{i,S_M} be the set of all in-trees in E_M that have s_i as their root and span S_M. Let $\Psi_i := \sum_{T \in IT_{i,S_M}} w(T)$, let $\Psi = (\Psi_1, \Psi_2, \ldots, \Psi_n)$.

Theorem 4. *(Markov chain tree theorem) Given a finite state irreducible Markov chain, For every $s_i \in S_M$, the unique Stationary Distribution Π_i equals to $\hat{\Psi}_i$. namely:*

$$\forall s_i \in S_M, \Pi_i = \hat{\Psi}_i = \frac{\Psi_i}{\sum_{j=1}^{n} \Psi_j} \tag{12}$$

Our approach is based on exploiting the structural similarity between Γ and Ψ. Indeed, let $G_N[A']$ be the graph G_N induced on the set A', then for every $1 \leq i \leq |A'|$, Γ_i is the summation of weights of spanning out-trees in $G_N[A']$, and Ψ_i is the summation of weights of spanning in-trees in a Markov chain. We thus first convert the social network $G_N[A']$ into a Markov chain G_M. This conversion includes the inversion of all the edges. That is, each node $v_i \in G_M$ is represented by a state s_i, and each edge (v_i, v_j) in $G_N[A']$ is converted to an edge (s_j, s_i) in G_M. Therefore, every spanning out-tree in $G_N[A']$ corresponds to a spanning in-tree in G_M. We then compute the complete stationary distribution Π, of the Markov chain G_M, obtaining $\hat{\Psi}$ (by Theorem 4). We then use $\hat{\Psi}$ to restore $\hat{\Gamma}$. Finally, we output the node with the maximal $\hat{\Gamma}$ value.

Observe that converting the social graph into a Markov chain must be performed carefully. Specifically, it requires that the transition probabilities are valid, i.e., for each state s_i, $\sum_{j=1}^{k} q^{ij} = 1$. Moreover, it requires that the $\hat{\Gamma}$ values can be efficiently restored from $\hat{\Psi}$.

The stationary distribution Π can be computed in polynomial time, and so can finding the set A' (see for example [44]), and the conversation of the social graph into a Markov chain. Therefore, our heuristic can be efficiently computed.

2.7 Conversion Methods

Now we detail how to produce the Markov chain graph $G_M = (S_M, E_M)$. Each node $v_i \in A'$ is represented by a state $s_i \in S_M$, and each directed edge $(v_i, v_j) \in E(A')$ is represented by the reversed edge $(s_j, s_i) \in E_M$. We get that in the Markov chain, each edge is pointing from a node to all its possible activators. Therefore, a naive approach for converting the social graph into a Markov chain is to divide the weights of each edge of the Markov chain, by the sum of all weights of the incoming edges of the original node. That is, each node $v_i \in A'$ is for each edge $(s_j, s_i) \in E_M$ set $q^{ji} = \frac{p^{ij}}{w_{in}(v_j)}$.

However, merely normalizing the edge probabilities is not enough, since we lose the distinction between nodes with different $w_{in}(\cdot)$ values. (And clearly, *ceteris paribus*, a node with a low $w_{in}(\cdot)$ value is more likely to be the source). For example, consider the social network in Figure 6a. In this example, the possible sources are $A' = \{v_1, v_2, v_3, v_4\}$, and the naive normalization assigns a weight of 1 to all the edges of the Markov chain. This results with a stationary distribution in

which $\Pi_1 = \Pi_2 = \Pi_3 = \Pi_4 = 0.25$ [6]. However, when computing the exact probabilities, we get that $p(R_1|\mathcal{A}') = p(\mathcal{A}'|R_1) \cdot p(R_1)/p(\mathcal{A}') = 0.1 \cdot 0.3 \cdot 0.6 \cdot \frac{1}{z}$, where z is the same for every R_i. Similarly, $p(R_2|\mathcal{A}') = 0.3 \cdot 0.6 \cdot 0.2 \cdot \frac{1}{z}$, $p(R_3|\mathcal{A}') = 0.6 \cdot 0.2 \cdot 0.1 \cdot \frac{1}{z}$, $p(R_4|\mathcal{A}') = 0.2 \cdot 0.1 \cdot 0.3 \cdot \frac{1}{z}$. Thus, $z = 0.072$ and we the vector of probabilities is $(0.25, 0.5, 0.167, 0.083)$, in which $p(R_i|\mathcal{A}')$ is in the i-th position. That is, v_2 is much more likely than any other node to be the source, but the naive approach fails to identify v_2 as the most likely source node.

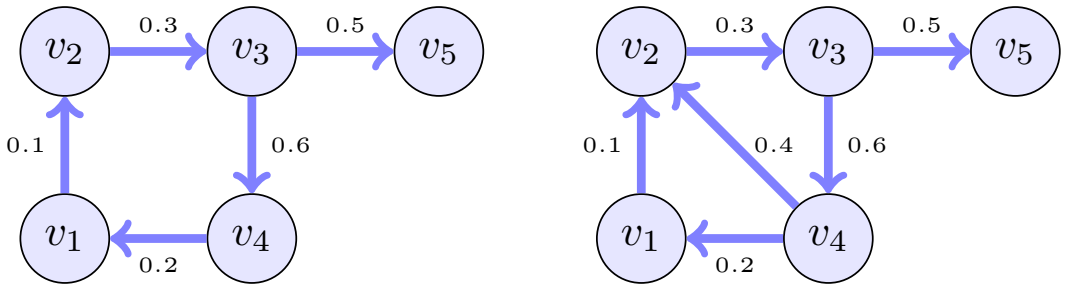

Figure 6: Graph examples.

We thus present two methods for converting the social graph into a Markov chain, which consider (among other things) the difference between the $w_{in}(\cdot)$ values.

2.7.1 The Self-Loops Method

In our first approach, *Self-loops*, after converting all the edges of $G_N[A']$, we add self-loops to all states. This allows us to normalize the edge probabilities by dividing all of the weights by the same number. Specifically, let $max_{in} = \max_{v_i \in A'} w_{in}(v_i)$ (We compute w_in in the graph $G_N[A']$). The self-loops methods works as follows;

1. Convert each node $v_i \in G_n[A']$ into a state s_i and each edge (v_i, v_j) into a reversed edge (s_j, s_i) with $q^{ji} = \frac{p^{ij}}{max_{in}}$.
2. For each s_i, add a self loop (s_i, s_i) with $q^{ii} = \frac{max_{in} - w_{in}(v_i)}{max_{in}}$
3. Compute the stationary distribution Π.
4. $\hat{\Gamma}$ is assigned the values of Π.

[6] The Markov chain is $S_M = \{s_1, s_2, s_3, s_4\}$, $E_M = \{(s_1, s_4), (s_4, s_3), (s_3, s_2), (s_2, s_1)\}$ and all the transition probabilities equal 1, therefore the stationary distribution is $\Pi = (0.25, 0.25, 0.25, 0.25)$.

Clearly, the transition probabilities are valid. We need to show that $\hat{\Gamma}$ is restored correctly.

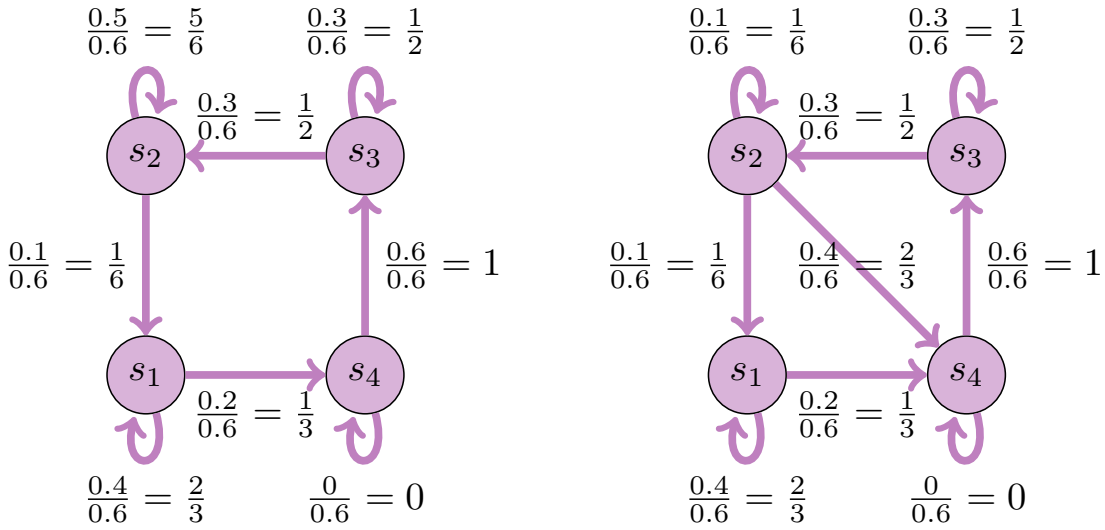

Figure 7: The Markov chains that are obtained by applying the self loops method on the graphs of Figure 6.

Theorem 5. *(Self-loops method)*

For the self-loops method, for every $1 \leq i \leq |A'|$ it holds that $\Psi_i = \alpha \cdot \Gamma_i$, where α is a constant. In other words, For every node $v_i \in G_N[A']$, the sum of weights of all out-trees spanning A' and rooted at v_i, is proportional to the sum of weight of all in-trees in G_M that span all states and are rooted at the corresponding state s_i.

Proof. Observe that every spanning out-tree $T \subseteq G_N[A']$, with weight $w(T) = \prod_{e \in T} p^e$ has a corresponding spanning in-tree $T' \subseteq G_M$ with weight $w(T')$. In addition, in-trees do not contain self loops, and have exactly $n-1$ edges (where $n = |A'|$). Therefore,

$$w(T') = \prod_{e \in T'} q^e = \prod_{e \in T} \frac{p^e}{max_{in}}$$

$$= \frac{1}{(max_{in})^{n-1}} \cdot \prod_{e \in T} p^e = \alpha \cdot w(T).$$

Thus,

$$\Psi_i = \sum_{T' \in IT_{i,S_M}} w(T') = \alpha \cdot \sum_{T \in OT_{i,A'}} w(T) = \alpha \cdot \Gamma_i.$$

□

Clearly, it can be concluded that $\hat{\Psi} = \hat{\Gamma}$. Note that the stationary distribution Π that is calculated by the self loops method, is equal to $\hat{\Psi}$ (by Theorem 4), and is also equal to $\hat{\Gamma}$. Therefore, by using the self loops method for the conversion and selecting the node v_i with the maximal $\hat{\Gamma}_i$, we obtain a good estimation for v^*.

To demonstrate the self-loops method and Theorem 5, we consider the social network given in Figure 6b, which is slightly more complex than the graph in Figure 6a, as it includes an additional edge between v_4 and v_2. Now, assume that the set of active nodes is $A = \{v_1, v_2, v_3, v_4, v_5\}$. Clearly, $A' = \{v_1, v_2, v_3, v_4\}$, The self-loop method computes the Markov chain that is shown in Figure 7a, and the stationary distribution of this Markov chain is $\Pi = (0.125, 0.25, 0.417, 0.208)$. Recall that $\Pi = \hat{\Psi}$, (according to Theorem 4), and $\hat{\Psi} = \hat{\Gamma}$ (according to Theorem 5). Indeed, using the out-tree counting method for directly calculating the values of Γ leads to the same result. Specifically, v_1 has one possible spanning out-tree with $\Gamma_1 = 0.1 \cdot 0.3 \cdot 0.6 = 0.018$. v_2 has one possible spanning out-tree with $\Gamma_2 = 0.3 \cdot 0.6 \cdot 0.2 = 0.036$. v_3 has two possible spanning out-tree with $\Gamma_3 = 0.6 \cdot 0.2 \cdot 0.1 + 0.6 \cdot 0.2 \cdot 0.4 = 0.06$ and v_4 has two possible spanning out-tree with $\Gamma_4 = 0.2 \cdot 0.1 \cdot 0.3 + 0.2 \cdot 0.4 \cdot 0.3 = 0.03$ Therefore, $\Gamma = (0.018, 0.036, 0.06, 0.03)$, and $\hat{\Gamma}$ equals Π. Overall, the self-loops method outputs the vertex v_3 as the most likely source node.

Note that the *precise* brute force calculation outputs the values $(0.1315, 0.2631, 0.4035, 0.2017)$, and thus it also determines that v_3 is the most likely source node. Furthermore, the correlation between the exact probabilities to Π is 0.9966.

Returning to the example in Figure 6a, the corresponding Markov chain that is obtained by the self loops method is shown in figure 7b, and the stationary distribution for this Markov chain is $\Pi = (0.25, 0.5, 0.167, 0.083)$. That is, in this example the self-loops method finds the exact probabilities, since for every sub-graph of $G_N[A']$ there is at most one spanning out-tree.

2.7.2 The no-loops Method

Recall that the self-loops method returns the exact values of $\hat{\Gamma}$ when Π is computed directly. However, when Π is estimated by sampling, the addition of the self loops to the graph might require longer random walks, as many of the random steps are "wasted" on the self loops. This, in-turn, may affect the efficiency of the sampling. Therefore, we present our second method, *no-loops*, which does not add self-loops to the states. Instead, it first converts the graph and computes the edge probabilities using the naive method. Once the stationary distribution Π is computed, the no-loops method restores the correct $\hat{\Gamma}$ values from Π by dividing each Π_i by $w_{in}(v_i)$ and normalizing the result.

Specifically, the no-loops method is executed as follows:

1. Convert each node $v_i \in G_n[A']$ into a state s_i and each edge (v_i, v_j) into a reversed edge (s_j, s_i) with $q^{ji} = \frac{p^{ij}}{w_{in}(v_j)}$.

2. Compute the corresponding stationary distribution Π.

3. Let $\Pi^{corr} = (\Pi_1^{corr}, \ldots, \Pi_{|A'|}^{corr})$, where for every $1 \leq i \leq |A'|$, $\Pi_i^{corr} = \frac{\Pi_i}{w_{in}(v_i)}$.

4. $\hat{\Gamma}$ is assigned the values of $\hat{\Pi}^{corr}$.

We now show that $\hat{\Gamma}$ is restored correctly. Indeed, the no-loops method converts the social network to a Markov chain using the naive method. We show that with this conversion, the weight of each spanning out-tree is divided by a value that depends only on the root node. Therefore, in order to restore $\hat{\Gamma}$ we must multiply each Π_i by this value. Indeed, as we show, instead of multiplying by this value, it is sufficient to divide by $w_{in}(v_i)$.

Theorem 6. *(No-loops method)*

For the no-loops method, for every $1 \leq i \leq |A'|$ it holds that $\Pi_i^{corr} = \alpha \cdot \Gamma_i$, where α is a constant.

Proof. Let T be a spanning out-tree in $G_N[A']$, rooted at v_r with weight $w(T) = \prod_{(v_i,v_j)\in T} p^{ij}$. Additionally, let T' be the spanning in-tree in G_M that corresponds to T. Each edge $(s_j, s_i) \in T'$ has a weight $q^{ji} = \frac{p^{ij}}{w_{in}(v_j)}$. Therefore, the weight of T' is:

$$w(T') = \prod_{(s_j,s_i)\in T'} q^{ji} = \prod_{(v_i,v_j)\in T} \frac{p^{ij}}{w_{in}(v_j)}$$

Since in the out-tree T, the root node v_r does not have an in-edge, and each of the other nodes in the out-tree has exactly one in-edge, the denominator is a multiplication of all the $w_{in}(v_i)$ values except for $w_{in}(v_r)$:

$$\begin{aligned} w(T') &= \frac{1}{\prod_{v_i\in A', i\neq r} w_{in}(v_i)} \cdot \prod_{(v_i,v_j)\in T} p^{ij} \\ &= \frac{1}{\prod_{v_i\in A', i\neq r} w_{in}(v_i)} \cdot w(T). \end{aligned}$$

Dividing both sides by $w_{in}(v_r)$ gives:

$$\frac{w(T')}{w_{in}(v_r)} = \frac{w(T)}{\prod_{v_i\in A'} w_{in}(v_i)} = \alpha \cdot w(T), \tag{13}$$

where $\alpha = \frac{1}{\prod_{v_i\in A'} w_{in}(v_i)}$, a value that does not depend on v_r.

Now, for Π that is calculated in stage (2) of the no-loops method, we get that for every $1 \leq i \leq |A'|$, $\Pi_i = \sum_{T' \in IT_{i,S_M}} w(T')$, according to Theorem 4. Therefore,

$$\Pi_i^{corr} = \frac{\Pi_i}{w_{in}(v_i)} = \sum_{T' \in IT_{i,S_M}} \frac{w(T')}{w_{in}(v_i)}$$

$$= \sum_{T \in OT_{i,A'}} \alpha \cdot w(T) = \alpha \cdot \Gamma_i.$$

□

It can be concluded that $\hat{\Pi}^{corr} = \hat{\Gamma}$. Therefore, by using the no-loops method for the conversion and selecting the node v_i with the maximal $\hat{\Gamma}_i$, we obtain a good estimation for v^*.

To demonstrate the no-loops method and Theorem 6 we return to the example in Figure 6b. The Markov chain that is obtained by stage (1) of the no-loops method for the social network graph in 6b, is shown in Figure 8. The stationary distribution of this Markov chain is $\Pi = (0.0625, 0.3125, 0.3125, 0.3125)$. Therefore, $\Pi^{corr} = (\frac{0.0625}{0.2}, \frac{0.3125}{0.5}, \frac{0.3125}{0.3}, \frac{0.3125}{0.6})$. Finally, after normalization we obtain $\hat{\Pi}^{corr} = (0.125,\ 0.25,\ 0.417,\ 0.208)$, which is equal to the output we obtained with the self-loops method.

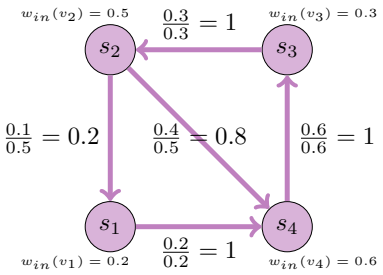

Figure 8: The Markov chain that is obtained by stage (1) of the no-loops method for the social network graph in 6b.

2.8 Experiments

For the evaluation of the performance of the self-loops and the no-loops methods, and comparing it to other baselines heuristics, we use 14 types of directed random graphs that have diffusion probabilities on their edges. Each graph is composed using the tuple $(n, Density, p_{range})$: n is the number of nodes, $Density$ is the probability that a directed edge exists between any two nodes (v_i, v_j), and for every directed edge (v_i, v_j), p^{ij} is a random number uniformly drawn from the

range $[0, p_{range}]$. $Density$ and p_{range} were chosen such that the average weighted out-degree of each node is slightly greater than 1. Table 5 in the Appendix summarizes the parameters of the 14 graph types.

	G_1	G_2	G_3	G_4	G_5	G_6	G_7	G_8	G_9	G_{10}	G_{11}	G_{12}	G_{13}	G_{14}	**Average**
Self-loops (direct calc.)	127	151	170	141	89	114	106	96	90	98	77	59	69	66	**103.78**
10 steps	72	93	101	73	36	63	58	59	55	58	40	36	35	34	58.07
100 steps	86	108	110	91	47	82	65	67	58	59	62	39	54	34	68.71
1000 steps	107	150	140	128	82	99	91	85	85	91	69	52	65	61	93.21
10000 steps	126	151	162	136	90	112	106	95	91	96	74	51	66	66	101.57
No-loops (direct calc.)	127	151	170	141	89	114	106	96	90	98	77	59	69	66	**103.78**
10 steps	82	97	101	83	41	65	69	72	50	61	37	35	43	41	62.64
100 steps	100	155	151	128	73	96	86	85	73	82	68	55	49	56	89.78
1000 steps	124	156	163	142	86	104	99	95	95	96	76	54	72	66	102
10000 steps	128	154	166	147	89	109	105	98	88	99	72	55	70	69	103.5
Naive	36	63	48	42	24	47	34	40	50	55	34	30	28	26	39.78
Random	20	24	28	37	19	26	18	26	25	28	21	9	8	11	21.42
Max out-deg	56	42	53	49	27	37	34	42	33	57	33	22	43	38	40.42
Min in-deg	49	37	50	42	19	27	30	35	62	52	36	26	40	33	38.42
Max (out/in)-deg	80	57	63	50	28	50	37	44	68	64	58	42	54	44	52.78
IM based	54	65	57	41	32	51	37	36	42	52	38	25	30	35	42.5

Table 4: The number of times in which each method finds the correct source node.

For each of the graph types we sampled a graph, selected a random source node, and ran a simulation of a diffusion according to the IC model, obtaining an active set A. If the size of the active set was less than 20 nodes or the size of the set A' was 1, the graph was discarded. We continued sampling graphs until we obtained 1000 graph instances for each of the 14 graph types. The average sizes of the set A', the average number of edges in the sub-graph induces by A', and the number of times we discarded a sampled graph are summarized in Table 6 in the Appendix.

We evaluate the performance of the self-loops and the no-loops methods, using a direct calculation of the stationary distribution. In addition, we evaluated these methods when the stationary distribution is estimated by random walks with 10, 100, 1000, or $10,000$ steps. Finally, we evaluate the performance of the following baseline methods:

- **Naive:** The Markov chain approach with the naive conversion method.
- **Random:** A random selection of a node in A'.
- **Max out-degree:** The node with the maximal weighted out-degree is selected.
- **Min in-degree:** The node with the minimal weighted in-degree is selected.

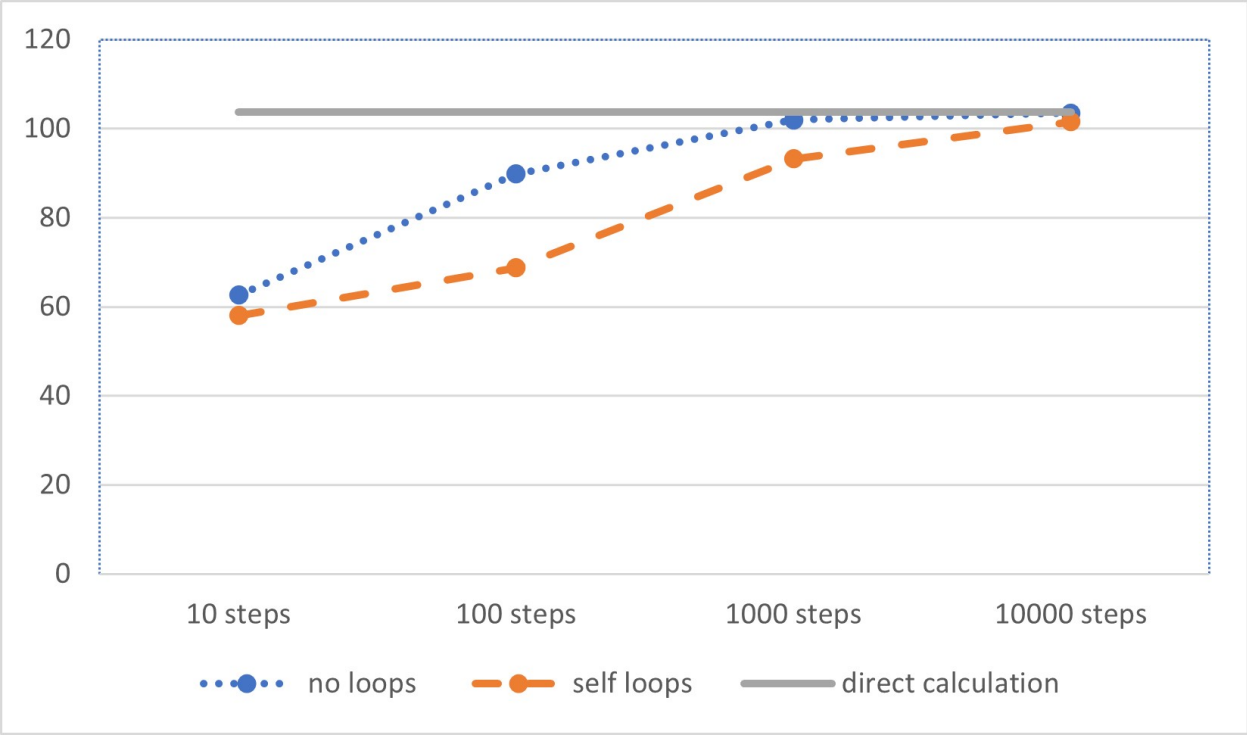

Figure 9: The average number of times in which the no-loops and the self-loops methods, using direct calculation and using random walks, find the correct source node.

- **Max (out/in) degree:** The node with the maximal weighted out-degree divided by its weighted in-degree is selected.
- **IM based:** For each node, we simulate 1000 diffusions, and the node with the maximal average size of the active set is selected.

We also evaluate the performance of the method proposed by [56]. However, unfortunately, this method takes extensive time to run and does not perform as well as our simple heuristics (max out-degree and min in-degree). Therefore, we do not report these results.

Our results are presented in Table **??**. As expected, the self-loops and no-loops methods using a direct calculation of the stationary distribution achieve the exact same results and outperform all other methods (on average). We note that, at times, our methods using random walks may perform slightly better than when using the direct calculation; however, this is only due to the inherent randomness of the problem, and, on average, this does not happen. Moreover, it can be observed that the no-loops method using random walks requires fewer steps than the self-loops method using random walks to approach the performance of our methods when using a direct calculation of the stationary distribution. This result is further demonstrated by Figure 9. This result is in line with our claim in Section 2.7.2 that the self-loops method might require longer walks and will thus be less efficient when the stationary distribution is estimated by sampling.

2.9 Conclusions and Future Work

In this paper, we study the problem of identifying the source of a given diffusion on a network. We use the common IC model, and utilize the MLE principle. Our approach is based on computing the stationary distribution of a Markov chain. Interestingly, with this approach, rather than computing the likelihood of every node to be the source separately, the values for all nodes are derived from the same stationary distribution. We propose two approaches for converting the network to a Markov chain, and demonstrate the effectiveness of one of them, the no-loops method, even when using random walks to estimate the stationary distribution.

For future work, we would like to extend our Markov chain approach to the setting in which there are several source nodes that initiate the diffusion. In addition, we would like to extend our approach to other diffusion models, such as the continuous time independent cascade model [23].

Graphs	n	Density	p_{range}	Average no. of edges
G_1	500	0.1	0.0416	24957
G_2	1000	0.1	0.0204	99,909
G_3	2000	0.1	0.0101	399,785
G_4	3000	0.1	0.0071	899,721
G_5	4000	0.1	0.0052	1,599,576
G_6	5000	0.1	0.0041	2,499,461
G_7	500	0.0416	0.1	10,404
G_8	1000	0.02	0.1	20,022
G_9	2000	0.0101	0.1	40,394
G_{10}	3000	0.0067	0.1	60,392
G_{11}	4000	0.0052	0.1	84,185
G_{12}	5000	0.0041	0.1	104,138
G_{13}	10000	0.002	0.1	204,092
G_{14}	15000	0.0013	0.1	304,001

Table 5: Graphs types details

Graph type	Average $\lvert A' \rvert$	Average $e(G[A'])$	no. times $\lvert A \rvert = 1$	no. diffusions too small
G_1	71.162	728.594	33	4,058
G_2	93.907	1438.516	21	4,527
G_3	119.167	2675.412	21	4,467
G_4	263.945	12418.689	13	3,280
G_5	257.102	13200.164	24	3,305
G_6	265.526	15136.812	20	3,736
G_7	77.595	410.53	154	4,440
G_8	92.328	376.151	469	6,800
G_9	148.714	550.316	804	7,589
G_{10}	174.998	608.171	1,139	9,284
G_{11}	371.221	1504.982	860	6,616
G_{12}	386.578	1460.45	1,073	7,630
G_{13}	536.922	1718.632	2,163	12,667
G_{14}	607.511	1783.66	3,026	17,039

Table 6: Diffusion details

3 On Predicting the Outcome of Public Goods Games on Networks

3.1 Abstract

Nash equilibrium is a well-established concept for predicting the outcome of a strategic game when the players are fully rational agents. While it is widely accepted that humans do not always behave as fully rational agents, our understanding of humans' prediction of the outcome of strategic games remains limited. This study attempts to bridge this gap by examining human subjects' prediction of the outcome of Public Goods in Networks (PGN) games.

In this study, we explore participants' ability to predict PGN games' outcomes without prior knowledge of game theory or graph theory concepts. Therefore, we conduct a survey involving 96 participants, in which we request their predictions for PGN games' outcomes. Surprisingly, our findings indicate that participants, even in the absence of explicit knowledge regarding stability or equilibrium, tend to predict outcomes that align with a Nash equilibrium. This suggests that, in certain scenarios, the Nash equilibrium is in correspondence with human intuition and reasoning in strategic games.

3.2 Introduction

Game theory involves the systematic analysis of strategic interactions between rational agents [34]. It provides a framework for understanding a variety of real-world scenarios, where the actions of individual agents influence the overall outcome. In the context of game theory, strategies are the defined plans or courses of action that an agent decides to follow. Game theory has far-reaching implications in various disciplines, including economics, political science, and computer science, and it may help to predict the outcomes of strategic interaction.

One of the most fundamental concepts in game theory is the Nash equilibrium. The Nash equilibrium refers to a state in a game where no agent can benefit from unilaterally changing their strategy, assuming the strategies of all other players remain fixed [37]. Fortunately, as proved by Nash, every finite simultaneous game has at least one Nash equilibrium, making it a crucial tool for analyzing and predicting the behavior of strategic agents.

Human behavior often exhibits patterns that closely align with the concept of Nash equilibrium. Under certain conditions, individuals and groups have been observed to follow strategies that would be predicted by the equilibrium concept, lending empirical support to its applicability in real-world situations [8]. However, human behavior often does not conform to the predictions of

the Nash equilibrium. Indeed, various studies have shown that human agents do not act according to the equilibrium strategies, which can be attributed to factors such as bounded rationality, errors in judgment, and social preferences [32, 15]. While empirical evidence shows that human behavior can both align with and deviate from Nash equilibrium, it is less clear whether ordinary humans anticipate other humans to adhere to the equilibrium.

The game of Public Goods in Network (PGN) is a theoretical model, in which agents within a network decide whether to produce a specific product that benefits themselves and their immediate neighbors [6]. That is, given a specific product, the utility of every agent depends on her decision to produce the product, as well as the decision of her immediate neighbors (whether to produce the product or not). Each PGN game defines a rule, which determines the utility of each agent. For example, a natural PGN rule is that it is worthwhile for each agent to produce a product if none of her neighbors produce it.

In PGN games, the network structure has a substantial effect on the behavior of the agents and thus, on the Nash equilibrium of the game. For example, if the PGN game is defined with the previously mentioned rule, and is played on a star graph (see Figure 14), there exist two pure strategy Nash equilibria. Namely, either the middle agent (G) produces the product and the other agents do not, or all other agents produce the product but the middle agent does not. Note that given a specific rule, it might be that a Pure Strategy Nash Equilibrium (PSNE) does not exist for some networks. Furthermore, given a specific rule, finding a PSNE might be computationally hard [14].

In this paper, we aim to investigate whether humans predict a PSNE as the outcome of PGN games. To that end, we conduct experiments with human participants. We first explain to the participants the basic concepts of the PGN game, such as the graph representation of the network and the PGN rule. Notably, we do not mention the terms "stability" or "Equilibrium" to the participants. Then, the participants are given three different networks and are asked to predict who they believe is producing the product and who is not. After providing their answers to the third network, the participants are shown two proposals that indicate who in the network is producing the product and who is not. The first proposal is a PSNE, and the second proposal is not. The participants were asked to compare each proposal to the prediction they have proposed. Finally, the participants were asked whether they were familiar with the concept of Nash equilibrium in Game Theory.

Our findings reveal that, in the simplest setting, a significant proportion of participants (61.5%) proposed a PSNE, even though they were unfamiliar with the concept. However, in the more complex networks, the percentage of participants predicting a PSNE decreased, ranging from 44.8% to

49%. It is noteworthy that, of those who proposed a PSNE for the third network, a large majority (72.3%) correctly recognized that a proposal not constituting a Nash Equilibrium is inferior to their own prediction.

Our results demonstrate that the concept of PSNE is intuitive to most humans. When the context is adequately simplified, participants anticipate that others will behave in alignment with a PSNE. This inherent understanding is not dependent on prior familiarity with the equilibrium concept. Even as the networks become more complex, a significant percentage continue to anticipate a PSNE, indicating robustness in their intuitive reasoning. Further, a strong majority among those proposing a PSNE could correctly differentiate between proposals that did and did not adhere to this equilibrium.

To summarize, the contribution of this paper is threefold. Firstly, it extends our understanding of how ordinary individuals intuitively predict PSNE in the context of Public Goods in Networks (PGN) games, shedding light on human anticipation of strategic behavior. Secondly, it evaluates this understanding across different complexities, showcasing the resilience of human intuition in more intricate network settings. Lastly, it provides empirical evidence of individuals' ability to recognize and distinguish between equilibrium and non-equilibrium states. Through these findings, our study contributes valuable insights into the intersection of human behavior and game theory, advancing our comprehension of decision-making processes in strategic interactions.

3.3 Related Work

Research in public goods provision within network structures has significantly progressed over the years. A foundational study by [6] lays out the groundwork by examining how network topology can influence individual contributions and overall efficiency of public goods provision, discussing certain configurations that foster higher levels of cooperation and welfare.

There have been several other proposes for incentivizing cooperation. [22] introduce a mechanism where rewards are given based on an individual's contribution and that of their neighbors. Their investigation highlights how this strategy influences cooperation levels across various network structures.

Algorithmic explorations form another core area in this domain. [54] delve into the algorithmic aspects of public goods games, establishing the NP-completeness of determining pure-strategy Nash equilibrium in these contexts. In addition, they identify tractable instances based on utility function and graph structure restrictions. Finally, they propose a heuristic method for approximating equilibria and show its efficacy in empirical evaluation.

Another approach is proposed by [53], who examine the complexity of computing action profiles that form a Nash equilibrium and those that maximize social welfare. Despite the general NP-hardness, they propose that the problem can be polynomial-time solvable when the network falls within certain restricted domains.

Similarly, [41] address the complexity of finding Nash equilibria in directed and undirected networks in public goods games. They present a comprehensive complexity dichotomy, suggesting polynomial-time algorithms may be feasible under bounded conditions.

[31], apply parameterized complexity theory to explore the existence of pure strategy Nash equilibrium (PSNE). Their study underscores the challenge of finding PSNE but offers an improved understanding of conditions under which PSNE can be a reliable solution. Similar to their work, we focus on pure strategy Nash Equilibria, rather than mixed strategy Nash Equilibria.

Building on this rich body of literature, our work aims to bring a human behavioral perspective to these mathematical and computational discussions on public goods games and Nash equilibria. We therefore turn to survey works that discuss Nash Equilibrium in environments where humans are directly involved.

The exploration of human behavior in strategic interactions, particularly in the context of game theory, has been the subject of extensive research. Our work builds upon these previous studies, with a specific focus on Public Goods in Networks (PGN) games.

[51] seminal work, critically evaluated a range of models from behavioral game theory. Their research highlighted the limitations of the Nash equilibrium in predicting initial human behavior in normal-form games, suggesting modifications to improve the practical prediction of human play. Our study extends this line of inquiry by examining how individuals intuitively predict Nash Equilibrium in the context of PGN games.

The study [48], provided insights into human decision-making in non-cooperative strategic interactions. Similarly, our work delves into the strategic behavior of individuals, albeit in a different game setting.

[7] assessed people's expectations of generosity in controlled experiments using the dictator game. This work, along with ours, contributes to the understanding of human anticipation of strategic behavior.

[18] discussed the inadequacies of the Nash equilibrium in predicting outcomes for a vast set of games. Our research complements this by examining the resilience of human intuition in predicting Nash Equilibrium across different complexities.

Lastly, the work of [55], modeled a problem in a game-theoretic framework of trust. While their focus was on robot behavior, our study similarly uses a game-theoretic approach to understand

human behavior.

In summary, our research builds upon these foundational studies, extending the understanding of human behavior in game theory, particularly in the context of PGN games. Our work contributes to this field by providing empirical evidence of individuals' ability to recognize and distinguish between equilibrium and non-equilibrium states, thereby advancing our comprehension of decision-making processes in strategic interactions.

3.4 Experimental Design

To determine whether humans predict the PSNE as an outcome, we conduct an experiment with human participants within the settings of PGN games. We choose PGN games for two primary reasons. First, we believe that the context of the PGN games represents a natural and intuitive setting for the participants. The game design is simpler to understand, as compared to the normal form game representation, which is a matrix of actions and utilities. This increased intuitiveness is anticipated to improve the participants' comprehension and engagement in the experiment, thus potentially producing more accurate and genuine responses. Second, the PGN game setup offers an opportunity to test human predictions in a situation where most possible predictions are not PSNE. Therefore, if the participants predict a PSNE, it is unlikely that they do so by chance.

We adopt the PGN rule stating that each agent finds it worthwhile to produce a product if none of their neighbors do so. We select this rule due to its simplicity and intuitiveness. In addition, this PGN rule ensures that a PSNE exists in every network, and it is also computationally easy to find a PSNE [6].

For the prediction of a Nash equilibrium, we focus on pure strategy Nash Equilibria (rather than on the general mixed strategy Nash Equilibria) since it appeals to human participants. PSNE is inherently straightforward: each player selects a single action. This aligns well with how individuals typically approach decision-making in real-life scenarios, where they often select a single best course of action based on available information. In contrast, mixed strategy Nash equilibrium involves players randomizing over multiple actions, with each action chosen with a certain probability. This concept can be counterintuitive for many human participants and requires a more advanced grasp of probability theory and strategic thinking.

We consider a binary production case, where individuals in the network either produce the product or not, preventing partial product production. This aligns with the findings by [6], asserting that when utilizing an extension of our PGN rule, the PSNEs do not incorporate strategies involving partial product production.

3.5 Experimental Settings

This section describes the experimental procedure we employ involving human participants. We conduct a survey on the Mechanical Turk platform [40]. The survey is composed of several instruction screens, a quiz, and the main task.

At the beginning of the survey we inform the participants that the survey studies the production of products in networks. We state that the participant's goal is to determine for each person in a given network, whether she believes that person will produce a product or not. We show the participants a simple graph with three nodes (A, B, and C) and edges between them (see Figure 10).

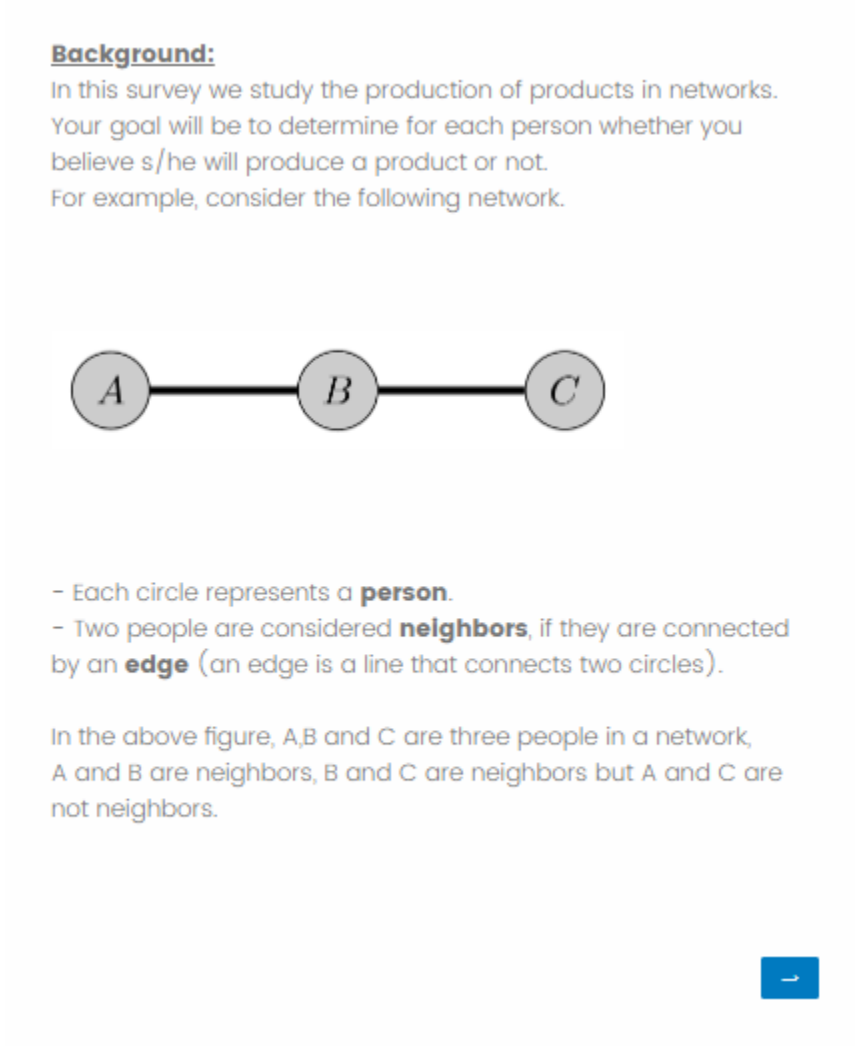

Figure 10: A screenshot from the survey providing the required background.

We then explain that each circle (i.e, node) represents an individual person, and that the connections between the circles, (i.e., edges), symbolize that the people are neighbors.

Next, the survey introduces a simple color code we used to easily show the production status of the individuals in the network. For a person that is producing, their representative circle is colored green. If they are not producing, their circle is colored red, and circles that are colored gray denote individuals for whom we do not know their production status. We then provided the example in Figure 11 to make sure this was well understood as well.

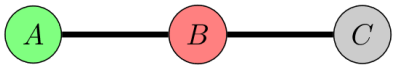

Figure 11: An example for a simple network in which person A is producing (colored green), person B is not producing (colored red), and it is unknown whether C is producing or not (colored grey).

In the next screen, the experiment outlines the decision-making process for production. The production choice of a person is influenced by their neighbors' production status. Specifically, it is worthwhile for a person to produce if none of their neighbors are producing. We explain to the participants, referring back to the example in Figure 11, that for person A, it is worthwhile to produce since their only neighbor, B, is not producing. Similarly, it is worthwhile for B not to produce, as their neighbor, A, is producing. For C, although their production status is unknown, it would be worthwhile for them to produce since their only neighbor, B, is not producing.

We then present another scenario where person B is not producing, while the production status of A and C is unknown (see Figure 12). In this situation, since the only neighbor of both A and C is producing, it is worthwhile for both of them to not produce.

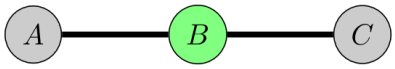

Figure 12: An example for a simple network in which person B is producing (colored green), and it is unknown whether A and C are producing or not (colored grey).

In the following screen, the participants must pass a quiz, demonstrating their understanding of the provided information. They are given the network in Figure 13 and are asked to state that C is producing, B is not producing and it is unknown whether A is producing. Finally, the participants are asked whether it is worthwhile for A to produce (with the answer being "Yes, because his or

her only neighbor (person B) is **not producing**"). The participants are required to answer all of the questions correctly to proceed to the main task.

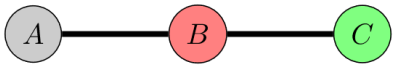

Figure 13: The network that was shown to the participants as a quiz.

The main task is composed of three different networks, in which all people are marked as unknown (i.e., are gray). In each network, the participants are required to state who they believe is producing and who they believe is not producing.

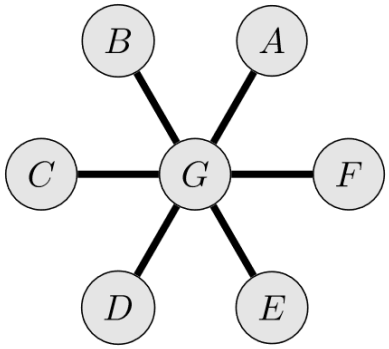

Figure 14: The Star network

The first network is shown in Figure 14. In this network, there are exactly two possible PSNEs: either the middle person G is producing and all other people are not producing, or vice versa (i.e., G is not producing and all the other people are producing). Therefore, 2 out of 2^7 (1.5%) states are PSNE. Finding a PSNE in this network is fairly simple, since it does not contain any triangle, and all but the middle person have exactly one neighbor.

The next network is shown in figure 15. In this network, since it is a complete graph, the only possible PSNE is when exactly one of the people in the network is producing and all of the others are not producing. Therefore, there are exactly five PSNEs (out of the total of 2^5 possible states). Finding a PSNE in this network is slightly confusing, since it is a complete graph, and all people have all the possible connections.

The third network we provided is shown in Figure 16. In this network, there are two types of PSNEs. The first is that either C or D is producing (but not the two of them) and all the others do not produce. The second PSNE is that either E and A or E and B are producing, and the others are

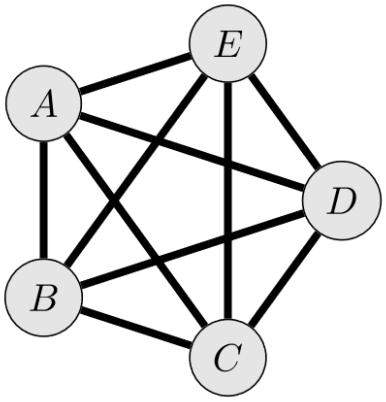

Figure 15: The Complete network

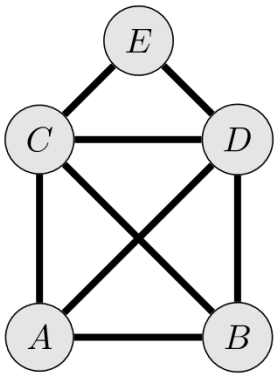

Figure 16: The House network

not. Overall, there are exactly four possible PSNEs in this network, out of a total of 2^5 possible states. Finding a PSNE in this network is also non-trivial.

Finally, we present two proposals for the states of the third network (Figure 16), and ask the participants to indicate whether these proposals are better or worse than their own prediction. The first proposal is shown in Figure 17, where E and B are producing. This prediction is a PSNE, since no one has an incentive to change her decision.

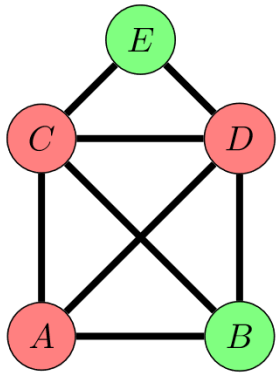

Figure 17

The second proposal is shown in Figure 18, where C and D are producing. Since C and D are neighbors, this state is a *non PSNE*, as for both C and D it is worthwhile to change their decision and to not produce.

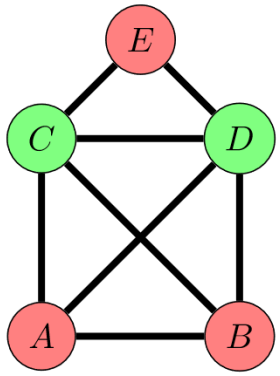

Figure 18

In the final screen of the survey, we ask the participants to provide their gender and year of birth. The participants are also asked whether they have Science, Technology, Engineering and

Mathematics (STEM) background, and whether they have heard of the concept of Nash Equilibrium.

3.6 Results

Overall, there were 96 participants, out of which 48 were females and 48 were males. The average age of the participants was 40.53 years. The average time that took the participants to answer the entire survey was 12.95 minutes.

Regarding the participants background elicited in the final questionnaire, 50 participants (52.1%) said that they have a background in the STEM fields, and 44 (47.9%) said that they have no background in the STEM fields. 30 participants (31.3%) stated that they were familiar with the concept of Nash Equilibrium, 49 (51.1%) were not familiar with this concept and 17 (17.7%) stated that they have heard of concept of Nash Equilibrium but they do not remember the details.

We begin with the analysis of the results for the star graph (Figure 14). In this setting, 61.45% (59 out of 96) predicted an outcome that is a PSNE. That is, even though we did not ask the participants to arrive at a stable outcome, most of them predicted that a PSNE would be the outcome of the game. Recall that there are two PSNEs in this graph, one in which the middle agent (G) produces the product and the other agents do not, or all other agents produce the product but the middle agent does not. In our experiment, within the 59 participants who predicted a PSNE, 35 (59.3%) predicted that the middle agent (G) is producing and all the others do not produce, while 24 (40.7%) predicted that all other agents produce and the middle agent does not. Note that since we (intentionally) did not provide exact information regarding the utility of the agents, there is no reason to prefer one PSNE over the other.

We now turn to analyze the results for the complete graph (Figure 15). In this graph, in every PSNE, one agent is producing and all other agents are not producing. In this setting, 44.79% (43 out of 96) of the participants predicted an outcome that is a PSNE.

Finally, for the House graph (Figure 16), 48.95% (47 out of 96) predicted an outcome that is a PSNE. This graph has two main types of PSNEs, one in which two people are producing (either E and A, or E and B), and one in which only one person is producing (either C or D). In our setting, 85.1% (40 out of 47) predicted the outcome in which two people are producing, and only 14.9% (7 out of 47) predicted the outcome in which one person is producing.

Table 7 provides a detailed breakdown of the frequency of PSNE predictions by the participants, categorized by their gender, age, STEM background, and familiarity with Nash Equilibrium. As depicted by the table, 32.29% (31 out of 96) predicted a PSNE in all three networks. 14.58%

(14 out of 96) predicted a PSNE twice and a non PSNE once, and 29.16% (28 out of 96) predicted a PSNE once and a non PSNE twice. Finally, only 23.95% (23 out of 96) predicted a non PSNE in all the networks.

Furthermore, regarding the parameters of gender and STEM background, it appears that there is no correlation between these factors and the ability to predict a PSNE. However, when taking the age parameter into account, a noticeable trend emerges: as participants get older, their ability to predict PSNE increases. Nevertheless, it is crucial to conduct further investigation before reaching any definitive conclusions.

Interestingly, participants without prior knowledge of Nash Equilibrium performed exceptionally well. Of the 31 who accurately predicted PSNEs for all three networks, 21 reported not being familiar with Nash Equilibrium. This surprising result warrants further investigation to confirm if it represents a genuine trend.

As noted earlier, our participant group was split nearly evenly, with 47 participants predicting a PSNE for the house network, and the remaining 49 predicting a non-PSNE. Our analysis, as presented in Tables 8 and 9, sheds light on how these two subsets of participants evaluated the quality of their own predictions in relation to our given proposals for the house network (Figures 17 and 18).

The perceptions of participants who predicted a PSNE are detailed in Table 8. The majority of these participants viewed the PSNE proposal as equivalent to their own predictions. A smaller subset, curiously, rated our proposal as superior to theirs, despite the identical nature of their prediction and ours. Those few participants who deemed our PSNE proposal worse than their own all shared a common prediction: a scenario where only one of the network's individuals was active, either C or D. This might suggest the influence of individual moral views concerning the distribution of production, even though we did not explicitly introduce this consideration in the task.

When presented with the non-PSNE proposal, the majority of this PSNE-predicting group were able to correctly identify its nature and considered it inferior to their own prediction. Some participants rated our non-PSNE proposal as good as theirs, with a few justifying this stance by pointing out that both proposals had two individuals active in the network. Only a small fraction claimed that the non-PSNE proposal was superior to their own. Taken together, these responses suggest a strong confidence level among those participants who correctly identified a PSNE in the network.

In contrast, Table 9 presents the perspectives of participants who initially proposed a non-PSNE solution. Unfortunately, this group did not demonstrate a clear preference for the PSNE proposal

	Predicted PSNE				
	0 times	**1 time**	**2 times**	**3 times**	**Total**
Total	23	28	14	31	96
Percentage	23.95%	29.16%	14.58%	32.29%	100%
Gender					
Male	9	16	6	17	48
Female	14	12	8	14	48
Age					
Under 40	15	16	6	11	48
Over 40	8	12	8	20	48
Background					
STEM	12	15	5	18	50
Non-STEM	11	13	9	13	46
Knowledge of NE					
Has knowledge	12	12	3	3	30
No knowledge	8	12	8	21	49
Heard of NE	3	4	3	7	17

Table 7: Distribution of responses. Each column under 'Predicted PSNE' indicates the number of times participants predicted a PSNE (out of the 3 networks presented to them). Abbreviations used: NE - Nash Equilibrium.

	PSNE Proposal	**Non-PSNE Proposal**
Better than their offer (+2)	7	4
As good as their offer (+1)	37	9
Worse than their offer (0)	3	34
Mean score	1.09	0.36

Table 8: Evaluations of proposal quality by the 47 participants who proposed a PSNE for the House network. The two proposals in question are depicted in Figures 17 and 18. In general, participants rated the PSNE proposal more highly than the non-PSNE proposal. Ratings were scored as follows: 2 points were assigned if the participant deemed the proposal superior to their own offer, 1 point for parity with their offer, and 0 points if they found the proposal worse than their offer. The 'Mean Score' row reports the average of these ratings.

over their initial non-PSNE prediction, even after the proposals were revealed.

	PSNE Proposal	Non-PSNE Proposal
Better than their offer (+2)	17	21
As good as their offer (+1)	25	18
Worse than their offer (0)	7	10
Mean score	1.02	1.22

Table 9: Quality assessment of the two proposals provided by the 49 participants who proposed a non-PSNE for the house network. Unfortunately, these participants did not favor the PSNE proposal over the non-PSNE proposal.

Overall, the majority of participants predicted a PSNE as the outcome of PGN games at least once. Participants without prior knowledge of Nash Equilibrium performed surprisingly well in predicting PSNE outcomes, suggesting that intuitive reasoning and strategic thinking may play a significant role in decision-making within game theory networks.

Gender, background in STEM fields, and familiarity with the concept of Nash Equilibrium did not appear to have a substantial impact on the participants' ability to predict PSNE outcomes, indicating that these factors may not be strong predictors of success in finding PSNE in PGN games.

3.7 Discussion

In this paper, we show that many humans predict the PSNE as an outcome in PGN games. However, we do not attempt to answer *how* people arrive at the conclusion that a PSNE is a suitable prediction. Indeed, many real-world problems can be naively solved using a greedy algorithm [**parulekar2013automatic**], and it was shown that students tend to rely on a greedy algorithmic approach when solving problems, even in situations where this approach is not suitable or leads to incorrect solutions [**ginat2003greedy**].

In our setting, the following greedy algorithm can be used to find a PSNE. First, the algorithm initializes an empty set, denoted as the independent set (IS). Then, it iterates once over the nodes set of the network. At each iteration, the algorithm checks if the current node has any connections to the nodes already included in the IS. If there are no such connections, the current node is added to the IS. If there is a connection, the node is skipped, and the algorithm continues to the next node. This process repeats until all nodes in the network have been considered. The end result is a maximal IS, where no additional nodes can be added without creating a connection between nodes in the set.

This maximal IS represents a PSNE for the game. Specifically, each player in the IS produces the product, finding it beneficial since none of their neighbors produce it, while players outside the IS do not produce it, as at least one of their neighbors does. Hence, no player has an incentive to unilaterally deviate from their chosen strategy, satisfying the definition of a Nash Equilibrium.

3.8 conclusion and future work

In this study, we explore the ability of human participants to predict the outcome of PGN games without any prior knowledge of the concept of Nash equilibrium. Our findings suggest that even in the absence of explicit information about stability or equilibrium, participants tend to give predictions that align with the PSNE in the simplest setting. This suggests that human intuition and reasoning are capable of converging towards a Nash equilibrium in certain scenarios.

Further research could explore the impact of additional factors, such as individual risk preferences or cultural influences, on prediction outcomes in (PGN). Additionally, studying the decision-making processes in more complex network structures or incorporating real-life scenarios could provide valuable insights into practical applications of game theory.

References

[1] Ameya Agaskar and Yue M Lu. "A fast Monte Carlo algorithm for source localization on graphs". In: *Wavelets and Sparsity XV*. Vol. 8858. SPIE. 2013, pp. 429–434.

[2] Suman Banerjee, Mamata Jenamani, and Dilip Kumar Pratihar. "A survey on influence maximization in a social network". In: *arXiv preprint arXiv:1808.05502* (2018).

[3] Nicola Barbieri, Francesco Bonchi, and Giuseppe Manco. "Topic-aware social influence propagation models". In: *Knowledge and information systems* 37.3 (2013), pp. 555–584.

[4] Amitrajeet A Batabyal. "Markov chains: Models, algorithms and applications". In: *Interfaces* 36.6 (2006), p. 609.

[5] Allan Borodin, Yuval Filmus, and Joel Oren. "Threshold models for competitive influence in social networks". In: *International workshop on internet and network economics*. Springer. 2010, pp. 539–550.

[6] Yann Bramoullé and Rachel Kranton. "Public goods in networks". In: *Journal of Economic theory* 135.1 (2007), pp. 478–494.

[7] Pablo Brañas-Garza, Ismael Rodríguez-Lara, and Angel Sánchez. "Humans expect generosity". In: *Scientific Reports* 7.1 (2017), pp. 1–9.

[8] Colin F Camerer. "Behavioral game theory: Plausible formal models that predict accurately". In: *Behavioral and Brain Sciences* 26.2 (2003), pp. 157–158.

[9] Wei Chen, Tian Lin, and Cheng Yang. "Efficient topic-aware influence maximization using preprocessing". In: *CoRR, abs/1403.0057* (2014).

[10] Samik Datta, Anirban Majumder, and Nisheeth Shrivastava. "Viral marketing for multiple products". In: *2010 IEEE International Conference on Data Mining*. IEEE. 2010, pp. 118–127.

[11] Mario Diani. "Social movements, contentious actions, and social networks:'From metaphor to substance'". In: *Social movements and networks: Relational approaches to collective action* (2003), pp. 1–20.

[12] Pedro Domingos and Matt Richardson. "Mining the network value of customers". In: *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2001, pp. 57–66.

[13] Harold N Gabow and Eugene W Myers. "Finding all spanning trees of directed and undirected graphs". In: *SIAM Journal on Computing* 7.3 (1978), pp. 280–287.

[14] Matan Gilboa and Noam Nisan. “Complexity of public goods games on graphs”. In: *Algorithmic Game Theory: 15th International Symposium, SAGT 2022, Colchester, UK, September 12–15, 2022, Proceedings*. Springer. 2022, pp. 151–168.

[15] Jacob K Goeree and Charles A Holt. “Ten little treasures of game theory and ten intuitive contradictions”. In: *American Economic Review* 91.5 (2001), pp. 1402–1422.

[16] Jacob Goldenberg, Barak Libai, and Eitan Muller. “Talk of the network: A complex systems look at the underlying process of word-of-mouth”. In: *Marketing letters* 12.3 (2001), pp. 211–223.

[17] Manuel Gomez-Rodriguez, Jure Leskovec, and Andreas Krause. “Inferring networks of diffusion and influence”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 5.4 (2012), pp. 1–37.

[18] Christian Hilbe. “Equilibrium notions and framing effects”. In: *arXiv preprint arXiv:1012.1188* (2010).

[19] Jiaojiao Jiang et al. “Identifying propagation sources in networks: State-of-the-art and comparative studies”. In: *IEEE Communications Surveys & Tutorials* 19.1 (2016), pp. 465–481.

[20] Rong Jin and Weili Wu. “Schemes of propagation models and source estimators for rumor source detection in online social networks: A short survey of a decade of research”. In: *Discrete Mathematics, Algorithms and Applications* (2021), p. 2130002.

[21] 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*. ACM. 2003, pp. 137–146.

[22] David Kempe, Sixie Yu, and Yevgeniy Vorobeychik. “Inducing equilibria in networked public goods games through network structure modification”. In: *arXiv preprint arXiv:2002.10627* (2020).

[23] Jinha Kim, Wonyeol Lee, and Hwanjo Yu. “CT-IC: Continuously activated and time-restricted independent cascade model for viral marketing”. In: *Knowledge-Based Systems* 62 (2014), pp. 57–68.

[24] Ankit Kumar, Vivek S Borkar, and Nikhil Karamchandani. “Temporally agnostic rumor-source detection”. In: *IEEE Transactions on Signal and Information Processing over Networks* 3.2 (2017), pp. 316–329.

[25] Theodoros Lappas et al. “Finding effectors in social networks”. In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2010, pp. 1059–1068.

[26] F Leighton and Ronald Rivest. “Estimating a probability using finite memory”. In: *IEEE Transactions on Information Theory* 32.6 (1986), pp. 733–742.

[27] Jure Leskovec et al. “Cost-effective outbreak detection in networks”. In: *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2007, pp. 420–429.

[28] David A Levin and Yuval Peres. *Markov chains and mixing times*. Vol. 107. American Mathematical Soc., 2017.

[29] Weihua Li et al. “Modelling multiple influences diffusion in on-line social networks”. In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems. 2018, pp. 1053–1061.

[30] Yuchen Li et al. “Influence maximization on social graphs: A survey”. In: *IEEE Transactions on Knowledge and Data Engineering* 30.10 (2018), pp. 1852–1872.

[31] Arnab Maiti and Palash Dey. “On parameterized complexity of binary networked public goods game”. In: *arXiv preprint arXiv:2012.01880* (2020).

[32] Richard D McKelvey and Thomas R Palfrey. “Quantal response equilibria for normal form games”. In: *Games and economic behavior* 10.1 (1995), pp. 6–38.

[33] Michel Minoux. “Accelerated greedy algorithms for maximizing submodular set functions”. In: *Optimization techniques*. Springer, 1978, pp. 234–243.

[34] Roger B Myerson. *Game theory: analysis of conflict*. Harvard university press, 1991.

[35] In Jae Myung. “Tutorial on maximum likelihood estimation”. In: *Journal of mathematical Psychology* 47.1 (2003), pp. 90–100.

[36] Ramasuri Narayanam and Amit A Nanavati. “Viral marketing for product cross-sell through social networks”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2012, pp. 581–596.

[37] John F Nash Jr. “Equilibrium points in n-person games”. In: *Proceedings of the national academy of sciences* 36.1 (1950), pp. 48–49.

[38] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. “An analysis of approximations for maximizing submodular set functions—I”. In: *Mathematical programming* 14.1 (1978), pp. 265–294.

[39] Naoto Ohsaka and Yuichi Yoshida. “Monotone k-submodular function maximization with size constraints”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 694–702.

[40] Gabriele Paolacci, Jesse Chandler, and Panagiotis G Ipeirotis. “Running experiments on amazon mechanical turk”. In: *Judgment and Decision making* 5.5 (2010), pp. 411–419.

[41] Christos Papadimitriou and Binghui Peng. “Public goods games in directed networks”. In: *Proceedings of the 22nd ACM Conference on Economics and Computation*. 2021, pp. 745–762.

[42] Devavrat Shah and Tauhid Zaman. “Detecting sources of computer viruses in networks: theory and experiment”. In: *Proceedings of the ACM SIGMETRICS international conference on Measurement and modeling of computer systems*. 2010, pp. 203–214.

[43] Devavrat Shah and Tauhid Zaman. “Rumors in a network: Who’s the culprit?” In: *IEEE Transactions on information theory* 57.8 (2011), pp. 5163–5181.

[44] Micha Sharir. “A strong-connectivity algorithm and its applications in data flow analysis”. In: *Computers & Mathematics with Applications* 7.1 (1981), pp. 67–72.

[45] Sushila Shelke and Vahida Attar. “Source detection of rumor in social network–a review”. In: *Online Social Networks and Media* 9 (2019), pp. 30–42.

[46] Guangmo Amo Tong et al. “Effector detection in social networks”. In: *IEEE Transactions on Computational Social Systems* 3.4 (2016), pp. 151–163.

[47] Sonja Utz. “The (potential) benefits of campaigning via social network sites”. In: *Journal of computer-mediated communication* 14.2 (2009), pp. 221–243.

[48] Zhijian Wang, Bin Xu, and Hai-Jun Zhou. “Social cycling and conditional responses in the Rock-Paper-Scissors game”. In: *Scientific reports* 4.1 (2014), pp. 1–7.

[49] Justin Ward and Stanislav Živnỳ. “Maximizing k-submodular functions and beyond”. In: *ACM Transactions on Algorithms (TALG)* 12.4 (2016), p. 47.

[50] David Wills and Stuart Reeves. “Facebook as a political weapon: Information in social networks”. In: *British Politics* 4.2 (2009), pp. 265–281.

[51] James Wright and Kevin Leyton-Brown. “Beyond equilibrium: Predicting human behavior in normal-form games”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 24. 1. 2010, pp. 901–907.

[52] Wan-Shiou Yang et al. “Mining social networks for targeted advertising”. In: *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*. Vol. 6. IEEE. 2006, 137a–137a.

[53] Yongjie Yang and Jianxin Wang. “A refined study of the complexity of binary networked public goods games”. In: *arXiv preprint arXiv:2012.02916* (2020).

[54] Sixie Yu et al. “Computing equilibria in binary networked public goods games”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 02. 2020, pp. 2310–2317.

[55] Zahra Zahedi, Sailik Sengupta, and Subbarao Kambhampati. “Inference of Human's Observation Strategy for Monitoring Robot's Behavior based on a Game-Theoretic Model of Trust”. In: *arXiv preprint arXiv:1903.00111* (2019).

[56] Xuming Zhai, Weili Wu, and Wen Xu. “Cascade source inference in networks: a Markov chain Monte Carlo approach”. In: *Computational social networks* 2.1 (2015), pp. 1–17.

[57] Le Zhang et al. “A Markov chain monte Carlo approach for source detection in networks”. In: *Chinese National Conference on Social Media Processing*. Springer. 2017, pp. 77–88.

אבסטרקט

עבודת הדוקטורט הזו חוקרת שלושה היבטים נפרדים של רשתות חברתיות: מקסום השפעה, זיהוי מקור וחיזוי של תוצאות משחקי טובין ציבוריים.

בפרק הראשון, אנו ניגשים לבעיית מקסום ההשפעה ברשתות חברתיות. בעיה חישובית זו עוסקת בבחירת קבוצה ראשונית של משתתפים אשר תביא למקסימום את ההתפשטות ברשת. אנו ניגשים לבעיה זו מנקודת מבט תועלתנית, שבה נלקחים בחשבון שני סוגי הודעות בעלות הסתברויות שונות של התפשטות ברשת ורמת תועלת שונה. באמצעות ניתוח של פונקציות בי-סובמודולריות, אנו מציעים אלגוריתם חמדני עם יחס קירוב הדוק של 1/2. יתר על כן, אנו מציגים אלגוריתם מבוסס תכנות דינמי העולה על הביצועים של הגישה החמדנית בסימולציות נרחבות.

נעבור לפרק השני, פה אנו ניגשים לבעיה הפוכה. אנו מתמקדים בזיהוי המקור הכי סביר של התפשטות מידע ברשת, בהינתן קבוצת המשתמשים אשר נחשפו להודעה. אנו מציגים שיטה יעילה המבוססת על חישוב ההתפלגות הסטציונרית של שרשרת מרקוב כדי להתמודד עם בעיה זו. באמצעות סימולציות, אנו מדגימים את היעילות והאפקטיביות של השיטה שלנו בהשוואה לטכניקות מתקדמות קיימות.

בפרק השלישי, אנו מתעמקים ביכולתם של נבדקים אנושיים לחזות את התוצאות של משחקי טובין ציבוריים ברשתות, ללא ידע מוקדם של תורת המשחקים ומושגי תורת הגרפים. אנו רואים מגמות מסקרנות שבהן רוב המשתתפים מספקים תחזיות המתואמות עם שיווי משקל נאש באסטרטגיות טהורות. הממצאים שלנו מצביעים על כך שבתרחישים מסוימים, שיווי המשקל של נאש מתכתב עם האינטואיציה וההיגיון האנושיים במשחקים אסטרטגיים.

עבודת גמר זו תורמת להבנה מקיפה של רשתות חברתיות על ידי התייחסות לאתגרים חישוביים במקסום השפעה וזיהוי מקור, וכן לשפוך אור על קבלת החלטות אנושית בחיזוי תוצאות המשחק.

אוניברסיטת אריאל בשומרון

חילחול מידע וקבלת החלטות קולקטיבית ברשתות חברתיות

חיבור זה הוגש כחלק מדרישות התואר
"דוקטור לפילוסופיה"

מאת

יעל סבתו

עבודה זו נכתבה בהנחיית (שמות המנחים)
פרופ' עמוס עזריה
ד"ר נועם חזון

מוגש לסנאט אוניברסיטת אריאל בשומרון

תאריך יוני 2023