# Targeted and Budgeted Influence Maximization in Social Networks under Deterministic Linear Threshold Model

A thesis submitted to the
Graduate School of Natural and Applied Sciences

by

Furkan GÜRSOY

in partial fulfillment for the
degree of Master of Science

in
Data Science

İSTANBUL
ŞEHİR
ÜNIVERSITY

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science in Data Science.

**APPROVED BY:**

| | |
|---|---|
| Assist. Prof. Dr. Dilek Günneç Danış (Thesis Advisor) | [illegible] |
| Prof. Dr. Necati Aras | [illegible] |
| Assist. Prof. Dr. Barış Arslan | [illegible] |

This is to confirm that this thesis complies with all the standards set by the Graduate School of Natural and Applied Sciences Graduate School of Natural and Applied Sciences of İstanbul Şehir University:

**DATE OF APPROVAL:** 10/08/2017

**SEAL/SIGNATURE:** [illegible]

# Declaration of Authorship

I, Furkan GÜRSOY, declare that this thesis titled, 'Targeted and Budgeted Influence Maximization in Social Networks under Deterministic Linear Threshold Model' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: [illegible]

Date: 10.08.2017

# Targeted and Budgeted Influence Maximization in Social Networks under Deterministic Linear Threshold Model

Furkan GÜRSOY

## Abstract

We define the new Targeted and Budgeted Influence Maximization under Deterministic Linear Threshold Model problem by extending the original influence maximization problem to a targeted version where nodes might carry heterogeneous profit values, and to a budgeted version where nodes might carry heterogeneous costs for becoming seed nodes. As a solution to this problem, we develop a novel and scalable general algorithm which utilizes a set of alternative methods for different operations: TArgeted and BUdgeted Potential Greedy (TABU-PG) algorithm.

TABU-PG works in an iterative and greedy fashion where nodes are compared at each iteration and the best one(s) are chosen as seed. The main idea behind TABU-PG is to invest in potential future gains which are hoped to be materialized at later iterations. Alternative methods are provided for calculating potential gain, and for comparing nodes. Some methods are taken from the literature while others are novel methods introduced by us. In comparing nodes, we propose a hybrid model which considers both gain and efficiency. In calculating potential gains, we propose methods which dynamically assign suitable weights to potential gains based on remaining budget. We also propose a new method which ignores the potential gains which are results of partial influences under a parameterized ratio. Moreover, we equip TABU-PG with novel scalability methods which reduces runtime by limiting the seed node candidate pool, or by selecting more nodes at each iteration; trading-off between runtime and spread performance. In addition, we suggest new data generation methods for influence weights on links; and threshold, profit, and cost values for nodes which better mimics the real world dynamics.

Extensive computational experiments with 8 different dataset on 4 real-life networks show that TABU-PG heuristics perform significantly better than benchmark heuristics. Moreover, runtime can be reduced with very limited reduction in final influence spread.

**Keywords:** Social Networks, Influence Maximization, Diffusion Models, Targeted Marketing, Greedy Algorithm

Sosyal Ağlarda Belirlenimci Doğrusal Eşik Modeli altında
Hedefli ve Bütçeli Etki Enbüyükleme

Furkan GÜRSOY

# Öz

Orijinal etki enbüyükleme problemini, düğümlerin farklı fayda değerleri taşıyabildiği hedefli ve düğümlerin tohum düğüm olmak için farklı maliyet değerleri taşıyabildiği bütçeli bir problem versiyonuna genişleterek, yeni Belirlenimci Doğrusal Eşik Modeli altında Hedefli ve Bütçeli Etki Enbüyükleme problemini tanımlıyoruz. Çözüm olarak, çeşitli işlemler için alternatif yöntemler barındıran, özgün ve ölçeklenebilir bir genel algoritma geliştiriyoruz: Hedefli ve Bütçeli Potansiyel Açgözlü (TABU-PG) Algoritma.

TABU-PG döngülü ve açgözlü bir biçimde çalışır. Her döngüde düğümler karşılaştırılır ve en iyisi/iyileri tohum düğüm olarak seçilir. TABU-PG'nin ana fikri, sonraki döngülerde somutlaştırılabilecek potansiyel kazançlara yatırım yapmaktır. Potansiyel kazançları hesaplamak ve düğümleri karşılaştırmak için alternatif yöntemler sağlanmıştır. Kimi yöntemler literatürden alınmışken, diğer yöntemler bizim tarafımızdan önerilen özgün yöntemlerdir. Düğümleri karşılaştırırken, hem kazancı hem verimliliği dikkate alan melez bir yöntem öneriyoruz. Potansiyel kazançları hesaplarken, potansiyel kazançlar için uygun ağırlıkları, kalan bütçe miktarından yola çıkarak dinamik biçimde atayan özgün yöntemler öneriyoruz. Aynı zamanda, parametreyle kontrol edilen bir değerin altında kalan kısmi etki oranlarından kaynaklanan potansiyel kazançları yoksayacak bir yöntem de öneriyoruz. Ayrıca, tohum düğüm aday havuzunu daraltarak veya her bir döngüde daha fazla düğüm seçerek, TABU-PG'nin çalışma süresini önemli ölçüde düşüren özgün ölçekleme yöntemleri sunuyoruz. Bu ölçekleme yöntemleri, çalışma süresi ve yayılma performansı arasında ödünleşerek çalışır. Ek olarak, bağlar üzerindeki etki ağırlıkları ve düğümler üzerindeki eşik, fayda ve maliyet değerleri için gerçek hayat dinamiklerini daha iyi yansıttığını düşündüğümüz yeni veri türetim yöntemleri öne sürüyoruz.

Gerçek hayattaki 4 sosyal ağ baz alınarak oluşturduğumuz 8 farklı veri setinde uygulanan kapsamlı sayısal deneyler gösteriyor ki; TABU-PG buluşsal algoritmaları, denektaşı buluşsal algoritmalarına göre önemli ölçüde daha iyi performans gösteriyor. Ek olarak, nihai etki yayılmasındaki kısıtlı bir düşüş karşılığında, çalışma süresi de anlamlı bir biçimde düşürülebiliyor.

**Anahtar Sözcükler:** Sosyal Ağlar, Etki Enbüyükleme, Yayılma Modelleri, Hedefli Pazarlama, Açgözlü Algoritma

*To my parents, Gülsüm and Ahmet, who are always there for me.*

# Acknowledgments

First and foremost, I would like to express my gratitude to my advisor, Prof. Dilek Günneç Danış, for the continuous support in my research. Without her valuable feedback and guidance, this thesis would not have been become what it is now.

Besides my advisor, I would like to thank the rest of my thesis committee, Prof. Necati Aras and Prof. Barış Arslan, for their insightful comments.

I also thank Filip Rak for his help in crawling one of the social network datasets we employ in this study.

The author is supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under 2210-A Program.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

As social process of influence intensely and frequently takes place in networks of people, marketers try to take advantage of it in order to increase the adoption of their products - or ideas and behaviours. With the development of online social networks, it has become more feasible as the structure of networks and other relevant information can now be collected and analyzed.

Social media in particular and social networks in general play an increasingly important role in our daily lives, and in the world in general. Politics, business, and other matters of our lives are significantly shaped by social influence we are exposed to via the social networks we are part of. One might argue that it was the case even hundreds of years ago with physical social networks, however the social networks are not bounded anymore by the smaller social circle which consists of people who we mostly know personally. The world has evolved to be a place where information can spread very quickly and easily to far distances when compared to even few decades ago. It might further be argued that information in our age is not different from before only in terms of its source, but also in terms of its potential use and social effects due to the democratization and industrialization that has been seen over the past few centuries.

Given the significance of diffusion of information in modern social networks; we study how influence spreads, and how the spread and adoption of a desired product or idea can be maximized over a given social network by finding influential user sets and subsidizing them to promote the product or idea. The general methodology we develop in this work, in part or in full, can be of use in other diffusion problems on networks.

## 1.1 Problem Statement

In a viral marketing campaign executed over a social network, ideally, subsidizing a few influential people to promote a certain product will create a cascade in the network where influential friends, friends of friends and so on will adopt the product. Therefore, the problem is to select a set of influentials in such a way where influence spread is maximized while the cost of subsidizing the influentials is kept within a given budget.

We define the particular problem we are studying as the Targeted and Budgeted Influence Maximization under Deterministic Linear Threshold Model. Basically, the aim is to maximize the total profit (i.e., influence spread) by activating selected nodes as seeds via external intervention (e.g., subsidizing them to become seed nodes) on a network where the activation function is defined by the Deterministic Linear Threshold Model.

A more formal and detailed definition of the problem is presented in Section 4.1.

## 1.2 Our Contribution

In this study, we define the new Targeted and Budgeted Influence Maximization under Deterministic Linear Threshold problem. This problem differs from the existing studies in the literature by extending the original influence maximization problem[1] to a targeted version of the problem where nodes might carry heterogeneous profit values, and extending to a budgeted version of the problem where nodes might carry heterogeneous costs for becoming seed nodes; in deterministic networks.

Our main contribution is the development of a new algorithm named Targeted and Budgeted Potential Greedy (TABU-PG) for the problem we defined. The algorithm employs a variety of method options for different operations. Therefore, selecting different methods results in different TABU-PG heuristics sharing the same framework.

Moreover, we propose novel methods to enable TABU-PG heuristics to run on very large networks in a significantly shorter amount of time by trading spread performance (i.e., total profit) with better runtime. For example, by utilizing the scalability methods we introduced, a TABU-PG heuristic which runs on a network with over a million nodes and over 14 million links takes approximately 1 hour instead of 11 hours while performing nearly the same as original.

In addition, we propose new methods for generating influence weights for links; and threshold, profit, and cost values for nodes. In our opinion, in many cases, our methods reflect the real world more accurately than the existing methods employed in the literature.

Lastly, we provide empirical evaluations of TABU-PG heuristics and benchmark heuristics such as closeness, betweenness, pagerank, hub, authority, eigenvector, and benchmark heuristics. With extensive computational experiments we show how all heuristics perform with 8 different datasets on 4 different real-life networks.

## 1.3 Thesis Structure

This work is organized as follows.

In Chapter 2, we review how people's decisions and behaviours are shaped by social influence. We then approach the topic from a social network analysis perspective. Background on basic concepts on networks which are used in the rest of the work, hence helpful for understanding this work, is given. Lastly, most popular diffusion models in social networks, which are essential to comprehend the related works and also our methodology, are explained.

In Chapter 3, we review how influence maximization problem emerged and developed. We cover the existing solution methods in the literature for influence maximization problem as well as its extensions. We complete the chapter with comparative analysis of our problem and existing problems in the literature, and present our contribution.

In Chapter 4, we begin with formally defining our problem. Next, we propose a novel algorithm, explain it in detail, and illustrate how it works. Then, the graphs (i.e., social networks) which are used in this work are given along with descriptions. Lastly, we present our data generation methods which we use to create non-existing data which are required for our algorithm, or to transform the existing data into a desired form.

In Chapter 5, we present experimental results and provide a comprehensive discussion on the results. First, data preprocessing steps which are utilized in generation of final datasets are explained. Then, experimental results on 8 different datasets which are created from 4 different social networks are presented along with initial comments. The chapter ends with an overall and comparative discussion on performance and runtime results of our heuristics and benchmark heuristics.

The conclusion and final remarks are given in Chapter 6.

# Chapter 2

# Background on Diffusion in Social Networks

In this chapter, we will review basic concepts which are necessary to understand the significance and fundamentals of the problem and solution we study. In the first section, we explore how social influence occurs in social networks and shapes people's decisions. Next, we review basic concepts in social networks, particularly focusing on the concepts which we are going to refer in later sections. In the last section, we explain the diffusion models in social networks which are studied in the relevant literature.

## 2.1 Social Influence

In a connected world, people's decisions and behaviours are influenced by others. Social influence can be observed in sales, marketing, persuasion, peer pressure, conformity, and leadership. Social influence can be categorized into two categories: informational social influence and normative social influence [2].

Informational social influence occurs when one accepts arguments provided by others when the person in question is uncertain about the subject. On the other hand, normative social influence occurs when one conforms, with or without private acceptance, in order to be liked by a group. Decisions and behaviours, such as purchasing of a product or adopting a particular lifestyle, are shaped by both of these types of social influence.

### 2.1.1 Role of influence in decision making

The influence may occur either at a conscious level or at an unconscious level. In the case of influence happening at a conscious level, people choose whether to be influenced or not as a result of some rational decision making process. On the other hand, one does not always have the control over whether to be influenced or not as influence can also happen at an unconscious level.

For the influence at a conscious level, there are two types of benefits in imitating the decisions of others: the direct-benefit effect and the informational effect.

**Direct-benefit effect** takes place when one's payoff from her own action is directly affected by other people's actions. This phenomenon is also called as network effect. In a setting with network effects, one's adoption decision is affected both by her own intrinsic interest and by quantity and composition of other people who have already adopted.

The composition refers to wide-ranging information on the people's roles and relations in their social network. For example, if some of one's colleagues start using a certain *Product B*, one's payoff in switching from *Product A* to *Product B* increases given that collaborating with them will become much easier.

A product with network effects needs to be adopted by a significant portion of all consumers before larger crowds start to adopt the product. Therefore, there exist tipping points (i.e., certain share of the market) such that once it is reached, the upward demand will easily increase the product's market share. This can be illustrated by the historical adoption rates of fax machine which quickly peaked after it slowly reached to a tipping point [3].

In the case where multiple products compete for the same market, if the role of network effects is significant, usually one of the competing products dominate the market at the end. Assuming that these products are comparable, the first product which successfully gets adopted by larger portion of the consumers has a significant advantage over others to become the market leader since a new consumer will tend to choose it because her payoff will be higher due to network effects, even though the functionality of the competing products is basically the same. Windows' domination in operating system market for PCs illustrates this phenomenon.

**The informational effect**, also called indirect-effect, occurs when collective information is more powerful than one's private information. In a setting where one has limited information on which action to prefer, the decision is more likely to be made by mimicking others' decisions. In fact, that is the rational thing to do since collective information is likely to be more accurate than one's own limited private information.

The informational effect is different from direct-benefit effect since one's payoff is not necessarily changed due to others' actions. Rather, it is one's information that is changed. Consider the following famous example: there are two neighbouring restaurants; one restaurant has a long line of customers waiting and the other restaurant has virtually no customer. A newly arriving customer will think that there must be a good reason that people fall in line for one restaurant and not for the other. Her private, thus prior, information about which restaurant to prefer will be influenced by the information that the majority seems to have.

This behaviour is called as herding. A person cannot directly observe the private information of others but can make inferences from their actions. So that, herding behavior occurs when people's decisions depend on the inferences from the earlier decisions of other people. The interesting part is that a single person can be the only reason behind a whole cascade. In the restaurant example; the second customer makes the same choice as the first customer, the third customer makes the same choice as the second customer and so on. Choice of the first customer, possibly a random choice, might result in most people preferring the same restaurant as her.

Despite the fact that herding behavior is the rational thing to do in the case where one has a limited or non-existent information about the decision to be made, the results in a herding model might not be optimal at the end. The earlier decisions might be taken under random circumstances or might be manipulated on purpose. Therefore, a small number of people at the beginning can affect the direction of the whole cascade. This phenomenon explains how certain businesses or products rise to prominence although they are not superior to their competitors.

Contrary to influence at a conscious level, people can also be influenced without any conscious rational decision making process. Social conformity, mirroring and other psychological effects are few explored examples of such situations [4].

### 2.1.2 Influence in social networks

Along with advancements in scientific methods and with the aid of computers, the process of influence has started to be studied from the perspective of social network analysis. Understanding how influence spreads in social networks plays a crucial role in modeling and developing solutions to the problems in social network analysis.

Theoretical roots of Social Network Analysis lie in Psychology, Anthropology, and Mathematics [5]. Moreno developed Sociometry and aimed to explore the relationship between psychological well-being and social configurations, and formation of groups [6]. Euler's Konigsberg Bridge problem [7], which later enabled the development of Markov probability chains in Statistics, also can be given as an example of the roots of Social Network Analysis in Mathematics. Today, Social Network Analysis is a well-developed area with increasing interdisciplinary approach.

The earliest studies about influence propagation in social networks took place in the middle of the 20th century. In his famous book, Diffusion of Innovations [8], Rogers brings together number of studies which study how innovations in agricultural methods and tools spread in the rural communities. He paved the way for the development of notions such as strength of weak ties, tipping point, phases of adoption, and categories of technology adopters. These notions have all been extensively studied later and become central themes in social network analysis and other fields.

Diffusion of innovation in the social network and adoption are not necessarily equivalent. One might be aware of a new product however might not immediately adopt it. When it comes to adoption, there are five groups of consumers: innovators, early adopters, early majority, late majority, and laggards.

Innovators are often the visionaries and enthusiasts who are willing to adopt new innovations immediately without requiring anyone in their network to priorly adopt the innovation. They make up 2.5% of the population. Early adopters are like Innovators but they are not as fast as them when adopting an innovation. They wait until they see a few people adopt before adopting themselves. They make up 13.5% of the population.

Early majority is the pragmatists who notice the benefits of adoption after a certain portion of people adopt the product. With 34% of the population, they constitute a large group of people who wait more than the previous two groups before adoption and only adopt after more careful consideration.

Late majority is the conservatives who resist to change unless significant portion of the population adopts the change. They are the same size of the early majority group. Laggards are the remaining 16% of the population and they are very unlikely to adopt an innovation even though majority of the population already adopted.

However, it might be misleading to assume that these groups are strictly separated from each other. Rather, each consumer has a position in the scale from innovative to laggard. A person in the early majority group can be closer to early adopters or late majority. Figure 2.1 shows the distribution of consumers in terms of tendency to adopt an innovation.

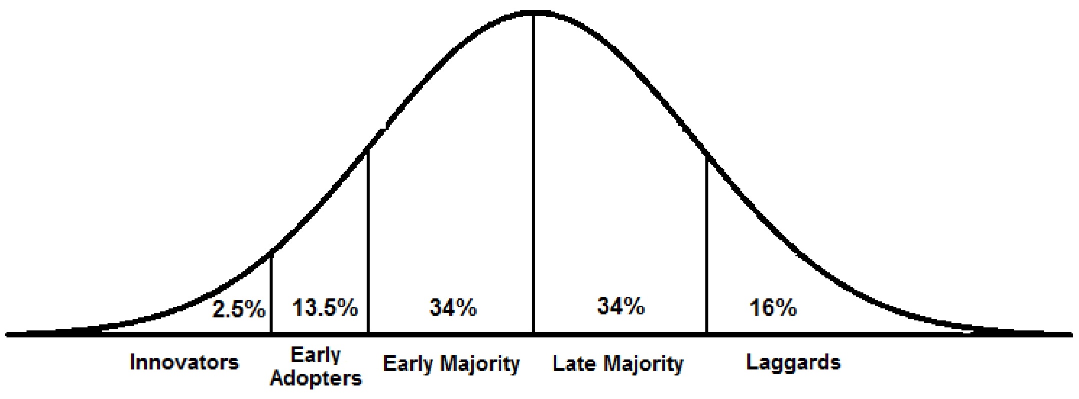

FIGURE 2.1: Five Groups of Consumers in Adoption of Innovation

## 2.2 Basic Concepts on Networks

A graph is a diagram which depicts the system of any relations between any two or more things. On the other hand, a social network is a social structure comprised by actors and social interactions between the nodes (i.e., the actors). A social network can be represented as a social graph using graph notations. We will use the term social network for the rest of this study. The two main constituents of social networks are nodes which represent the actors and links which represent the interactions. A social network with 12 nodes and 19 links is shown in Figure 2.2.

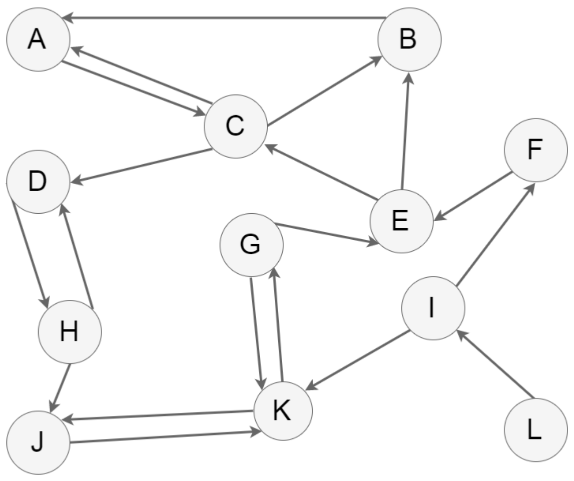

FIGURE 2.2: A sample social network

Nodes, commonly referred as vertices in graph theory, represent things such as people, webpages, places and anything else which can be modelled as actors in a system. Nodes might have properties associated with them. Values for these properties can be assigned externally or can be calculated based on the structure of the network. Demographics can be given as an example of externally assigned property whereas centrality measures are calculated based on the structure of the network.

Links, commonly referred as edges in graph theory, are the representations of interactions between nodes. Friendships, co-authorships, rating the same product are some examples of links. A link constitutes a neighborship between the two nodes it connects. Links can be directed or undirected (also called as bi-directional). For example, on Twitter, the *follow* action constitutes a directional link from the follower to the followed user. On the contrary, friendship links on Facebook are undirected since $A$ is always a friend of $B$ given that $B$ is a friend of $A$. A social network is called a directional social network if at least one of the links is directed and called a bidirectional social network if all of the links are undirected.

### Centrality measures

Usually, a tiny fraction of nodes plays a disproportionately large role in diffusion mechanisms in social networks (i.e., *power law distribution*). Therefore, finding methods of identifying such influential nodes is a notable research problem. It might be intuitive to assume that nodes which are more central in a social network are more important and thus more influential. There are different ways of measuring centrality and most of them are based on the relative position of a node in the network. degree, closeness, betweenness, pagerank, hub, authority, and eigenvector centrality scores are among the most popular measures of centrality.

**Degree centrality** is the measure of number of links a node has. The intuition is that a node is more important if a node has more links, therefore connections, with other nodes. For a directional network, indegree centrality and outdegree centrality is measured by the number of incoming links and outgoing links, respectively. Compared with other centrality measures, degree centrality is a simpler approach because it does not consider the relative location of a node in the network. [9] showed that degree is not necessarily an indicator of influence in microblogging networks.

**Closeness centrality** is the measure of how easy it is for a node to reach other nodes. The easiness implies the shortness of the paths. Therefore, closeness is average length of the shortest paths from the node to all other nodes. The intuition is that a node is more important if it is close to all other nodes on average since it can reach them more easily.

**Betweenness centrality** is the measure of how frequently a node is on the shortest paths between all node pairs in a social network. The intuition is that a node is more important if it plays a larger role in connecting other nodes in shorter paths.

**PageRank score** is based on the probability distribution of random walk on graphs. In a simplistic intuitive view, a node which has a link to another node is effectively casting for a vote for the latter node. Thus, the network can be represented as nodes voting other nodes which they have link to. However, not all votes are equal. Importance of the vote is determined by the importance of the voting node, and a node with higher PageRank score is more important. Therefore, PageRank score is calculated in a recursive manner. The PageRank algorithm was initially developed by Page et al. [10] to rate Web pages objectively and mechanically. It has been a significant part of Google's search algorithm.

**Hub** score is higher for a node if it has more outgoing links to more authoritative nodes. **Authority** score is higher for a node if it has more incoming links from more hub nodes. Therefore, they exhibit a mutually reinforcing relationship. Using this relationship, hub scores and authority scores can be calculated in a recursive manner. The HITS algorithm is developed by Kleinberg [11] to calculate Hub and Authority scores.

**Eigenvector centrality**, also known as eigencentrality, is a measure of influence where a node is more central if it is connected to nodes which are more central, measured by the size of their eigencentrality values. Hence, Eigenvector centrality utilizes a reinforcing relationship in calculation of eigencentrality. For example, PageRank score is a variation of Eigenvector Centrality. However, throughout this study, we will refer to the version in [12] as Eigenvector centrality.

## 2.3 Diffusion Models in Social Networks

Information might spread in three ways: epidemics, herding and information cascades in networks [13].

Epidemic models generally assume implicit network and unknown connections among nodes. Epidemic models are largely used to model spread of diseases. However, rather than focusing on network structure, it generally focuses more on infection and recovery rates at a global level. Although contact networks, a type of epidemic models, assume connections between nodes; these nodes are not necessarily close in terms of real-world proximity. Sharing the same air by chance with a total stranger can be considered a connection while modelling the spread of a flu virus.

Herd behavior describes the way people align their actions in a group without previous planning. There are two prerequisites for a herd behavior to occur: connections between individuals and a method to transfer behavior among individuals or to observe their behavior. An example is when you divide a group of people into two and ask them to guess the population of a random country. First group is instructed to write their answers on a paper without showing it to others. The second group is instructed to answer in a way that everyone in the group can hear. In this experiment, the people in the second group approximate their original answers to the previous answers of others while the first group gives more varying answers.

For herding, there is no need for a network in theory, however a well-connected network exists in practice. As the network should be a near-complete graph for herding behaviour to take place, herding behavior does not apply to most social networks where underlying network is rather sparse. However, it can still be observed in densely-connected clusters which are part of a sparse network. Also, the trending section of microblogs (such as Twitter) can be an example of where herding takes place as users become aware of what many other people talk about although these people are not necessarily in their immediate network.

In contrast to the epidemics and herding, people's interactions predominantly take place at the local level rather than at a global level. People in general are more interested in opinions of their own network than opinions of the whole population. This phenomenon also explains people aligning with their friends although they are a minority in the global population. Thus, structure of network plays a significant role in spread of information, or diffusion of innovation.

Information cascades in social networks can be modelled by employing Markov random fields, voter models, Independent Cascade Model, and Linear Threshold Model. The latter two are investigated in detail in the following sections.

### 2.3.1 Independent Cascade Model

Independent Cascade Model (ICM) [1] assumes that diffusion time steps are discrete. At any time, a node can be either active (i.e., influenced) or inactive. An active node may attempt to activate a neighbouring inactive node only once (i.e., it has a single chance), and a node cannot become inactive later once it is active (i.e., a progressive model).

The process starts with initially active nodes which serve as the seed nodes. A node $v$ that is activated at time $t$ tries to activate its inactive neighbour node $u$ at time $t+1$. The attempt is successful with probability $p_{vu}$. The process runs until the time step where no more nodes get activated. Independent Cascade Model can be thought of as a sender-centric model. An example of the diffusion process in ICM is illustrated in Figure 2.3.

In Figure 2.3, $B$ is selected as the seed node and activated at $t = 0$. It then attempts to activate its neighbors. $A$, $C$, and $E$ have activation probabilities of $p_{BA}$, $p_{BC}$, and $p_{BE}$ respectively. At $t = 1$, only $A$ is activated by $B$; and $B$ cannot activate any of its neighbors anymore. $A$ then proceeds to activate $D$ in a similar fashion. After $D$ attempts to activate its neighbors and activate $G$, the diffusion terminates since there does not remain any active node which can attempt to activate neighbors.

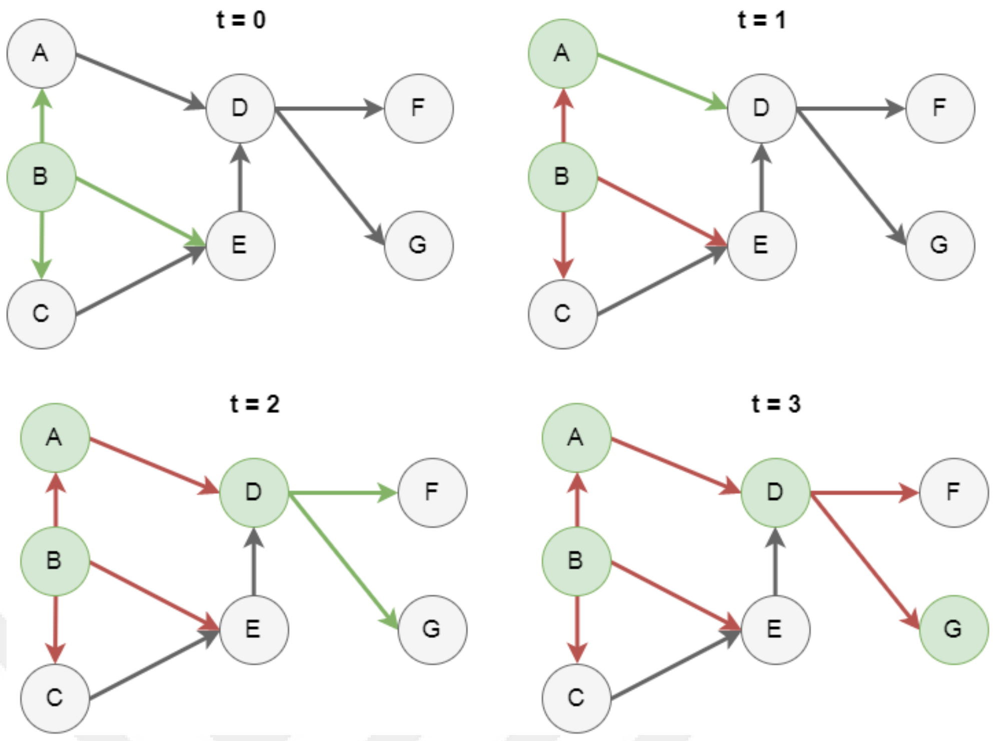

FIGURE 2.3: Independent Cascade Model

### 2.3.2 Linear Threshold Model

Linear Threshold Model (LTM) [1] holds the same assumptions as the ICM except that single chance of activation attempt rule is replaced with another activation model. Each node, in a way, contributes to activation of their neighbours rather than attempting to activate them at once. So, LTM can be thought of as a receiver-centric model in contrast to ICM's sender-centric model. In LTM, each link is assigned a weight $w_{vu}$ representing the influence of node $v$ towards the target node $u$. Each node has an assigned threshold $\theta_u$ to get activated.

The process starts with initially active nodes which serve as the seed nodes. At any time step $t$, for node $u$, if weights of links from neighbouring active nodes exceed the threshold $\theta_u$, then $u$ becomes active. The process runs until the time step where no more nodes get activated. The process in LTM is demonstrated in Figure 2.4

In Figure 2.4, all nodes are assumed to have fixed thresholds of 0.5 for demonstration purposes. At $t = 0$, $B$ is selected as the seed node and activated. At $t = 1$, $A$ and $C$ get activated since sum of incoming influence weights are greater than their thresholds. In the next time step, $D$ and $E$ are activated in a similar fashion. Since $F$ and $G$ cannot be activated, the diffusion terminates after $t = 3$.

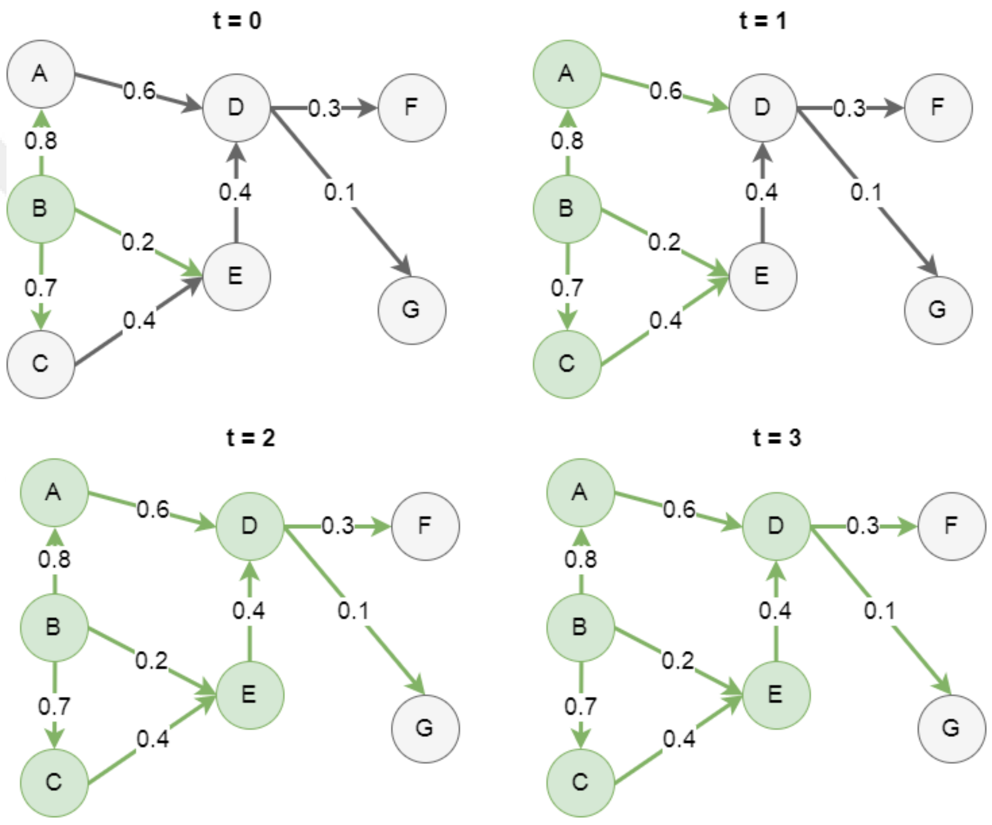

FIGURE 2.4: Linear Threshold Model

# Chapter 3

# Literature Review on Solution Algorithms

## 3.1 Influence Maximization Problem

Domingos and Richardson [14] popularized the concept of *network value* of customers. By approaching the market as a set of connected entities rather than independent entities, they shifted the approach to considering the extra value which might emerge as a result of influences between entities instead of considering only the intrinsic value of each entity. Their study introduced the fundamental problem of *Influence Maximization*, that is how to choose seed nodes so that particular influence spread functions in social networks are maximized.

Kempe, Kleinberg and Tardos [1] formulated the problem that is posed by [14] as a stochastic discrete optimization problem under Independent Cascade Model (ICM), Linear Threshold Model (LTM) or their special cases. The Independent Cascade Model takes its roots from the studies on interacting particle systems in probability theory. Linear Threshold Model is based on Granovetter's study [15] on threshold models of collective behaviour. Detailed information about ICM and LTM are given in Section 2.3.1 and 2.3.2

They formulated the problem as follows: given a social network, a set of influence weights for links, a random threshold function, an integer budget, and a diffusion model; which nodes should be selected as seeds so that the final count of activated nodes is maximized.

Extensions to the problem include budgeted version of the problem where costs of nodes are heterogeneous (see Section 3.3.1), and targeted version of the problem where nodes have different profit values associated with them (see Section 3.3.2).

## 3.2 Solution Methods

### 3.2.1 Original Greedy Algorithm

For the stochastic discrete optimization problem they formulated, [1] proposes a greedy approximation algorithm with an approximation guarantee of $(1-1/e)$. By randomizing the spread of influence instead of using a deterministic version, they obtain a submodular objective function. Using properties of submodular functions, they are able to secure a provable approximation guarantee under both ICM and LTM for their algorithm.

Algorithm 1 summarizes the greedy algorithm of [1]. Since a stochastic function is used for the diffusion of influence, Monte Carlo simulations are employed to estimate the influence spread.

**Algorithm 1** General Greedy Algorithm

```
Input: Graph $N(V,E)$, size of the desired seed set (i.e., budget) $k$, submodular and
    monotone influence spread function $f$
1: $S \leftarrow \emptyset$
2: for $i = 1$ to $k$ do
3:     $u \leftarrow argmax_{w \in V \setminus S\{f(S \cup \{w\} - f(S)\}}$
4:     $S \leftarrow S \cup \{u\}$
5: end for
Output: $S$
```

The General Greedy Algorithm requires $k$ iterations. At each iteration, the algorithm estimates the influence spread of $S \cup v$ for every $v \notin S$. Obtaining an accurate estimate of the influence spread requires a large number of Monte Carlo simulations, typically 10,000 times. Therefore, the original greedy algorithm takes a very long time to complete. For instance, it takes multiple days to select 50 seeds in a network of 30K nodes [16].

To sum up, the original greedy algorithm is inefficient for two reasons: $(i)$ Monte Carlo simulations are called $kn$ times given a budget $k$ and number of nodes in the network $n$, and $(ii)$ large number of Monte-Carlo simulations are required. In order to improve the long running time of the original greedy algorithm, [1] is followed by number of studies which suggested modifications on their original greedy algorithm. [17–19] focus on improving upon the first limitation by employing lazy evaluation techniques. [16, 20–22] try to overcome the second limitation by developing heuristic algorithms instead of a Monte Carlo Greedy (MC-Greedy) algorithm.

### 3.2.2 Lazy Greedy Algorithms

Leskovec et al. [17] propose the Cost-Effective Lazy Forward (CELF) optimization for the given influence maximization problem. Utilizing the submodularity property of the model used in [1], the number of evaluations for influence spread estimations (i.e., calls for Monte Carlo simulations) is reduced. The idea behind their approach is that a node's marginal gain in the current iteration cannot be better than its marginal gain in the previous iterations due to submodularity. Therefore, if any node $v$'s marginal gain in previous iterations is smaller than marginal gain of the current best node in the current iteration, it is unnecessary to evaluate the marginal gain of $v$ in the current iteration. It is empirically shown that CELF is up to 700 times faster than the naive greedy algorithm.

Goyal et al. [18] propose CELF++ algorithm to further reduce the running time of CELF algorithm. In CELF++, when a node $v$'s marginal gain is computed with regard to current seed set $S$, node $v$'s marginal gain with regard to $S \cup u$ is also computed, given that $u$ is the best candidate so far in the current iteration. If $u$ is selected as the best candidate in the current iteration, it will not be necessary to compute $v$'s marginal gain in the next iteration. The authors report 35% to 55% running time improvement compared to CELF algorithm.

Zhou et al. [19] propose Upper-Bound Lazy Forward (UBLF) algorithm to further improve the running time of CELF. They derive an upper bound for influence spread and use that to reduce the number of calls to Monte-Carlo simulations. The algorithm is designed for ICM. The authors explain that when the propagation probability is relatively small and number of nodes in the network is large enough, the upper bound asymptotically approximates the real value of influence spread. In comparison to CELF, UBLF is able to reduce the number of Monte-Carlo calls up to 95% and is faster 2-5 times.

In this group of studies, running times are reduced while maintaining the influence spread. Although the running times are significantly improved, they are yet to be scalable for selecting a relatively larger seed set in large networks.

### 3.2.3 Other Heuristics

An array of heuristic methods have been proposed to further improve the running time; avoiding the Monte Carlo simulations altogether while trying to nearly match the influence spread.

Chen et al. [20] improve both the original greedy and CELF algorithms by employing an efficient randomized algorithm to estimate the size of reachable sets for all nodes in batches under the Independent Cascade Model and Weighted Cascade Model. By doing so, they avoid the independent Monte-Carlo simulations which are required at each iteration. Their improved greedy algorithm (New Greedy) achieves 15% to 34% running time improvement compared to CELF algorithm while matching the influence spread.

Chen et al. [20] developed the Degree Discount Heuristics as well. The heuristic is derived from the ICM with uniform influence probabilities. In simple terms, the proposed heuristic works by reducing the degree count of nodes based on their neighbors which are already activated. The heuristic is more than six magnitudes faster than the greedy algorithms since it does not employ a greedy approach nor require Monte Carlo simulations. The influence spread obtained by using Degree Discount Heuristics is reported to be close to the influence spread obtained by using greedy algorithms, although not matching. The Degree Discount Heuristic can be considered as a special case of Maximum Influence Arborescence heuristic [21] with uniform probabilities and all arborescences having depth one.

Chen et al. [21] propose a new heuristic algorithm, Maximum Influence Arborescence (MIA) under ICM. In their model, influence computations are restricted in the local influence regions of nodes instead of the whole network. Size of local influence regions is established by a user defined tunable trade-off parameter. Arborescence structures (i.e., a tree graph, in which there is exactly one directed path from the root node to other nodes) are created for each node based on shortest-paths between the nodes. These structures represent the local influence regions where the computations happen. In addition, they propose Prefix excluded MIA (PMIA) algorithm which modifies the arborescence structures to provide alternative paths between nodes in certain cases. It outperforms other heuristics such as PageRank and Degree Discount Heuristic in terms of influence spread; and it completes in seconds where Greedy Algorithm completes in hours.

Chen et al. [16] propose Local Directed Acyclic Graphs Algorithm under LTM, in contrast to the previous heuristics which are designed for ICM. In their algorithm, influence computations happen in the local regions, namely local directed acyclic graphs (LDAG). Note that, every arborescence is a DAG, but not every DAG is an arborescence. Therefore, this approach can be seen as a more general version of the approach in MIA. Local DAG of a particular node is created by adding all other nodes whose influence on that particular node is above a parameterized threshold. The influence computations can be done in LDAGs accurately under LTM. Restricting the calculations in LDAGs, the algorithm improves the running time of greedy algorithm by three order of magnitudes while reportedly its influence spread almost matches that of the Greedy Algorithm.

Goyal et al. [22] propose SIMPATH model under LTM. Instead of using local DAG for each node, it uses all simple paths between the node and nearby nodes. The size of the neighborhood is controlled by a parameter. Integrating other techniques such as Vertex Cover Optimization and Look Forward Optimization, they are able to further improve the running time. SIMPATH improves the running time by $22-67\%$, memory usage by $63-81\%$, and influence spread by $2-9\%$ over LDAG algorithm.

In this group of studies, running times are dramatically reduced while trying to match the influence spread. The improvement in running time is usually enabled by restricting the calculation of influence spread in local regions, assuming that most of the influence diffusion happens in local neighborhoods. Theoretically, there is no approximation guarantee for the algorithms given in this section. Nevertheless, their influence spreads nearly match the influence spreads of the greedy algorithm in many experiments.

## 3.3 Extensions

### 3.3.1 Budgeted Influence Maximization

The studies we have visited so far assume that initial activation costs of potential seed nodes are uniform, therefore result in a cardinality constraint in selection of seed sets. However, in practice, the nodes (e.g. accounts in social networking websites) have varying self-perceived values and different pricing strategies for becoming a seed node. Consequently, the nodes are expected to have heterogeneous initial activation costs. Hence, Budgeted Influence Maximization (BIM) problem focuses on the case where the nodes have heterogeneous initial activation costs and the budget is monetary rather than an integer count. In BIM problem, the cardinality constraint is transformed into a knapsack constraint.

Leskovec et al. [17] enable heterogeneous costs in their CELF algorithm. In selecting seed nodes, both the nodes with highest density (i.e., the ratio of gain[1] to cost, also called as *efficiency*), and the nodes with best marginal gain are evaluated and stored as two separate solutions. At the end, the two solutions are compared in terms of estimated influence spread and the algorithm outputs the solution with better score.

Nguyen et al. [23] develop a greedy solution for BIM, using DAGs for estimating influence spread under ICM. The algorithm selects the seed nodes based on estimated spread to cost ratio (i.e., density or efficiency). Let's say, the density-based selection method results in seed set $S_1$. Assuming that there is no node with a cost greater than the budget, let

[1]Terms *gain* and *marginal gain* are used interchangeably in this study.

the node with the maximum spread among all nodes be $s_{max}$. They proceed to compare influence spreads of $S_1$ and $s_{max}$, and then output whichever has a greater value. By doing so, they report to obtain an approximation guarantee of $(1 - 1/\sqrt{e})$.

Du et al. [24] study a version of the problem which has a budget constraint along with timing and user constraints. They assume a continuous-time ICM for the diffusion process. We are interested in how they handle the budget constraint. The main ideas behind their approach are (*i*) spend the budget efficiently, which means selecting only the seeds with relatively high density, and (*ii*) spend the budget as much as possible. In selecting a new seed node at each iteration, the nodes whose density values are above the current density threshold and whose marginal gains are above a parameterized threshold are selected. The density threshold satisfies the first idea, and a decreasing marginal gain threshold as the process unfolds satisfies the second idea. They also show that there exists a density threshold achieving a balance between the two main ideas which might seem contradicting at first.

Singer [25] designs mechanisms which elicit individuals' costs and provides desirable approximation guarantees under many popular models in the literature including ICM and LTM. Mechanical Turk is employed to learn initial activation costs of users by launching a competition where users are asked to specify the number of their friends on Facebook and how much they would like to be rewarded in exchange for posting a commercial content on their Facebook profile. Interestingly, they found no evidence of correlation between the demanded reward and number of Facebook friends. However, this finding should not be generalized without further evidence since the sample of Facebook users might not accurately represent all Facebook users. In the experimental design, they use the distribution that is obtained from the Mechanical Turk experiment to assign costs to nodes. In selecting seed nodes, the following rule is employed: After sorting the nodes based on the highest density, nodes are included in the solution if the ratio between their cost and the budget is smaller than half of the ratio between their marginal contribution and the value of the subset already selected (i.e., proportional share rule).

In this section, we are particularly interested in the methods other studies employ to handle the budget constraint rather than focusing on other features of their algorithms. In a nutshell, there are four general methods employed as a seed selection method: (*i*) highest density, (*ii*) highest marginal gain, (*iii*) highest marginal gain above a certain density threshold, (*iv*) highest density above certain marginal gain threshold. In a sense, the last two methods are a hybrid of the first two methods.

### 3.3.2 Targeted Influence Maximization

Most of the studies we looked into estimate the spread of influence in terms of number of activated nodes. However, in practice, each node carries different value for the marketer. The promoted product might be relevant for only nodes of certain types or nodes in certain locations. The expected profit might be different for each node since the nodes' perceived values of the product might differ or because of the varying purchasing power of the nodes. Thus, it is an obvious direction to assign heterogeneous profit values to nodes and modify the objective function to account for these values.

Li et al. [26] propose Labeled New Greedy Algorithm based on New Greedy Algorithm of [20], Labeled Degree Discount Heuristic based on Degree Discount Heuristic of [20], and their own Maximum Coverage Greedy Algorithm based on offline computations of proximities between nodes and online finding of the seed nodes for given target labels. All proposed algorithms are under ICM. The algorithms, as inputs, require a label for each node, and designated profit values for labels. They modify the objective function to sum up profit values of activated nodes rather than simply counting them. In experiments, they use data derived from Internet Movie Database (abbreviated as IMDb). Actors and actress of movies during 1994-1995 are collected and are considered as nodes in the network. The labels associated with each node are derived from the categories of their involved movies. Each label is then assigned a profit value arbitrarily.

Lu and Lakshmanan [27] propose the Linear Threshold model with user Valuations (LT-V) by introducing nodes' valuations of the marketed product, and a new state called *adopted* in addition to *influenced*. An influenced node adopts only in the case where her valuation is lower than or equal to the price of the product. For the profit maximization problem they formulated, they aim to find the optimal pair of seed set and price vector. Their Price Aware GrEedy (PAGE) algorithm assigns prices dynamically based on the potential profits of candidate seeds.[2]

Li et al. [28] study the Real-time Targeted Influence Maximization problem under ICM for online advertising. In their work, topics are extracted from the available data on nodes (i.e., tweets) and a user profile is represented by a term vector in the topic space. The advertisement is also represented by a term vector. Then, impact of an advertisement to an end user is calculated as the similarity between two term vectors. To find the seed set, they propose an online sampling Reverse Influence Set method named Weighted Reverse Influence Set (WRIS) which returns a solution with $(1 - 1/e - \epsilon)$ approximation ratio.

[2]It should be noted that our algorithm does not extend to the case of offering different prices to each node.

Lee et al. [29] study Targeted Influence Maximization problem under ICM. They assign binary values to nodes as *targeted* or *not targeted.* Categorical data available in datasets is used to determine which nodes are to be selected as target nodes, when such data exists. In other cases, to assign the binary values, they first determine the target nodes to all nodes ratio $p_1$. Next, they randomly select a node and perform a breadth-first search starting from that node. For every visited node, they establish them as a target node with probability $p_2$. With probability $p_3$, another node is selected uniformly at random as a target node. Their aim is to maximize the influence on the targeted nodes instead of on whole network. As a solution algorithm, they develop Independent Maximum Influence Paths-based Expectation (IMIP) Model which is based on PMIA of [21].

Song et al. [30] study Location Targeted Influence Maximization problem. Utilizing a network where users share their locations, each node is assigned with her most frequent check-in location as its location. Then, for each node, a distance value is calculated based on the node's distance to the target location. Based on the distances, suitable target fitness values are assigned to the nodes. They employ a version of ICM and develop a Weighted Reverse Influence Set based method like [28] do.

In summary, all of the algorithms modify the objective function to include so-called profit or target fitness values. In assigning such values, they either benefit from the data already available in the datasets or generate values arbitrarily.

## 3.4 Deterministic Linear Threshold Model

In ICM and LTM, the influence spread is stochastic. In ICM, the randomness is naturally obtained since activations of nodes depend on influence probabilities on edges. In LTM, the randomness is obtained by assigning nodes random thresholds which are uniformly distributed between 0 and 1. [1] showed that an equivalency between ICM and LTM can be established. The resulting influence functions are submodular in both models. Therefore, theoretical approximation guarantees can be obtained for both by employing the General Greedy Algorithm. In both models, the exact computation of influence spread is shown to be P-hard [21].

In this work, we employ Deterministic Linear Threshold Model (Deterministic LTM). The main idea behind Deterministic LTM is that, in practice, threshold values of nodes can be learned via surveys or data mining techniques. Thus, threshold values can be viewed as an input to the model instead of assuming a random threshold function. For LTM, this approach results in deterministic version of the problem where the influence spread is based on deterministic rules.

However, when the threshold values are fixed, the number of activated nodes is not a submodular function of the target set [1]. In fact, Lu et al. [31, 32] showed that there is no $n^{1-\epsilon}$ factor polynomial time approximation for the problem unless $P = NP$. On the other hand, the exact computation of the influence spread under Deterministic LTM can be solved in linear time [32].

Acemoglu et al. [33] study the dynamics of Deterministic LTM and aim to answer whether different types of networks in terms of structures make diffusion more easy in them. They show that eminently clustered networks are not necessarily more advantageous over less structured networks because it is more difficult to penetrate into tight communities unless there exists a seed node inside them.

Another type of problem under Deterministic LTM covers the problems relating to Positive Influence Dominating Sets. A Positive Influence Dominating Set (PIDS) is a subset $S$ of all nodes $V$, where each node in $V - S$ has at least a fraction $p$ of its neighbours in $S$. In Minimum PIDS (MPIDS) problem, the aim is to find the minimum sized set $S$, satisfying the condition that $S$ is a PIDS. The diffusion model in this problem can be viewed as an LTM with uniform weights and uniform thresholds. [34–37] study the MPIDS problem, its complexities and employ greedy algorithms to find a such seed set. Although the underlying diffusion model is a version of LTM, the objective of MPIDS problem is different than the objective of the problem we study. MPIDS is a *set cover* problem where the aim is to find the minimum-sized set influencing all nodes in the network, while our problem is a *maximum coverage* problem where the aim is to influence as many nodes as possible given an upper limit on size of the seed set.

Given a subset of nodes called as a snapshot, Askalidis et al. [38] seek to understand whether there exists a seed set of size at most $k$ which can result in the activation of the given snapshot under variations of Deterministic Linear Threshold Model. The problem they study is a variation of *set cover* problem.

Xu [39] proposes a sparse optimization technique with a linear algebraic approach for the Influence Maximization problem under Deterministic LTM. The author shows that the proposed $L_p$-norm $(0 < p < 1)$ non-convex relaxation method achieves better results than the $L1$-norm convex relaxation approach. It should be noted that this study differs from the rest of the literature in terms of its solution method, and the empirical analysis is performed with experiments in very small networks.

Swaminathan [40] proposes Threshold Difference Greedy (TDG) algorithm for the Influence Maximization problem under Deterministic LTM. The main idea behind his algorithm is as follows. When comparing nodes in the seed selection steps, a node should also be rewarded for partially influencing other nodes. Such a reward is obtained by calculating marginal gain based not only on activated nodes but also the nodes that are partially influenced. Thresholds of the partially influenced nodes are decreased by the influence exerted on them. Consequently, it will be easier to activate them in the next iterations. In our study, we name this concept as potential gain. The author reports that empirical results show that TDG performs better than other methods such as eigenvector centrality, betweenness, Degree Discount, PMIA, and LDAG under Deterministic LTM.

The deterministic activation of nodes was first thought to be a highly simplistic view in the Influence Maximization Problem [1]. However, in line with the ever-increasing speed of the technological advancements in terms of data mining techniques and data collection abilities; it becomes more feasible to predict people's behaviour in product adoption which in turn results in the estimation of node threshold values and link influence weights in LTM. Once these values are learned, better fitting algorithms can be enabled for the deterministic version of LTM; replacing most of the algorithms in the literature which assume random thresholds.

## 3.5 Our Contribution

The problems which are studied in the literature differ between themselves with respect to certain classifications such as whether it is an Influence Maximization problem or Set Cover problem, whether influence propagates deterministically or stochastically, the underlying diffusion model, and which extensions it does cover.

Table 3.1 shows the comparison of the problem definition in our work with that of existing studies in the literature in terms of diffusion model, diffusion type, extensions, and problem type. A mark symbolizes relatively strong focus on the given aspect. Our work is given in the last row and it is shown to be unique in the literature according to the given classifications.

On top of the differences in problem definition, the studies also differ in terms of their solution methods. For instance; while the early studies employ a greedy algorithm to ensure the theoretical approximation guarantee, a number of following studies aim at reducing the run time by sacrificing the theoretical approximation guarantee.

Our contribution in problem definition is to expand the Influence Maximization problem under Deterministic LTM to cover (*i*) Budgeted IM problem and (*ii*) Targeted IM problem. Enabling these extensions brings the problem closer to the real life problems. Our formal problem definition is given in Section 4.1.

Table 3.1: Problem Comparison with the Related Work

| | **Diffusion Model** | | | **Diffusion Type** | | **Extensions** | | **Problem Type** | |
|---|---|---|---|---|---|---|---|---|---|
| **Ref.** | **LTM** | **ICM** | **Other** | **Deter.** | **Stoch.** | **Targ.** | **Budg.** | **Inf.Max.** | **Other** |
| [14] | - | - | X | - | X | - | - | X | - |
| [1] | X | X | - | - | X | - | - | X | - |
| [16] | X | - | - | - | X | - | - | X | - |
| [17] | X | X | - | - | X | - | X | X | - |
| [18] | X | X | - | - | X | - | - | X | - |
| [19] | - | X | - | - | X | - | - | X | - |
| [20] | - | X | - | - | X | - | - | X | - |
| [21] | - | X | - | - | X | - | - | X | - |
| [22] | X | - | - | - | X | - | - | X | - |
| [23] | - | X | - | - | X | - | X | X | - |
| [24] | - | X | - | - | X | - | X | X | - |
| [25] | X | X | - | - | X | - | X | X | - |
| [26] | - | X | - | - | X | X | - | X | - |
| [27] | X | - | - | - | X | X | - | X | - |
| [28] | - | X | - | - | X | X | - | X | - |
| [29] | - | X | - | - | X | X | - | X | - |
| [30] | - | X | - | - | X | X | - | X | - |
| [33] | X | - | - | X | - | - | - | - | X |
| [35] | X | - | - | X | - | - | X | - | X |
| [36] | X | - | - | X | - | - | - | - | X |
| [37] | X | - | - | X | - | - | - | - | X |
| [38] | X | - | - | X | - | - | - | - | X |
| [39] | X | - | - | X | - | - | - | X | - |
| [40] | X | - | - | X | - | - | - | X | - |
| **Us** | **X** | **-** | **-** | **X** | **-** | **X** | **X** | **X** | **-** |

Our main contributions in solution method are (*i*) improving the use of potential gain, (*ii*) introducing a new node selection method, and (*iii*) scaling the solution to very large graphs while nearly matching the same influence spread. Details of our algorithm are given in Section 4.2.

In addition, when compared with the methods used in the literature, we have used novel data generation methods to assign threshold, profit (i.e., target fitness), and cost values to nodes; and influence weights to links. The main purpose of the new methods we employ is to better mimic the real world dynamics. The detail on our data generation methods are given in Section 4.4.

# Chapter 4

# Methodology

## 4.1 Formal Problem Statement

In this section, we provide a formal statement of the problem we propose to solve, namely the Targeted and Budgeted Influence Maximization problem under Deterministic Linear Threshold Model.

Let $N = (V, E)$ be a directed network where $V$ is the set of nodes with $|V| = n$ nodes, and $E$ is the set of links with $|E| = m$ links. Each node $v \in V$ is associated with a threshold value $\theta_v$, an activation cost for being a seed node $c_v$, and a profit value $p_v$. Each directed link has an influence weight $i_{uv}$ representing the degree of influence node $u$ has on node $v$. The budget is denoted by $B$.

At any time step, a node can only be in one of the two states, inactive or active, represented by $\sigma_v \in 0, 1$. $f(v)$ describes transition of the state of node $v$ and it is solely based on Deterministic Linear Threshold Model described earlier in Section 3.4. The model is assumed to be progressive so that once a node becomes active, it can never go back to the inactive state.

The problem aims to find a set of seed nodes $S$ under the constraint $\sum_{v \in S} c_v \leq B$, such that activating the nodes in $S$ is expected to maximize the total profit $P = \sum_{v \in V \setminus S} p_v \sigma_v$ over the social network.

## 4.2 Proposed Algorithm

This section covers our TArgeted and BUdgeted Potential Greedy Algorithm (TABU-PG) for the Targeted and Budgeted Influence Maximization problem under Deterministic

Linear Threshold Model.

TABU-PG works in an iterative and greedy fashion. At each iteration, nodes which satisfy the given constraints are compared based on the objective measure of the given selection method. The node with the maximum value for the given selection method is chosen and added to the seed set. In next iterations, the algorithm keeps adding new nodes to the seed set in the same way. Iterations continue until the remaining budget cannot afford selecting any new node. Ultimately, the seed set and its influence spread is found as the output of the algorithm.

The aforementioned constraints and objective measures are parameterized, therefore different versions of the algorithm can be tested and comparative analysis can be carried out between the different versions.

### 4.2.1 Gain Calculation

In order to compare nodes against each other, gain values for all nodes are calculated. *gain* consists of two components: *actual gain* and *potential gain*. Actual gain immediately realizes itself and increases the total profit whereas potential gain represents the potential future profits which the algorithm invests on.

At first, gain values are calculated for all nodes. At later steps, gain values are calculated only for the nodes whose gain values will be affected by the updates in the network. This approach avoids redundant calculations and significantly improves the runtime. The procedure in determining the nodes whose gain values are to be recalculated is further explained in Section 4.2.5

**Actual gain** of a node is calculated by measuring the increase in the total actual profit which emerges due to the node's capability of fully influencing (i.e., activating) neighboring nodes, and also other nodes by passing influence via the nodes it activated. For instance, assume that node $u$ exerts influence towards node $v$ whose current threshold is lower than the incoming influence from $u$. Therefore, $u$ is able to activate $v$. If $v$ is the only neighbor $u$ can activate and $v$ cannot activate any other node, then actual gain of $u$ is equal to $p_v$. If $v$ also is able to activate a single node $w$, then actual gain of $u$ is equal to $p_v + p_w$.

**Potential gain** is related to the concept of partial influence, which is an important theme of our algorithm. Partial influence can be described as influence exerted by a node towards another node, which decreases the threshold of the latter but cannot eliminate it in full. In this case, the latter node is not immediately activated but is more likely to be activated in later iterations since it now has a lower threshold.

Minimum Potential Gain Ratio parameter $MinPGR$ and Potential Gain Calculation Method parameter $PGCMthd$ together specify how potential gains are accounted for while calculating the potential gain values for nodes.

Only the potential gains satisfying constraint of $i_{uv}$ to $\theta_v$ ratio being greater than or equal to $MinPGR$ are accounted for. If the ratio is lower than $MinPGR$, the potential gain is ignored. However, note that, $\theta_v$ is decreased by $i_{uv}$ in both cases due to the nature of LTM.

When the ratio of $i_{uv}$ to $\theta_v$ is greater than or equal $MinPGR$, the potential gain is calculated by multiplying $p_v$, $i_{uv}/\theta_v$, and $PGMltp$. $PGMltp$ is calculated based on the selection of parameter $PGCMthd$.

Parameter $PGCMthd$ specifies how $PGMltp$ is calculated. In this work, we employ four methods. In Method 1, $PGMltp$ is set to 0 thus effectively removing the potential gain from the algorithm. In Method 2, $PGMltp$ is set to 1 thus effectively eliminating the impact of $PGMltp$. In Method 3, $PGMltp$ is dynamically assigned the value of $1-E/B$ each time given that $E$ is the amount expensed so far. For instance, when half of the budget is exhausted, $PGMltp$ is set to 0.5. In Method 4, $PGMltp$ is dynamically set to $1-(E/B)^2$ similar to the previous method. In this case, for instance, $PGMltp$ is set to 0.75 when half of the budget is exhausted.

The intuition behind the third and fourth method is as follows. Since the potential gain represents the investment to the future profits, the value of $PGMltp$ should decrease when the chances of reaping these future profits decrease. Due to the fact that the number of future steps is limited by the remaining budget, the chances decrease and eventually converges to zero as budget is exhausted.

Equation 4.1 and 4.2 shows how *actual gain* and *potential gain* components are calculated at any given node that is exposed to an influence originating from the node which we calculate gain for. The resulting values from the formula add up and produce the actual gain and potential gain values. *gain*, which is the main component of the measures on which nodes are compared, is the sum of *actual gain* and *potential gain*.

$$Actual\ Gain = \begin{cases} p_v, & \text{if } i_{uv}/\theta_v \geq 1 \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

$$Potential\ Gain = \begin{cases} p_v(i_{uv}/\theta_v)(PGMltp), & \text{if } MinPGR \leq i_{uv}/\theta_v < 1 \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

### 4.2.2 Node Selection

The Node Selection Method parameter $NSMthd$ specifies the measure which nodes are compared on, and also specifies any additional constraints. In all methods, the value which algorithm tries to maximize at each iteration is closely related to the gain values calculated for each node.

We currently employ three node selection methods. Method 1 compares nodes solely based on gain and selects the node with maximum gain. Method 2 compares nodes solely on efficiency, that is the ratio of gain to cost $g_v/c_v$. This ratio of efficiency is also called as density in the literature.

Method 3 is a type of combination of the first two methods. In this method, top three candidate nodes are selected based on efficiency (i.e., applying Method 2 but choosing three nodes instead of one). Then, among the three, the node with maximum gain (i.e., applying Method 1 but comparing only three nodes instead of many) is selected as the seed node. It is possible to combine the first two methods in other ways such as calculating scores for both methods and averaging them, or choosing top three nodes based on gain and selecting the seed among them based on efficiency. However, our preliminary experiments showed that they do not perform better than Method 2. Therefore, we do not employ such additional methods in this work.

### 4.2.3 Algorithm Illustration

To illustrate how TABU-PG works, consider the following network. $A$ influences $B$ and $D$; $B$ influences $A$, $C$, and $E$; $C$ influences $E$; $D$ does not influence any node; and $E$ influences $B$ and $D$. Figure 4.1 illustrates the network and execution of TABU-PG. Node threshold, profit, and cost values are given inside the nodes. Influence weights are given on links. Nodes selected as seed are shown with a bold font type. Active nodes are depicted with green color. The campaign budget is set to 6.

In the first iteration, gain values will be calculated for all nodes. Potential gain and actual gain calculations are shown below. Note that, $PGMltp$ is not shown in the initial potential gain calculations. Instead, it is shown in the calculation of gain as $g = g^a + g^p(PGMltp)$. Since none of the nodes can activate another node at this iteration, actual gain values are equal to zero for all nodes.

${g^p}_A = (0.2/0.3)1 + (0.3/0.8)2 = 1.42, \quad {g^a}_A = 0$

${g^p}_B = (0.3/0.6)3 + (0.2/0.8)3 + (0.5/0.6)3 = 4.75, \quad {g^a}_B = 0$

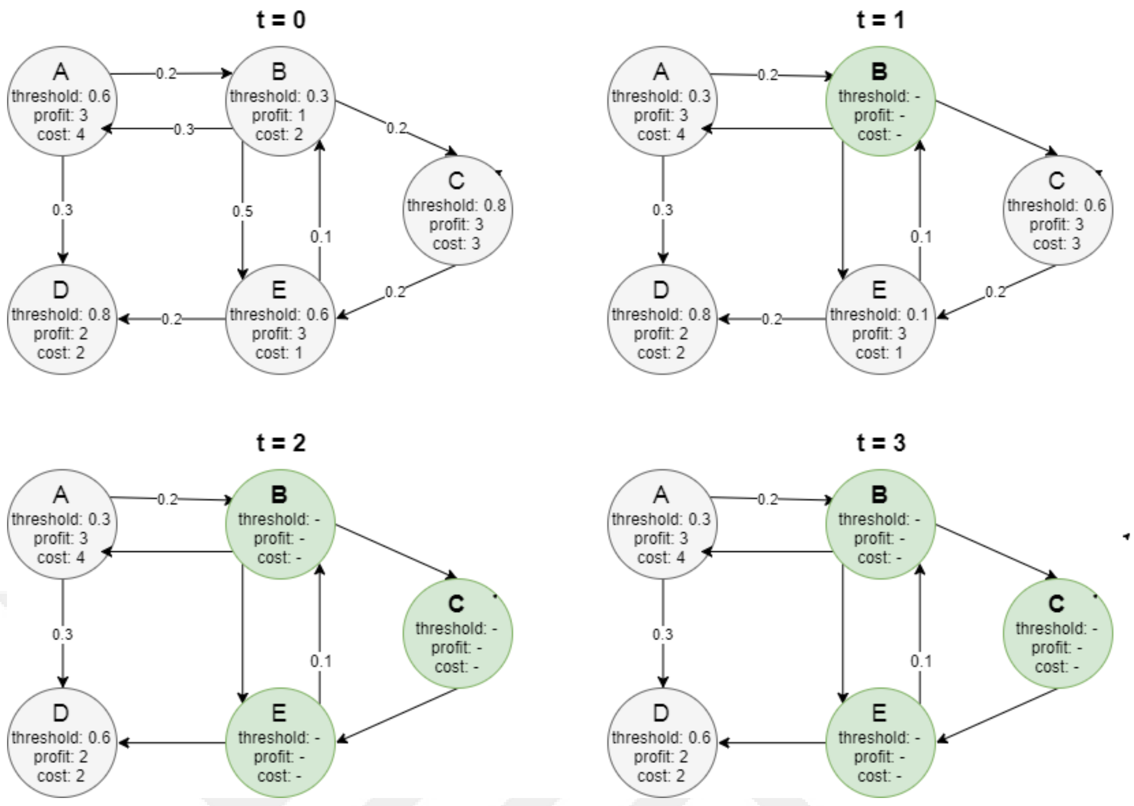
t = 0
A threshold: 0.6 profit: 3 cost: 4
B threshold: 0.3 profit: 1 cost: 2
C threshold: 0.8 profit: 3 cost: 3
D threshold: 0.8 profit: 2 cost: 2
E threshold: 0.6 profit: 3 cost: 1
0.2
0.3
0.2
0.3
0.5
0.1
0.2
0.2
t = 1
A threshold: 0.3 profit: 3 cost: 4
B threshold: - profit: - cost: -
C threshold: 0.6 profit: 3 cost: 3
D threshold: 0.8 profit: 2 cost: 2
E threshold: 0.1 profit: 3 cost: 1
0.2
0.3
0.1
0.2
0.2
t = 2
A threshold: 0.3 profit: 3 cost: 4
B threshold: - profit: - cost: -
C threshold: - profit: - cost: -
D threshold: 0.6 profit: 2 cost: 2
E threshold: - profit: - cost: -
0.2
0.3
0.1
t = 3
A threshold: 0.3 profit: 3 cost: 4
B threshold: - profit: - cost: -
C threshold: - profit: - cost: -
D threshold: 0.6 profit: 2 cost: 2
E threshold: - profit: - cost: -
0.2
0.3
0.1

FIGURE 4.1: TABU-PG Illustration

$g^p{}_C = (0.2/0.6)3 = 1, \quad g^a{}_C = 0$

$g^p{}_D = 0, \quad g^a{}_C = 0$

$g^p{}_E = (0.1/0.3)1 + (0.2/0.8)2 = 0.83, \quad g^a{}_E = 0$

When selecting the seed node, we have three options as explained in Section 4.2.2. For this illustration, we will employ $NSMthd = 2$, the second method that is choosing the most efficient node. Accordingly, the efficiency values are found as $A : 0.36$, $B : 2.38$, $C : 0.33$, $D : 0$, and $E : 0.83$, by dividing total gain ($g = g^a + g^p(PGMltp)$) by cost of the node. Note that $PGMltp$ is equal to 1 at the first iteration except for the case when $PGCMthd = 1$, that is to ignore potential gains altogether. For this illustration, we will employ $PGCMthd = 3$, that is decreasing the share of potential gain as budget is exhausted. $MinPGR$ is assumed to be 0 for illustration purposes. See 4.2.1 for details on gain calculation.

$B$ is selected as the first seed node since it has the maximum value for the given objective measure. The network is updated accordingly: relevant thresholds are lowered and relevant nodes are activated if there is any. The total amount spent is 2, and the remaining budget is 4.

After the very first gain calculation, potential and actual gain values will only be calculated for those nodes which are not already activated and whose gain values are affected by the updates in the network. Therefore, in the second iteration, potential gain and actual gain values will be calculated only for $A$ and $E$ since they influence $B$; and for $C$ since it influences $E$ whose threshold is lowered by $B$.

$C$ can activate $E$, therefore the gains obtained on other nodes via $E$ is also accounted for.

$g^p{}_A = (0.3/0.8)2 = 0.75, \quad g^a{}_A = 0$

$g^p{}_C = (0.2/0.8)2 = 0.5, \quad g^a{}_C = 3$

$g^p{}_E = (0.2/0.8)2 = 0.5, \quad g^a{}_E = 0$

Then, gain values are calculated by multiplying potential gains by $(6-2)/6 = 0.67$ since we employ $PGCMthd = 3$. For the nodes whose gain values are not recalculated, the gain values from the previous iterations are used since their potential gain and actual gain values are not changed. The gain values for the candidate nodes are calculated as follows.

$g_A = (0.75)(0.67) + 0 = 0.5$

$g_C = (0.5)(0.67) + 3 = 3.34$

$g_D = (0)(0.67) + 0 = 0$

$g_E = (0.5)(0.67) + 0 = 0.34$

Then the efficiency values are found as $A : 0.13$, $C : 1.11$, $D : 0$, and $E : 0.34$. $C$ is selected as the seed node at this iteration since it has the maximum value for the given objective measure. The network is updated accordingly: relevant thresholds are lowered and relevant nodes are activated if there is any. The total amount spent is increased to 5, and the remaining budget is 1. Since there is no node in the network with cost not greater than the remaining budget; a new seed node cannot be selected in the next iteration and the algorithm halts.

### 4.2.4 Scaling for Larger Networks

In order to make the algorithm scalable to very large networks, parameter $NumBefReCalc$ specifying the number of nodes to select before recalculating gain values, and parameter *TopMltp* specifying the multiplier for determining the number of top nodes are defined. $NumBefReCalc$ specifies how many nodes are to be selected as seed nodes based on the current gain values for nodes; before recalculating the gains for the updated network. When $NumBefReCalc$ is set to 1 which is the default, the gains are calculated before each time a new seed node is selected. When it is set to $Inf$, the gains are calculated only once at the beginning, and all seed nodes are selected based on these gain values. When, for example, it is set to 5; after a gain calculation, up to five more nodes can be selected before recalculating the gain values.

$TopMltp$ takes part in determining the number of top nodes for which gains will be calculated for after the very first calculation which is done for all nodes. The intuition is that if a node's gain or efficiency is very small at first; it is very unlikely that the gain or efficiency values which will be calculated for that node will be large enough for that node to be selected as a seed at later steps. Therefore, the idea is to calculate gain values or efficiency values only for the nodes who are more likely to be selected as seed nodes.

Before recalculating the gain values, number of top nodes which gain will be calculated for is calculated based on $TopMltp$, $NumBefReCalc$, and current size of $S$ which is the number of nodes which have been selected as seeds so far. The formula for determining the number is given in Equation 4.3. $NumBefReCalc$ parameter is included in the formula to ensure that there will be enough number of nodes to consider when $NumBefReCalc$ is larger and thus gain is calculated less frequently. Utilizing this method, we limit the candidate pool for seed nodes to a number of top nodes instead of all nodes.

$$N.\ of\ Top\ Nodes = \begin{cases} n, & \text{if } NumBefReCalc = Inf \\ (|S| + NumBefReCalc)TopMltp, & \text{otherwise} \end{cases} \tag{4.3}$$

Introducing these two parameters enables a trade-off between runtime and final influence spread, therefore making it possible to run the algorithm on very large graphs in a reasonable amount of time. Note that, in some cases, there could be large improvements in run time without any worse performance in influence spread results at all.

### 4.2.5 The Algorithm

The algorithm we propose, TABU-PG, is described in Algorithm 2. After initializing required variables in $Line$ $1-4$, algorithm starts the iterative process in $Line$ 5 as long as remaining budget allows. Potential Gain Multiplier ($PGMltp$) is determined based on Potential Gain Calculation Method, in $Line$ $6-14$.

**Algorithm 2** TABU-PG for IM under Deterministic LTM

```
Input: Graph $N(V, E)$, set of threshold values for nodes $\Theta$, set of profit values for nodes
    $P$, set of activation costs for nodes $C$, set of influence weights for links $I$, maximum
    budget $B$, node selection method $NSMthd \in \{1, 2, 3\}$, Minimum Potential Gain
    ratio to qualify for potential gain calculation $MinPGR$, potential gain calculation
    method $PGCMthd \in \{1, 2, 3, 4\}$, number of nodes to select for seed set before recal-
    culating the gain $NumBefReCalc$, multiplier for determining the limit on number
    of top nodes whose gain will be recalculated $TopMltp$
 1: $S \leftarrow \emptyset$, $A \leftarrow \emptyset$, $E \leftarrow 0$    // $S$: seed nodes, $A$: active nodes, $E$: amount spent so far
 2: $SortInitGain \leftarrow TRUE$    // to enable sorting based on very first gain calculation
 3: $T \leftarrow V$, $U \leftarrow V$    // $T$: top nodes, $U$: nodes whose gain need to be updated
 4: $G^a$, $G^p$, $G$    // initiate $G^a$: actual gains, $G^p$: potential gains, $G$: total gains
 5: while $B \geq E$ do
 6:   if $PGCMthd = 1$ then
 7:     $PGMltp \leftarrow 0$
 8:   else if $PGCMthd = 2$ then
 9:     $PGMltp \leftarrow 1$
10:   else if $PGCMthd = 3$ then
11:     $PGMltp \leftarrow 1 - E/B$
12:   else if $PGCMthd = 4$ then
13:     $PGMltp \leftarrow 1 - (E/B)(E/B)$
14:   end if
15:   for $y \in (\mathrm{U} \cap T) \backslash A$ do
16:     $U.remove(y)$
17:     $(g^a)_y \leftarrow 0$, $(g^p)_y \leftarrow 0$    // reset gain value for the node
18:     $A^t \leftarrow A$    // $A^t$: temporary copy of $A$
19:     $\Theta^t \leftarrow \Theta$    // $\Theta^t$: temporary copy of $\Theta$
20:     $Q \leftarrow \emptyset$    // initiate a list for holding -temporarily- active nodes
21:     $Q.push(y)$, ${\theta^t}_y \leftarrow 0$, $A^t \leftarrow A^t \cup y$    // temporarily activate the node
22:     while $Q \neq \emptyset$ do
23:       $u \leftarrow Q.pop()$
24:       for $v \in (outNeighbors(u) \backslash A)$    // $u$ influences $outNeighbors(u)$ do
25:         if $i_{uv} \geq {\theta^t}_v$ then
26:           ${g^a}_y \leftarrow {g^a}_y + p_v$, ${\theta^t}_v \leftarrow 0$, $A^t \leftarrow A^t \cup v$
27:           $Q.push(v)$
28:         else
29:           if $i_{uv}/{\theta^t}_v \geq MinPGR$ then
30:             ${g^p}_y \leftarrow {g^p}_y + p_v(i_{uv}/{\theta^t}_v)$
31:           end if
32:           $\theta^t \leftarrow {\theta^t}_v - i_{uv}/{\theta^t}_v$
33:         end if
34:       end for
35:     end while
36:   end for
37:   $G \leftarrow G^a + G^p(PGMltp)$
```

```
38:   for $j = 1$ to $NumBefReCalc$ do
39:     if $NSMthd = 1$ then
40:       $s \leftarrow argmax^1_{v \in T \setminus A, c_v \leq B-E}\{g_v\}$          // $argmax^k$: top $k$ nodes
41:     else if $NSMthd = 2$ then
42:       $s \leftarrow argmax^1_{v \in T \setminus A, c_v \leq B-E}\{g_v/c_v\}$
43:     else if $NSMthd = 3$ then
44:       $s \leftarrow argmax_{v \in argmax^3{}_{v \in T \setminus A, c_v \leq B-E}\{g_v/c_v\}}\{g_v\}$
45:     end if                                  // $s$: node selected as a seed
46:     $Q_2 \leftarrow \emptyset$      // $Q_2$: list to hold affected nodes. $U$ is generated using this list
47:     $Q_3 \leftarrow \emptyset$                        // $Q_3$: a list also used in generation of $U$
48:     $Q \leftarrow \emptyset$, $Q.push(s)$, $\theta_y \leftarrow 0$, $A \leftarrow A \cup s$
49:     while $Q \neq \emptyset$ do
50:       $u \leftarrow Q.pop()$
51:       for $v \in outNeighbors(u) \setminus A$ do
52:         $Q_2.push(v)$
53:         if $i_{uv} \geq \theta_v$ then
54:           $\theta_v \leftarrow 0$, $A^t \leftarrow A^t \cup v$
55:           $Q.push(v)$
56:         else
57:           $\theta_v \leftarrow \theta_v - i_{uv}/\theta_v$
58:         end if
59:       end for
60:     end while
61:   end for
62:   while $Q_2 \neq \emptyset$ do
63:     $u \leftarrow Q.pop()$
64:     for $v \in inNeighbors(u) \setminus (A \cup U)$      // $u$ is influenced by $inNeighbors(u)$ do
65:       $U \leftarrow U \cup v$
66:       $Q_3.push(v)$  // to check if the change in $g_v$ causes a change in $g_{inNeighbors(v)}$
67:     end for
68:   end while
69:   while $Q_3 \neq \emptyset$ do
70:     $u \leftarrow Q_3.pop()$
71:     for $v \in inNeighbors(u) \setminus (A \cup U)$ do
72:       if $i_{vu} \geq \theta_u$ then
73:         $U \leftarrow U \cup v$        // $g_v$ should be updated since the change in $g_u$ affects $g_v$
74:         $Q_3.push(v)$
75:       end if
76:     end for
77:   end while
78:   if $SortInitGain = TRUE$ then
79:     $G^i \leftarrow G$                    // $G^i$: initial gain values after the very first calculation
80:     $SortInitGain \leftarrow FALSE$
81:   end if
82:   if $TopMltp \neq Inf$ then
83:     if $NSMthd = 0$ then
84:       $T \leftarrow argmax^{(size(S)+NumBefReCalc)TopMltp}_{v \in V}\{g^i{}_v\}$
85:     else
86:       $T \leftarrow argmax^{((size(S)+NumBefReCalc)TopMltp}_{v \in V}\{g^i{}_v/c_v\}$
87:     end if
88:   end if
89: end while
Output: $S$,$A$
```

From *Line* 15 to *Line* 36, gain values for nodes who are in both of the to-be-updated list $U$ and top nodes list $T$ are calculated. To calculate the gain for a node, the node which gain is calculated for is removed from the to-be-updated list $U$, its gain is reset, temporary copies of active node list $A$ and thresholds $\Theta$ is created, and a list $Q$ is created to hold temporarily active nodes. Then, the node is temporarily activated to simulate the diffusion process as if the node was selected as seed. By this way, the algorithm calculates the gain which would result from the activation of the node. *Line* $22 - 35$ shows how influence diffuse in the network, and how actual gain and potential gain are calculated. Finally, in *Line* 37, the gain value which nodes will be compared on is calculated. Note that, since $PGMltp$ might be changed in every iteration, it actually requires gain values for all nodes to be changed. By placing this gain summation outside of the loop, we avoid actual gain and potential gain calculations for nodes whose values are changed only due to the change in $PGMltp$.

After gain values are calculated for nodes, new seed nodes are selected and the network is updated accordingly in *Line* $38 - 61$. Before going on with the rest of the algorithm, $NumBefReCalc$ nodes are selected as seed by iterating the process $NumBefReCalc$ times. Depending on the Node Selection Method, the node which satisfies the constraints and has the maximum value is selected as a seed in *Line* $39 - 45$. Then, in *Line* $48 - 60$, the seed node is activated, influence spread is calculated, and the network is updated.

Since updates in thresholds of nodes change the gain values of nodes who exert influence upon them, list $U$ must be generated to hold nodes whose gains to be updated. For this purpose, two lists are initialized in *Line* $46 - 47$. Nodes whose thresholds have been changed are added to $Q_2$ in *Line* 52. After selecting the seed nodes and before recalculating the gains, the nodes whose gains need to be updated are determined and added to $U$ in *Line* $62 - 77$. In the first part, in *Line* $62 - 68$, nodes who influence the nodes whose thresholds are changed are added to $U$. In the second part, in *Line* $69 - 77$, nodes who are able to activate any node in $U$ by themselves are added to $U$. This is because gains will be recalculated for the nodes in $U$, and if a node can fully activate a node in $U$, then its gain value also needs to be updated.

In *Line* $78 - 87$, all nodes are sorted according to their initial gain or efficiency value based the node selection method. Certain number of nodes are selected as top nodes based on the number of nodes have been selected so far, the number of nodes to select before recalculating the gain values, and the multiplier for determining the number of top nodes. It results in a list of top nodes $T$, which presents a constraint on nodes for which gain will be calculated in the next calculation.

## 4.3 Graphs

For the empirical analysis, the following public network datasets are employed: Epinions[41, 42], Academia.edu [43, 44], and Pokec[41, 45]. In addition, we crawled the underlying social network of Inploid, a social network where users can ask questions and answer questions of others in Turkish.

Epinions is a consumer review website where users can register for free and share their reviews for a variety of products. In order to prevent deceiving reviews, a trust system is put into place where users can specify whether other users are trustworthy or not. It results in a social network where nodes are the users and directed links are indicators of trust between the users. The dataset originally consists of $75,879$ nodes and $508,837$ directed links and no other information. This data set has been used in number of studies including [16, 21, 27, 29, 40].

Academia.edu is a social networking website for academics where users can share papers and follow each other. It originally consists of $200,169$ nodes and $1,398,063$ directed links and no other information. The dataset is crawled by Ben Gurion University Social Networks Security Research Group and used in number of studies including [43, 44].

Inploid is a social question & answer website in Turkish with total number of nearly $40,000$ registered users. Users can follow others and see their questions and answers on the main page. Each user is associated with a reputability score which is influenced by feedback of others about questions and answers of the user. Each user can also specify interest in topics. The data is crawled in June 2017 and consist of $39,750$ nodes and $57,276$ directed links between them. In addition, for each user, reputability scores and top five topics are included in the dataset.

Pokec is a Slovakian online social networking website where users can share information about themselves, post pictures on their profiles, and chat with other users. The information about users include age, gender, physical appearance, marital status, hobbies, political view, and many other fields. The dataset covers the whole network and originally consists of $1,632,803$ nodes and $30,622,564$ directed links, and profile information of users for near 60 fields. The dataset is used in various studies including [29, 45].

## 4.4 Data Generation

Our algorithm requires influence weights for links; and threshold, cost, and profit values for nodes. Most of the time, at least one of them is not available in the social networks.

Therefore, we employ suitable data generation methods to fill the missing parts with synthetic yet meaningful data.

The following three methods are used in data generation: (i) a fixed value, (ii) a discrete distribution, or (iii) continuous distribution. For continuous distribution we use either a uniform distribution or a normal distribution. Truncated normal distribution based on [46] is also used when a lower or upper limit need to be established.

### 4.4.1 Influence Weights on Links

Influence weights on links represent the degree influence one node has on another. For some networks, influence values in some forms might be available in the dataset due to the nature of the given social network (i.e., trust degrees). However, in most cases, social networks do not have mechanisms to assign such values to the links. Therefore, it is necessary to assign proper values as influence weights to the links.

In the literature, influence weights (or influence probabilities in ICM) are assigned in the following ways: fixed value, arbitrary, or ratio model. In fixed value method, all links are assumed to have the same fixed weight such as 0.05 or 0.1. When this method is employed, it effectively ignores any differentiation in influence capabilities of links; which is not a proper representation of mechanisms in the real world. This method is used in studies including [26].

In arbitrary selection method, influence weights are sampled randomly from a set of values such as $\{0.01, 0.05, 0.1\}$ or from an interval such as $[0, 1]$. This is a better representation of the real world compared to fixed value method since it acknowledges that different links might have different influence capabilities. This method is used in studies including [23, 27].

In ratio model, influence weight is assigned by dividing the count of edges by the number incoming links to target node. Supposing that $\vec{uv}$ represents a directed link where influence goes from $u$ to $v$, the equation for influence weight is given in Equation 4.4. Various studies including [1, 16, 40] employ this method.

$$i_{uv} \;=\; |\vec{uv}| \;/\; indegree(v) \tag{4.4}$$

When one of the first two methods are employed, a node becomes active based on count of its neighbors which are already activated. For instance, at least five neighbors with influence weight of 0.1 on the links are required to activate a node with a threshold of 0.5. On the other hand, in the third method, a node becomes active based on proportion of its active neighbors instead of count. For example, to activate a node with threshold of 0.5, at least 50% of its neighbors are required to be active given that links have equal weights.

The methods which are based on count makes it relatively difficult to activate nodes with smaller degrees. For instance, a node with threshold of 0.6 has no chance to be activated if it only has three neighbors given that all links have influence weights of less than 0.2. On the contrary, the methods which are based on proportion makes it difficult to activate the nodes with larger degrees (i.e., nodes with many incoming influences). For example, a node with relatively low threshold of 0.3 and 100 neighbors requires at least 30 active neighbors whereas a node with relatively high threshold of 0.7 and 10 neighbors require only 7 active neighbors, given that influence weights are uniform.

Considering a such trade-off, we develop a hybrid method which is a fusion of arbitrary selection method and ratio model, as explained as follows. First, average degree of the network is calculated by dividing number of links to number of nodes. Then, 1 is divided by average degree and a fixed value is found. For each link, this fixed value is then multiplied by a value sampled from $\{a_1, a_2, ...\}$ with respective probabilities of $\{p_1, p_2, ...\}$. In effect, it is similar to arbitrary selection method. However, we introduce sampling probabilities and the practice of dividing 1 by average degree of the network.

Then, for each link, geometric mean of the results of the above method and results of the ratio model is calculated and assigned as the influence weight. The formula is given in Equation 4.5. In the case where the influence weight resulted from ratio model is 0.2, and the influence weight resulted from the above method is 0.45; our hybrid model results in influence weight of 0.3, their geometric mean.

$$i_{uv} = \sqrt{(|\vec{uv}|/indegree(v))((1/(|E|/|V|))(rand(\{a_1, a_2, ..\}, \{p_1, p_2, ..\}))} \tag{4.5}$$

### 4.4.2 Node Thresholds

Node thresholds are not readily available neither in any of the datasets we use nor in any datasets in the studies in the literature, to the best of our knowledge. Therefore, node thresholds need to be synthetically generated.

In most of the literature, thresholds are assigned randomly between 0 and 1 to satisfy the submodularity requirement for the original LTM. In studies about PIDS problems, thresholds are assigned a fixed value, usually 0.5. In [40], fixed values 0.8 and 0.5; or random values between 0.1 and 0.9, or between 0.3 and 0.7 are used. Most of studies use ratio model in assignment of influence weights; and in such models, a node with a threshold above 1 can never be activated.

Since submodularity does not hold in Deterministic LTM, we are not limited to drawing thresholds randomly between 0 and 1. On the other hand, assigning all nodes a same fixed value is an oversimplification. Instead, we develop a new approach which mimics the real world dynamics of diffusion of innovations.

As Rogers put it in his seminal work [8], there are five groups of consumer when it comes to adoption of innovation: innovators (2.5%), early adopters (13.5%), early majority (34%), late majority (34%), and laggards (16%). See Section 2.1.2 for details. Although share of markets are given for five discrete groups, they do not follow a discrete distribution but a normal distribution, as illustrated in Figure 2.1. Therefore, to assign threshold values, we employ a normal distribution with a limit on lowest value (i.e., truncated normal distribution) to mimic the real world dynamics.

### 4.4.3 Node Costs

In the literature, studies on BIM problem embraced the following ways of assigning cost values. [24] employ two methods in assigning cost values to nodes. First one is to assign a fixed uniform cost to all nodes. The second method is assigning costs based on degree of nodes. Similarly, [17] use fixed uniform costs to all nodes (so-called unit cost model) as the first model, or assign costs to blogs (i.e., nodes) based on number of blog posts they contain. [23] employed the method of selecting costs randomly from an interval in addition to employing unit cost model.

Instead of assigning random or fixed values as costs for nodes, we develop a new method similar to that of [24], that considers the indegree (i.e., number of outgoing influences) of a node while estimating the cost of that node[1]. The intuition is as follows. In typical social networks, nodes (e.g. users) are not likely to know their true network value. Instead, we assume, a node's self-perceived value is mostly based on the number of its followers. It is a simple metric on which users can compare themselves with others, and estimate their own value in the market.

However, degree is not a deterministic factor by itself. Thus, we also employ a random variation in addition to the value find based on indegree. Moreover, we specify a fixed value added to costs of all nodes, which represents any possible fixed costs in real life (e.g. cost of time required for communication with any node, legal costs, cost of sample products, and etc.).

[1]A node with large indegree value means that number of its followers is large. Direction of an edge and direction of the influence on that edge is opposite and should not be confused.

In order to normalize the cost values, square root of indegree is used instead of indegree. The found value then is multiplied with a random value between $a$ and $b$, to represent the variation in users' methods of self-perception. Finally, a fixed value of $z$ is added to every node. The formula for cost calculation is given in Equation 4.6

$$c_v = \sqrt{v_{indeg}} rand(a, b) + z \tag{4.6}$$

### 4.4.4 Node Profits

Using heterogeneous profit values on nodes extends our problem to a targeted version of the Influence Maximization problem. In real life, profit values can be used for targeting certain demographics where profit values are generated according to fitness of nodes to target demographics. It can also be used for simply profit maximization by generating profit values based on estimated profit (e.g. socioeconomic status of nodes). For instance, if incomes of nodes (i.e., users) are known, then profit can be estimated based on it. A third option is to combine these two approaches to represent both fitness to target demographics and estimated profit values.

In the literature, profit values are assigned to nodes in the following ways. [27] employs a product-user network and derives profit values by combining the item ratings of users and prices of the items. [30] uses a location based social network and derives profit values (or so called target fitness values) by measuring the distance of users to the target location. [26] employs a movie-actor network and creates profit values based on the genres of the movies in which actors have played. [28] studies online advertising and assigns profit values based on similarity between the two term vectors he created for advertisements and users. [29] used profile data of users to target users in certain categories.

Generating such profit values is not always possible when the dataset lacks necessary information. In those cases, profit values need to be created synthetically.

Profits can be drawn from a discrete distribution $\{a_1, a_2, ...\}$ and $\{p_1, p_2, ...\}$ where $p_1$ is probability of selecting $a_1$ and so on. For instance, if a campaign only targets the males living in urban areas, then $a_1$, $a_2$ and $p_1$, $p_2$ should be 1, 0 and 0.25, 0.75 respectively, given that half of people live in urban areas and male to female ratio in urban areas is 1.

Profit can also be generated based on the assumed continuous profit distribution. Since the distribution is highly dependent on the product and context; for simplicity, we will assume a log-normal distribution as a representation of income inequality.

The third method which combines the discrete distribution method and continuous distribution method can be achieved by multiplying the outputs of the two.

# Chapter 5

# Experimental Results and Discussion

The goal of the experiments is to present the performance of our algorithm on various datasets. In this chapter, we first explain the procedures in data preprocessing step which include preliminary operations on graphs, generation of influence weights for links and generation of threshold, profit and cost values for nodes. For each of Epinions, Academia.edu, Inploid, and Pokec networks, two different datasets are generated via using different data generation methods which are explained in Section 4.4. Experiments are categorized based on underlying networks. Numerical results and initial comments are provided for each experiment in Section 5.2.

An overall discussion with regard to different value settings for parameters and their impacts on influence spread (i.e., total profit)[1] and runtime are given in Section 5.3.

All experiments are run on a computer with Intel[2] Core i5-5200U CPU @ 2.20 Ghz, and 8 GB memory. R is employed for data preprocessing and data generation operations. Centrality scores to be used in benchmark heuristics are calculated in R as well. Java is employed to run our TABU-PG algorithm, and to obtain influence spread results of benchmark heuristics. The upper limit on memory allocated to Java program is set as 6500MB.

[1]Total profit and influence spread are used interchangeably throughout this study.
[2]Intel is a trademark of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

## 5.1 Data Preprocessing

Various data preprocessing operations are undertaken for all graphs which are presented in Section 4.3: Epinions, Academia.edu, Inploid, and Pokec. In this section, we go over the data preprocessing steps and explain how final datasets to be used in experiments are generated from the original network datasets.

In order to choose seed sets based on the benchmark heuristics, *igraph* package of R is utilized. Weights on edges are not taken into account when calculating centrality scores. Directions of edges are considered. Cutoff points[3] are set as 3 in estimation of betweenness and closeness scores for all experiments except for the experiments on Pokec. For Pokec, the cutoff points are set as 2 to prevent it to take a very long time to estimate since the network is very large. In calculation of PageRank score, for all experiments, damping factor[4] is set as 0.85 which is an assumed default value in most PageRank applications.

Following data preprocessing procedures is applied to all graphs. Any self-loops and multiple (i.e., recurring) links are removed from the graph. Largest connected component in each network is found and other components are removed from the network. Therefore, basically, the experiments are done on the largest connected component for each network. This is preferred because disconnected components are, in a sense, like multiple different networks rather than a single network.

Additional data preprocessing steps are taken for Pokec graph. The nodes whose follower information was not public, approximately one third of all nodes, are excluded from the network.

The information on resulting graphs before and after data preprocessing operations are performed are summarized in Table 5.1.

Table 5.1: Graphs

| | **Before Data Preprocessing** | | **After Data Preprocessing** | |
|---|---|---|---|---|
| **Graphs** | **# of nodes** | **# of links** | **# of nodes** | **# of links** |
| **Epinions** | 75,879 | 508,837 | 75,877 | 508,836 |
| **Academia.edu** | 200,169 | 1,398,063 | 200,167 | 1,397,620 |
| **Inploid** | 39,750 | 57,276 | 14,360 | 57,100 |
| **Pokec** | 1,632,803 | 30,622,564 | 1,080,251 | 14,662,846 |

[3]A cutoff point is the maximum path length to consider when calculating the betweenness or closeness score.

[4]Please refer to [10] for details.

Data generation methods which are employed in creating influence weights for links; and threshold, cost, and profit values for nodes are explained below and in their respective sections when necessary.

In all experiments, threshold values are assigned randomly by using a truncated normal distribution. This method aims to reflect the groups of consumers in adoption of innovation, which are shown earlier in 2.1. The parameters of the distribution is given for each experiment in their respective sections.

In odd numbered experiments, influence weights are assigned by the ratio model (see Section 4.4.1) which is the most widely used method in the literature. Briefly, in this model, sum of weights of incoming influences is equal to 1.

In even numbered experiments, influence weights are assigned by the hybrid model which is explained in Section 4.4.1. $x$ being the average degree of nodes, for each node, a value from $\{a_1x,\ a_2x,\ a_3x,\ a_4x\}$ is selected with respective probabilities of $\{p_1,\ p_2,\ p_3,\ p_4\}$. Then, geometric mean of the selected value and the value which would be resulted from ratio model is found. The geometric mean is set as the influence weight of the node. The values for $\{a_1,\ a_2,\ a_3,\ a_4\}$ and $\{p_1,\ p_2,\ p_3,\ p_4\}$ are specified for each experiment in their respective sections.

It is worth noting that values of influence weights or thresholds become meaningful only in comparison to values of the other. For instance, a threshold of 0.8 can be seen as relatively high when most of the influence weights are smaller than 0.1, however it can be seen as relatively low when most of the influence weights are greater than 0.4. Thus, influence weights and thresholds should be viewed as strongly interrelated components.

In all experiments except for experiments on Inploid, cost values are assigned according to following formula we suggest: $c_v = 1 + \sqrt{indegree(v)}rand(min = 0.5,\ max = 1.5)$. As explained in Section 4.4.3; 1 represents the fixed cost, $indegree(v)$ represents the assumption that the cost of a node is correlated to number of followers the node has, and $rand(min = 0.5,\ max = 1.5)$ represents the distinctness in users' self-evaluations in determining their costs and also other unaccounted factors. Square root is employed for normalization purposes.

For the experiments on Inploid, cost values are assigned by using the already available information in the dataset. Details of the calculation of cost values for nodes in Inploid network is given in the respective section.

In experiments on Epinions and Academia.edu networks, profit values are assigned by multiplying the following two values: a selection in $\{0, 1, 2, 3\}$ with respective probabilities of $\{0.25,\ 0.25,\ 0.25,\ 0.25\}$, and a random selection from the log-normal distribution of $ln(mean = 1,\ sd = 0.3)$. The first component aims to represent different demographics (e.g., four equal sized target demographics). It is assumed that one group does not bring any profit at all, and the other three groups have respective importance degrees (e.g., profit potentials) of 1, 2, and 3. The second component is assumed to represent the income distribution among people in the network, creating further variations in profit values.

In experiments on Pokec and Inploid, the profit values are assigned by using the already available information in the dataset. Details of the calculation for profit values for nodes in Pokec and Inploid networks are given in the respective sections.

When two experiments are using the same data generation method, it does not mean that both are using the same data since methods include random components or network specific calculations, hence outputs vary at each run.

## 5.2 Experiments

For each experiment; degree[5], closeness, betweenness, pagerank, hub, authority, and eigenvector heuristics (see Section 2.2) are employed as benchmarks in addition to the random heuristic where seed sets are selected randomly.

When displaying different parameter settings, parameter values are added as a suffix to the name of our algorithm. For example, TABU-PG_2_1_0.05_5_$Inf$ states that our algorithm utilizes second method in node selection (i.e., $NSMthd$), utilizes first method in potential gain calculation (i.e., $PGCMthd$), employs minimum potential gain ratio of 0.05 (i.e., $MinPGR$), chooses 5 seed nodes before recalculating gain values (i.e., $NumBefReCalc$), and does not put a limit on maximum number of nodes to gain calculate for (i.e., $TopMltp$).

The parameters manage which methods are to be employed in the algorithm. Therefore, different parameter settings of TABU-PG result in different TABU-PG heuristics sharing the same framework which is explained in Chapter 4.

Detailed information about parameters and the way parameters work can be found in Section 4.2. The parameters are briefly explained in Table 5.2.

In all experiments, the budget is set as 3000.

[5]Since networks are directed, indegree is calculated. Higher indegree equates to higher number of followers.

Table 5.2: Methods Used in the Experiments

| | |
|---|---|
| ***NSMthd*: Node Selection Method** | |
| **1** | selects the node with maximum gain |
| **2** | selects the node with maximum efficiency |
| **3** | selects the node with maximum efficiency among top three nodes based on gain |
| ***PGCMthd*: Potential Gain Calculation Method** | |
| **1** | *PGMltp* is set to 0, effectively removes the potential gain |
| **2** | *PGMltp* is set to 1, effectively eliminates the impact of *PGMltp* |
| **3** | *PGMltp* is set to $1 - (E/B)$ |
| **4** | *PGMltp* is set to $1 - (E/B)(E - B)$ |
| ***MinPGR*: Minimum Potential Gain Ratio** | |
| $\boldsymbol{x}$ | if $i_{uv}$ to $\theta_v$ ratio lower than $x$, the potential gain is ignored. |
| ***NumBefReCalc*: Number of nodes to select before recalculating** | |
| ***Inf*** | all seed nodes are to be selected without recalculating gain values |
| $\boldsymbol{x}$ | up to $x$ nodes are to be selected before recalculating gain values |
| ***TopMltp*: Multiplier for determining number of top nodes** | |
| ***Inf*** | gain calculations are not limited by any number of top nodes |
| $\boldsymbol{x}$ | gain calculations are limited to top nodes, determined by multiplier $x$ |

For all experiments, results are presented in the following way. Final influence spreads (i.e., total profit) of all heuristics including TABU-PG heuristics and benchmark heuristics are given in a figure. However, these heuristics do not include different values for the scaling parameters (i.e., *NumBefReCalc* and *TopMltp*). Instead, only the default values of 1 and *Inf* are employed. Hence, whenever last two parameters are not displayed when presenting a TABU-PG heuristic, the default values should be assumed. In these figures, performance of the worst performing parameter setting for TABU-PG is set as 100%. Performances of other parameter settings and benchmark heuristics are presented by measuring their relative performances.

There are only three heuristics employing Potential Gain Calculation Method 1 (i.e., ignoring potential gains altogether) since Minimum Potential Gain Ratio does not play any role when potential gains are ignored altogether, therefore such heuristics are not multiplied for different values of *MinPGR*. There are 12 heuristics each for Potential Gain Calculation Method 2, 3, and 4. Thus, total of 39 heuristics are initially presented for each experiment except for experiments on Pokec where a smaller number of heuristics are presented due to the fact that experiments on Pokec take long time to complete.

Then, results of different TABU-PG heuristics are briefly compared with benchmark heuristics along with a comparison among themselves. The different values for first three parameters (i.e., *NSMthd*, *PGCMthd*, and *MinPGR*) are compared by averaging their performances. While calculating average, TABU-PG heuristics given in the figures are taken into account. When calculating average values for different values of *MinPGR*, heuristics which employs Potential Gain Calculation Method 1 are not accounted for since that method ignores potential gains altogether. Additional comments on results are also provided.

Next; best, median, and worst performing TABU-PG heuristics (excluding non-default values for scaling parameters) in terms of total profit are selected. Two benchmark heuristics are also selected, one being the best performing benchmark heuristic. Detailed diffusion results of the selected five heuristics are given in a chart. x-axis shows the amount spent (i.e., $E$), and y-axis shows the total profit obtained. Relevant comments are provided.

Subsequently, effects of different values of scaling parameters on the best performing heuristic and the median performing heuristic is illustrated in terms of total profit and also runtime. The exception is Experiment 8 where only one heuristic is employed instead of two. In these figures, primary axis and columns represent the spread performance (i.e., total profit) whereas secondary axis and lines represent the runtime. The performance of the heuristic with default values for the scaling parameters is set to 100%, in order to serve as a benchmark. Comments are provided on the results.

Setting $NumBefReCalc$ to $Inf$ would not be relevant to Potential Gain Calculation Method 3 and 4, however it is still included in figures which employ Method 3 or 4. These two methods assign value to $PGMltp$ according to the remaining budget whereas $NumBefReCalc = Inf$ selects all nodes at once in the first iteration instead of spending budget overtime. Therefore, Potential Gain Methods 3 and 4 is equivalent to Method 2 whenever $NumBefReCalc$ is set to $Inf$.

Overall, the methodology and structure for presenting experiment results is nearly the same for all experiments. On the other hand, experiments on Pokec can be considered as exceptions since they, to a certain extent, diverge from the standard structure we use to present our results. For this reason, in Section 5.3, results of experiments on Pokec are treated differently when presenting and discussing overall results.

### 5.2.1 Experiments on Epinions

#### Experiment 1

Threshold values are assigned randomly by using the following truncated normal distribution: $tn(mean = 0.65,\ sd = 0.3,\ lower = 0.1,\ upper = Inf)$. Profit values and cost values are assigned by utilizing the methods which are explained in Section 5.1. Influence weights are assigned by the ratio model.

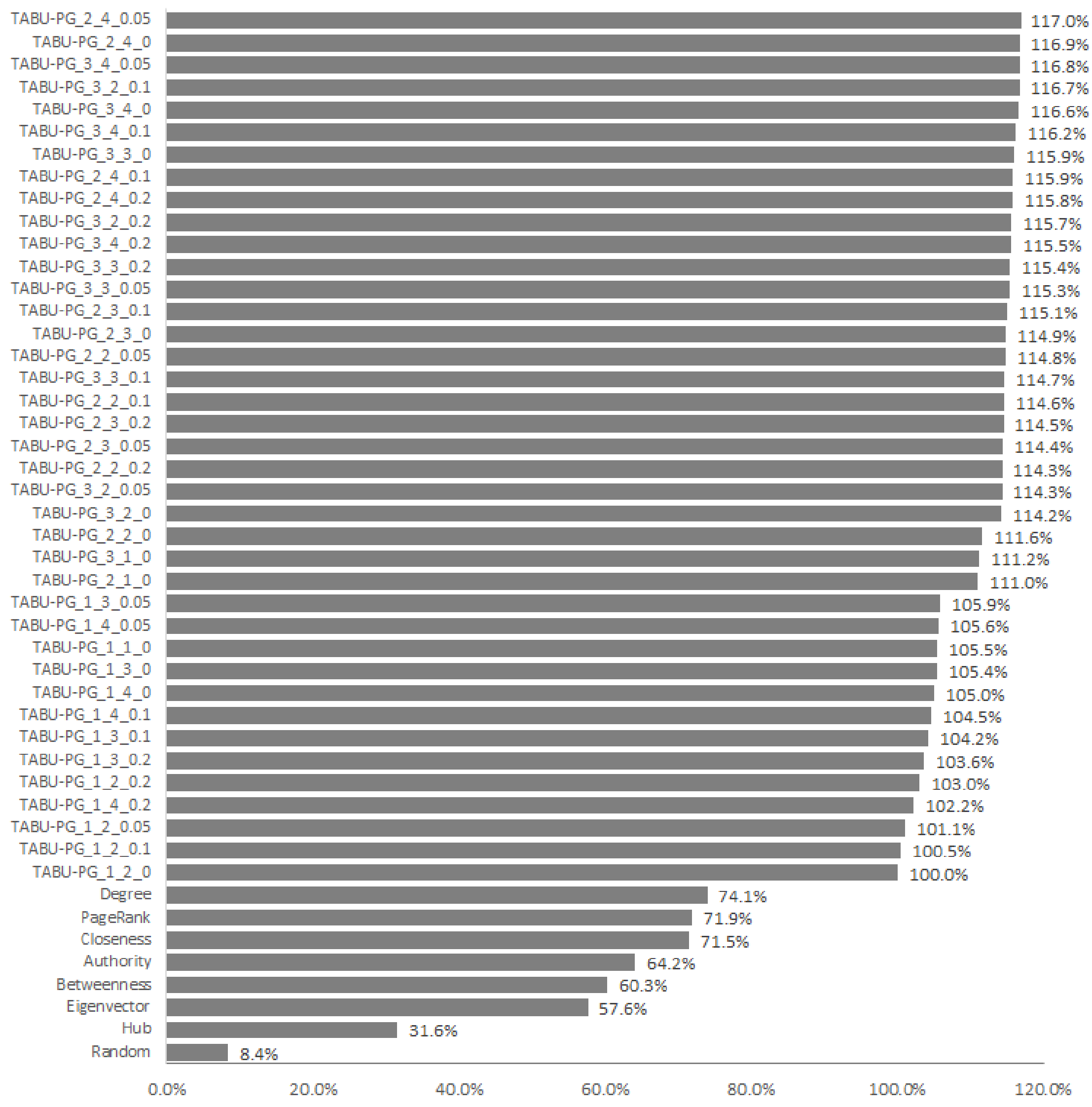

FIGURE 5.1: Comparison of All Heuristics in Experiment 1

Figure 5.1 depicts the comparison of different parameter settings of TABU-PG along with benchmark heuristics, in terms of achieved influence spread (i.e., total profit). Performance of the worst performing parameter setting for TABU-PG, which is TABU-PG_1_2_0 in this case, is set as 100%.

TABU-PG with different parameter settings perform 35% to 58% better than the best benchmark heuristic, that is degree heuristic in this case.

When different parameter settings of TABU-PG are compared between themselves, there is %17 increase in total profit, from the worst parameter setting to the best one.

Node Selection Method 1 (i.e., based on gain) performs consistently worse than the Method 2 and Method 3 (i.e., efficiency based methods). On average, Method 1 obtains a performance of 103.6% while Method 2 obtains 114.7% and Method 3 obtains 115.3%.

Among all; the best two heuristics employ Method 2, whereas the following five heuristics employ Method 3.

Potential Gain Calculation Method 3 and Method 4, which are introduced by us, usually perform better than Method 1 and Method 2. On average, Method 1, 2, 3, and 4 achieve performances of 109.2%, 110.1%, 111.6%, and 112.3% respectively. Out of the best 10 heuristics, 7 employ Method 4.

A significant and consistent difference between performances of different Minimum Potential Gain Ratios is not observed in this experiment. Yet, there exist slight improvements gained by tuning $MinPGR$. Minimum Potential Gain Ratio of 0, 0.05, 0.1, and 0.2 achieve average performances of 111.2%, 111.7%, 111.4%, and 111.1% respectively.

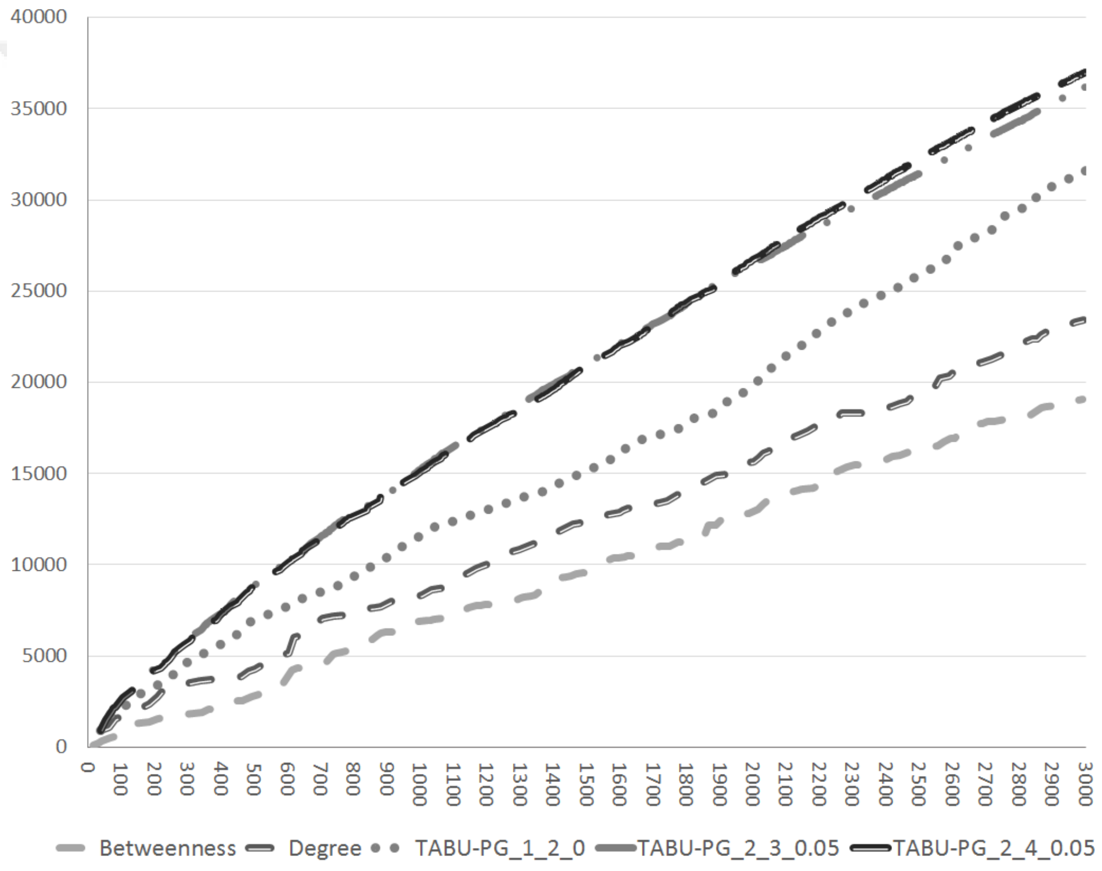

FIGURE 5.2: Detailed Diffusion of Selected Heuristics in Experiment 1

Figure 5.2 shows detailed diffusion results with regard to budget for the selected heuristics: betweenness, degree, the worst parameter setting TABU-PG_1_2_0, the median parameter setting TABU-PG_2_3_0.05, and the best parameter setting TABU--PG_2_4_0.05. As can be interpreted from the figure, there is no observation of any sudden increase in influence spread as budget is exhausted over time. The influence spread is close to a linear function of the budget.

Figure 5.3 shows the effects of different values of scaling parameters on the best performing heuristic TABU-PG_2_4_0.05, in terms of runtime and total profit. As depicted in the figure, increasing the value of $NumBefReCalc$ from 1 to 5, and to 25 improves the runtime from 341 to 212, and to 88 seconds respectively. However, as a trade-off, the total profit decreases by 7.2% and 17.5% respectively.

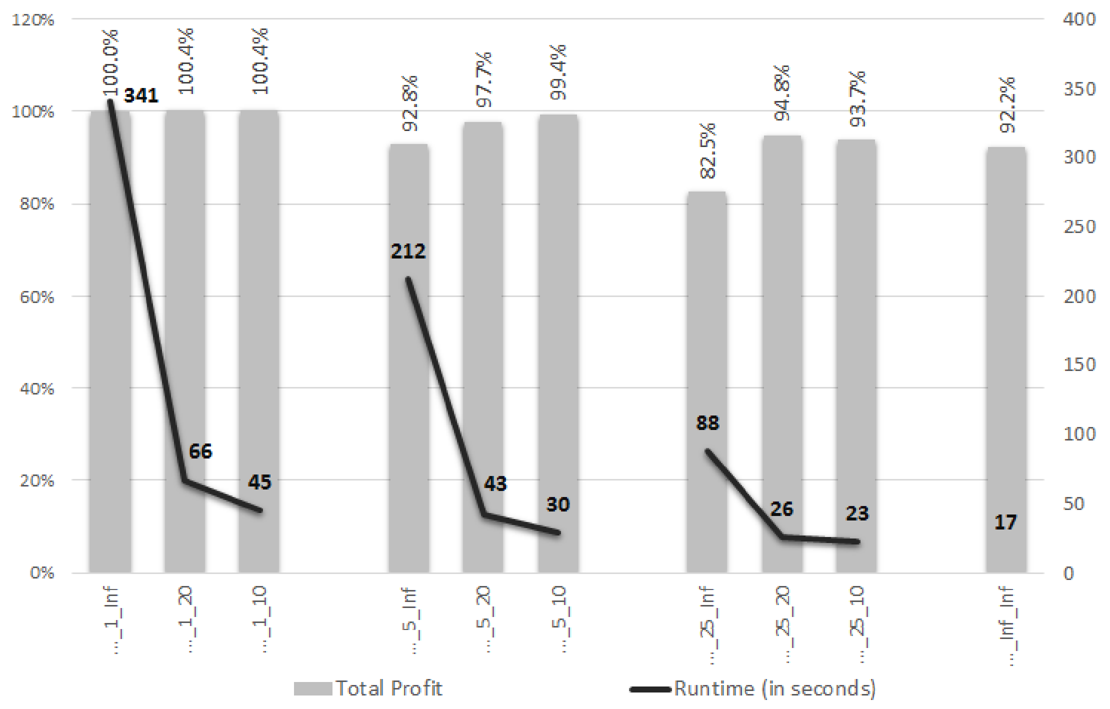

FIGURE 5.3: Scaling TABU-PG_2_4_0.05 in Experiment 1

Counter-intuitively, limiting calculations to top nodes by decreasing the value of $TopMltp$ improves the total profit while also significantly reducing the runtime. The improvement is very slight when $NumBefReCalc$ is set to 1 however is more significant when $NumBefReCalc$ is set to greater values. On the other hand, decreasing $TopMltp$ from 20 to 10, thus creating a even smaller pool of candidates, reduces the extent of improvement, or eliminates it altogether. This result of improved performance with limited pool of candidates might be because of the non-optimal nature of the solution due to the complexity of the problem, hence an exception. Or, it might be a rule which does not seem intuitive at first. We will refer to this as the counter-intuitive result in Experiment 1, and examine whether this result hold true in other experiments.

A comparison of scaling parameters' impact for the median performing heuristic TABU-PG_2_3_0.5 is given in Figure 5.4. Roughly, the same conclusions for Figure 5.3 apply to this one as well. However, further limiting the pool of candidates by setting $TopMltp$ to 10 instead of 20 does not necessarily reduce or eliminate the improvement in performance, as it does in Figure 5.3.

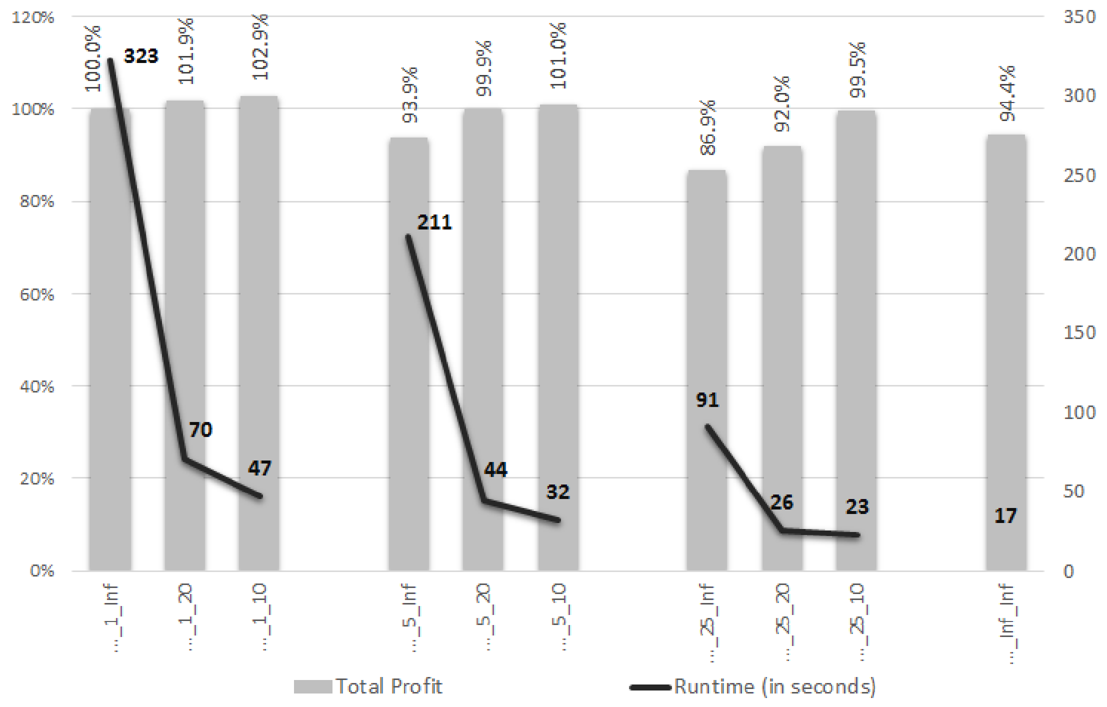

FIGURE 5.4: Scaling TABU-PG_2_3_0.5 in Experiment 1

**Experiment 2**

Threshold values are assigned randomly by using the following truncated normal distribution: $tn(mean = 1, sd = 0.5, lower = 0.1, upper = Inf)$. Profit values and cost values are assigned by utilizing the methods which are explained in Section 5.1.

Influence weights are assigned by the hybrid model. $x$ being the average degree of nodes, for each node, a value from $\{x/4,\ x/2,\ x,\ 3x/2\}$ is selected with respective probabilities of $\{0.25,\ 0.25,\ 0.25,\ 0.25\}$. Then, geometric mean of the selected value and the value which would be resulted from ratio model is set as the influence weight of the node. In this way, influence weights are more compatible with real life as we explained in Section 4.4.1.

Figure 5.5 shows comparison of different parameter settings of TABU-PG along with benchmark heuristics, in terms of achieved influence spread (i.e., total profit obtained). The results show that TABU-PG with different parameter settings perform 19% to 32% better than the best benchmark heuristic, that is closeness heuristic in this case.

When different parameter settings of TABU-PG are compared between themselves, there is %11 increase in total profit, from the worst parameter setting to the best one.

Node Selection Method 1 (i.e., based on gain) performs worse than Method 1 and Method 2 (i.e., the efficiency based methods) at almost all times. Method 1 achieves an average

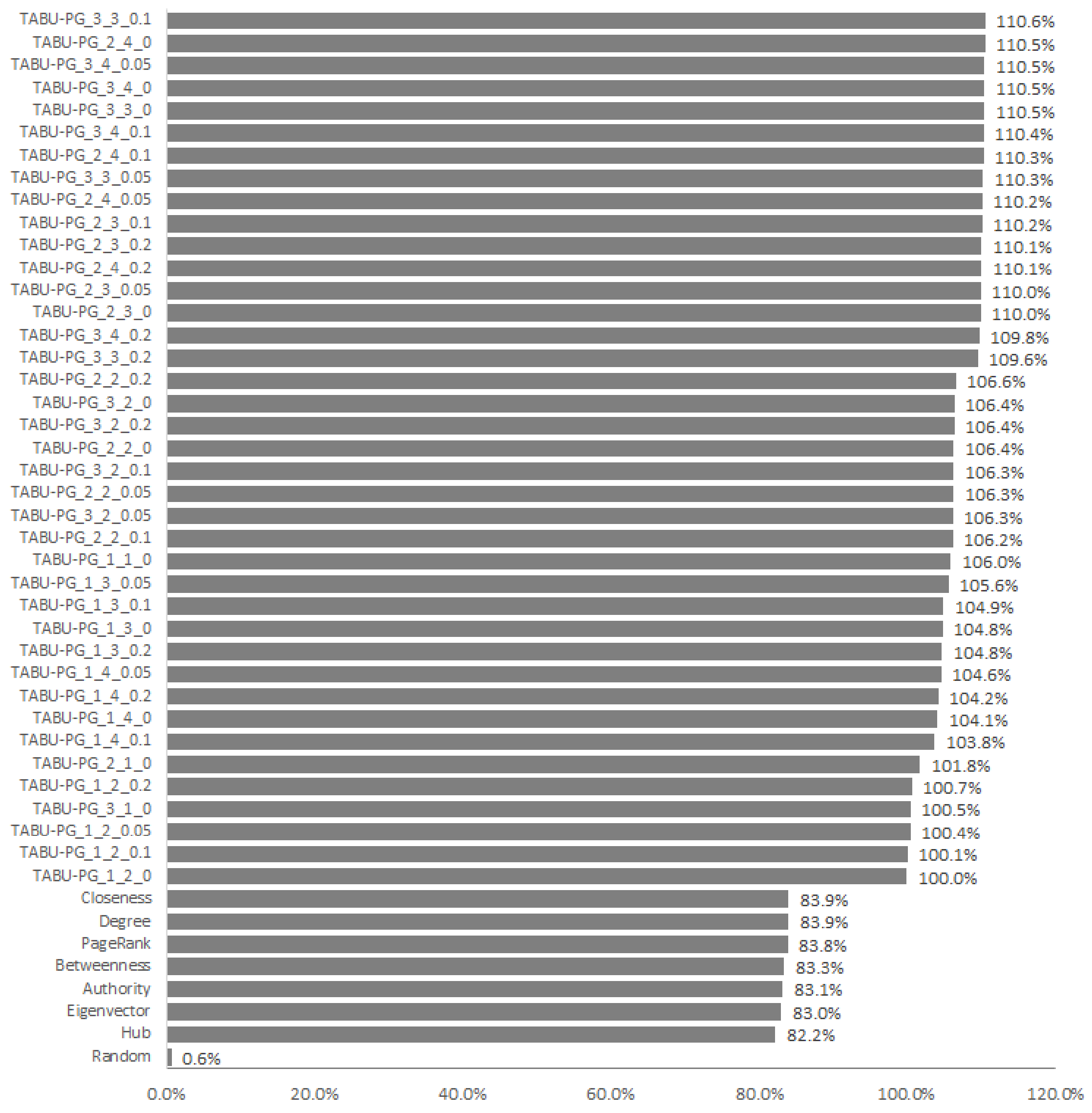

FIGURE 5.5: Comparison of All Heuristics in Experiment 2

performance of 103.4% while Method 2 achieves 108.4% and Method 3 achieves 108.3%. Among all heuristics, 4 out of the best 5 heuristics utilizes Method 3, and the remaining one utilizes Method 2.

Potential Gain Calculation Method 3 and Method 4 usually perform better than Method 1 and Method 2. On average, Method 1, 2, 3, and 4 achieve performances of 102.7%, 104.4%, 108.5%, and 108.3% respectively. All of the best 10 heuristics employ Method 3 or 4.

A meaningful and consistent difference between performances of different Minimum Potential Gain Ratios is not observed in this experiment as well. Minimum Potential Gain Ratio of 0, 0.05, 0.1, and 0.2 achieve average performances of 107%, 107.1%, 107%, and 106.9% respectively.

Figure 5.6 shows detailed diffusion results with regard to budget for the selected heuristics: hub, closeness, the worst parameter setting TABU-PG_1_2_0, the median parameter setting TABU-PG_2_2_0, and the best parameter setting TABU-PG_3_3_0.1.

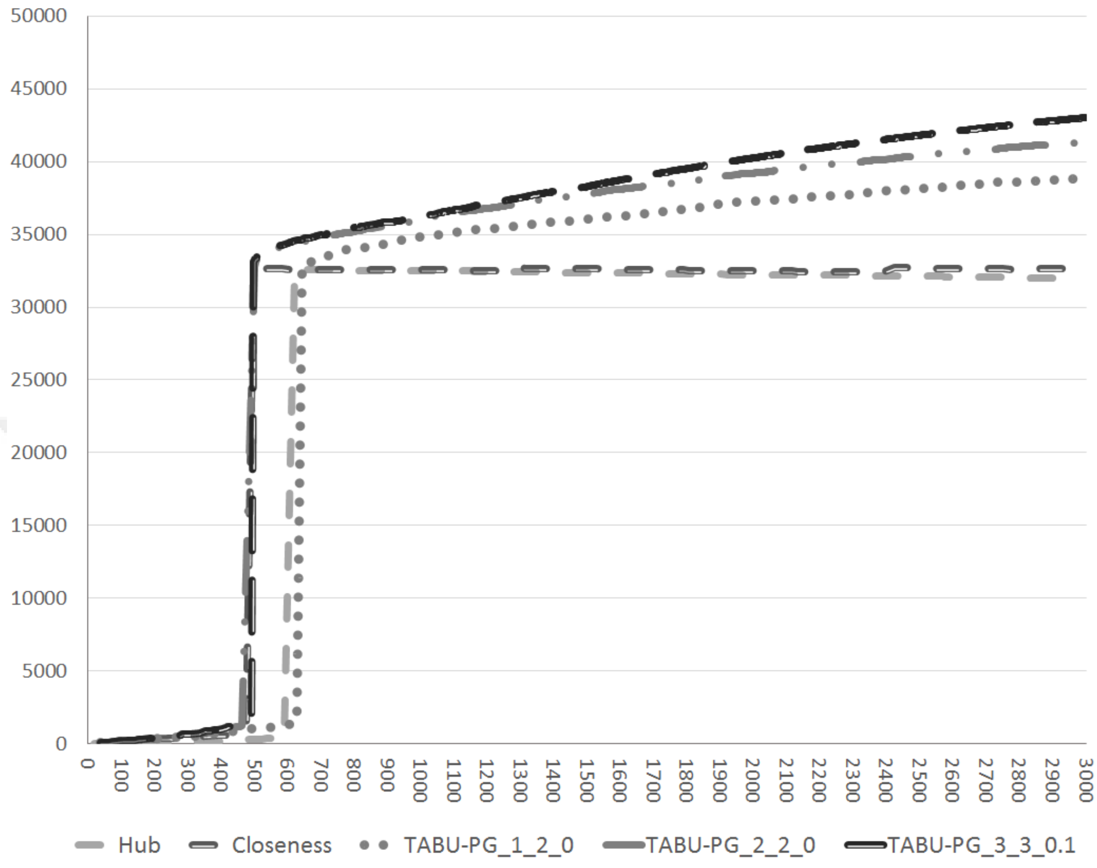

FIGURE 5.6: Detailed Diffusion of Selected Heuristics in Experiment 2

As it can be seen in the figure, total profit dramatically increases after reaching a certain point. It is a reflection of the *tipping point* phenomenon. Some of the algorithms in the figure reach the tipping point before others in terms of exhausted budget. Therefore, in a case where the budget is close to the tipping points, parameter tuning can result in extraordinarily large improvements. If the algorithm was terminated between 500 and 600 instead of 3000, TABU-PG_3_3_0.1 would perform tens of times better than TABU-PG_1_2_0 instead of only providing an %11 improvement.

Figure 5.7 shows the effects of different values of scaling parameters on the best performing heuristic TABU-PG_3_3_0.1. Increasing the value of $NumBefReCalc$ from 1 to 5, and to 25 improves the runtime from 105 seconds to 57 and 59 seconds respectively. Moreover, the total profit decreases only by 1.9% and by 6.9% respectively.

The counter-intuitive result we encountered in Experiment 1 does not hold true in this case, except for the observed improvement from ..._25_$Inf$ to ..._25_20. Compared to

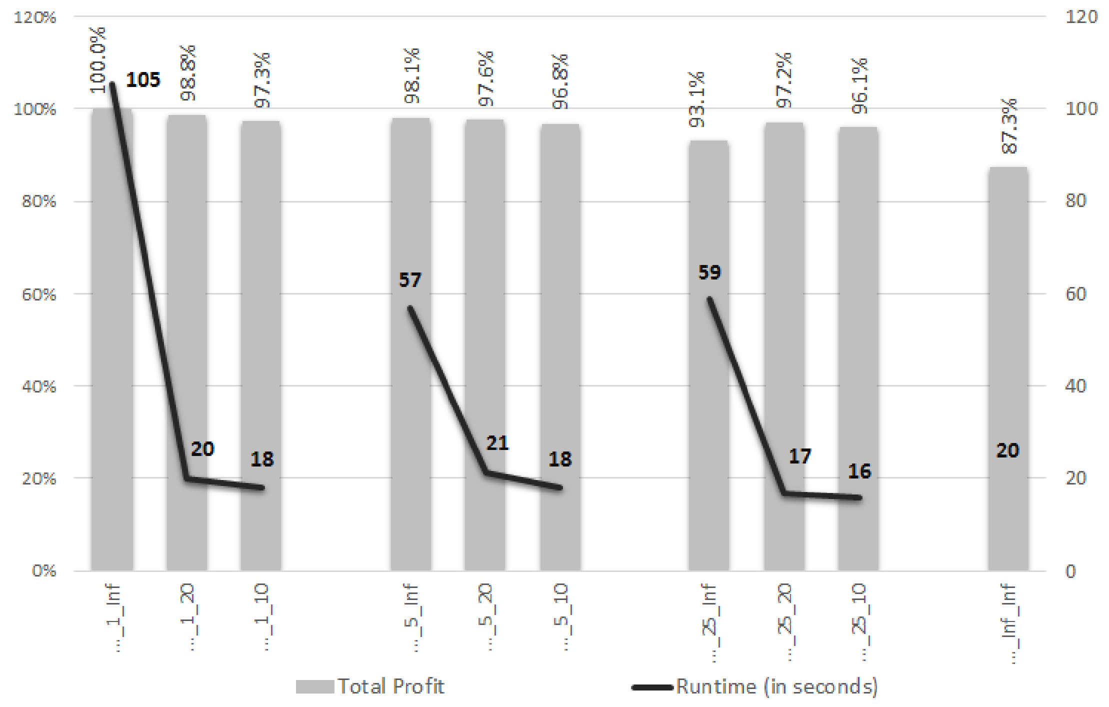

FIGURE 5.7: Scaling TABU-PG_3_3_0.1 in Experiment 2

..._1_$Inf$, ..._1_20 takes less than one fifth of the time while the performance in terms of total profit decreases only by 1.2%. The parameter setting which takes the shortest time, except for the setting of ..._$Inf$_$Inf$, reduces the spread performance by less than 4%.

Figure 5.8 shows the effects of different values of scaling parameters on the median performing heuristic TABU-PG_2_2_0. More or less, the same conclusions for Figure 5.7 are valid for this figure as well.

### 5.2.2 Experiments on Academia.edu

#### Experiment 3

Threshold values are assigned randomly by using the following truncated normal distribution: $tn(mean = 0.65,\ sd = 0.3,\ lower = 0.1,\ upper = Inf)$. Profit values and cost values are assigned by utilizing the methods which are explained in Section 5.1. Influence weights are assigned by the ratio model.

Figure 5.9 displays a comparison of different parameter settings of TABU-PG along with benchmark heuristics, in terms of achieved influence spread (i.e., total profit obtained). The results show that TABU-PG with different parameter settings perform 11% to 37%

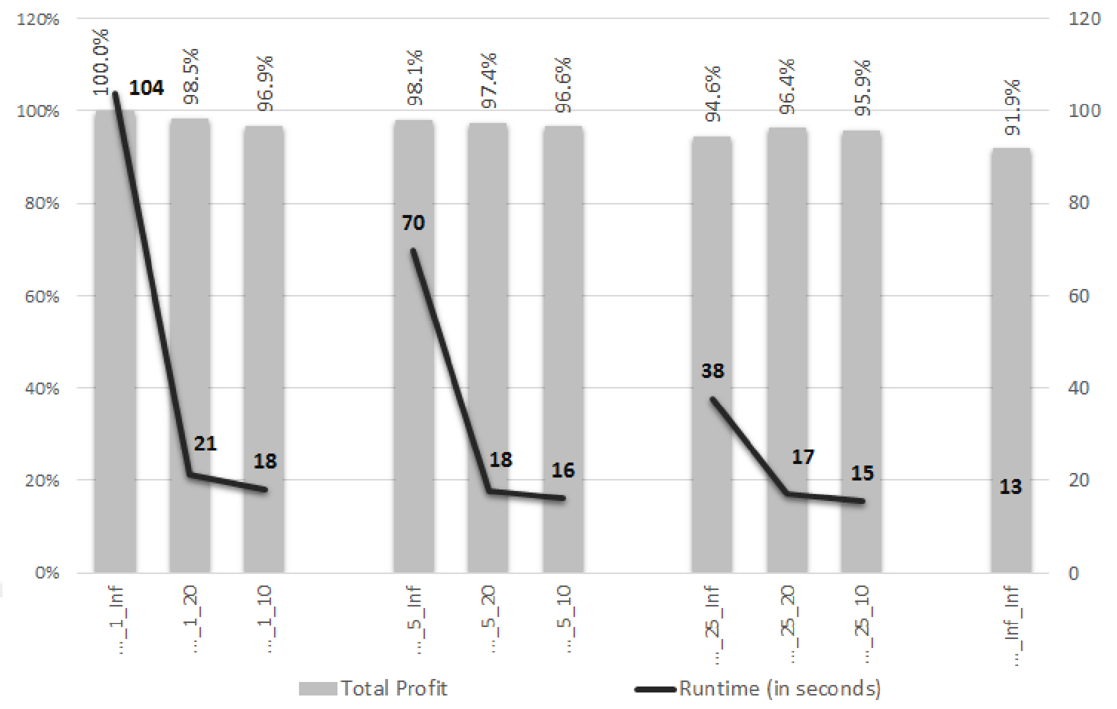

FIGURE 5.8: Scaling TABU-PG_2_2_0 in Experiment 2

better than the best benchmark heuristic, that is pagerank heuristic in this case. When compared with the second best benchmark heuristic, which is degree heuristic, TABU-PG performs 107% to 156% better.

When different parameter settings of TABU-PG are compared between themselves, there is %24 increase in total profit, from the worst parameter setting to the best one.

Node Selection Method 1 (i.e., based on gain) performs always worse than Method 1 and Method 2 (i.e., the efficiency based methods). Method 1 achieves an average performance of 108.4% while Method 2 achieves 121.6% and Method 3 achieves 121.6% as well. As depicted in the figure, Method 2 and Method 3 achieves almost the same results although slightly different at times. This is probably due to the fact that both methods end up choosing almost the same seed set.

On average, Potential Gain Calculation Methods 1, 2, 3, and 4 achieve performances of 119.5%, 113.2%, 119.2%, and 118.7% respectively. Out of the best 10 heuristics, 8 employ Method 3 or 4. In contrast to Experiment 1 and 2, Method 1 is not outperformed by other methods. Method 3 and Method 4 performs better than Method 2, as in Experiment 1 and 2.

Minimum Potential Gain Ratio of 0, 0.05, 0.1, and 0.2 achieve average performances of 116.5%, 116.7%, 117.1%, and 117.8% respectively. Therefore, tuning $MinPGR$ provides slight improvements in total profit.

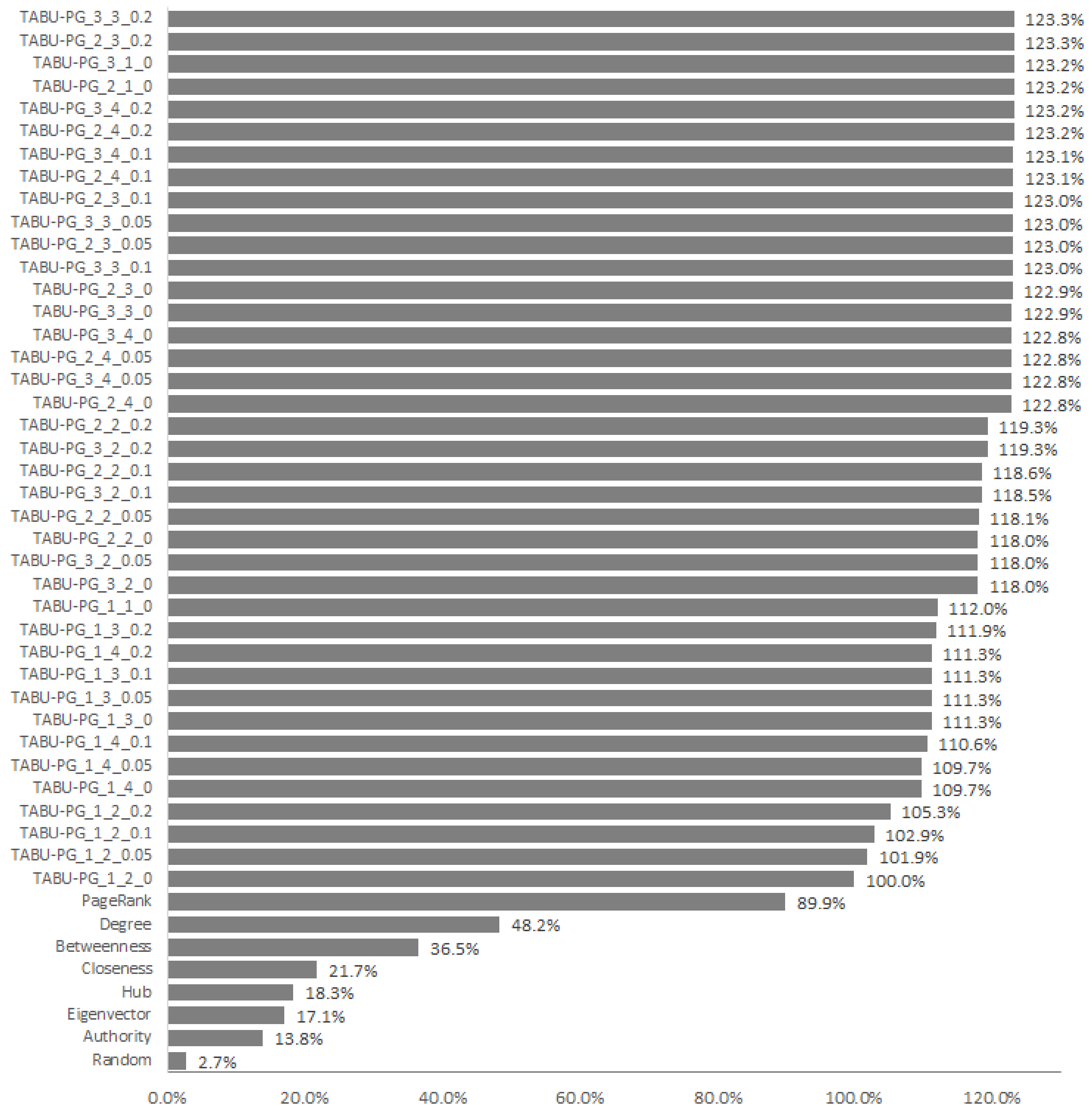

Figure 5.9: Comparison of All Heuristics in Experiment 3

Figure 5.10 shows detailed diffusion results for the selected heuristics with regard to budget: closeness, pagerank, the worst parameter setting TABU-PG_1_2_0, the median parameter setting TABU-PG_3_2_0.2, and the best parameter setting TABU-PG_3_3_0.2.

As can be seen in the figure, there is no observation of any sudden increase in influence spread as budget is exhausted over time. The advantage of Partial Gain Calculation Method 3 over Method 2 is clearly displayed. Both methods perform very similar at the beginning while Method 3 positively distinguishes itself from Method 2 as it gets closer to the end of budget. This is because Method 3 weighs less on potential gains and more on actual gains as it gets closer to the end.

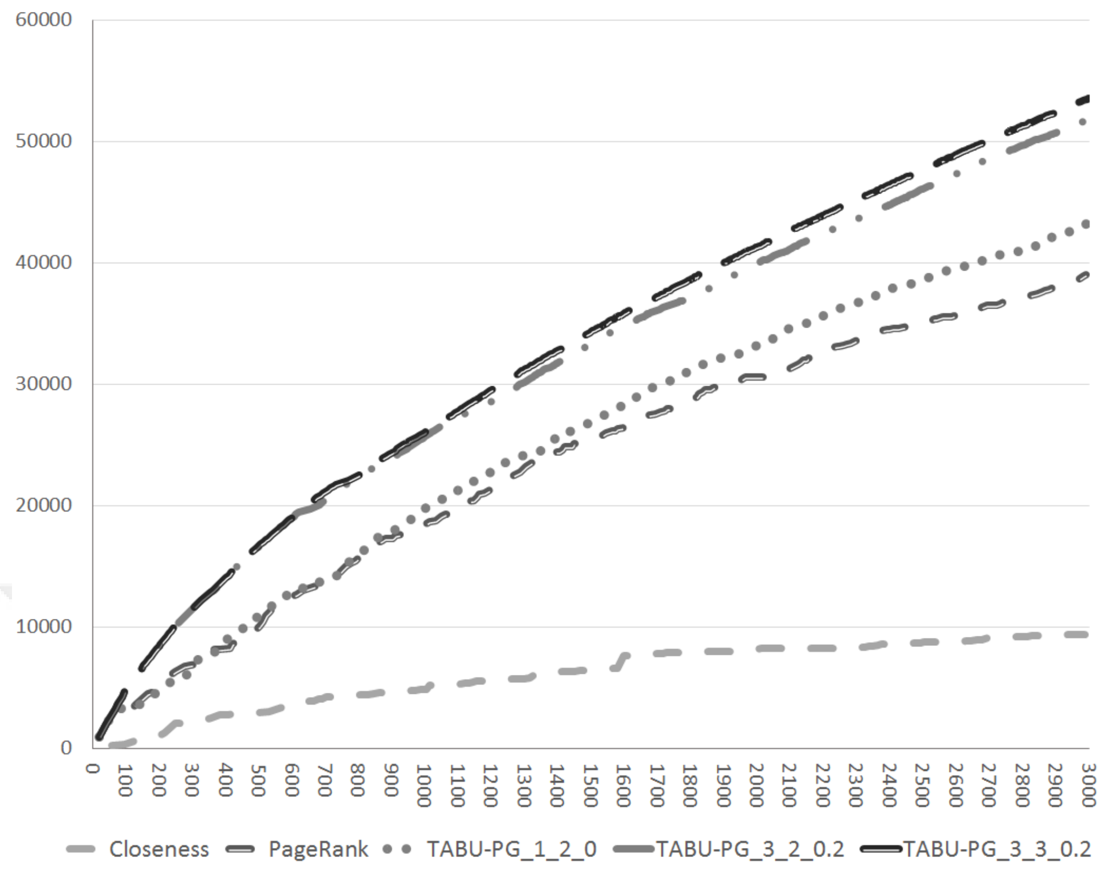

FIGURE 5.10: Detailed Diffusion of Selected Heuristics in Experiment 3

Figure 5.11 shows the effects of different values of scaling parameters on the best performing heuristic TABU-PG_3_3_0.2. Increasing the value of $NumBefReCalc$ from 1 to 5, and to 25 improves the runtime from 1477 seconds to 1212 and 634 seconds respectively. Moreover, the total profit decreases only by 0.2% in both cases.

The counter-intuitive result we encountered in Experiment 1 does not hold true in this case as well, except for small improvements of 0.2% and 0.1% in heuristics which sets $TopMltp$ to 25. The parameter setting taking the shortest runtime takes approximately one ninth of the time required for the heuristic with default scaling parameter values while the performance in terms of total profit are virtually the same in both cases.

Figure 5.12 shows the effects of different values of scaling parameters on the median performing heuristic TABU-PG_3_2_0.2. Nearly, the same conclusions for Figure 5.11 hold true for results given in this figure as well.

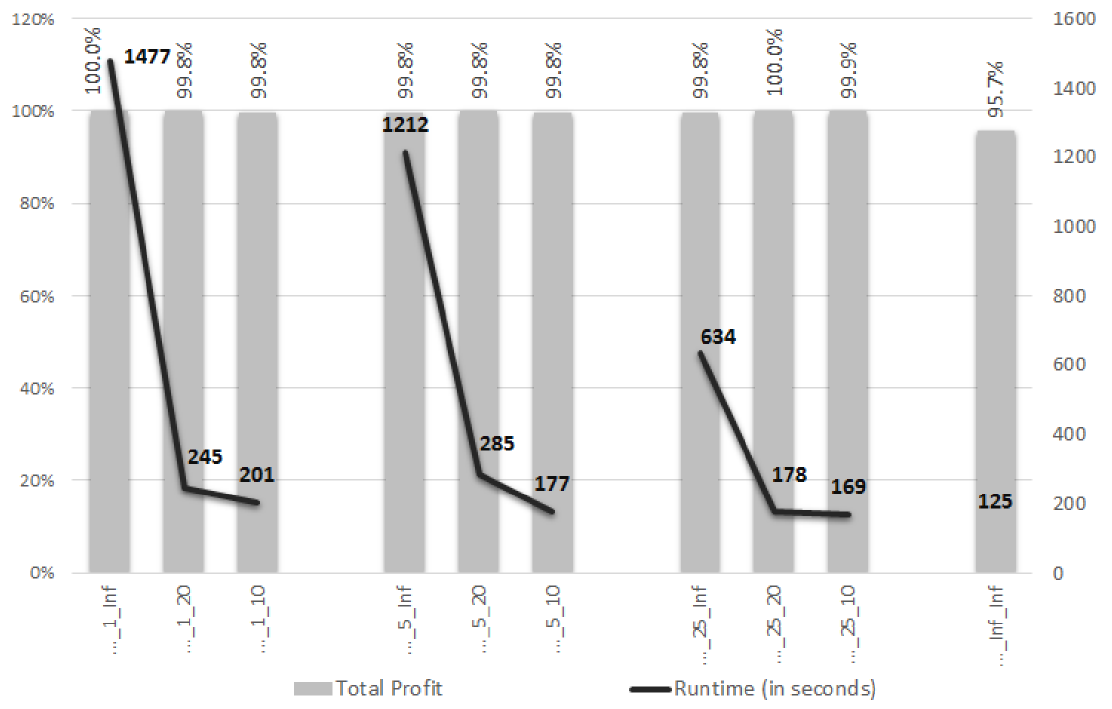

FIGURE 5.11: Scaling TABU-PG_3_3_0.2 in Experiment 3

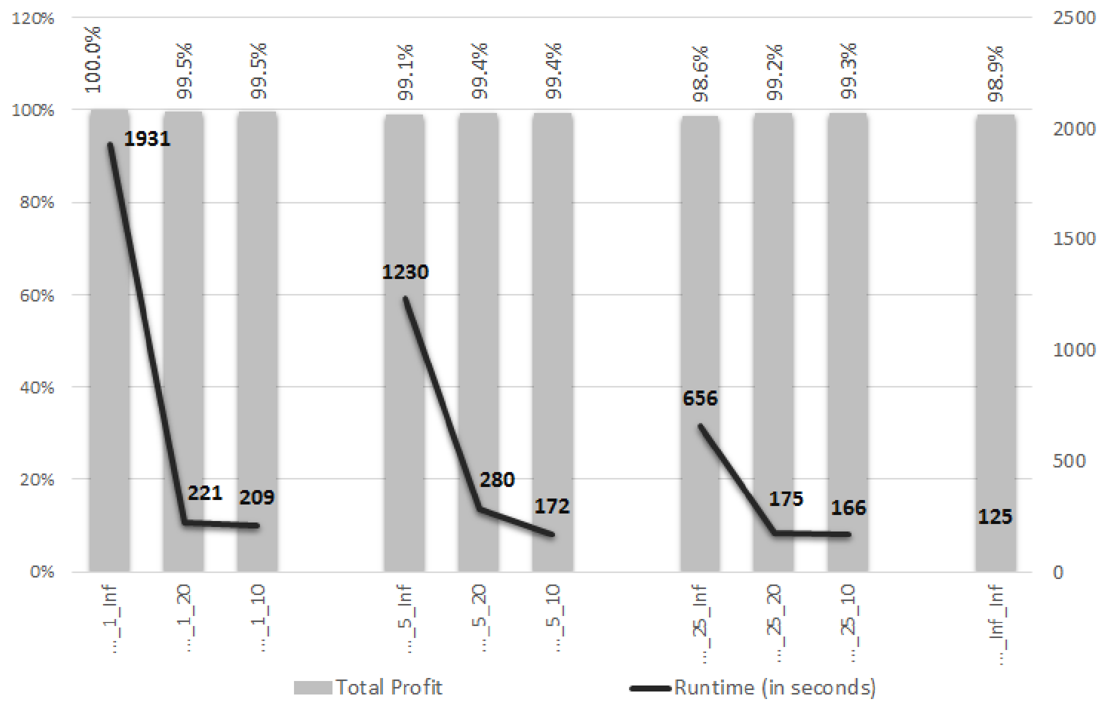

FIGURE 5.12: Scaling TABU-PG_3_2_0.2 in Experiment 3

## Experiment 4

Threshold values are assigned randomly by using the following truncated normal distribution: $tn(mean = 0.8, sd = 0.4, lower = 0.05, upper = Inf)$. Profit values and cost values are assigned by utilizing the methods which are explained in Section 5.1.

Influence weights are assigned by the hybrid model. $x$ being the average degree of nodes, for each node, a value from $\{x/3,\ x/2,\ x,\ 2x\}$ is selected with respective probabilities of $\{0.1,\ 0.2,\ 0.3,\ 0.4\}$. Then, geometric mean of the selected value and the value which would be resulted from ratio model is set as the influence weight of the node.

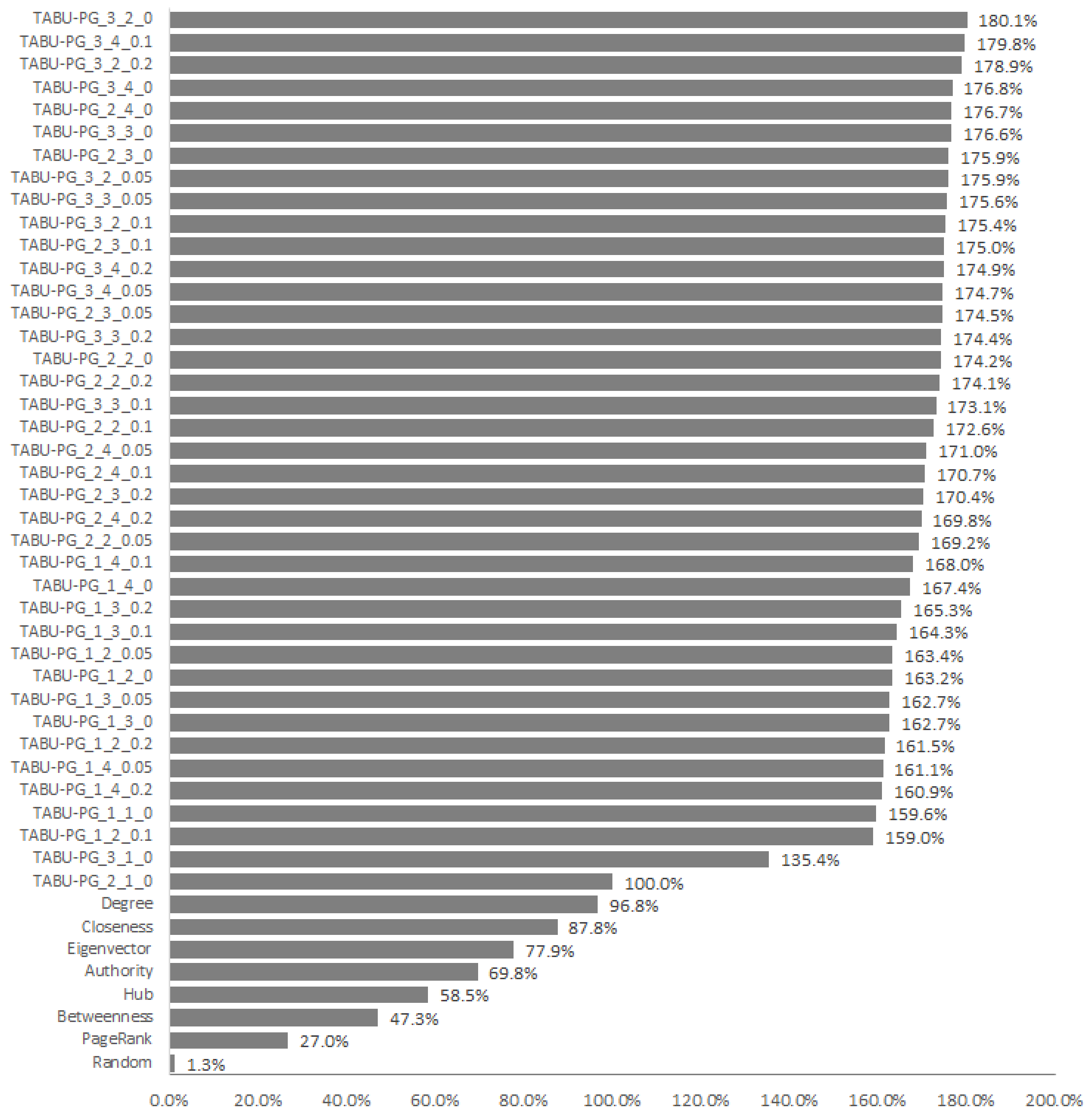

Figure 5.13: Comparison of All Heuristics in Experiment 4

Figure 5.13 shows comparison of different parameter settings of TABU-PG along with benchmark heuristics, in terms of achieved influence spread (i.e., total profit obtained). The results show that TABU-PG with different parameter settings perform 3% to 86% better than the best benchmark heuristic, that is degree heuristic in this case. When compared with the second best benchmark heuristic, which is closeness heuristic, TABU-PG performs 14% to 105% better.

When different parameter settings of TABU-PG are compared between themselves, there is %80 increase in total profit, from the worst parameter setting to the best one.

Node Selection Method 1 (i.e., based on gain) usually performs worse than Method 1 and Method 2 (i.e., the efficiency based methods). Method 1 achieves an average performance of 163% while Method 2 achieves 167.2% and Method 3 achieves 173.2%. Among top 5 heuristics, 4 employ Method 3 and the remaining one employs Method 2.

Potential Gain Calculation Methods 1, 2, 3, and 4 achieve average performances of 131.7%, 170.6%, 170.9%, and 171% respectively. Among the top 9 heuristics; Method 2, 3, and 4 are utilized by equal number of heuristics.

Minimum Potential Gain Ratio of 0, 0.05, 0.1, and 0.2 achieve average performances of 172.6%, 169.8%, 170.9%, and 170% respectively. In contrast to previous experiments, increasing the value of $MinPGR$ slightly decreases the performance in terms of total profit. Overall, the performance differences are very small thus insignificant.

Figure 5.14 shows detailed diffusion results with regard to budget for the selected heuristics: authority, degree, the worst parameter setting TABU-PG_2_1_0, the median parameter setting TABU-PG_2_4_0.05, the best parameter setting TABU-PG_3_2_0.

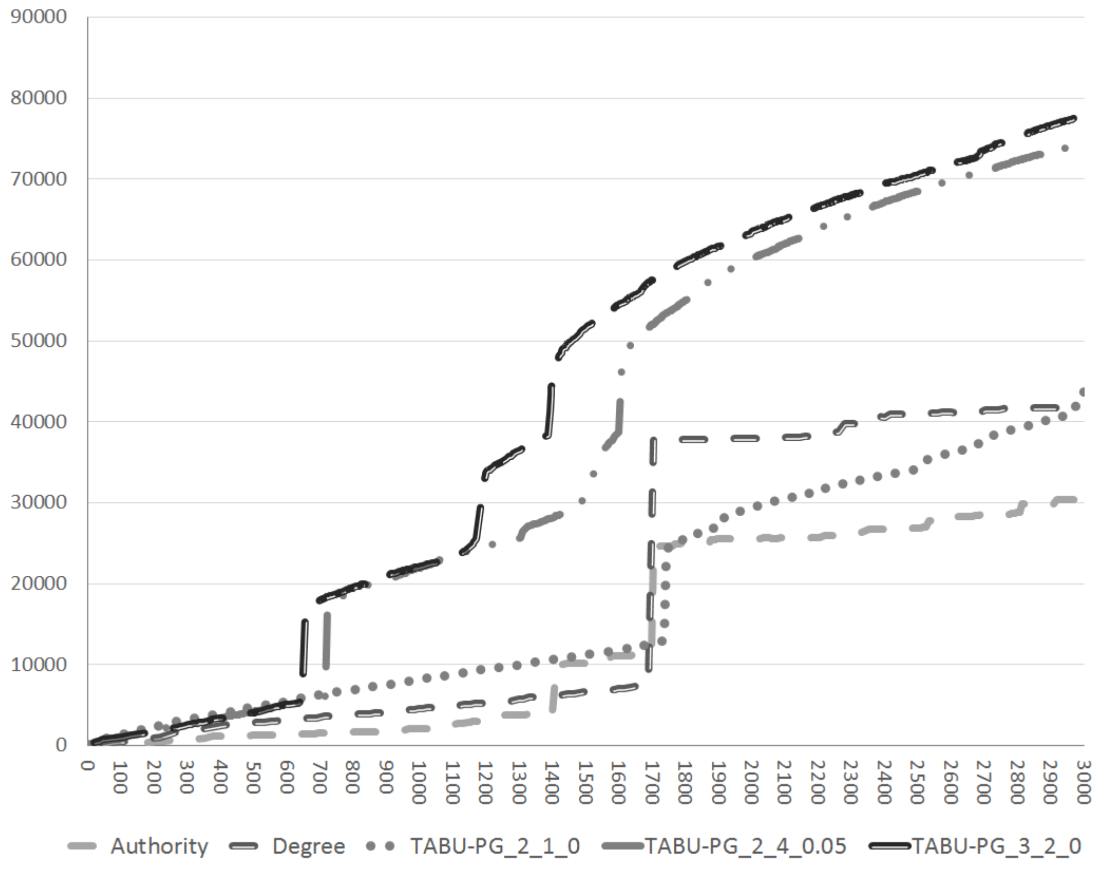

FIGURE 5.14: Detailed Diffusion of Selected Heuristics in Experiment 4

As can be seen in Figure 5.14, tipping points exist in this experiment as well. For the best performing two heuristics, there exist two tipping points for each. One is around 700 for both. Other one is through 1200 and 1400 for TABU-PG_3_2_0 and through 1500 and 1700 for TABU-PG_2_4_0.05. As can be recalled from Experiment 2; in a case where the budget is close to the tipping points, parameter tuning can result in extraordinarily large improvements.

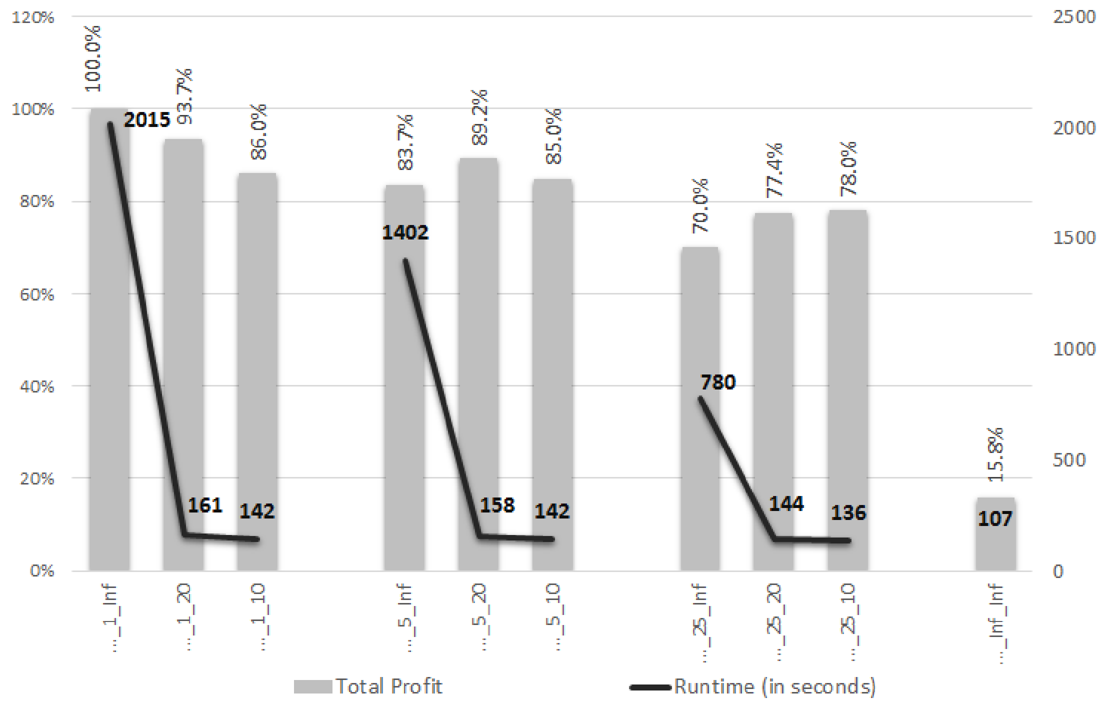

FIGURE 5.15: Scaling TABU-PG_3_2_0 in Experiment 4

Figure 5.15 shows the effects of different values of scaling parameters on the best performing heuristic, TABU-PG_3_2_0. Increasing the value of from 1 to 5, 25, and $Inf$ improves the runtime from 2015 seconds to 1402, 780, and 107 seconds respectively. As a trade-off, the total profit decreases by 16.3%, 30%, and 84.2% respectively. The reductions in performance is extraordinarily high in this experiment when compared to other experiments. Especially the large decrease which emerges when all nodes are selected at once (i.e., $NumBefReCalc = Inf$) seems to be an exception since experiments on other networks do not provide a similar result, hence should be viewed as such. However, we do not suggest that these exceptions are not expected at all.

When $NumBefReCalc$ is kept as 1, setting the $TopMltp$ as 20 decreases the runtime from 2015 seconds to 161 seconds. As a trade-off, the total profit is worsened by 6.3%. On the other hand, the counter-intuitive result we encountered in Experiment 1 neither holds true nor is falsified in this experiment.

Figure 5.16 shows the effects of different values of scaling parameters on TABU-PG_2_4_0.05. Overall, the same conclusions for Figure 5.11 are accurate for this figure as well.

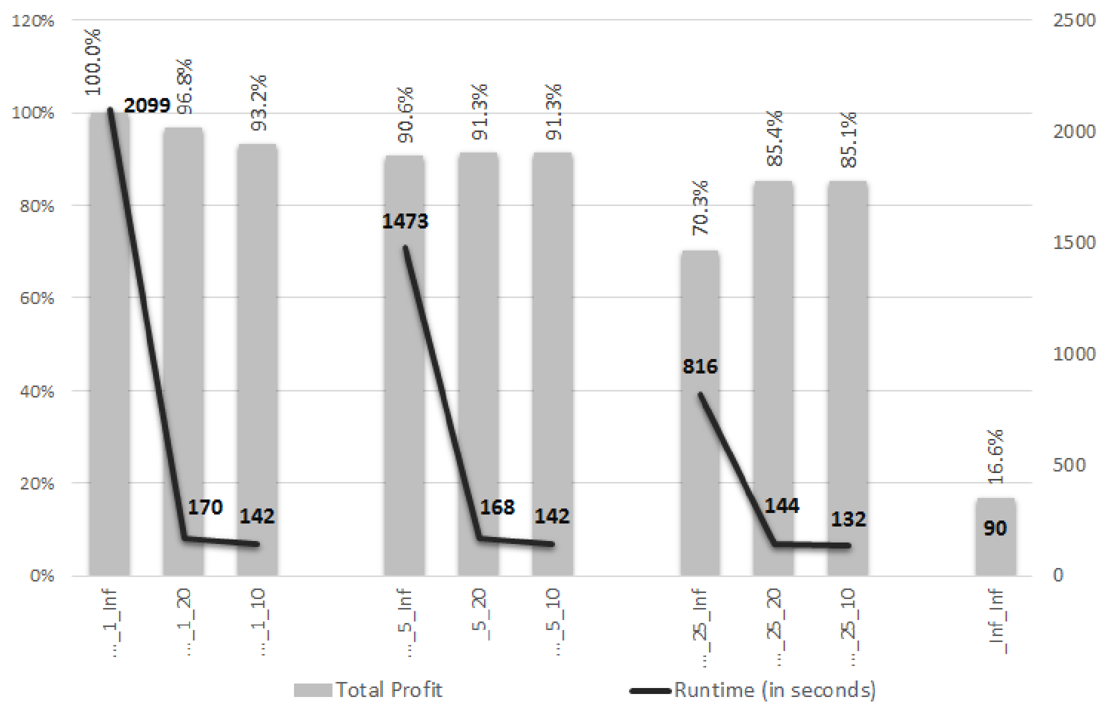

Figure 5.16: Scaling TABU-PG_2_4_0.05 in Experiment 4

### 5.2.3 Experiments on Inploid

For experiments on Inploid network, profit values are assigned in the following way. There exist over 3000 unique topics which users are shown interest in. STEM (Science Technology Engineering Mathematics) related topics are manually flagged by us. In the dataset, a user can have at most 5 associated topics. Number of STEM-flagged topics are counted and assigned as profit values. If a user is interested in 3 STEM-related topics, its profit value is assigned as 3. For users who did not specify any topic on their profile or who are not interested in any STEM-related topic are assigned a profit value of 0.5. Overall, this profit assignment method might reflect a case where a product's primary target group is people who are interested in STEM.

For experiments on Inploid network, cost values are assigned according to following formula: $c_v = 1 + \sqrt[3]{indegree(v)} + \sqrt[3]{v_{rep}}$. 1 represents the fixed cost, $indegree(v)$ represents the assumption that the cost of a node is correlated to the number of followers the node has, and $v_{rep}$ represents reputability scores of users. Cube root is employed for normalization purposes. Overall, this cost assignment method might reflect a case where users determines their costs by their reputability scores along with the number of their followers.

**Experiment 5**

Threshold values are assigned randomly by using the following truncated normal distribution: $tn(mean = 1,\ sd = 0.33,\ lower = 0.1,\ upper = Inf)$. Profit values and cost values are assigned by utilizing the methods which are explained in Section 5.2.3. Influence weights are assigned by the ratio model.

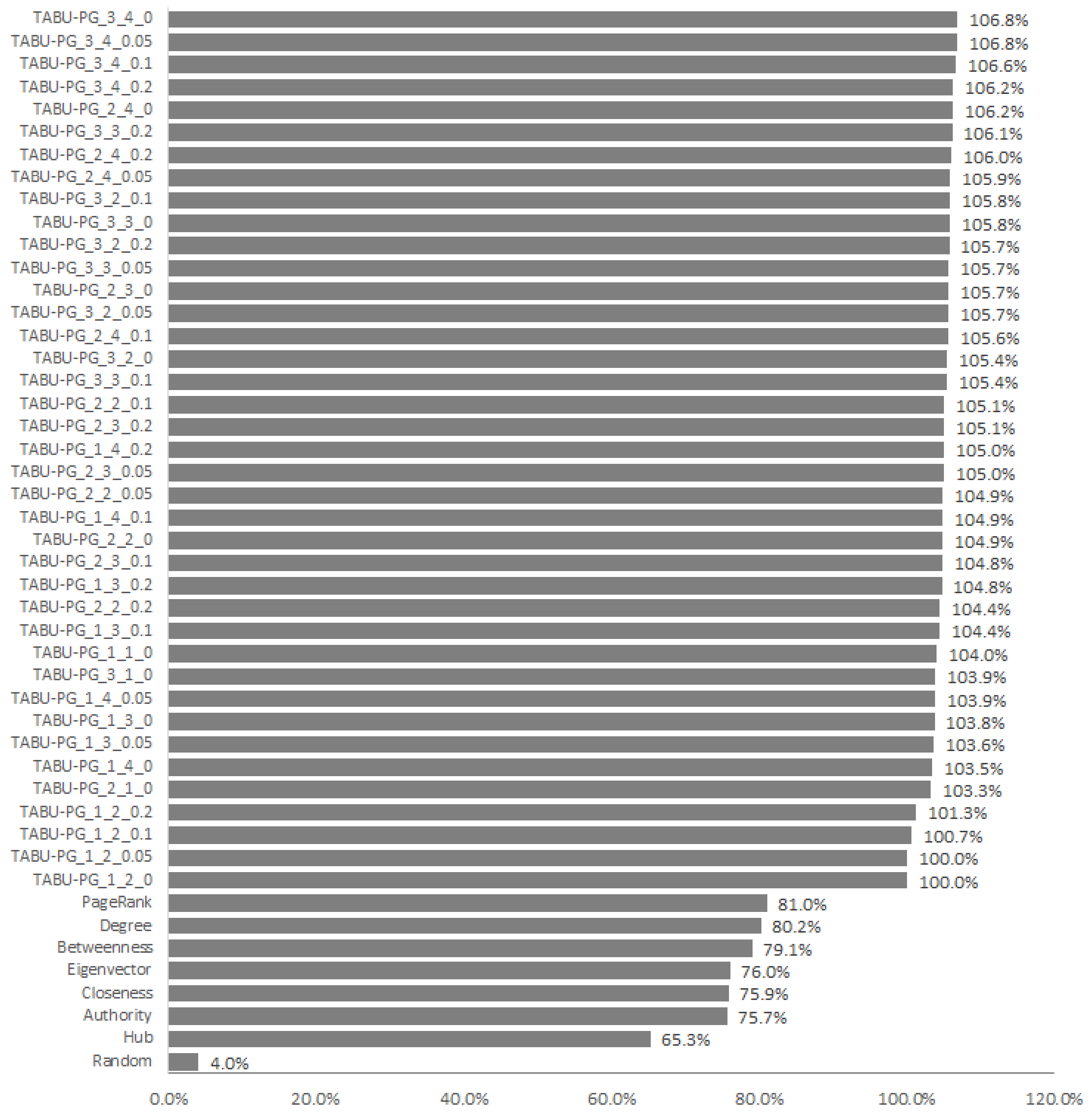

FIGURE 5.17: Comparison of All Heuristics in Experiment 5

Figure 5.17 displays a comparison of different parameter settings of TABU-PG along with benchmark heuristics, in terms of achieved influence spread (i.e., total profit obtained).

The results show that TABU-PG with different parameter settings perform 23% to 32% better than the best benchmark heuristic, that is pagerank heuristic in this case.

When different parameter settings of TABU-PG are compared between themselves, there is %7 increase in total profit, from the worst parameter setting to the best one.

Node Selection Method 1 (i.e., based on gain) performs usually worse than Method 1 and Method 2 (i.e., the efficiency based methods). Method 1 achieves an average performance of 103.1% while Method 2 achieves 105.2% and Method 3 achieves 105.8%. Among all, top 4 heuristics employ Method 3.

On average, Potential Gain Calculation Methods 1, 2, 3, and 4 achieve performances of 103.8%, 103.7%, 105%, and 105.6% respectively. Out of the best 10 heuristics, 7 employ Method 4.

Minimum Potential Gain Ratio of 0, 0.05, 0.1, and 0.2 achieve average performances of 104.7%, 104.6%, 104.8%, and 105% respectively.

Figure 5.18 shows detailed diffusion results with regard to budget for the selected heuristics: eigenvector, pagerank, the worst parameter setting TABU-PG_1_2_0, the median parameter setting TABU-PG_1_4_0.2, and the best parameter setting TABU-PG_3_4_0.

As can be seen in Figure 5.18, the influence spread does not contain any sudden jumps. Although all heuristics perform similar at beginning, TABU-PG heuristics distinguish themselves from benchmark heuristics as budget is spent.

Figure 5.19 shows the effects of different values of scaling parameters on the best performing heuristic TABU-PG_3_4_0. Increasing the value of $NumBefReCalc$ from 1 to 5, and to 25 improves the runtime from 13 seconds to 8 and 6 seconds respectively. As a trade-off, the total profit decreases by near 5% and 10% respectively.

On the other hand, limiting the candidate pool by setting $TopMltp$ to 20 and 10 reduces runtime from 13 seconds to 4 and 3 seconds respectively, without any loss in performance in terms of total profit.

Figure 5.20 shows the effects of different values of scaling parameters on the median performing heuristic TABU-PG_1_4_0.2. In general, the same conclusions for Figure 5.19 apply to this figure as well except that there are near 1% and 2% decrease in performance when setting $TopMltp$ to 20 and 10 respectively. However, near 5% and 10% lose of performance which is encountered when $NumBefReCalc$ is set to 5 and 25 are now reduced to approximately 1% and 7%.

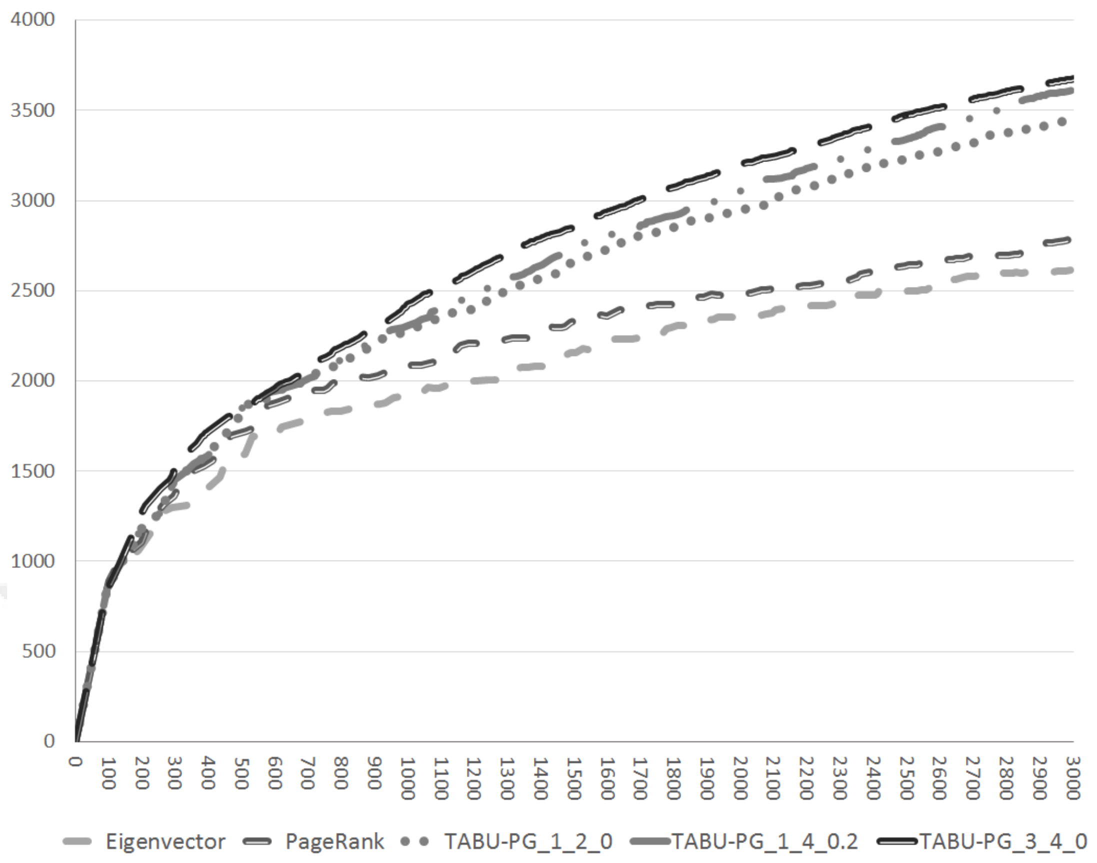

FIGURE 5.18: Detailed Diffusion of Selected Heuristics in Experiment 5

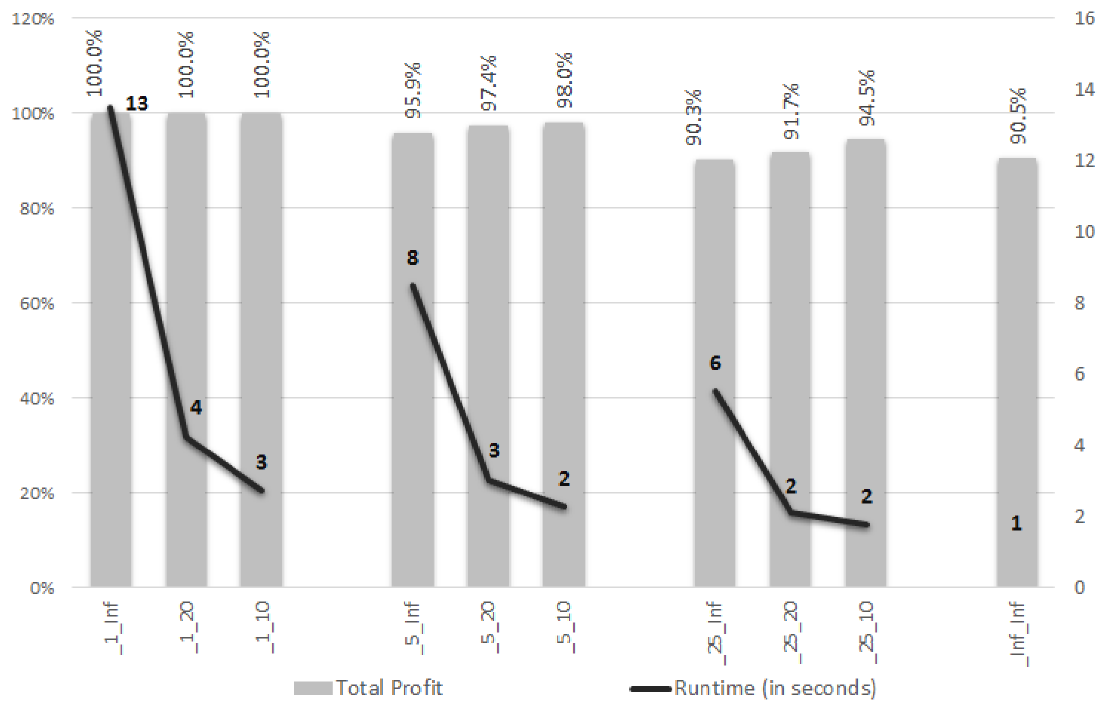

FIGURE 5.19: Scaling TABU-PG_3_4_0 in Experiment 5

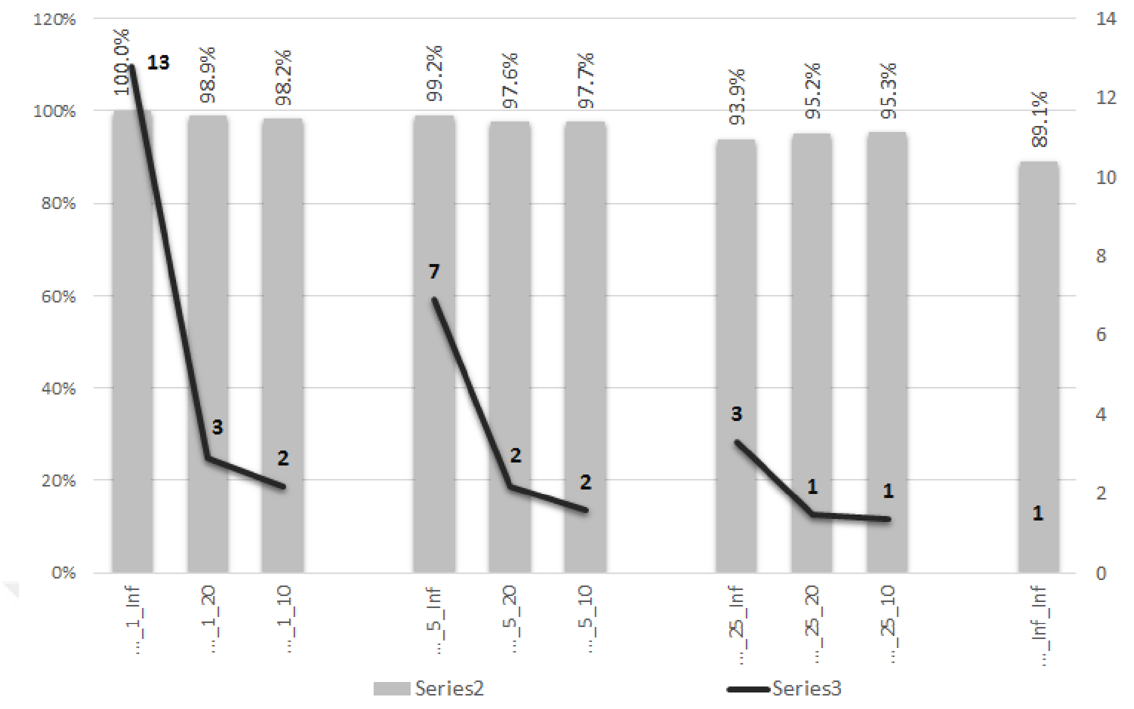

FIGURE 5.20: Scaling TABU-PG_1_4_0.2 in Experiment 5

## Experiment 6

Threshold values are assigned randomly by using the following truncated normal distribution: $tn(mean = 0.8,\ sd = 0.4,\ lower = 0.1,\ upper = Inf)$. Profit values and cost values are assigned by utilizing the methods which are explained in Section 5.2.3.

Influence weights are assigned by the hybrid model. $x$ being the average degree of nodes, for each node, a value from $\{x/3,\ x/2,\ x,\ 3x/2\}$ is selected with respective probabilities of $\{0.3,\ 0.3,\ 0.2,\ 0.2\}$. Then, geometric mean of the selected value and the value which would be resulted from ratio model is set as the influence weight of the node.

Figure 5.21 displays a comparison of different parameter settings of TABU-PG along with benchmark heuristics, in terms of achieved influence spread (i.e., total profit obtained). The results show that TABU-PG with different parameter settings perform 72% to 91% better than the best benchmark heuristic, that is degree heuristic in this case.

When different parameter settings of TABU-PG are compared between themselves, there is %11 increase in total profit, from the worst parameter setting to the best one.

Node Selection Method 1 (i.e., based on gain) performs usually worse than Method 1 and Method 2 (i.e., the efficiency based methods). Method 1 achieves an average performance of 102.8% while Method 2 achieves 108.5% and Method 3 achieves 108.8%. Among the top 10 heuristics, 6 employ Method 3 and 4 employ Method 2.

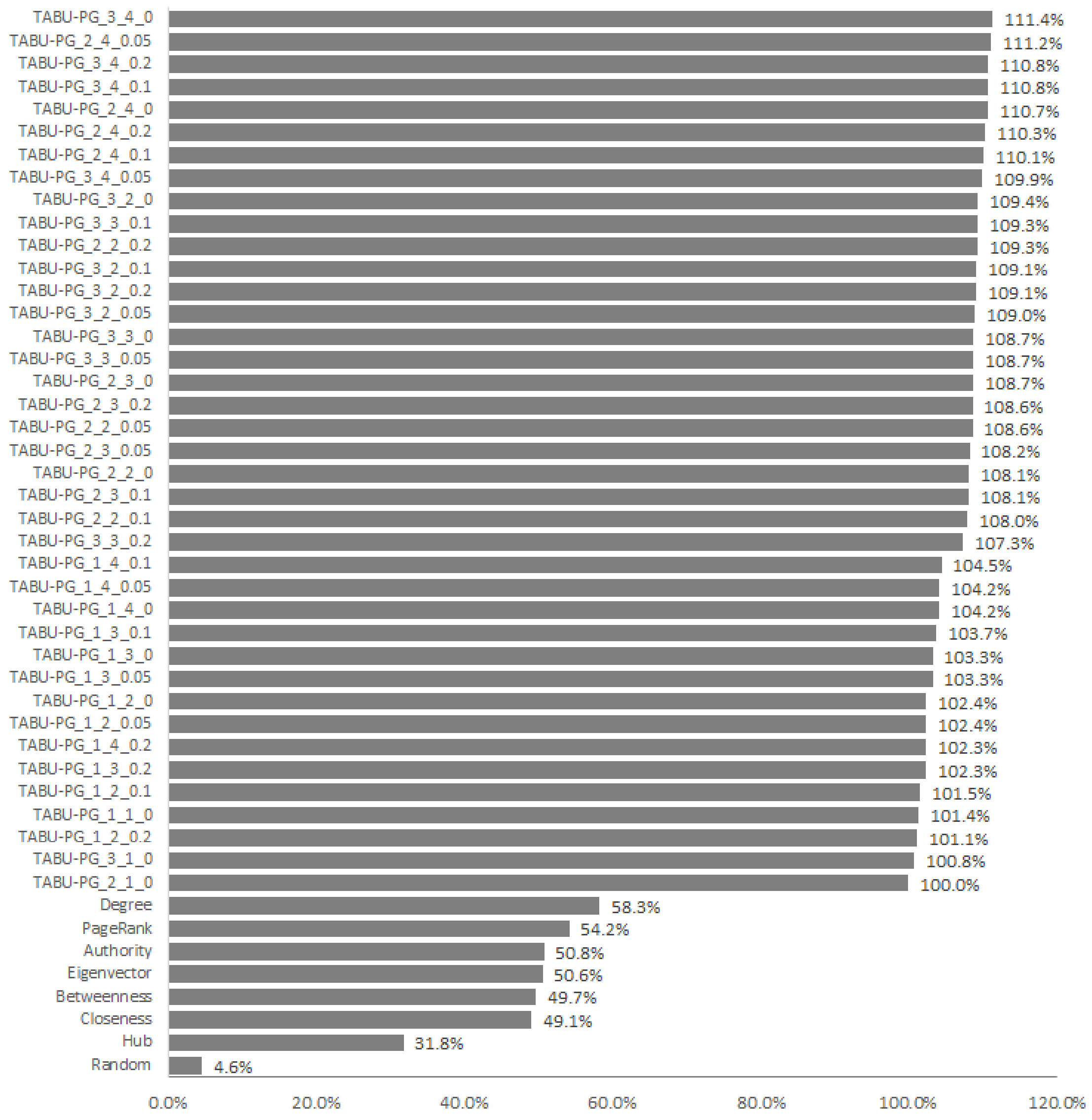

FIGURE 5.21: Comparison of All Heuristics in Experiment 6

On average, Potential Gain Calculation Methods 1, 2, 3, and 4 achieve performances of 100.7%, 106.5%, 106.7%, and 108.4% respectively. Among all, top 4 heuristics employ Method 4.

Minimum Potential Gain Ratio of 0, 0.05, 0.1, and 0.2 achieve average performances of 107.4%, 107.3%, 107.2%, and 106.8% respectively.

Figure 5.22 shows detailed diffusion results for the selected heuristics with regard to budget: authority, degree, the worst parameter setting TABU-PG_2_1_0, the median parameter setting TABU-PG_2_3_0.05, and the best parameter setting TABU-PG_3_4_0.

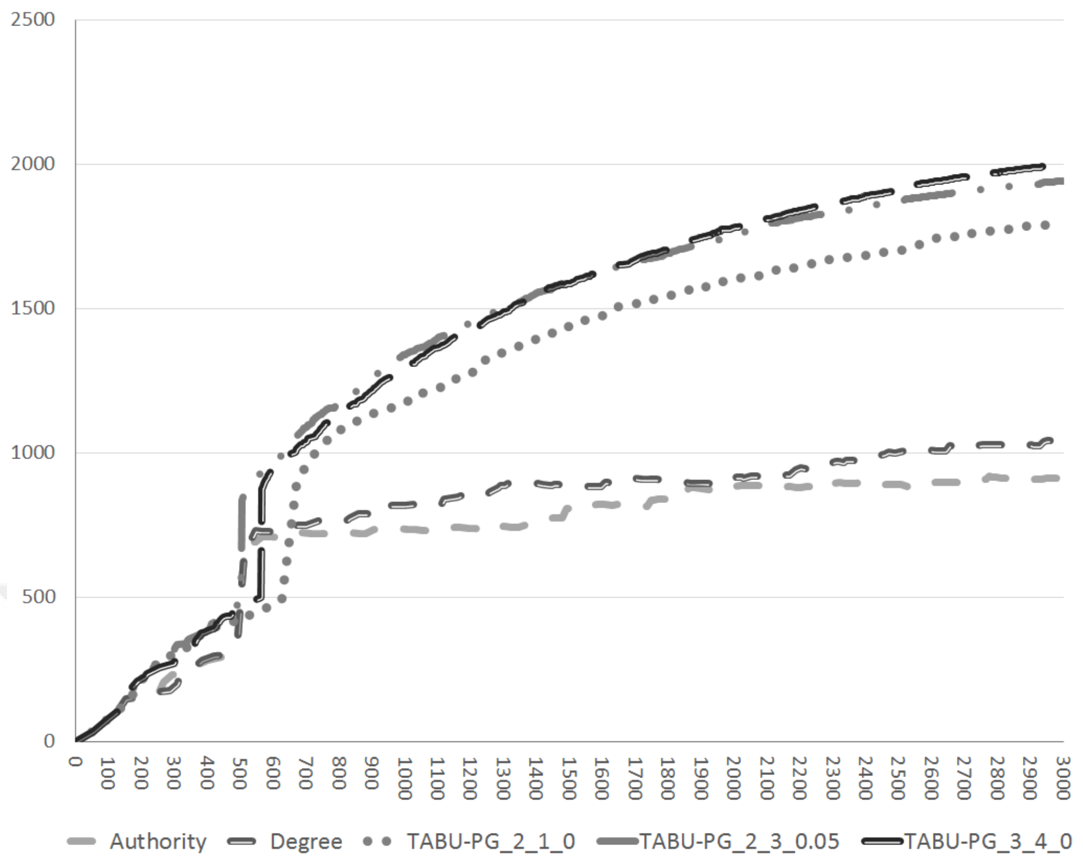

FIGURE 5.22: Detailed Diffusion of Selected Heuristics in Experiment 6

As can be seen in Figure 5.22, there exist tipping points. Although all heuristics perform similar at the beginning, some better algorithms reach tipping points before others. Moreover, after the sudden jump effect of tipping points fades away, TABU-PG heuristics continue to increase while benchmark heuristics almost stays the same.

Figure 5.23 shows the effects of different values of scaling parameters on the best performing heuristic TABU-PG_3_4_0. Increasing the value of $NumBefReCalc$ from 1 to 5, and to 25 improves the runtime from 11 seconds to 6 and 4 seconds respectively. As a trade-off, the total profit decreases by 5.5% and 11% respectively.

On the other hand, limiting the candidate pool by setting $TopMltp$ to 20 and 10 reduces runtime from 11 seconds to 3 and 23 seconds respectively, with performance loss of only 0.6% and 2.1% respectively. The counter-intuitive result in Experiment 1 is not valid here except for setting $TopMltp$ to 20 when $NumBefReCalc$ is 5.

Figure 5.24 shows the effects of different values of scaling parameters on the median performing heuristic TABU-PG_2_3_0.05. In general, the same conclusions on $NumBefRecalc$ parameter in Figure 5.23 apply to this figure. However, results on $TopMltp$ is usually

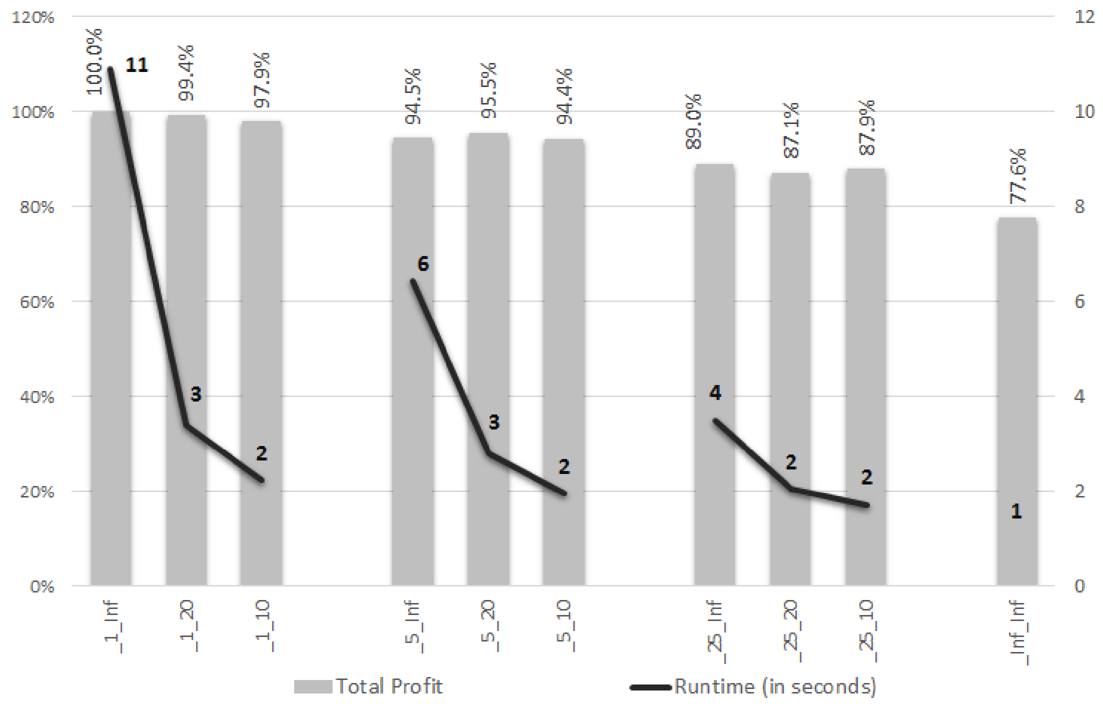

FIGURE 5.23: Scaling TABU-PG_3_4_0 in Experiment 6

inline with the counter-intuitive result in Experiment 1, that is limiting the candidate pool results in better performance.

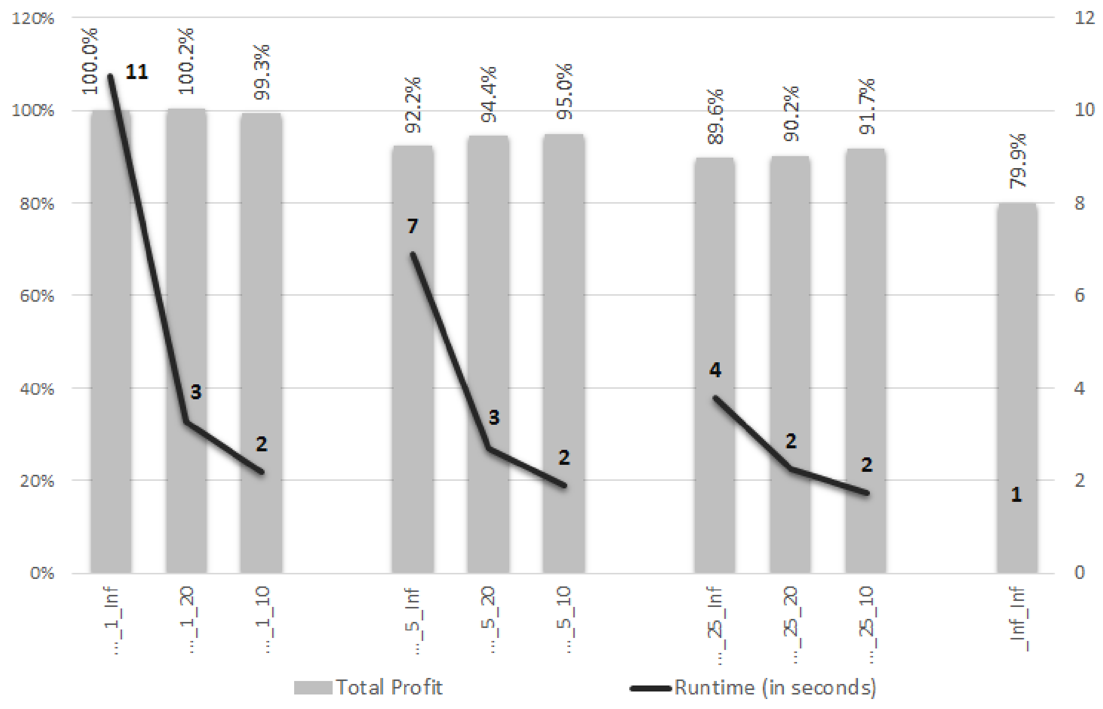

FIGURE 5.24: Scaling TABU-PG_2_3_0.05 in Experiment 6

### 5.2.4 Experiments on Pokec

In Pokec social network, not all users have specified their ages on their profile pages. To fill the missing age data, random integer values between 18 to 45 are assigned.

Profit values are assigned by targeting specific demographics with specified importance values. Females aged between 18 to 34 are assigned the profit value of 5, males aged between 18 to 24 are assigned the profit value of 3, males aged between 25 to 34 are assigned the profit value of 2, males aged between 35 to 44 are assigned the profit value of 1, and the rest is assigned the profit value of 0. The values are created based on an arbitrary hypothetical case where given demographics carry specified degrees of importance or potential profits for the marketer.

**Experiment 7**

Threshold values are assigned randomly by using the following truncated normal distribution: $tn(mean = 0.65,\ sd = 0.3,\ lower = 0.1,\ upper = Inf)$. Cost values are assigned by utilizing the methods which are explained in Section 5.1. Profit values are assigned by utilizing the method and values given in Section 5.2.4. Influence weights are assigned by the ratio model.

Figure 5.25 shows comparison of different parameter settings of TABU-PG along with benchmark heuristics, in terms of achieved influence spread (i.e., total profit obtained). Note that, there are 12 TABU-PG heuristics shown instead of 39 which is the case in experiments on other networks. This is because parameter settings are not multiplied for different values of $MinPGR$ for experiments on Pokec.

The results show that TABU-PG with different parameter settings perform 96% to 263% better than the best benchmark heuristic, that is pagerank heuristic in this case.

When different parameter settings of TABU-PG are compared between themselves, there is %85.2 increase in total profit, from the worst parameter setting to the best one.

Node Selection Method 1 (i.e., based on gain) consistently performs worse than Method 1 and Method 2 (i.e., the efficiency based methods). Method 1 achieves an average performance of 116.7% while Method 2 achieves 177.8% and Method 3 achieves 177.7%.

On average, Potential Gain Calculation Methods 1, 2, 3, and 4 achieve performances of 165.4%, 139.6%, 164.0%, and 160.8% respectively.

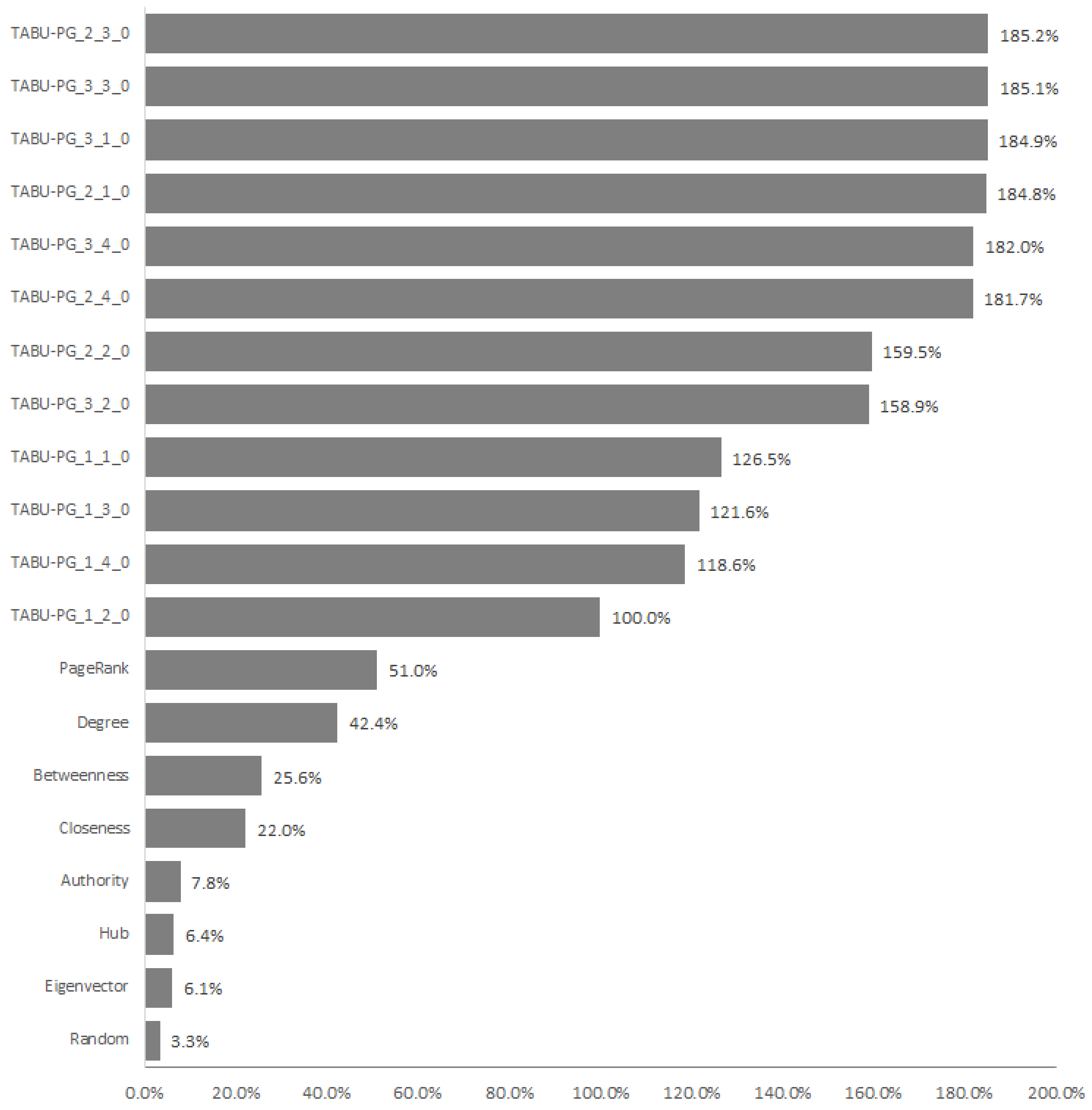

FIGURE 5.25: Comparison of All Heuristics in Experiment 7

Impact of $MinPGR$ is only inspected for the best performing heuristic TABU-PG_2_3_0. The findings are illustrated in Figure 5.26. Although very slight differences can be achieved by changing the parameter value, the changes are negligible for the most part.

Figure 5.27 shows detailed diffusion results for the selected heuristics with regard to budget: betweenness, pagerank, the worst parameter setting TABU-PG_1_2_0, the median parameter setting TABU-PG_2_2_0, the best parameter setting TABU-PG_2_3_0. The median parameter setting is not technically median but one of the two median parameter settings since the number of TABU-PG heuristics is even in this case.

As can be seen in Figure 5.27, total profit is more or less a linear function of the budget. The advantage of Partial Gain Calculation Method 3 over Method 2 is evidently shown. Both methods perform very similar at the beginning while Method 3 positively

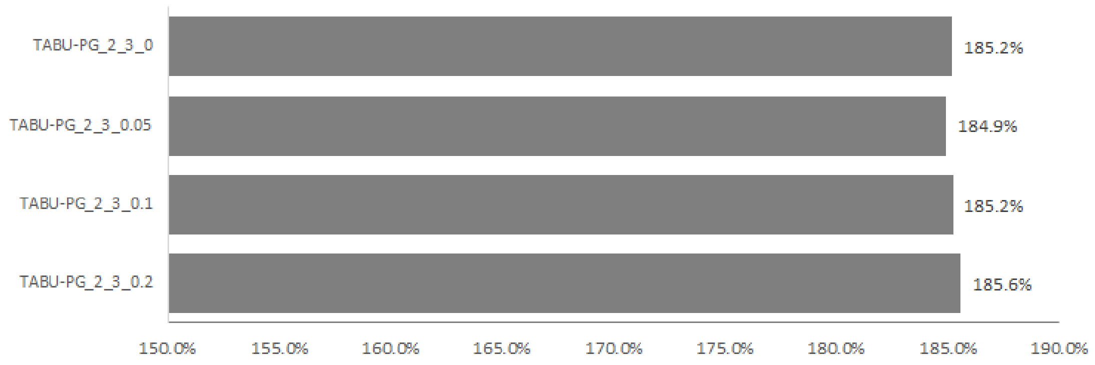

FIGURE 5.26: Impact of $MinPGR$ on TABU-PG_2_3 in Experiment 7

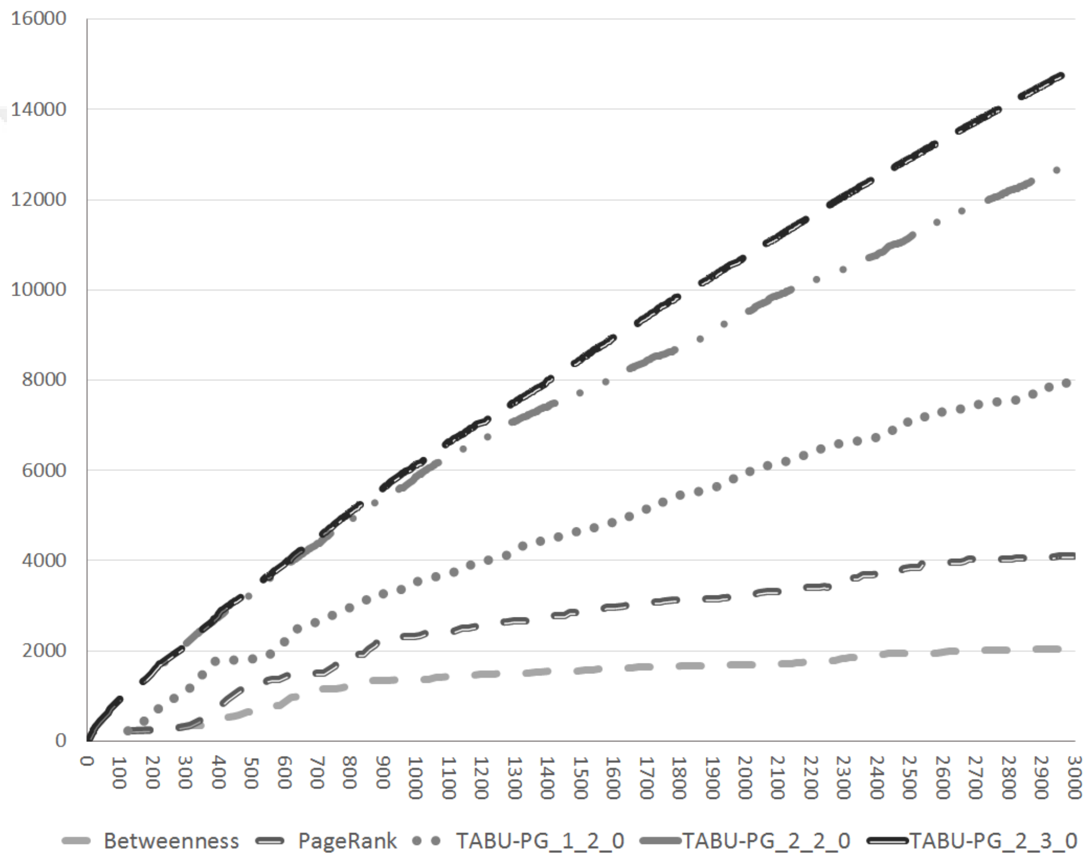

FIGURE 5.27: Detailed Diffusion of Selected Heuristics in Experiment 7

distinguishes itself from Method 2 as it gets closer to the end of budget. This is because Method 3 weighs less on potential gains and more on actual gains as it gets closer to the end.

Figure 5.28 shows the effects of different values of scaling parameters on the best performing heuristic, TABU-PG_2_3_0. Increasing the value of $NumBefReCalc$ from 1 to 5, and 25 improves the runtime from 11000 seconds to 8382, and 7427 seconds respectively. As a trade-off, the total profit decreases by 1.5%, and 3% respectively.

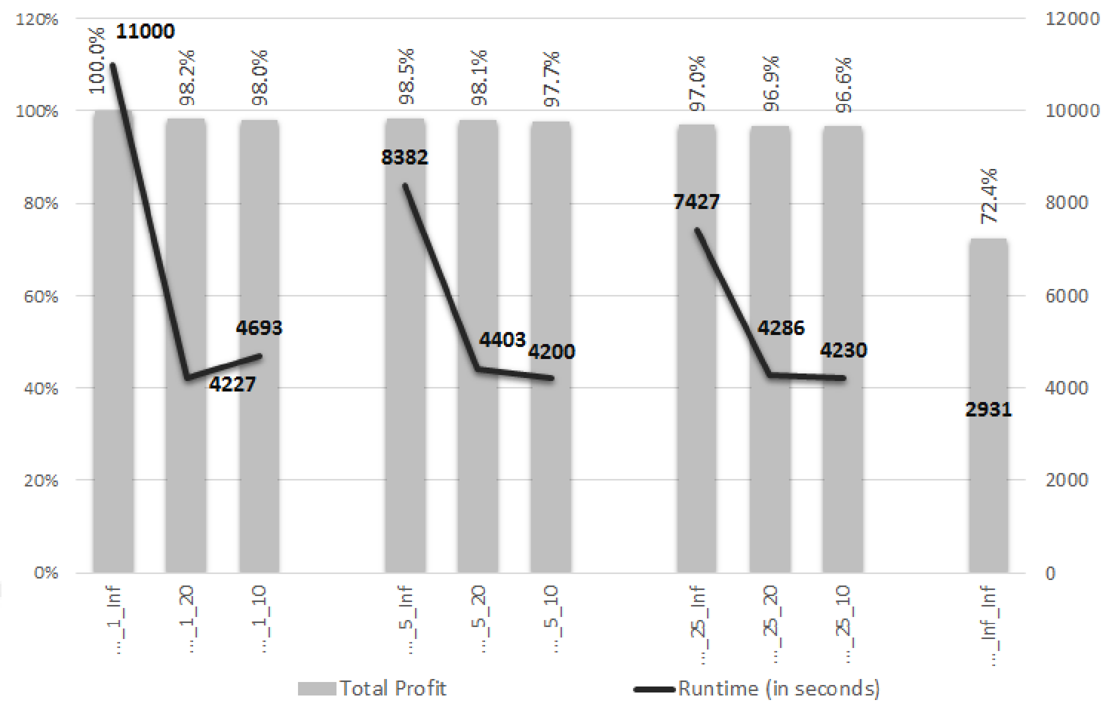

FIGURE 5.28: Scaling TABU-PG_2_3_0 in Experiment 7

Setting $TopMltp$ to 20 instead of $Inf$ reduces the runtime by approximately half while providing a spread performance only 1.8% worse than that of the original. The counter-intuitive result we encountered in Experiment 1 does not hold true in this case since limiting the seed node candidate pool to top nodes results in slight decreases in spread performances as intuitively expected.

Figure 5.29 shows the effects of different values of scaling parameters on TABU-PG_2_2_0. In general, the same conclusions for Figure 5.7 are valid for this figure as well.

## Experiment 8

Threshold values are assigned randomly by using the following truncated normal distribution: $tn(mean = 0.65,\ sd = 0.3,\ lower = 0.1,\ upper = Inf)$. Cost values are assigned by utilizing the methods which are explained in Section 5.1. Profit values are assigned as explained in Section 5.2.4.

Influence weights are assigned by the hybrid model explained in Section 4.4.1. $x$ being the average degree of nodes, for each node, a value from $\{x,\ 2x,\ 3x\}$ is selected with respective probabilities of $\{0.25,\ 0.5,\ 0.25\}$. Then, geometric mean of the selected value and the value which would be resulted from ratio model is set as the influence weight of the node.

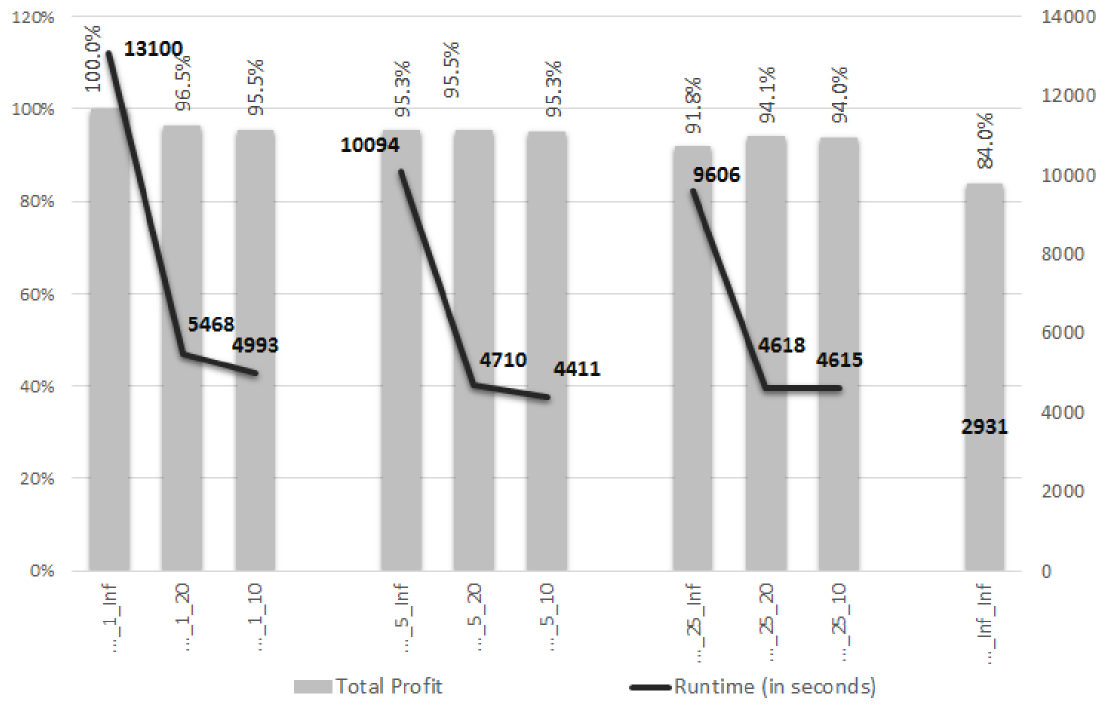

FIGURE 5.29: Scaling TABU-PG_2_2_0 in Experiment 7

Figure 5.30 shows comparison of different parameter settings of TABU-PG along with benchmark heuristics, in terms of achieved influence spread (i.e., total profit obtained). Note that, there are 12 TABU-PG heuristics shown instead of 39 which is the case in other experiments except for experiments on Pokec. This is because parameter settings are not multiplied for different values of *MinPGR* for experiments on Pokec. In addition, results are presented for heuristics which sets *TopMltp* to 10 due to the fact that original heuristics takes very long time without scaling parameters. For instance, TABU-PG_2_2_0 takes more than 11 hours.

The results are quite unusual when compared to other experiments. Some of the benchmarks perform better than some of TABU-PG heuristics. However, the best TABU-PG heuristic performs 313% better than the best benchmark heuristic, that is pagerank heuristic in this case. Since a network with tipping points is employed in this experiment, heuristics which cannot reach to a tipping point before others can suffer greatly in terms of performance when compared to other heuristics which are able to reach a tipping point earlier.

Figure 5.31 and 5.32 present detailed diffusion results for the selected heuristics: closeness, pagerank, the worst parameter setting TABU-PG_1_1_0_1_10, the median parameter setting TABU-PG_1_2_0_1_10 (this is not technically median but one of the two median parameter settings since the number of TABU-PG heuristics is even in this

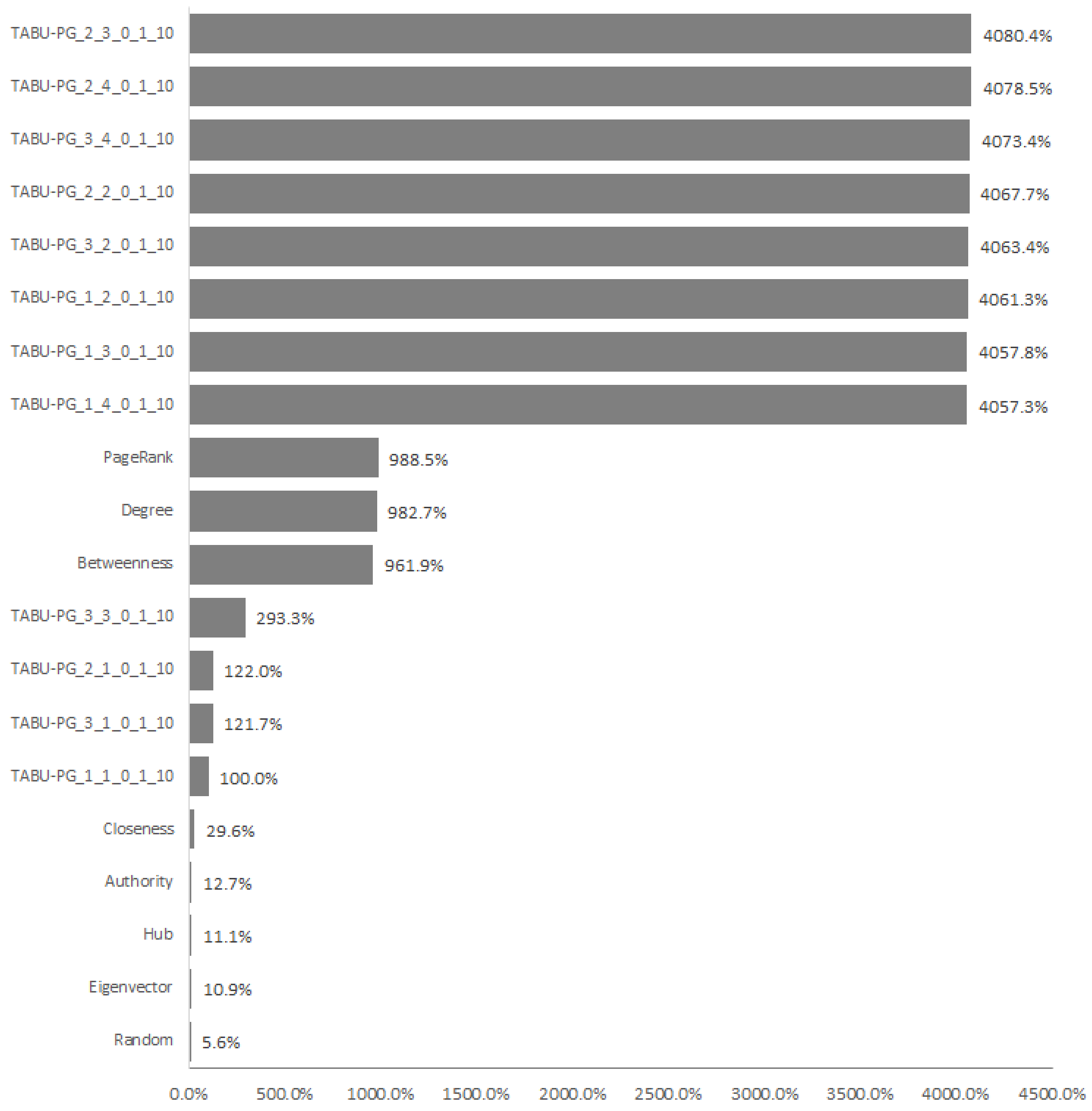

FIGURE 5.30: Comparison of All Heuristics in Experiment 8

case), and the best parameter setting TABU-PG_2_3_0_1_10. The latter figure is the same figure except that y-axis is limited by maximum value of 20, 000.

Figure 5.31 shows that very large differences in terms of performance are caused by sudden jumps after reaching tipping points. Note that, pagerank heuristic would not be able to reach its tipping point if the budget was only around 5% lower. Among the best performing two heuristics in the figure, TABU-PG_2_3_0_1_10 and TABU--PG_1_2_0_1_10 , one reaches tipping point clearly before the other although their final spread results are very similar.

Figure 5.32 shows that although TABU-PG_1_1_0_1_10 performs very similar to other TABU-PG heuristics at the beginning, it is impressively outperformed by them since it is not able to reach a tipping point inside the given budget.

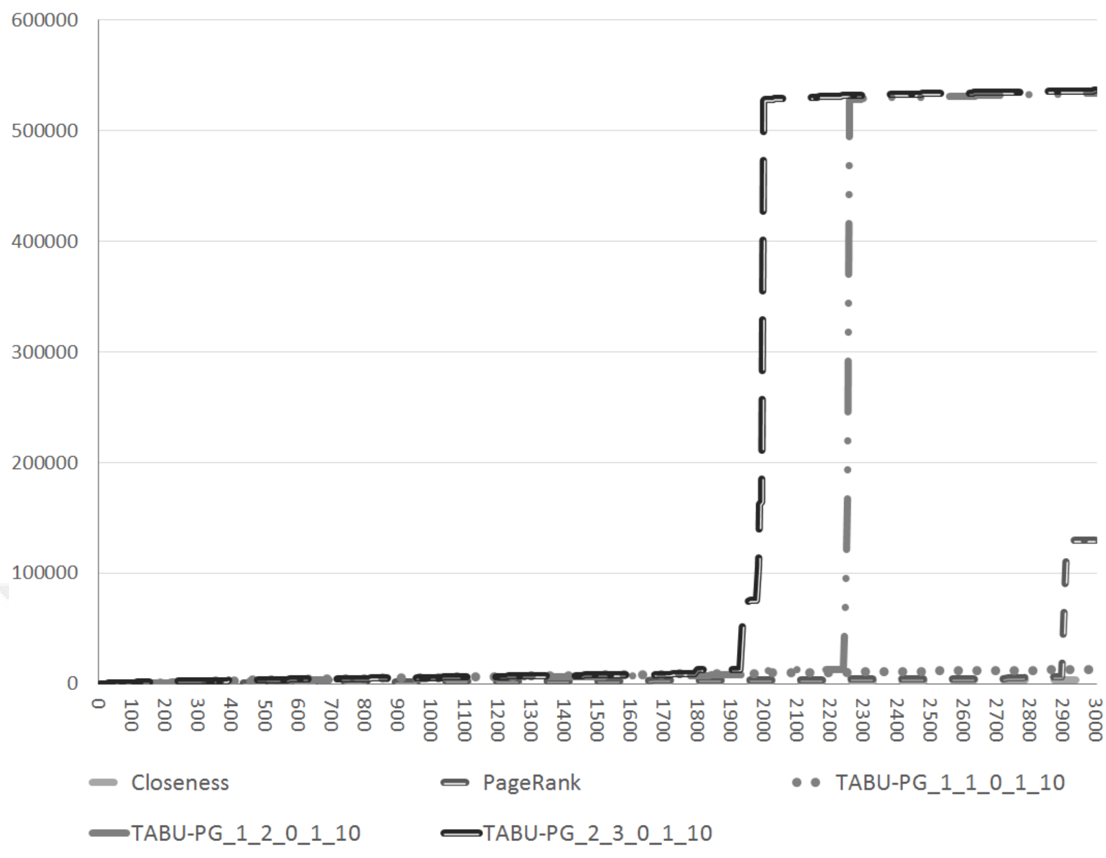

Figure 5.31: Detailed Diffusion of Selected Heuristics in Experiment 8

Figure 5.33 shows the effects of different values of scaling parameters on the best performing heuristic, TABU-PG_2_2_0. Increasing the value of $NumBefReCalc$ from 1 to 5, and 25 improves the runtime from $40,212$ seconds to $37,582$, and $22,412$ seconds respectively. As a trade-off, the total profit decreases by 1.6%, and 4.5% respectively.

Setting $TopMltp$ to 20 instead of $Inf$ reduces the runtime to approximately 11% of what it originally takes while providing a spread performance only 3.4% worse than that of the original, given that $NumBefReCalc$ is set to 1. The counter-intuitive result we encountered in Experiment 1 does not hold true in this case since limiting the seed node candidate pool to top nodes results in slight decreases in spread performances as intuitively expected. Overuse of scaling parameters results in very low performance in terms of total profit, as worse as less than 1% of the original in the case of ..._$Inf$_$Inf$. A similar result is also valid for ..._25_10. This is because extensively utilizing both methods of limiting the candidate pool and selecting greater number of nodes at once results in a heuristic which misses the tipping points. Therefore, such heuristics are not able to perform reasonably close to the default version.

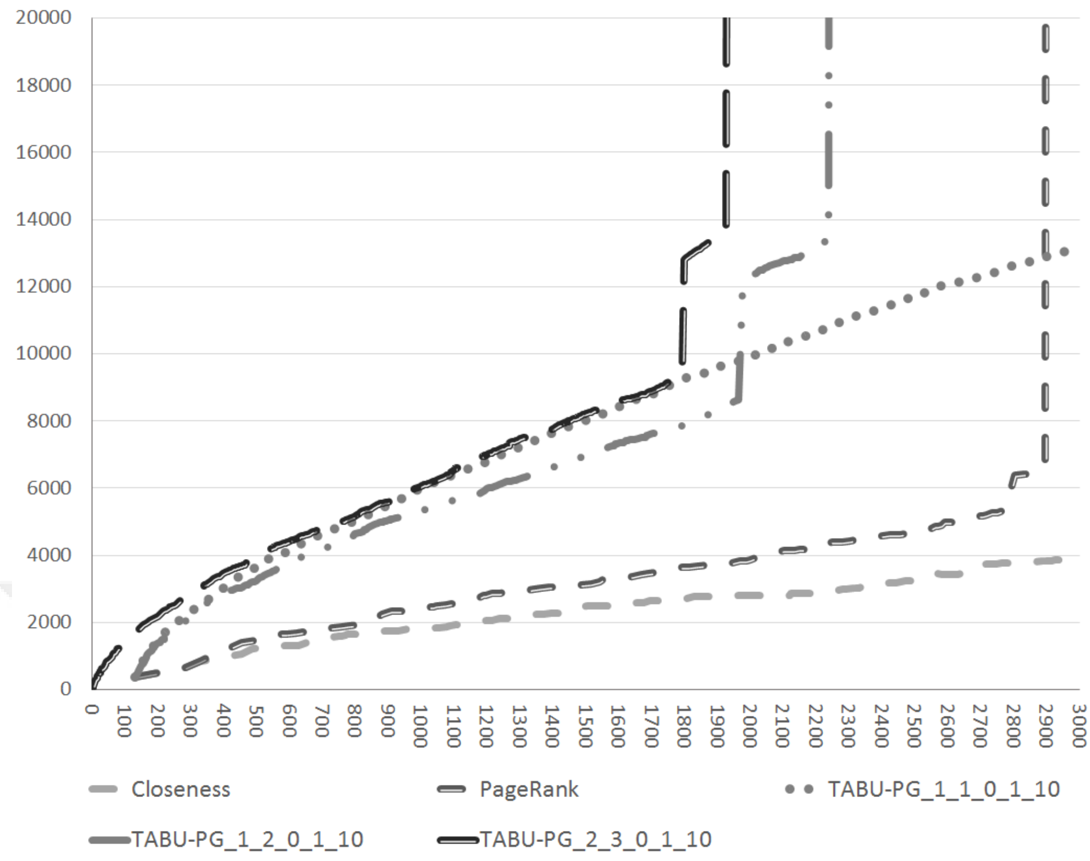

FIGURE 5.32: Zoom on Detailed Diffusion of Selected Heuristics in Experiment 8

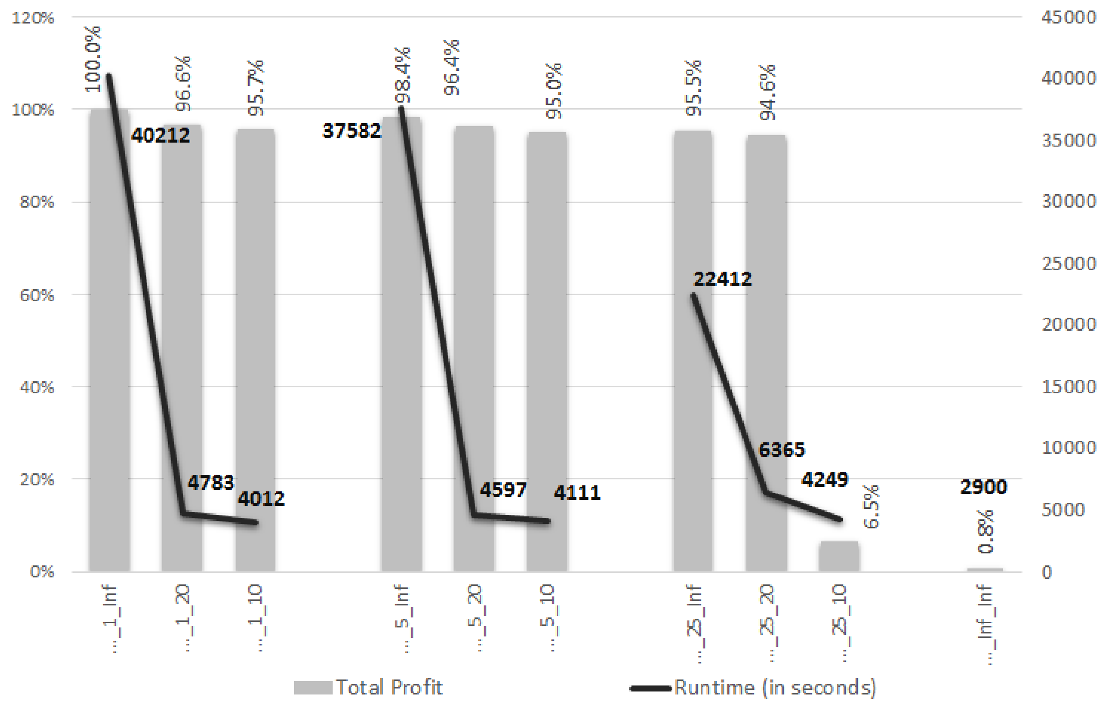

FIGURE 5.33: Scaling TABU-PG_2_2_0 in Experiment 8

## 5.3 Discussion

Performances of all heuristics is summarized in Table 5.3. Average performances ($\mu$) and standard deviation of performances ($\sigma$) are calculated without considering experiments on Pokec since not all heuristics are run on Pokec. The heuristics are sorted based on their average performances relative to the worst performing TABU-PG heuristic in the given experiment. Experiments are named with letter E suffixed by the experiment number.

Table 5.4 and 5.6 show performances of TABU-PG heuristics and benchmarks heuristics respectively. For each experiment, standard scores[6] (i.e., z-scores) of heuristics are calculated and presented. For each heuristic, its average performance ($\mu$) and standard deviation of its performances ($\sigma$) over given experiments are also calculated and presented, in terms of their standard scores. Table 5.4 does not include experiments on Pokec since only selected experiments are run on Pokec. In both tables, heuristics are sorted based on their average performance. In these tables and in all other tables, negative values are shown inside parenthesis.

As shown in Table 5.4, Node Selection Method 3 is utilized by 4 of the 5 top performing heuristics, followed by Method 2. When TABU-PG_3_1_0 and TABU-PG_3_2_0 are excluded, a heuristic which employs Method 1 never performs better than any heuristic which employs Method 2 or Method 3. On average, Method 1,2, and 3 obtain performances of $-1$, 0.41, and 0.59.

Potential Gain Calculation Method 4 is employed by top performing heuristics followed by Method 3. On average, Method 1,2,3 and 4 achieve performances of $-1.0$, $-0.32$, 0.21, and 0.36.

On average, $MinPGR$ values 0, 0.05, 0.1, and 0.2 achieves performances of 0.09, 0.07, 0.09, and 0.08. Therefore, setting Minimum Potential Gain Ratio to values other than 0 does not result in significant difference in performance. However, it results in a reduction in performance for heuristics which employ a combination of Node Selection Method 2 and 3, and Potential Gain Calculation Method 3 and 4. For many other heuristics, it provides better performance. This is most likely because Potential Gain Calculation Method 3 and 4 weigh lesser on potential gain as remaining budget gets smaller, therefore already accounting for potential gains which are not likely to be ever realized.

[6]Standard score is the deviation from the mean score of the group in units of standard deviation.

Table 5.3: Overall Comparison of All Heuristics

| Values as percentage (%) | **E1** | **E2** | **E3** | **E4** | **E5** | **E6** | **E7** | **E8*** | $\sigma$ | $\mu$ |
|---|---|---|---|---|---|---|---|---|---|---|
| **TABU-PG_3_4_0.1** | 116 | 110 | 123 | 180 | 107 | 111 | - | - | 25 | 124 |
| **TABU-PG_3_4_0** | 117 | 111 | 123 | 177 | 107 | 111 | 182 | 4073 | 24 | 124 |
| **TABU-PG_2_4_0** | 117 | 111 | 123 | 177 | 106 | 111 | 182 | 4079 | 24 | 124 |
| **TABU-PG_3_4_0.05** | 117 | 111 | 123 | 175 | 107 | 110 | - | - | 23 | 124 |
| **TABU-PG_3_4_0.2** | 116 | 110 | 123 | 175 | 106 | 111 | - | - | 24 | 123 |
| **TABU-PG_3_3_0** | 116 | 110 | 123 | 177 | 106 | 109 | 185 | 293 | 24 | 123 |
| **TABU-PG_3_3_0.05** | 115 | 110 | 123 | 176 | 106 | 109 | - | - | 24 | 123 |
| **TABU-PG_2_4_0.05** | 117 | 110 | 123 | 171 | 106 | 111 | - | - | 22 | 123 |
| **TABU-PG_2_3_0** | 115 | 110 | 123 | 176 | 106 | 109 | 185 | 4080 | 24 | 123 |
| **TABU-PG_3_3_0.2** | 115 | 110 | 123 | 174 | 106 | 107 | - | - | 24 | 123 |
| **TABU-PG_3_3_0.1** | 115 | 111 | 123 | 173 | 105 | 109 | - | - | 23 | 123 |
| **TABU-PG_2_3_0.1** | 115 | 110 | 123 | 175 | 105 | 108 | - | - | 24 | 123 |
| **TABU-PG_2_4_0.1** | 116 | 110 | 123 | 171 | 106 | 110 | - | - | 22 | 123 |
| **TABU-PG_2_4_0.2** | 116 | 110 | 123 | 170 | 106 | 110 | - | - | 22 | 123 |
| **TABU-PG_2_3_0.05** | 114 | 110 | 123 | 175 | 105 | 108 | - | - | 24 | 123 |
| **TABU-PG_3_2_0.2** | 116 | 106 | 119 | 179 | 106 | 109 | - | - | 26 | 123 |
| **TABU-PG_3_2_0** | 114 | 106 | 118 | 180 | 105 | 109 | 159 | 4063 | 26 | 122 |
| **TABU-PG_2_3_0.2** | 114 | 110 | 123 | 170 | 105 | 109 | - | - | 22 | 122 |
| **TABU-PG_3_2_0.1** | 117 | 106 | 119 | 175 | 106 | 109 | - | - | 24 | 122 |
| **TABU-PG_3_2_0.05** | 114 | 106 | 118 | 176 | 106 | 109 | - | - | 25 | 122 |
| **TABU-PG_2_2_0.2** | 114 | 107 | 119 | 174 | 104 | 109 | - | - | 24 | 121 |
| **TABU-PG_2_2_0.1** | 115 | 106 | 119 | 173 | 105 | 108 | - | - | 24 | 121 |
| **TABU-PG_2_2_0** | 112 | 106 | 118 | 174 | 105 | 108 | 159 | 4068 | 24 | 121 |
| **TABU-PG_2_2_0.05** | 115 | 106 | 118 | 169 | 105 | 109 | - | - | 22 | 120 |
| **TABU-PG_1_4_0.1** | 105 | 104 | 111 | 168 | 105 | 104 | - | - | 23 | 116 |
| **TABU-PG_1_4_0** | 105 | 104 | 110 | 167 | 103 | 104 | 119 | 4057 | 23 | 116 |
| **TABU-PG_1_3_0.1** | 104 | 105 | 111 | 164 | 104 | 104 | - | - | 22 | 115 |
| **TABU-PG_1_3_0.2** | 104 | 105 | 112 | 165 | 105 | 102 | - | - | 22 | 115 |
| **TABU-PG_1_3_0.05** | 106 | 106 | 111 | 163 | 104 | 103 | - | - | 21 | 115 |
| **TABU-PG_1_3_0** | 105 | 105 | 111 | 163 | 104 | 103 | 122 | 4058 | 21 | 115 |
| **TABU-PG_1_4_0.05** | 106 | 105 | 110 | 161 | 104 | 104 | - | - | 21 | 115 |
| **TABU-PG_1_1_0** | 105 | 106 | 112 | 160 | 104 | 101 | 126 | 100 | 20 | 115 |
| **TABU-PG_1_4_0.2** | 102 | 104 | 111 | 161 | 105 | 102 | - | - | 21 | 114 |
| **TABU-PG_3_1_0** | 111 | 100 | 123 | 135 | 104 | 101 | 185 | 122 | 13 | 113 |
| **TABU-PG_1_2_0.2** | 103 | 101 | 105 | 161 | 101 | 101 | - | - | 22 | 112 |
| **TABU-PG_1_2_0.05** | 101 | 100 | 102 | 163 | 100 | 102 | - | - | 23 | 112 |
| **TABU-PG_1_2_0** | 100 | 100 | 100 | 163 | 100 | 102 | 100 | 4061 | 23 | 111 |
| **TABU-PG_1_2_0.1** | 100 | 100 | 103 | 159 | 101 | 102 | - | - | 22 | 111 |
| **TABU-PG_2_1_0** | 111 | 102 | 123 | 100 | 103 | 100 | 185 | 122 | 8 | 107 |
| **Degree** | 74 | 84 | 48 | 97 | 80 | 58 | 42 | 983 | 16 | 74 |
| **PageRank** | 72 | 84 | 90 | 27 | 81 | 54 | 51 | 988 | 22 | 68 |
| **Closeness** | 72 | 84 | 22 | 88 | 76 | 49 | 22 | 30 | 23 | 65 |
| **Eigenvector** | 58 | 83 | 17 | 78 | 76 | 51 | 6 | 11 | 23 | 60 |
| **Authority** | 64 | 83 | 14 | 70 | 76 | 51 | 8 | 13 | 23 | 60 |
| **Betweenness** | 60 | 83 | 36 | 47 | 79 | 50 | 26 | 962 | 17 | 59 |
| **Hub** | 32 | 82 | 18 | 59 | 65 | 32 | 6 | 11 | 22 | 48 |
| **Random** | 8 | 1 | 3 | 1 | 4 | 5 | 3 | 6 | 3 | 4 |

*For Experiment 8, results are reported for heuristics with $TopMltp$ as 10 instead of $Inf$.

TABLE 5.4: Overall Comparison of TABU-PG Heuristics

| | **E1** | **E2** | **E3** | **E4** | **E5** | **E6** | $\sigma$ | $\mu$ |
|---|---|---|---|---|---|---|---|---|
| **TABU-PG_3_4_0** | 0.95 | 1.10 | 0.81 | 0.65 | 1.25 | 1.34 | 0.24 | 1.02 |
| **TABU-PG_3_4_0.1** | 0.90 | 1.08 | 0.86 | 0.87 | 1.15 | 1.16 | 0.13 | 1.00 |
| **TABU-PG_2_4_0** | 1.01 | 1.11 | 0.80 | 0.65 | 0.92 | 1.14 | 0.17 | 0.94 |
| **TABU-PG_3_4_0.05** | 0.99 | 1.10 | 0.80 | 0.50 | 1.24 | 0.90 | 0.23 | 0.92 |
| **TABU-PG_3_4_0.2** | 0.77 | 0.90 | 0.86 | 0.51 | 0.93 | 1.17 | 0.20 | 0.86 |
| **TABU-PG_2_4_0.05** | 1.04 | 1.02 | 0.81 | 0.23 | 0.71 | 1.27 | 0.33 | 0.85 |
| **TABU-PG_2_4_0.2** | 0.82 | 0.98 | 0.86 | 0.15 | 0.79 | 1.04 | 0.29 | 0.77 |
| **TABU-PG_3_3_0** | 0.84 | 1.09 | 0.83 | 0.64 | 0.66 | 0.58 | 0.17 | 0.77 |
| **TABU-PG_2_4_0.1** | 0.83 | 1.05 | 0.86 | 0.21 | 0.56 | 0.97 | 0.29 | 0.75 |
| **TABU-PG_3_3_0.05** | 0.74 | 1.04 | 0.84 | 0.57 | 0.60 | 0.58 | 0.17 | 0.73 |
| **TABU-PG_2_3_0** | 0.66 | 0.95 | 0.83 | 0.59 | 0.59 | 0.56 | 0.14 | 0.70 |
| **TABU-PG_3_3_0.1** | 0.62 | 1.14 | 0.84 | 0.39 | 0.41 | 0.74 | 0.26 | 0.69 |
| **TABU-PG_3_3_0.2** | 0.76 | 0.84 | 0.88 | 0.48 | 0.87 | 0.18 | 0.26 | 0.67 |
| **TABU-PG_2_3_0.1** | 0.69 | 1.02 | 0.84 | 0.52 | 0.06 | 0.40 | 0.31 | 0.59 |
| **TABU-PG_2_3_0.05** | 0.57 | 0.97 | 0.84 | 0.49 | 0.17 | 0.44 | 0.26 | 0.58 |
| **TABU-PG_2_3_0.2** | 0.59 | 0.99 | 0.88 | 0.19 | 0.23 | 0.55 | 0.30 | 0.57 |
| **TABU-PG_3_2_0.2** | 0.80 | (0.08) | 0.31 | 0.81 | 0.62 | 0.67 | 0.32 | 0.52 |
| **TABU-PG_3_2_0.1** | 0.98 | (0.10) | 0.19 | 0.56 | 0.68 | 0.69 | 0.36 | 0.50 |
| **TABU-PG_3_2_0** | 0.54 | (0.08) | 0.11 | 0.90 | 0.45 | 0.76 | 0.34 | 0.45 |
| **TABU-PG_3_2_0.05** | 0.55 | (0.12) | 0.11 | 0.59 | 0.57 | 0.65 | 0.29 | 0.39 |
| **TABU-PG_2_2_0.2** | 0.56 | (0.02) | 0.31 | 0.46 | (0.14) | 0.74 | 0.31 | 0.31 |
| **TABU-PG_2_2_0.1** | 0.61 | (0.14) | 0.20 | 0.35 | 0.26 | 0.38 | 0.22 | 0.28 |
| **TABU-PG_2_2_0.05** | 0.64 | (0.11) | 0.12 | 0.10 | 0.15 | 0.55 | 0.27 | 0.24 |
| **TABU-PG_2_2_0** | 0.08 | (0.10) | 0.12 | 0.46 | 0.11 | 0.41 | 0.20 | 0.18 |
| **TABU-PG_1_4_0.1** | (1.18) | (0.84) | (0.96) | 0.01 | 0.12 | (0.63) | 0.49 | (0.58) |
| **TABU-PG_1_3_0.1** | (1.23) | (0.51) | (0.86) | (0.26) | (0.14) | (0.84) | 0.38 | (0.64) |
| **TABU-PG_1_3_0.2** | (1.34) | (0.56) | (0.77) | (0.19) | 0.05 | (1.25) | 0.51 | (0.68) |
| **TABU-PG_1_3_0.05** | (0.94) | (0.32) | (0.86) | (0.37) | (0.65) | (0.97) | 0.26 | (0.69) |
| **TABU-PG_1_3_0** | (1.03) | (0.54) | (0.86) | (0.37) | (0.55) | (0.97) | 0.24 | (0.72) |
| **TABU-PG_1_4_0.05** | (0.99) | (0.60) | (1.09) | (0.49) | (0.47) | (0.70) | 0.24 | (0.73) |
| **TABU-PG_1_4_0** | (1.09) | (0.74) | (1.09) | (0.03) | (0.71) | (0.70) | 0.35 | (0.73) |
| **TABU-PG_1_1_0** | (1.01) | (0.21) | (0.75) | (0.60) | (0.40) | (1.50) | 0.42 | (0.74) |
| **TABU-PG_1_4_0.2** | (1.59) | (0.72) | (0.86) | (0.51) | 0.18 | (1.25) | 0.56 | (0.79) |
| **TABU-PG_3_1_0** | 0.01 | (1.79) | 0.87 | (2.36) | (0.46) | (1.66) | 1.13 | (0.90) |
| **TABU-PG_2_1_0** | (0.03) | (1.42) | 0.87 | (4.94) | (0.81) | (1.90) | 1.83 | (1.37) |
| **TABU-PG_1_2_0.2** | (1.45) | (1.73) | (1.73) | (0.46) | (2.00) | (1.57) | 0.49 | (1.49) |
| **TABU-PG_1_2_0.05** | (1.79) | (1.81) | (2.22) | (0.32) | (2.77) | (1.23) | 0.77 | (1.69) |
| **TABU-PG_1_2_0.1** | (1.89) | (1.90) | (2.08) | (0.64) | (2.40) | (1.47) | 0.56 | (1.73) |
| **TABU-PG_1_2_0** | (1.98) | (1.93) | (2.49) | (0.34) | (2.79) | (1.23) | 0.81 | (1.79) |

On the other hand, for individual heuristics in individual experiments, tuning $MinPGR$ can also result in better performance. For instance, TABU-PG_3_4_0.1 obtains standard score of 0.87 while TABU-PG_3_4_0 obtains 0.65 in Experiment 4. Note that TABU-PG_3_4_0.1 is the second best performing heuristic for this experiment.

Table 5.5 shows the rankings of places of 39 TABU-PG heuristic for each experiment, along with mean and standard deviation values. Except for TABU-PG_3_1_0 and TABU-PG_3_2_0 which performed unusually well in Experiment 3, standard deviations are all lower than 8.5 with a mean of 4.5 and median of 4.2. It suggests that performance of a heuristic over different experiments in different datasets are similar in general.

When benchmarks are compared among themselves, degree and pagerank heuristics are the best two performing heuristics on average. The best performing heuristic in each experiment is either degree heuristic or pagerank heuristic except for Experiment 2 where the result of degree heuristics is only very slightly less than the best benchmark heuristic.

Table 5.7 illustrates impacts of scaling parameters over different heuristics in different experiments in terms of spread performance. The first column specifies the experiment and heuristic. For instance, 4xM is the median TABU-PG heuristic in Experiment 4 and 5xB is the best TABU-PG heuristic in Experiment 5. Scaling parameter results for all heuristics which are summarized in this table are previously given in their respective sections. Since, only one heuristic is investigated for Experiment 8 and that heuristic is neither the best nor the median performing heuristic, it is specified as such in this table. $Inf$ is further shortened as I. For example, I_I represents $Inf\_Inf$ for scaling parameters. Heuristics with the default values for scaling parameters are set as 100% while others are assigned values based on their relative performances.

It can be depicted from Table 5.7 that the counter-intuitive result of improved result when candidate pool is limited usually holds true when $NumBefReCalc$ is larger and does not hold true when $NumBefReCalc$ is smaller. However, exceptions to this make it more appropriate to present this as an initial observation rather than a strong fact.

An explanatory idea for why this is the case could be as follows. When large number of nodes are selected at once, there is almost always a decrease in spread performance. As the value of $NumBefReCalc$ grows, the decrease in spread performance becomes larger. When candidate pool is limited by setting $TopMltp$ to values which are small enough but not too small, the decrease in performance due to the larger value of $NumBefReCalc$ is limited.

Table 5.8 illustrates impacts of scaling parameters over different heuristics in different experiments in terms of runtime. The columns and rows are same as Table 5.7, but displaying runtimes instead of total profit. The results show that the runtimes can be

TABLE 5.5: Rankings of TABU-PG Heuristics

| | **E1** | **E2** | **E3** | **E4** | **E5** | **E6** | $\sigma$ | $\mu$ |
|---|---|---|---|---|---|---|---|---|
| **TABU-PG_3_4_0.1** | 6 | 6 | 7 | 2 | 3 | 4 | 1.8 | 4.7 |
| **TABU-PG_3_4_0** | 5 | 4 | 15 | 4 | 1 | 1 | 4.7 | 5.0 |
| **TABU-PG_2_4_0** | 2 | 2 | 18 | 5 | 5 | 5 | 5.5 | 6.2 |
| **TABU-PG_3_4_0.05** | 3 | 3 | 17 | 13 | 2 | 8 | 5.6 | 7.7 |
| **TABU-PG_3_4_0.2** | 11 | 15 | 5 | 12 | 4 | 3 | 4.5 | 8.3 |
| **TABU-PG_2_4_0.05** | 1 | 9 | 16 | 20 | 8 | 2 | 6.9 | 9.3 |
| **TABU-PG_3_3_0** | 7 | 5 | 14 | 6 | 10 | 15 | 3.9 | 9.5 |
| **TABU-PG_2_4_0.2** | 9 | 12 | 6 | 23 | 7 | 6 | 6.0 | 10.5 |
| **TABU-PG_2_4_0.1** | 8 | 7 | 8 | 21 | 15 | 7 | 5.3 | 11.0 |
| **TABU-PG_3_3_0.05** | 13 | 8 | 10 | 9 | 12 | 16 | 2.7 | 11.3 |
| **TABU-PG_3_3_0.2** | 12 | 16 | 1 | 15 | 6 | 24 | 7.4 | 12.3 |
| **TABU-PG_3_3_0.1** | 17 | 1 | 12 | 18 | 17 | 10 | 5.9 | 12.5 |
| **TABU-PG_3_2_0.2** | 10 | 19 | 20 | 3 | 11 | 13 | 5.7 | 12.7 |
| **TABU-PG_3_2_0.1** | 4 | 21 | 22 | 10 | 9 | 12 | 6.5 | 13.0 |
| **TABU-PG_2_3_0** | 15 | 14 | 13 | 7 | 13 | 17 | 3.1 | 13.2 |
| **TABU-PG_2_3_0.1** | 14 | 10 | 9 | 11 | 25 | 22 | 6.1 | 15.2 |
| **TABU-PG_2_3_0.2** | 19 | 11 | 2 | 22 | 19 | 18 | 6.8 | 15.2 |
| **TABU-PG_3_2_0** | 23 | 18 | 26 | 1 | 16 | 9 | 8.4 | 15.5 |
| **TABU-PG_2_3_0.05** | 20 | 13 | 11 | 14 | 21 | 20 | 3.9 | 16.5 |
| **TABU-PG_3_2_0.05** | 22 | 23 | 25 | 8 | 14 | 14 | 6.1 | 17.7 |
| **TABU-PG_2_2_0.2** | 21 | 17 | 19 | 17 | 27 | 11 | 4.8 | 18.7 |
| **TABU-PG_2_2_0.1** | 18 | 24 | 21 | 19 | 18 | 23 | 2.4 | 20.5 |
| **TABU-PG_2_2_0.05** | 16 | 22 | 23 | 24 | 22 | 19 | 2.7 | 21.0 |
| **TABU-PG_2_2_0** | 24 | 20 | 24 | 16 | 24 | 21 | 2.9 | 21.5 |
| **TABU-PG_3_1_0** | 25 | 36 | 3 | 38 | 30 | 38 | 12.3 | 28.3 |
| **TABU-PG_1_4_0.1** | 32 | 33 | 33 | 25 | 23 | 25 | 4.2 | 28.5 |
| **TABU-PG_1_3_0.1** | 33 | 27 | 30 | 28 | 28 | 28 | 2.0 | 29.0 |
| **TABU-PG_1_3_0.05** | 27 | 26 | 29 | 31 | 33 | 30 | 2.4 | 29.3 |
| **TABU-PG_2_1_0** | 26 | 34 | 4 | 39 | 35 | 39 | 12.2 | 29.5 |
| **TABU-PG_1_3_0.2** | 34 | 29 | 28 | 27 | 26 | 34 | 3.2 | 29.7 |
| **TABU-PG_1_1_0** | 29 | 25 | 27 | 36 | 29 | 36 | 4.2 | 30.3 |
| **TABU-PG_1_3_0** | 30 | 28 | 31 | 32 | 32 | 29 | 1.5 | 30.3 |
| **TABU-PG_1_4_0.05** | 28 | 30 | 34 | 34 | 31 | 26 | 2.9 | 30.5 |
| **TABU-PG_1_4_0** | 31 | 32 | 35 | 26 | 34 | 27 | 3.3 | 30.8 |
| **TABU-PG_1_4_0.2** | 36 | 31 | 32 | 35 | 20 | 33 | 5.3 | 31.2 |
| **TABU-PG_1_2_0.05** | 37 | 37 | 38 | 29 | 38 | 31 | 3.6 | 35.0 |
| **TABU-PG_1_2_0.2** | 35 | 35 | 36 | 33 | 36 | 37 | 1.2 | 35.3 |
| **TABU-PG_1_2_0** | 39 | 39 | 39 | 30 | 39 | 32 | 3.8 | 36.3 |
| **TABU-PG_1_2_0.1** | 38 | 38 | 37 | 37 | 37 | 35 | 1.0 | 37.0 |

Table 5.6: Overall Comparison of Benchmark Heuristics

| | **E1** | **E2** | **E3** | **E4** | **E5** | **E6** | **E7** | **E8** | $\sigma$ | $\mu$ |
|---|---|---|---|---|---|---|---|---|---|---|
| **Deg.** | 0.88 | 0.40 | 0.67 | 1.28 | 0.54 | 0.89 | 1.29 | 1.30 | 0.33 | 0.91 |
| **Pgr.** | 0.78 | 0.40 | 2.28 | (1.04) | 0.57 | 0.64 | 1.79 | 1.31 | 0.94 | 0.84 |
| **Btw.** | 0.25 | 0.38 | 0.21 | (0.37) | 0.49 | 0.37 | 0.30 | 1.26 | 0.42 | 0.36 |
| **Cls.** | 0.76 | 0.40 | (0.36) | 0.98 | 0.36 | 0.33 | 0.08 | (0.74) | 0.53 | 0.23 |
| **Aut.** | 0.43 | 0.37 | (0.67) | 0.38 | 0.35 | 0.44 | (0.75) | (0.78) | 0.55 | (0.03) |
| **Eig.** | 0.12 | 0.37 | (0.54) | 0.65 | 0.37 | 0.42 | (0.85) | (0.78) | 0.56 | (0.03) |
| **Hub** | (1.08) | 0.34 | (0.49) | 0.01 | (0.08) | (0.71) | (0.84) | (0.78) | 0.46 | (0.45) |
| **Rnd.** | (2.15) | (2.65) | (1.10) | (1.90) | (2.60) | (2.38) | (1.02) | (0.79) | 0.70 | (1.82) |

Table 5.7: Overall Impact of Scaling Parameters on Performance

| **ExH** | **1_I** | **1_20** | **1_10** | **5_I** | **5_20** | **5_10** | **25_I** | **25_20** | **25_10** | **I_I** |
|---|---|---|---|---|---|---|---|---|---|---|
| **1xM** | 100% | 102% | 103% | 94% | 100% | 101% | 87% | 92% | 100% | 94% |
| **1xB** | 100% | 100% | 100% | 93% | 98% | 99% | 83% | 95% | 94% | 92% |
| **2xM** | 100% | 98% | 97% | 98% | 97% | 97% | 95% | 96% | 96% | 92% |
| **2xB** | 100% | 99% | 97% | 98% | 98% | 97% | 93% | 97% | 96% | 87% |
| **3xM** | 100% | 100% | 100% | 99% | 99% | 99% | 99% | 99% | 99% | 99% |
| **3xB** | 100% | 100% | 100% | 100% | 100% | 100% | 100% | 100% | 100% | 96% |
| **4xM** | 100% | 97% | 93% | 91% | 91% | 91% | 70% | 85% | 85% | 17% |
| **4xB** | 100% | 94% | 86% | 84% | 89% | 85% | 70% | 77% | 78% | 16% |
| **5xM** | 100% | 99% | 98% | 99% | 98% | 98% | 94% | 95% | 95% | 89% |
| **5xB** | 100% | 100% | 100% | 96% | 97% | 98% | 90% | 92% | 94% | 90% |
| **6xM** | 100% | 100% | 99% | 92% | 94% | 95% | 90% | 90% | 92% | 80% |
| **6xB** | 100% | 99% | 98% | 94% | 96% | 94% | 89% | 87% | 88% | 78% |
| **7xM** | 100% | 96% | 95% | 95% | 95% | 95% | 92% | 94% | 94% | 84% |
| **7xB** | 100% | 98% | 98% | 98% | 98% | 98% | 97% | 97% | 97% | 72% |
| **8x-** | 100% | 97% | 96% | 98% | 96% | 95% | 96% | 95% | 6% | 1% |
| $\sigma$ | **0%** | **2%** | **4%** | **4%** | **3%** | **4%** | **9%** | **6%** | **6%** | **26%** |
| $\mu$ | **100%** | **99%** | **98%** | **95%** | **96%** | **96%** | **89%** | **93%** | **93%** | **78%** |

reduced to up to 5% of what it originally takes. Limiting the candidate pool to top nodes reduces the runtime to one fifth on average. Selecting 5 or 25 nodes instead of 1 at a time reduces the runtime by 34% and 59% on average. Combining these two methods can further reduce the runtime.

In summary, the new methods and techniques we introduced perform better than the existing methods and techniques on average, or at least in certain settings as is the case with *MinPGR*. The use of scaling parameters which are also introduced by us significantly reduces the runtime while providing similar results.

Node Selection Method 1 is the most naive method which does not account for budget or efficiency but only for gain. Hence, it is outperformed by Method 2 which maximizes the efficiency before gain. Method 3 which is introduced by us performs better than both methods on average. The intuition behind Method 3 is that efficiency is indeed more

Table 5.8: Overall Impact of Scaling Parameters on Runtime

| **ExH** | **1_I** | **1_20** | **1_10** | **5_I** | **5_20** | **5_10** | **25_I** | **25_20** | **25_10** | **I_I** |
|---|---|---|---|---|---|---|---|---|---|---|
| **1xM** | 100% | 22% | 15% | 65% | 14% | 10% | 28% | 28% | 7% | 5% |
| **1xB** | 100% | 19% | 13% | 62% | 12% | 9% | 26% | 8% | 7% | 5% |
| **2xÃ** | 100% | 21% | 17% | 67% | 17% | 16% | 36% | 16% | 15% | 12% |
| **2xB** | 100% | 19% | 17% | 54% | 20% | 17% | 56% | 16% | 15% | 19% |
| **3xM** | 100% | 11% | 11% | 64% | 15% | 9% | 34% | 9% | 9% | 6% |
| **3xB** | 100% | 17% | 14% | 82% | 19% | 12% | 43% | 12% | 11% | 8% |
| **4xM** | 100% | 8% | 7% | 70% | 8% | 7% | 39% | 7% | 6% | 4% |
| **4xB** | 100% | 8% | 7% | 70% | 8% | 7% | 39% | 7% | 7% | 5% |
| **5xM** | 100% | 23% | 15% | 54% | 15% | 15% | 23% | 8% | 8% | 8% |
| **5xB** | 100% | 31% | 20% | 63% | 22% | 17% | 41% | 16% | 13% | 8% |
| **6xm** | 100% | 27% | 18% | 64% | 27% | 18% | 36% | 18% | 18% | 9% |
| **6xB** | 100% | 27% | 18% | 55% | 27% | 18% | 36% | 18% | 18% | 9% |
| **7xM** | 100% | 42% | 38% | 77% | 36% | 34% | 73% | 35% | 35% | 22% |
| **7xB** | 100% | 38% | 43% | 76% | 40% | 38% | 68% | 39% | 38% | 27% |
| **8x-** | 100% | 12% | 10% | 93% | 11% | 10% | 56% | 16% | 11% | 7% |
| $\boldsymbol{\sigma}$ | 0% | 10% | 10% | 8% | 9% | 9% | 14% | 10% | 10% | 7% |
| $\boldsymbol{\mu}$ | 100% | 22% | 18% | 66% | 20% | 16% | 41% | 15% | 15% | 11% |

important than immediate gain, however it could be the case that selecting the node with highest gain among the top efficient nodes could be more effective. Experiments showed that this intuition is a logical one.

Potential Gain Calculation Method 3 and 4 which are introduced by us outperform Method 1 and 2. Method 1 is the most naive method which do not account for potential gains altogether, thus it is outperformed by Method 2. Method 2 emphasizes on potential gains. However, investing in future potential gains when there is only a limited budget left is not wise. Our methods reflect that and dynamically change the weight between actual gain and potential gain, hence are able to perform better than other methods.

Minimum Potential Gain Ratio parameter proposed in this work does not result in any significant difference on average. The average results suggest that the role $MinPGR$ is supposed to play is already taken care of by the methods we introduced in Node Selection and Potential Gain Calculation. However, tuning $MinPGR$ is able to produce better performances in individual cases.

The scaling parameters we created, $NumBefReCalc$ and $TopMltp$, enables TABU-PG to run on very large networks in reasonable amounts of time. Especially, limiting the candidate pool for seed nodes by assigning an appropriate value to $TopMltp$ significantly reduces the runtime while nearly matching the same influence spread. The runtime can further be reduced by employing $NumBefReCalc$ by sacrificing a little more influence spread. However, $NumBefReCalc$ should not be set to values which are too large since it might cause a dramatic decrease in performance especially in the cases where there exist tipping points.

In addition, the hybrid influence weight generation method proposed in Section 4.4.1 creates tipping points in the network whereas the ratio model which is the most widely used method in the literature rather results in a decreasingly growing or a linear influence spread. This comparison further supports our claim that the hybrid model which is introduced by us is a better representation of the real world because tipping points also exist in most real life networks.

# Chapter 6

# Conclusion

In this thesis, we defined the new Targeted and Budgeted Influence Maximization under Deterministic Linear Threshold problem. As a solution to the defined problem, we developed Targeted and Budgeted Potential Greedy (TABU-PG) algorithm. The algorithm supplies different methods for different operations. Thus, each combination of method selection results in a distinct TABU-PG heuristic.

The idea behind TABU-PG algorithm is to invest on potential future gains instead of accounting for only immediate gains (i.e., actual gains). In order to obtain a better return on such investments, we equip TABU-PG with methods which dynamically control such investments by employing a set of potential gain calculation procedures. One method discounts the potential gains if the partially influenced node is only very slightly influenced, hence not likely to be activated in future iterations. Another method dynamically controls the weights of potential gains in comparison to actual gains. As the remaining budget gets closer to zero, the emphasize given on potential gains is reduced in favour of actual gains. This is because, the chances of reaping those potential gains decreases as the budget is spent over time.

Another idea behind TABU-PG is to consider the budget in the node selection step. Our problem presents a knapsack problem with nodes having cost values which are heterogeneous (but fixed over time) and returns which might change at each iteration. Therefore, there is a need to select nodes in a way that the total return is maximized without violating the budget constraint. For this purpose, we equip TABU-PG with different node selection (i.e., node comparison) methods which consider gain, efficiency (i.e., density), or a combination of both.

Extending the problem and solution to enable different nodes to carry different return values makes it possible to apply our solution to different problems depending on how the return values are generated. Assigning values based on estimated profits makes it a profit maximization problem whereas assigning values based on distances to the target location (in real world) makes it a location-based marketing problem. The term *Targeted* in TABU-PG is intended to cover all such problems which are mostly the same in essence.

Computational experiments demonstrated that even the worst TABU-PG heuristic performs better than the best benchmark heuristic excluding some rare cases where an average TABU-PG heuristic still performs better than all benchmarks. The results show that the difference in performances of TABU-PG heuristics and benchmark heuristics can vary from significant to very huge.

Some of the methods which are supplied as alternatives in TABU-PG are taken from the literature while the others are novel methods introduced by us. Experiments showed that the novel methods proposed by us usually provide better performance than the existing methods.

TABU-PG is readily designed to minimize the number of calculations by detecting redundant calculations and skipping them. Further to that, we proposed additional novel methods which significantly improve runtime by providing a trade-off between runtime and spread performance. The reduction in spread performance is shown to be often very slight. Utilizing these scalability methods, TABU-PG is able to run on very large networks in a reasonable amount of time.

Briefly, in this work, we defined a new influence maximization problem and offered a novel solution algorithm named TABU-PG that is equipped with novel methods along with methods taken from the literature. Additional methods are utilized for making the algorithm scalable to very large networks. Extensive computational experiments are carried out with 8 different datasets on 4 different real life networks. Experimental results showed that the novel methods we introduced are principally superior to the existing methods in the literature.

# Bibliography

[1] D. Kempe, J. Kleinberg, and É. 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*, pages 137–146, Washington, DC, USA, 2003. ACM.

[2] M. Deutsch and H. B. Gerard. A study of normative and informational social influences upon individual judgment. *The Journal of Abnormal and Social Psychology*, 51(3):629, 1955.

[3] M. Gladwell. *The tipping point: how little things can make a big difference.* Little Brown, New York, NY, USA, 2006.

[4] D. Kahneman. *Thinking, fast and slow.* Macmillan, New York, NY, USA, 2011.

[5] J. Scott and P. J. Carrington. *The SAGE handbook of social network analysis.* SAGE publications, Thousand Oaks, CA, USA, 2011.

[6] J. L. Moreno, H. Hall Jennings, et al. *Who shall survive?* Beacon House, Beacon, NY, USA, 1934.

[7] L. Euler. Leonhard euler and the königsberg bridges. *Scientific American*, 189(1): 66–70, 1953.

[8] E. M. Rogers. *Diffusion of innovations.* Free Press of Glencoe, New York, NY, USA, 1962.

[9] M. Cha, H. Haddadi, F. Benevenuto, and P. K. Gummadi. Measuring user influence in twitter: the million follower fallacy. In *4th International AAAI Conference on Weblogs and Social Media*, Washington, DC, USA, 2010.

[10] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: bringing order to the web. Technical report, Stanford InfoLab, 1999.

[11] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[12] P. Bonacich. Power and centrality: a family of measures. *American Journal of Sociology*, 92(5):1170–1182, 1987.

[13] D. Easley and J. Kleinberg. *Networks, crowds, and markets: reasoning about a highly connected world.* Cambridge University Press, Cambridge, UK, 2010.

[14] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 57–66. ACM, 2001.

[15] M. Granovetter. Threshold models of collective behavior. *American Journal of Sociology*, 83(6):1420–1443, 1978.

[16] W. Chen, Y. Yuan, and L. Zhang. Scalable influence maximization in social networks under the linear threshold model. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 88–97, Sydney, Australia, 2010. IEEE.

[17] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 420–429, San Jose, CA, USA, 2007. ACM.

[18] A. Goyal, W. Lu, and L. V. S. Lakshmanan. Celf++: optimizing the greedy algorithm for influence maximization in social networks. In *Proceedings of the 20th International Conference Companion on World Wide Web*, pages 47–48, Hyderabad, India, 2011. ACM.

[19] C. Zhou, P. Zhang, J. Guo, X. Zhu, and L. Guo. Ublf: An upper bound based approach to discover influential nodes in social networks. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 907–916, Dallas, TX, USA, 2013. IEEE.

[20] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 199–208, Paris, France, 2009. ACM.

[21] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1029–1038, Washington, DC, USA, 2010. ACM.

[22] A. Goyal, W. Lu, and L. V. S. Lakshmanan. Simpath: An efficient algorithm for influence maximization under the linear threshold model. In *Data Mining (ICDM),*

*2011 IEEE 11th International Conference on*, pages 211–220, Vancouver, Canada, 2011. IEEE.

[23] H. Nguyen and R. Zheng. On budgeted influence maximization in social networks. *IEEE Journal on Selected Areas in Communications*, 31(6):1084–1094, 2013.

[24] N. Du, Y. Liang, M. F. Balcan, and L. Song. Budgeted influence maximization for multiple products. *arXiv preprint arXiv:1312.2164*, 2013.

[25] Y. Singer. How to win friends and influence people, truthfully: influence maximization mechanisms for social networks. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, pages 733–742, Seattle, WA, USA, 2012. ACM.

[26] F. Li, C. Li, and M. Shan. Labeled influence maximization in social networks for target marketing. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pages 560–563, Boston, MA, USA, 2011. IEEE.

[27] W. Lu and L. V. S. Lakshmanan. Profit maximization over social networks. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 479–488, Brussels, Belgium, 2012. IEEE.

[28] Y. Li, D. Zhang, and K. Tan. Real-time targeted influence maximization for online advertisements. *Proceedings of the VLDB Endowment*, 8(10):1070–1081, 2015.

[29] J. Lee and C. Chung. A query approach for influence maximization on specific users in social networks. *IEEE Transactions on knowledge and Data engineering*, 27(2): 340–353, 2015.

[30] C. Song, W. Hsu, and M. L. Lee. Targeted influence maximization in social networks. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*, pages 1683–1692, Indianapolis, IN, USA, 2016. ACM.

[31] Z. Lu, W. Zhang, W. Wu, J. Kim, and B. Fu. The complexity of influence maximization problem in the deterministic linear threshold model. *Journal of Combinatorial Optimization*, 24(3):374–378, 2012.

[32] Z. Lu, W. Zhang, W. Wu, B. Fu, and D. Du. Approximation and inapproximation for the influence maximization problem in social networks under deterministic linear threshold model. In *Distributed Computing Systems Workshops (ICDCSW), 2011 31st International Conference on*, pages 160–165, Minneapolis, MN, USA, 2011. IEEE.

[33] D. Acemoglu, A. Ozdaglar, and E. Yildiz. Diffusion of innovations in social networks. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 2329–2334, Orlando, FL, USA, 2011. IEEE.

[34] F. Zou, Z. Zhang, and W. Wu. Latency-bounded minimum influential node selection in social networks. In *International Conference on Wireless Algorithms, Systems, and Applications*, pages 519–526, Boston, MA, USA, 2009.

[35] X. Zhu, J. Yu, W. Lee, D. Kim, S. Shan, and D. Du. New dominating sets in social networks. *Journal of Global Optimization*, 48(4):633–642, 2010.

[36] F. Wang, H. Du, E. Camacho, K. Xu, W. Lee, Y. Shi, and S. Shan. On positive influence dominating sets in social networks. *Theoretical Computer Science*, 412(3): 265–269, 2011.

[37] W. Zhang, W. Wu, F. Wang, and K. Xu. Positive influence dominating sets in power-law graphs. *Social Network Analysis and Mining*, 2(1):31–37, 2012.

[38] G. Askalidis, R. A. Berry, and V. G. Subramanian. Explaining snapshots of network diffusions: Structural and hardness results. In *International Computing and Combinatorics Conference*, pages 616–625, Atlanta, GA, USA, 2014. Springer.

[39] R. Xu. An lp norm relaxation approach to positive influence maximization in social network under the deterministic linear threshold model. In *International Workshop on Algorithms and Models for the Web-Graph*, pages 144–155, Montreal, Canada, 2013. Springer.

[40] A. Swaminathan. An algorithm for influence maximization and target set selection for the deterministic linear threshold model. Master's thesis, Virginia Polytechnic Institute and State University, 2014.

[41] J. Leskovec and A. Kreyl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/Data`, June 2014.

[42] M. Richardson, R. Agrawal, and P. Domingos. Trust management for the semantic web. *The Semantic Web-ISWC 2003*, pages 351–368, 2003.

[43] M. Fire, L. Tenenboim, O. Lesser, R. Puzis, L. Rokach, and Y. Elovici. Link prediction in social networks using computationally efficient topological features. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pages 73–80, Boston, MA, USA, 2011. IEEE.

[44] M. Fire, L. Tenenboim-Chekina, R. Puzis, O. Lesser, L. Rokach, and Y. Elovici. Computationally efficient link prediction in a variety of social networks. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(1):10, 2013.

[45] L. Takac and M. Zabovsky. Data analysis in public social networks. In *International Scientific Conference and International Workshop Present Day Trends of Innovations*, volume 1, 2012.

[46] C. P. Robert. Simulation of truncated normal variables. *Statistics and Computing*, 5(2):121–125, 1995.