N-HOP INFLUENCE MAXIMIZATION PROBLEM UNDER DETERMINISTIC LINEAR THRESHOLD MODEL

A Thesis

by

Sena Odabaşı

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Data Science

Özyeğin University
August 2022

N-HOP INFLUENCE MAXIMIZATION PROBLEM UNDER DETERMINISTIC LINEAR THRESHOLD MODEL

Approved by:

Assistant Professor Dilek Günneç Danış,
Advisor
Department of Industrial Engineering
Özyeğin University

Assoc. Professor Emre Keskin
Department of Industrial Engineering
Atatürk University

Assistant Professor Enis Kayış
Department of Industrial Engineering
Özyeğin University

Date Approved: 12 August 2022

To my siblings, Beyza & Tuna

ABSTRACT

The Influence Maximization Problem (IMP) finds a set of highly influential nodes within a social network in order to maximize the spread of influence. We consider that people can have influence on their direct (1-hop), 2-hop and 3-hop neighbors. IMP with extended influence transitivity is called n-hop IMP. In this paper, we study the problem under the deterministic linear threshold model and propose a heuristic solution. In our proposed heuristic model, there are two parts, extended seed set algorithm and local search. Main purpose of extended seed set algorithm is creating a node set and send it to local search which is based on trying to improve it via replacing the nodes in the given set. We used two different features for selecting the candidate nodes. We propose an equation to estimate the value of a node set without actually computing. We used real-life and synthetic networks to test our solution method and generated weight and threshold values in three different methods.

ÖZETÇE

Etki Enbüyükleme Problemi, bir sosyal ağ içinde etkinin yayılmasını en üst düzeye çıkarmak için bir dizi son derece etkili düğüm bulur. İnsanların doğrudan (1-adım), 2-adım ve 3-adım komşuları üzerinde etkileri olabileceğini düşünüyoruz. Genişletilmiş etki geçişliliğine sahip Etki Enbüyükleme Problemi, n-adım Etki Enbüyükleme Problemi olarak adlandırılır. Bu yazıda, bahsedilen problemi belirlenmiş doğrusal eşik modeli altında inceliyoruz ve sezgisel bir çözüm buluyoruz. Önerdiğimiz sezgisel modelimizde, genişletilmiş çekirdek küme algoritması ve yerel arama olmak üzere iki bölüm var. Genişletilmiş çekirdek küme algoritmasının temel amacı, bir düğüm kümesi oluşturmak ve onu verilen kümedeki düğümleri değiştirerek iyileştirmeye dayanan yerel aramaya göndermektir. Aday düğümleri seçmek için iki farklı özellik kullandık. Gerçekte hesaplama yapmadan bir düğüm kümesinin değerini tahmin etmek için bir denklem buluyoruz. Çözüm yöntemimizi test etmek için gerçek hayat ve sentetik ağları kullandık ve üç farklı yöntemde ağırlık ve eşik değerleri oluşturduk.

ACKNOWLEDGEMENTS

I would like to mention a number of special people who helped me throughout the research journey.

Firstly, I would like to express my deepest appreciation to my advisor Assistant Professor Dilek Günneç Danış. She assisted me in my studies by offering important guidance and in-depth explanations, created a safe environment in our feedback sessions with her insightful comments and suggestions during the work. This work would not have been possible without her invaluable guidance. I consider myself lucky to be under her marvelous supervision.

My warm and heartfelt thanks to my beloved family, especially my parents Canan and Adnan Odabaşı, who are always by my side, for everything they have done for me. I am grateful for my sister Beyza and brother Tuna for their keen interest, unconditional love and support. They kept me strong and gave me hope during my work. My family helped me with the roughest decisions, put up with me in stressful times throughout my life and they deserve endless gratitude.

I would like to thank to my classmates in Özyeğin University: Mihail, Batuhan, Eren, Ecem and Deniz for answering every little question I asked patiently and their inspiration. The assistance and cooperation they provided taught me so much.

Lastly, I'd like to mention my dearest friends Orçun and Mustafa who gave me their moral support constantly and who were always there for me. They have been great friends by cheering me up, motivating me and with their encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

In the age of social media, information diffusion happens over online platforms, especially in social media. Social media becomes more and more important each day and with the overgrow, it becomes more effective on people. Data companies use this effect for different purposes like marketing, political, etc. In marketing, it is important to find the right people so that the campaigns would be successful. Viral marketing via social media has become a key component of all types of campaigns.

People who use social media actively and have more than thousands of followers are called social media influencers. Influencers are idolized by their followers, their followers trust them and value their advice. Therefore, influencers can shape others' buying habits, fashion taste, style, or trends. The main channels of influencers in social media are Instagram, TikTok and Twitter. Instagram has a new feature called Instagram Shopping. People can promote, sell, buy, find new customers. Products can be tagged on photos or videos like a commercial. When companies suggest their own products, people may find it not genuine. However, when a person who is very similar to them and has many followers (therefore accepted by other people), makes the same suggestion it feels more heartfelt. Thus, the idolized influencer recommends a product to their followers while using it and their followers can easily reach and buy the item. Sharing, shopping, and influencing get easier and easier. Brands start to send free products or pay influencers to wear or promote them. But how to select these influencers in pursuance of gaining more profit and recognition? In order to persuade the maximum number of people into buying a product, the most effective group of people must be determined to make the promotion. Determining the seed

set and choosing the best nodes to influence as many individuals as possible is called the Influence Maximization Problem (IMP). The aim of the Influence Maximization Problem is to find the most influential set of nodes on a social graph.

Another Instagram feature is Instagram Discover which is a feature where users may find post recommendations based on their interests. On the Discover page, users see content they don't follow rather than ones they do. Twitter uses an algorithm that suggest Tweets recommending tweets based on the people you already follow, the topics you follow, popular ones and your network. A point that shapes individuals' behaviors is peer pressure. Peer pressure means that people, especially teenagers, have pressure or effect from their peer to perform some behaviors that they wouldn't perform otherwise. Affection can be both positive or negative. On the marketing angle, Instagram has a huge impact on fashion and trends. People tend to choose clothes according to their friends and environment.

Individuals are the means by which information is conveyed; they both receive and send information. One might leave their comments about a product on internet marketing services to inform others about it. Likewise, they can share that with their friends verbally or on social media. People can spread a comment or an information about a product whether they have bought it or not like a gossip or a rumor. For instance, a person buys something and recommends it to his friends. One of his friends does not buy it but tells his friend circle that his friend bought something and it is very good. Even though one does not buy the same item, one can promote it. It can be concluded that users may affect not only their friends, but also friends of their friends and their friends. This feature is added to influence maximization problem as n-hop influence maximization problem.

1.1 Our Contribution

In this work, we approached the IMP with extended influence transitivity which is called as the n-hop IMP. We have established algorithms to determine the effect of a node on another across a 2-hop and 3-hop distance. Two new metrics are introduced as Total Weight and Weighted-Neighbor degree. We have proposed a degree calculation method to estimate a set's value. Lastly, we have proposed a heuristic solution to n-hop IMP that consists of two sections and utilizes new metrics and degree calculation method. First part selects nodes and creates the set in order to be improved on the next section.

1.2 Thesis Structure

Rest of the thesis is structured as follows: We review the literature on IMP, n-hop IMP on how they have been studied, developed and solved in Chapter 2. We provide a detailed explanation of the problem in Chapter 3 along with an illustrative example and the mathematical model. Then we present our solution to the problem in Chapter 4. In Chapter 5, we show which networks are used for the computational study and how weights and thresholds are generated together with the numerical results. Lastly, in Chapter 6, we present our conclusions.

CHAPTER II

LITERATURE REVIEW

In this section, first we examine the literature on the influence maximization problem and cover the latest and most preferred algorithms to solve this problem. Next we focus on the n-hop IMP and the influence transitivity methods.

2.1 Influence Maximization Problem (IMP)

The Influence Maximization Problem (IMP) finds a set of highly influential nodes within a social network in order to maximize the spread of influence. Kempe et al. (2003) [1] proposes a solution to IMP using simple greedy algorithm. The greedy algorithm selects the node that supplies maximum marginal gain, and it provides a $(1 - 1/e - \varepsilon)$ approximation but it is time consuming. The authors propose two different diffusion models: Independent Cascade (IC) and the Linear Threshold (LT). In the IC model, nodes activate their neighbors with a probability $p_{i,j}$. The neighbor node becomes active according to that probability. In the LT model, each node i has a threshold value θ_i and a node gets activated if the total influence from its neighbors is higher than its threshold. Chen et al. (2009) [2] continue to enhance the greedy algorithm to increase its effectiveness. NewGreedy and MixedGreedy are the two methods where instead of calculating the real influence, an estimation is made. The authors, furthermore, propose a new heuristic called Degree Discount. Nodes with high degree gets selected. Nodes' degrees gets discounted if their neighbors are already in the seed set. Cost effective lazy forward (CELF) [3], which is 700 times faster than the greedy algorithm, takes advantage of submodularity while finding optimal nodes. An improved and even faster CELF method CELF++ is developed by Goyal et al. (2011) [4]. They estimate the influence of a node and selects the nodes according to

that.

There are some heuristic approaches to IMP. A discrete particle swarm optimization (DPSO) algorithm is proposed by Gong et al. (2016) [5]. Even though there is no hopping influence transmission between nodes, they use hops to evaluate a set's degree in LIE function which estimates how much influence a node can have on its neighbors, neighbors' neighbors and neighbors' neighbors' neighbors in a stochastic setting. LIE function also used in a discrete shuffled frog-leaping algorithm as an influence estimator [6].

2.2 n-hop IMP

n-hop Maximization is an extension of the IMP. It is based on the idea that people are influenced not only from their immediate neighbors, but also from their neighbors' neighbors. In our problem, we try to find the seed set with maximum influence with the when the influence model follows the n-hop structure. Jøsang & Pope (2005) [7] study the concept of transitive trust. If an individual A trusts individual B and individual B trusts individual C, that means that individual A will trust individual C throughout individual B. Expanding the idea of trust, Hang et al. (2008) [8] suggest that trust and influence move the similarly and influence transitivity proceeds the same way as trust transitivity does. If an individual A influences individual B and individual B influences individual C, that means that individual A will influence individual C throughout individual B.

2.2.1 n-hop IMP Diffusion Models

A new influence model is proposed by Xu et al. (2014) [9] which is based on using the similarity between people. If two people are connected, it is more likely that they have similar taste. People who have similar taste are tend to be friends with each other. If nodes A and B are connected and node B is connected to node C, then the influence probability from node A to node C is calculated by using the Equation 1.

When the hop number increases, the trust between people gets lower. The authors suggest that there are 3 phases: slow decay, fast decay and slow decay. Trust value decreases slowly in the first and the last 3 hops, but in the middle hops the decrease is faster.

$$P(A,C) = P(A,B)P(B,C) \tag{1}$$

Another influence estimation algorithm which is based on hops is given by [10]. This algorithm is based on probability of the nodes, not their weights. But it can work with the LT model. One hop activation probability is calculated via Equation (2), $p_{w,v}$ represents the probability for node v to be activated by node w. When a new node v added to the seed set S, the estimated influence spread to the 1-hop neighbors.

$$\pi_1^S(v) = \begin{cases} 1, & \text{if } v \epsilon S, \\ 1 - \prod_{w \epsilon I_v \cap S} (1 - p_{w,v}), & \text{otherwise.} \end{cases} \tag{2}$$

Newman et al. (2001) [11] uses Equation (3) on a co-authorship network, calculates a node's estimated distance to its neighbors and finds the total distance. They assume that two authors know each other better if they have worked together before. δ_i^k number shows the relationship between two authors in paper k, it becomes 1 if author i have worked on paper k otherwise 0. n^k represent number of coauthors who have participated in paper k.

$$w_{ij} = \sum_k \frac{\delta_i^k \delta_j^k}{n^k - 1} \tag{3}$$

Another way to calculate the transitivity influence is based on the speed of information distribution among individuals. A propagation velocity equation which is based on time and the distance between nodes is introduced [12]. When the distance increases, the trust between individuals decreases like the velocity of the information.

In order to define velocity, the equation 4. In our suggested propagation model, we used a very similar equation that based on the distance but without the constant parameters.

$$v_i = [\frac{1}{d^2}]e^{-\lambda * t} \tag{4}$$

Gulati & Eirinaki (2018) [13] considers the problem as a min-max problem. Instead of using k as a fixed parameter, the authors find the maximum influence with the minimum number of nodes in the initial seed set. Linear threshold is used like our study but only one hop neighbors are covered. For hopping factor a different approach is adopted as in Equation 5 where decay is a constant number which is 0.1. In our work, we define k as a fixed parameter and given to the algorithm as an input.

$$\text{Hopping factor} = 1 + hop * decay \tag{5}$$

2.2.2 n-hop IMP Solution Methods

There are many heuristic methods in the literature to find an optimal solution for the IMP. As mentioned before a simple greedy algorithm proposed by Kempe et al. (2003) [1]. The algorithm selects the node with maximum marginal gain and adds it to set. It is a hill-climbing method. Speeding up the greedy algorithm have been done using various methods. Nguyen et al. (2017) [14] proposed a new heuristic method called probability-based multi-hop diffusion while using the transitivity method [9] as the influence model which closely resembles to the DegreeDiscount [2]. They calculate the probability degree of each node using its degree and the effect of two-hops neighbors. Another solution method suggested is the seed set algorithm to select the seed nodes [9]. Seed set algorithm selects a seed set that maximizes the total weight between the seed set and non-seed set. In graph G, a cut must be put in optimal place that provides the total weight between cut points to be maximized.

A combination of greedy and heuristic solution is IMMRA [12]. IMMRA uses a degree method based on the degree of the nodes and their velocity. To calculate the velocity, they use the velocity attenuation they invented. Their time complexity is significantly lower than greedy algorithm [1].

The threshold-bounded influence propagation algorithm uses different ranking systems, then order the nodes according to their rank from largest to smallest [13]. Starting from the largest, if the node can activate others, it is added to seed set. It checks is the node has valid influence on its neighbors, instead of selecting them only for their degrees like DegreeDiscount [2].

Tang [10] calculates the influence spread and forms the seed set while maximizing the marginal gain of influence spread. Upper bounding technique works for one-hop and two-hops. Their technique works with linear threshold propagation model. This method reduces the time of greedy algorithm [1] significantly.

On dynamic social networks, Meng et al. (2019) [15] studies the IMP with n-hop while using the influence estimation model [10]. They calculate the influence spread and find the marginal gain.

Even though there are numerous work in the literature, most of uses stochastic network models.

CHAPTER III

PROBLEM DEFINITION

In this chapter, we explain the characteristics of the IMP and the n-hop IMP. We first define the problem, explain the n-hop influence spread and then provide an illustrative example on a sample network. Then, we give our the mathematical model for the problem.

3.1 Problem Statement

Influence Maximization Problem is the problem of selecting k nodes on a social network in such a way that their influence will result in the maximum spread of a product, an opinion or an idea. Graph $G = (V, E)$ is given as an input where V represents the vertices and E represents edges. An edge (i, j) represents the relationship between node i and node j. On directed networks if there is an edge from node i to node j, it means that node i has an influence on node j. On undirected networks influence goes on both ways, i.e. when node i and node j are connected, node i influences node j and node j influences node i. On each edge, there is a weight value that shows how much the node influences its neighbor on that edge. Weight value from node i to node j is shown as w_{ij}.

A node can take one of two status: active and inactive. Being active means that the node has bought the product and inactive nodes have not bought the product yet. An inactive node becomes active when the total influence from its active neighbors is bigger than its threshold value. An active node stays active. x_{it} represents whether node i is active at time t, it equals to 1 when node is active and 0 otherwise.

In this thesis, we study the n-hop IMP where the influence spread is further than the original IMP. In the n-hop influence setting, a node can be influenced by not only

its neighbors but by the nodes it can access through the neighbors of its neighbors. Here the influence travels to nodes that are not neighbors of the influencing node. The objective of the n-hop IMP is the same as the IMP, i.e., to find the initial set of nodes that maximizes the influence. The value k shows the number of the nodes to be selected in the initial seed set S and is given as a parameter. The nodes in the seed set S starts as active nodes at time 0. Every active node influences its neighbors with certain weight. The process ends when there are no nodes left to be activated. $\sigma(S)$ is the total influence of set S at the end of the process. A hopping factor is introduced as ω^h. Nodes has influence on its neighbors as well as the neighbors of their neighbors and so on. In order to calculate the hopping influence from a node its neighbors, the hopping factor is multiplied with the weight between the node and its neighbor.

The influence spreading model is based on Deterministic Linear threshold model. All weight and threshold values are given before starting to solve the problem. A node decides to buy when the total influence from neighbor nodes is larger than or equal to the threshold value of the node. The hopping weight values are also included in collective influence. Nodes activate their neighbors with a weight value. Each edge has a weight value. On directed networks, the out-coming edges of a node has an influence on its neighbors and nodes get influenced via in-coming edges. In order to be activated, the total influence/weight from its neighbors at timestamp $t-1$, hopping weight from neighbors of its neighbors, 2-hop neighbors, at timestamp $t-2$ and hopping weight from neighbors of its neighbors, 2-hop neighbors, at timestamp $t-2$ must be more than its threshold value at timestamp t.

3.2 n-hop Influence Structure

In this part, the structure of n-hop is explained in detail. There are two important concepts: hopping factor and hopping weight. Hopping factor is the ratio that is used

in influence transitivity. As mentioned in the earlier chapters, a node has an influence on its direct, 2-hop and 3-hop neighbors. When hopping factor is too small, then the transmitted influence becomes insignificantly small and when it is larger, then the diffusion becomes too fast. Finding an optimal hopping factor required many trials. Formula 6 shows the equation of how the hopping factor is calculated.

$$\text{Hopping factor(i) at hope h}\{2,3\} = \frac{1}{h^2} \tag{6}$$

After the hopping factor is found, the hopping weight can be calculated. There are 2 thing that are necessary and will be multiplied. First the weight from initial node to its direct neighbor is important because their influence and trust relationship reflex on the neighbors of the neighbors which are 2-hop and more extended 3-hop neighbors.

Hopping weight transmission can be seen on Figure 3.2. It is a directed network. Thick blue lines represent the weight, influence between nodes. It can be seen from node 0 to node 1, node 1 to node 4 and node 4 to node 5 with weights of 0.5, 0.6 and 0.5 respectively. Node 0 has an influence on node 2 through node 1, but the effect is smaller. The influence from node 0 to 1 is multiplied with the hopping factor and the result is 0.13. Likewise, in case of getting activated, node 1 will have a hopping weight on node 3. If node 1 does not become active, it will not have influence on its neighbors. But it can pass the influence from node 0 to nodes 2 and 3. Dashed orange line represent 2-hop influence from a node to neighbors of its neighbors. In this example, node 0 to node 2 throughout node 1 and node 1 to node 3 throughout node 2 are simulated as 2-hop influence transitivity. 3-hop weight from node 0 to node 3 is represented with dashed green line which is found by multiplying the weight from node 0 to node 1 and hopping factor when hop is 2: 0.5 * 1 / 9 = 0.06.

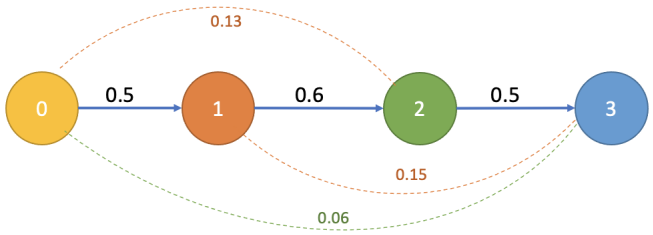


Figure 1: Hopping Weight Transmission

3.3 Acyclic Graphs

Networks especially in real-life may contain cycles. If the effect is not considered, the nodes in a cycle can influence each other over and over until each one is activated. This is an unhealthy scenario. We wanted to prevent this. All hopping weights are initially calculated and stored in an environment. While calculating the hopping weights, some rules and regulations are considered to prevent that. A node cannot reflect the influence it receives to its influencer. If there is a directed edge from node i to node j and a directed edge from node j to node i or the network is undirected and two nodes i and j have influence on each other. Seed node i is not its 2-hop neighbor. The same logic applies on 1 and 3-hop neighbors too. An influence cannot reflect the same direction it came from.

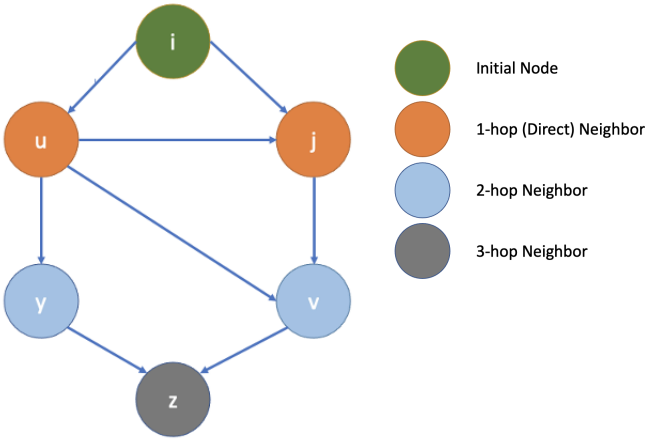


Figure 2: Sample Network

In Figure 3.3, nodes are colored according to their state. Green node i is selected as seed node and it is active at time 0. Yellow nodes j and u are its 1-hop neighbors. Node i has an influence on them as w_{ij} and w_{iu}. It there are bigger than their threshold values then they get activated. When a node gets activated at timestamp t, it will try to activate its neighbors at timestamp $t+1$. Whether nodes u and j gets activated or not, the influence from node i will reflect from them to their neighbors which are red nodes y and v. Blue node f is 3-hop neighbor of node i.

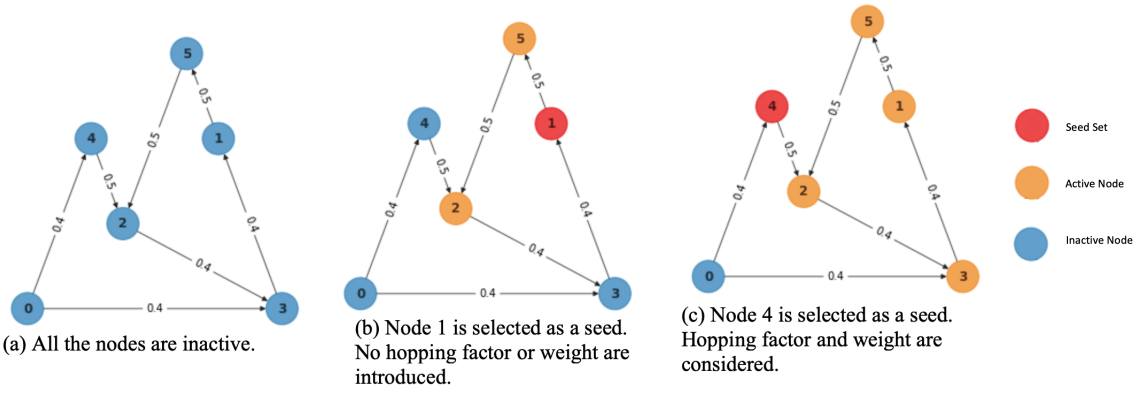
3.4 An Illustrative Example

In this section an example dataset given to demonstrate how the transmission of the influence in n-hop method happens and how does it differ from regular. In Figure 3, there is a directed graph which has 6 nodes that are numbered from 0 to 5 and 7 edges between them. If node 0 is selected as the seed set, nodes 3 and 4 are its 1-hop neighbors. Node 2 is 2-hop and node 3 and node 5 are 3-hop neighbors. Each node has a threshold of 0.5. Weight on edges can be either 0.4 or 0.5 randomly and can be seen in the figure on the related edge.

There are two different scenarios in two different settings. In the first setting, there is no hopping influence from a node to its hopping neighbors. A node only can affect its direct neighbors. The second setting is with n-hop influence, nodes have influence on their direct, 2-hop and 3-hop neighbors.

In Figure 3, (a) shows the graph in its initial state, all the nodes are inactive. (b) shows that what happens when node 1 is selected as seed set and no hopping factor or hopping weight is introduced. (c) is the networks when node 4 is selected as seed set and hopping weights implied while calculating the total influence.

Let's say that node 1 is selected as the initial influencer. The influence weight from node 1 to node 5 is 0.5. It will try to influence its neighbors which is node 1 with an influence of 0.5 and its 2-hop neighbors are the neighbors of node 1 which is



(a) All the nodes are inactive.

(b) Node 1 is selected as a seed. No hopping factor or weight are introduced.

(c) Node 4 is selected as a seed. Hopping factor and weight are considered.

Figure 3: An Illustrative Network

node 4. Even though there is no edge from node 0 to node 4, node 0 still can have an influence on node 4. This is called hopping weight/influence. The hopping weight from node 0 to node 4 is equal to the weight from node 0 to node 1 multiplied by $1/2 * (distance)$. Node 0 will try to influence node 0 with 0.5(its weight on node 1) $* 1/2^2$. We can see a path on the next figure. The influence weight from node 0 to node 1 is 0.5. After 3-hop the hopping weight becomes significantly small. Only 1-hop, 2-hop and 3-hop neighbors are influenced via hopping factor. When a node gets activated, it will influence its neighbors. The algorithm continuous until there is no node left to be activated.

Given the above, when there is no n-hop approach, selecting node 1 reaches 3 active nodes at the end of the period. On the other hand, with the hopping weights, node 4 as the seed set activates maximum number of nodes in the end. Same algorithm can miss most beneficial set.

Same seed set gives inconsistent influence spread on different settings. So, using the same method or algorithm to solve IMP and n-hop IMP may not be efficient. With n-hop method, the number of active nodes gets higher. Total influence is more and the impact goes beyond a no-hop circumstance. In this example, when node 4 is selected as the initial node set, 5 nodes will be active at the end of time. When hopping vector is not considered and only the total of the normal weight values coming

into a node reaches its threshold to be activated, then only 3 nodes are active at the end of the time. Node 1 activates node 5 and node 5 activates node 2. With the absence of hopping weight, total influence is less.

3.5 Mathematical Model

In this section the linear mathematical model of the n-hop Influence Maximization Problem is given for an indirected network. The objective of the problem is to maximize the number of active nodes at the end of the time period.

Sets:

V : Set of all nodes, $i \epsilon 1, .., |V|$

E : Set of all edges, $(i, j) \epsilon 1, .., |E|$

N_i : Set of neighbor nodes of node i

T : Set of all time periods, $t \epsilon 1, .., |T|$

Decision Variables:

x_{it}: 1 when node i is active at time t and 0 otherwise.

Parameters:

w_{ij}: Influence weight from node i to node j.

$\omega^h = \frac{1}{h^2}$, Hopping weight of node i at hope $h \epsilon$ {1, 2, 3}

θ_i: Threshold for node i.

k : # of seed nodes

a_{it}: 1 if $(i, j) \epsilon E$

$$\max \quad \sum_{i \in V} x_{i|T|} \tag{1}$$

$$\text{s.t.} \quad \sum_{i \in I} x_{i0} \leq k \tag{2}$$

$$x_{i(t-1)} \leq x_{it} \qquad \forall i \in V, \forall t \in T \backslash 0 \tag{3}$$

$$\sum_{j \in N_i} (x_{j(t-1)} w_{ji} + \sum_{k \in N_j} (x_{k(t-1)} w_{kj} \omega^2 + \sum_{u \in N_k} x_{u(t-1)} w_{uk} \omega^3)) \geq \theta_i x_{it} \quad \forall i \in V, \forall t \in T \backslash 0 \tag{4}$$

The objective is to maximize the total influence at the end of time T (1). Constraint (2) ensures that the maximum number of seed set S is less than or equal to k. Constraint (3) ensures to keep active nodes active until the end, once a node becomes active, it stays active in the future and cannot return to inactive state. Constraint (4) states that a node becomes active when the total influence a node receives from its neighbors and one-hop neighbors bigger than its threshold value.

We solved this mathematical model using CPLEX. Although it works successfully with small instances, when the networks get bigger, CPLEX works for longer hours and moreover gives inconsistent results. For an example, it sometimes finds a larger objective value with a smaller seed set. CPLEX takes 19 hours to find an initial seed set for a network where the number of nodes is 3000 and k is 1. For larger k values, it takes even longer to solve. In order to avoid such situations, in our CPLEX is given an hour limit to solve the problem.

CHAPTER IV

SOLUTION METHODS

In this paper, our main goal is to solve the n-hop IMP. We provide 5 different algorithms to approach the problem from different perspectives. Before we give details on these algorithms in Section 4.2, we first explain the metrics we use in these algorithms. Then we evaluate their performances in Chapter 5.

4.1 Features Used In our Solution Methods

In order to understand the important features of nodes, which are selected in the seed set, we ran a set of experiments on small size data. We observed the optimal solutions obtained by CPLEX.

We conducted a detailed examination on each node and a series of their features, these features included degree, in-degree, out-degree, neighbor degree, average weight, average threshold of neighbors, total weight and Weighted-Neighbor degree. Next, we explain each of these features. Degree of a node is simply calculated by the number of its direct neighbors. Neighbors are also called as 1-hop neighbors. In degree is the number of incoming edges into a node, if the networks are directed, otherwise, it is equal to the number of direct neighbors. In the n-hop problem a node can affect its 2-hop and 3-hop neighbors too. So, in standard degree calculation the influence from 2-hop and 3-hop neighbors is neglected even though they are important factors especially in online social platforms. In order to address this, we introduce a new feature called "Neighbor Degree" which is the summation of a node's degree and the number of its 2-hop and 3-hop neighbors. (Equation 7) Neighbor Degree provides the knowledge of how many other nodes it can reach to. Another introduced metric is called the "Total Weight", which is the summation of the weight coming out of a

node to its 1-hop, 2-hop and 3-hop neighbors. (Equation 8) It is a good metric to learn how effective the node is or how much power it holds for influencing others.

$$\text{Neighbor Degree}(i) = \sum_{j \epsilon N_i}(1 + \sum_{k \epsilon N_j}(1 + \sum_{u \epsilon N_k} 1))) \tag{7}$$

$$\text{Total Weight}(i) = \sum_{j \epsilon N_i}(w_{ij} + \sum_{k \epsilon N_j}(w_{ij}\omega^2 + \sum_{u \epsilon N_k} w_{ij}\omega^3))) \tag{8}$$

$$\text{Weighted-Neighbor Degree}(i) = \text{Neighbor Degree}(i)\text{Total Weight}(i) \tag{9}$$

When we investigate the seed nodes in the optimal sets, we can see that they tend to have high Weighted-Neighbor Degrees. But not necessarily the highest ones. Thus, selecting the top k nodes only according to Weighted-Neighbor Degree does not give a good solution. Seed nodes may have different characteristics, for example when a few of them are in the same group, the outcome may change. Total Weight is another feature with a high value in the seed nodes. It represents the total influence of a node and how much it affects its neighbors. Thus, when we select the top k-nodes according to Total Weight, the results are promising. We define the combination of these two features as the "Weighted-Neighbor Degree", where neighbor degree is used as a multiplier for the total weight. (It is calculated to reward the number of neighbors(1-hop) rather than 1 or 3-hop neighbors.) After our initial experimentation, we saw that these 3 features exist as a common feature among optimal seed nodes, however they are not strong enough to generate solutions. So, we provide 5 different algorithms, which we explain next, to solve this problem.

4.2 Solution Algorithms

After the problem definition is finalized, we started working on solution algorithms. There were several algorithms which we have implemented before the proposed heuristic was finished. In this section, we will explain them in detail.

There are 3 main categories under solutions algorithms. Firstly, we wanted to see whether machine learning applications would be successful and worked with clustering and regression models. Secondly, we implemented a constructive heuristic algorithm which starts with an empty set and adds nodes one by one until the seed set is completed. Lastly, we improved the constructive heuristic and made it our proposed method.

There were various solution techniques we tried before settling on heuristic methods. We wanted to see whether machine learning applications would be beneficial or not. First, we formed a data using the metric values of nodes including their degree, in-degree, total neighbor degree, average weight to its neighbors, average threshold value of its neighbors. It was unlabeled so only unsupervised methods would be useful. Clustering is an unsupervised technique and can be used on selecting a seed set. We can compare the clusters and select one of them to reduce the size of the problem from the total number of nodes to the nodes that are only in that cluster. First, we created multiple synthetic datasets in different sizes between 5 to 50. Since the datasets were small, number of clusters may be small too. We used elbow method which is used in k-means clustering to determine the number of clusters and found out that 2-3 clusters are enough. We tried them and decided on 3 would be better on analyzing. In experiments, the number of nodes in clusters were inconsistent and distributed unfairly. A cluster have many nodes and the other two had only a little. The nodes that were outliers and have high degree than others were in one cluster. Yet we wanted to see if the method works and found their exact solution to be a comparison with clustering method. We compared the nodes in the solution with their clusters but could not see a link between them. To summarise, clustering method was not efficient on the IMP. We have calculated the total influence spread of each node and realized that the data had a label and it can be used as a target. A regression model can learn the characteristics of each node and predict their influence spread.

The accuracy of the models were low.

Then we developed four different constructive heuristics. Similar to the greedy algorithm, earlier versions had only one part which selects nodes to create a seed set. The node with the highest total weight on its 1-hop, 2-hop and 3-hop neighbors is selected as the first node in the initial seed set. After selecting the first node, one of its neighbors which with the highest weight and is not activated yet gets added to the initial set. Same procedure is applied to lastly added node. This method calculates the real influence of the sets to compare. Therefore, especially in larger datasets, spent time is very high. Selecting the seed nodes and expanding the set is a difficult progress. Marginal gain changes according to characteristics of the nodes.

In CH-1 a set is created by the largest k-nodes according to their total weight. Then each node is replaced with its neighbor if the weight from node to its neighbor bigger than the node's threshold. It continues until there is no improvement on the set 10 times.

CH-2 is a base version of our proposed method (Extended seed set algorithm). Half of the seed set is filled by the highest $k/2$-nodes according to neighbor degree and a candidate set is formed by the highest k-nodes according to weighted-neighbor degree. For each node i, another node from the candidate set which has the greatest number of common neighbors with the node i is added to the seed set. When the iteration is completed, we check if the nodes in the seed set have a neighbor that may have high influence on them. If a node has a neighbor that can activate it, it gets replaced with the neighbor.

CH-3 is an early version of local search algorithm. It gets a seed set as an input and improves it. It creates a set of nodes to be used as a candidate set. Potential nodes are selected in this set according to their total weight. The algorithm iterates the candidate set to improve the seed set. The node in the seed set with the highest number of common neighbors with the candidate node is replaced by the candidate.

If the influence spread of the set gets higher after the replacement, algorithm holds the better or equal node in its memory. The nodes in the memory will be used later as the new candidates. After the iteration ends, same procedure starts with the nodes in the memory.

4.3 Extended Seed Set Algorithm (ESS)

We propose a new method which consists of two parts. The first part (construction) involves selecting a set of k-nodes. Then in the second part (improvement) a heuristic is used to change the set of nodes in the original set to achieve higher influence. The first part, creating a set of k-nodes is done by Extended Seed Set (ESS) algorithm. The purpose of this method is to establish a good k-node set before the local search. When initial node set is better, the local search heuristic can achieve higher influence spread.

The main idea behind the ESS algorithm is to select nodes in such a way that the union of the neighbors of the seed set nodes form the largest group. Thus, we seek to find the set of nodes, who altogether have the largest unique neighbors. In the experiments, both most similar and least similar nodes are selected together according to cosine similarity. Then the real influence values are calculated for each set. When we compare the results, the sets with least similar were better than the sets with most similar.

Algorithm starts with selecting the top, half of the given number k, nodes sorted by Total Weight. It shows that nodes that have higher total weight, the summation of all the weight from the node to its neighbors, are more likely to activate more nodes. After replacing process is finished, the empty places must be filled. In order to spread the initial influence to more and further nodes, least similar nodes that have less common neighbors are selected together. While implementing a solution to the problem cosine similarity which is the inner product space of two vectors is used to

measure the similarity between two nodes. Node similarity is based on the proportion of common friends of two nodes. The neighbors which two nodes have in common are called mutual neighbors. Nodes are similar if most of their neighbors are mutual. Cosine similarity between node i and node j is calculated by dividing the number of common neighbors to square root of degree of node i times square root of degree of node j. The equation is represented by:

$$CosineSimilarity(i,j) = \frac{\sum_{k \in N} E(i,k)E(j,k)}{\sqrt{Degree(i)}\sqrt{Degree(j)}} \tag{10}$$

After the node selection process is finished, we wanted to check whether a node in seed set can be activated by one of its direct neighbors or not. For example, if node i is in seed set, its neighbor node j is not in seed set and the weight from node j to node i is bigger than threshold of node i, it means that node i can be influenced by node j and become active. So, selecting node j would be a better move. At the end of the algorithm, it checks for each node whether there is neighbor node that may activate it. If there is node is replaced with its neighbor node.

ESS behaves differently if k is equal to 1. Selecting only one node is risky and more challenging. In order to choose 1, a candidate set of nodes which is combined with top 5 percent of nodes by weight and top 5 percent of nodes by weight neighbor degree. Then the node with maximum degree by degree calculation method is selected as the initial seed set. After the selection the same procedure to find eligible parent nodes like the greater k values is applied.

Algorithm 1: Extended Seed Set (ESS) Algorithm

```
Result: Set S
initialization;
Set S ← [ ]
Sort nodes according to Total Weight
Set S ← top (3k) node by descending total weight
if k > 1 then
    candidates ← nodes ordered by Weighted-Neighbor Degree in descending
     order and top (k/2) are selected
    for each node i in set S do
        the candidate that is least similar to the node according to cosine
         similarity is selected and added to set
    end
end
for each node i in set S do
    for each n in N_i do
        if weight( n, i) ≥ threshold(i) then
            Replace node i with node n in set S
        end
    end
end
return Set S
```

4.4 Local Search (LS)

After ESS algorithm comes up with a set of nodes, calculates its influence and gives it to local search. Local search algorithm takes two inputs; a graph G and a set of nodes that ESS algorithm provided before. In this part, the purpose and the logic behind the algorithm are explained.

Local search algorithm is a heuristic method. The aim of this heuristic is to improve the given set by replacing the nodes in the set. It tries to improve the set to best possible without spending too much time. Methods tend to spend more time when their solution improves. We have studied the previous works in section **??**.

In some situations, extended seed set algorithm can find a set that is able to influence whole network. In this case, local search is unnecessary since it cannot find a better solution. In such cases, after ESS, local search is not used.

As mentioned before, local search algorithm is based on replacing the nodes in the seed set. Choosing a candidate node from network to replace it with is complex and requires time. Instead of using the whole network and every node, we formed a candidate set from the nodes that have the highest Weighted-Neighbor Degree and not in the initial seed set. For k values smaller than 30, 60 highest nodes are selected as candidates. For the bigger k values double of the k values $2k$ highest nodes are selected as candidates.

In the beginning of the LS algorithm the candidate set is formed. Iteration starts and continues until there is no improvement 100 times in a row. In each iteration, each node in initial seed set is replaced 6 times. In order to improve, heuristic tries to swap each member of the initial set with one of the candidate nodes and checks whether the total influence at the end of T times is higher or not. Then highest one according to degree calculation is selected as the new seed set and iteration continues with new seed set. If these 5 sets have lower or equal degree calculation as the seed set, there will not be a change and algorithm will move onto the next node. First the node is replaced with one of its in-degree nodes. As mentioned before, selecting parent nodes is more beneficial than selecting leaf nodes so neighbor degree, total weight and their product neighbor weight degree is calculated and used in both parts. Among the in-degree nodes, the one that has the greatest number of mutual neighbors is selected. This new set is called S_2. Set S_3 is very similar to previous set S_2. This time the second node that has the greatest number of mutual neighbors is replaced with the node. If node has no in-degrees the random nodes from candidate set is swapped with the node. In the replacement for S_4 and S_5, cosine similarity is used. Two nodes from candidate set which have the highest and the lowest cosine similarity with the node is swapped with the node. And for the last set S_6, an arbitrary effect is added to escape the local maxima and reach to global maxima. A random node from the candidate set is selected to be replacement for the node.

In this algorithm some controversial methods are used together like choosing according to least and most cosine similarity. We wanted to cover each side. For instance, node i and j have the same group of neighbors and similar influence on them. If node i activates a subset of its neighbors then having node j in the set may be disadvantageous and unhelpful. On the other hand, two nodes together can be more effective and activate their neighbors with their combined influence. Both of the scenarios may be effective with different sets, different nodes, different weight or threshold values. So, using them both for each node, provides a more comprehensive solution.

In the proposed method, if the number of active nodes achieve to total number of nodes, then no more node is added, solution becomes the set, even though there are less nodes than k value. For the bigger datasets, it happens less, almost never.

In order to know the influence spread of a set, the summation of the weight and the hopping weight from the seed set to their neighbors is calculated and checked if the total influence is bigger than neighbors' thresholds. Then the process continues with newly activated nodes until there are no nodes left to be activated. Making this calculation requires a large amount of time, especially with larger networks. For instance, t takes approximately 5 minutes with HEP-TH dataset which has 5835 nodes. If an algorithm does 100 runs, it would take 100x5min=500 minutes. So, instead of calculating the influence of each set, an approximate influence is calculated with the summation of unique values of 1-hop, 2-hop and 3-hop neighbors which are presented by U_1, U_2 and U_3 respectively, then we multiply that value with the total weight from node set to other nodes. Instead of calculating the real influence, an estimated degree with the number of unique common neighbors is defined. It holds the information of unique neighbors for 1, 2 and 3 hops separately.

$$\text{Unique Common Neighbors Degree} = (U_1 + \frac{U_2}{4} + \frac{U_3}{9})totalweight \qquad (11)$$

When it was lower as 50, solutions were not as promising as when 100. But one the other hand, the algorithm can be quite long when k was bigger like 500 or 1000. A time limit was needed. To make it fair, the same time-limit is given to algorithm s CPLEX which is 1 hour. If searching time exceeds 1 hour, searching ends and the real influence of the top 5 sets are calculated. It also requires more time with bigger k values. These calculation times are also counted in total time spent of the algorithm.

ESS and Local Search does not provide an exact solution.

Algorithm 2: Local Search Algorithm

```
Set S
initialization
Set S is made by using initial set algorithm
Unique Common Neighbors Degree[S]= calculateDegree(S)
candidate nodes set ← top(2k or 60) nodes sorted by neighbor degree
while Seed Set S does not improve 100 times and time passed ≤ 1 hour do
    for each node i in set S do
        S_2 = change node i with its in-degree node that have most number of
         mutual neighbors
        S_3 = change node i with its in-degree node that have second most
         number of mutual neighbors
        S_4 = change node i with the most similar to the node according to
         cosine similarity
        S_5 = change node i with least similar to the node according to cosine
         similarity
        S_6 = change node i with a random node from the candidate set
        calculate the estimated influence of each set by Unique Common
         Neighbors Degree method
        S ← the set with the max estimated influence { S_2, S_3, S_4, S_5, S_6}
    end
end
max 5 sets holds the top 5 sets that have the maximum degree
for each set in set max 5 sets do
    calculate each set's real influence
end
Set S_final ← max real influence of max 5 sets
return S_final
```

CHAPTER V

COMPUTATIONAL EXPERIMENTS

In this section, we first explain how data used in computational experiments are generated. Then we present the numerical experiments to measure the performance of the proposed methods.

5.1 Dataset Generation

Before conducting the experiments, datasets are produced using real-life and synthetic data. Dataset generation consists of mainly two parts: networks, and influence weight/threshold parameters.

5.1.1 Networks

We used both simulated and real-world datasets to validate the performance of the proposed heuristic. There are 4 different real-life networks. In Adjnoun [16] dataset, words from the novel of Charles Dickens named David Copperfield are used as nodes. The adjectives and nouns are linked if they appear together in a sentence. So, the edges represent that the words are adjacent to each other. Celegansneural network [17, 16] is compiled using an experimental data of biological neural network named C. Elegans. It is a directed network where the nodes represent the neurons and the edges represent the synapses between them. Netscience [16] is a network of scientists and their coauthorship. The nodes represent scientists and the edges represent if they have worked before on network theories or experiments. Similar to Netscience, HEP-TH [18] is a network of scientists and their collaborated work. The nodes represent scientists and the edges represent if they posted preprints on the high-energy theory between years 1995-1999.

Table 1: Real-life networks: number of nodes and edges before and after preprocessing

	Before		After	
Networks	#Nodes	#Edges	#Nodes	#Edges
Adjnoun	112	425	112	425
Celegansneural	297	2,359	239	1,918
Netscience	1,589	2,742	379	914
HEP-TH	8,361	15,751	5,835	13,815

Using a network which has many unconnected components is not healthy for finding an optimal set of nodes. Therefore, the largest component from each network is gathered and used for the experiments. After subtracting little components, real networks got smaller by number both nodes and edges. On the other hand, there was no change in synthetic networks. The number of nodes and edges before and after the preprocessing of the real life networks can be found in Table 1.

Synthetic networks are generated by two different methods. Barabasi-Albert model [19] and Watts-Strogatz model [20]. After numerous test runs, optimal parameters required for the network generation are determined. In Barabasi-Albert model, nodes are generated one by one. When a new node is added to the network, the parameter is used to decide how many edges will be created to connect it to the existing nodes. We choose this parameter to be one. Watts-Strogatz model generates networks in three steps. Nodes are generated so that each node is connected to the nearest a neighbors and then nodes can change whom they are connected to randomly with a chance of p. The parameters a and p can be changed, but for our study, they are set to 2 and 40 after many experiments. There are 3 different sizes of generated network: small, medium and large which have 100, 1000 and 5000 nodes consequently. Networkx library is utilized to produce the networks. Table 2 presents the number of nodes and edges of the synthetic networks used.

Table 2: Synthetic networks: number of nodes and edges

Networks	#Nodes	#Edges
Barabasi-Small	100	198
Barabasi-Medium-1	1000	1998
Barabasi-Medium-2	2000	1999
Barabasi-Large-1	5000	9998
Barabasi-Large-2	10000	9998
Watts-Small	100	200
Watts-Medium-1	1000	2000
Watts-Medium-2	2000	2000
Watts-Large-1	5000	10000
Watts-Large-2	10000	10000

5.1.2 Weight and Threshold Generation

Weight of an arc represent how much a node can influence its neighbor. And threshold of a node represents the value of total influence a node needs to get from its neighbors in order to be activated. As expected, synthetic networks do not have any existing information for threshold and weight values. Celegansneural and HEP-TH networks are weighted originally with respect to a weight generation method which calculates the distance between nodes to value their relationship in [11] but do not have threshold. Most of the networks we use are not weighted and none has threshold values. So, we decided to generate threshold and weight values for each dataset.

Threshold values are synthetically generated for each node. In the majority of studies the thresholds are randomly generated between [0, 1] [21, 13]. But in real life people act according to their personalities and not necessarily randomly. When we generate thresholds, we consider that popular people who have more followers or famous people with many connections tend to be more difficult to influence. On the other hand, ordinary people can be easily influenced from people who are around them.

For computational experiments, we use two different methods to generate node thresholds. In the first method [22] named Degree-based method, the threshold values

are generated according to Equation (12) where the threshold of node i increases as the degree (the number of neighbors of a node) increases.

$$Threshold(i) = 1 - \frac{1}{Degree(i)} \tag{12}$$

In the second method [23] named the Hybrid method, thresholds are randomly generated according to a truncated normal distribution between 0.7 and 1. The upper and lower limits are selected in such a way that the influence diffusion at optimal speed without making it too slow or too fast. The networks generated with the second method are added 2 in their names.

If there were historical information about prior interactions between nodes, such as whether they had met, communicated, were mentioned, etc. it means that they have an edge between them and they have influence over one another. However, such information is not present for our networks. Thus, **weight** values are synthetically generated for each edge between nodes. Finding the influence from one node or one person to another is a challenging process. In the IC model, most of the weight values are found random. Likewise, Kempe (2003) [1] chooses a value which is less than 1. Another way to define weight values is giving uniform values like 0.1 [14] or 0.05 [15]. Everyone has a different personality in real world and uniform weight generation does not contain any characteristics.

Most articles from the computer science literature calculate a weight by calculating the proximity of nodes using pre-given data using methods such as regression or neural networking. In particular, some algorithms that calculates people's followers or affinities with people they follow, using Twitter's past retweet information are launched [24].

Arbitrary formulas can be unstable. So, we wanted to use a node dependent weight generation formula to be more realistic. The weight values are generated according to the following Equation (13).

$$Weight(i,j) = \begin{cases} \frac{1}{Degree(j)}, & \text{if } Degree(j) \geq Degree(i) \\ 1 - \frac{Degree(j)}{Degree(i)}, & \text{otherwise} \end{cases} \tag{13}$$

This method is an improved version of the ratio model, where in ratio model the weight between two nodes is found with Equation (15). Here the influence between two nodes is dependent only on the node being influenced.

$$AverageDegree = \frac{\#ofEdges}{\#ofNodes} \tag{14}$$

$$Weight(i,j) = \sqrt{\frac{1}{Degree(j)}(\frac{1}{AverageDegree}Random(0,1))} \tag{15}$$

In the improved method, the average degree of a network is found by dividing number of edges to number of nodes. It is a fixed value. To change the weight values, this average degree value is multiplied with a random number. Then the geometric mean of the weight in the ratio model and the weight in average degree method is found and used as the weight value.

In the earlier experiments, we generate the threshold and weight values randomly. All the weights are uniformly distributed and a random number is selected between the values 0.30 and 0.60. The threshold values of the nodes are uniformly distributed and a random number is selected between the values 0.50 and 0.75. When we examine the results, we have seen that random nodes that have little degrees have more influence than popular nodes.

The number of k, which shows the size of the set, is another parameter to be determined. In our early experiments, k is selected as 1, 2, 5, 10, 20, 30, 50, 100, 200, 500, 1000. But we see that in some cases k too big causing the spread to cover all of the network in a few steps. According to the size of dataset, bigger k values may not be used. We select k so that it cannot be bigger than half of the number of nodes. It prevents unnecessary experiments because selecting more than half of the nodes is

unrealistic and the total influence will probably be the whole set.

5.2 Test Instances

We conducted our computational experiments on many instances as given in Table 3. Network column represent which network is used as the base network. The synthetic networks Barabasi-Albert and Watts-Strogatz are represented with letters BA and WS. For the real-life networks named Adjnoun, Celegansneural, Netscience and HEP-TH, their names are used in the table. Columns V and E show the number of nodes and edges in the given network. Networks are considered to be small if the number of nodes between 100 and 500 (Instances 1-93). Medium networks contain 1000 to 2000 nodes (Instances 94-165). Lastly, large networks have more than 5000 nodes and 10,000 edges (Instances 166-258). As mentioned in the previous part, seed set size can be one of the following numbers: 1, 2, 5, 10, 20, 30, 50, 100, 200, 500, 1000. According to the number of nodes in a network, larger seed set sizes may not be applicable for small networks. For small networks seed set size changes from 1 to 30 or 100, for medium networks seed set size changes from 1 to 200 and for large networks seed set size changes from 1 to 1000.

Column WT shows the weight and threshold generation methods. There are three different categories in this feature. Hybrid, Deg and Random. Deg is an abbreviation of Degree-based method [22] mentioned in Section 5.1.2. In the hybrid method [23], thresholds take a value between 0.7 and 1 according to truncated normal distribution and weight values are calculating according to Equation (15). When WT method is Deg, it means that weight values are generated according to Equation (13) and threshold values are generated according to Equation (12). In the instances where the column WT says Rand, it means that weight and threshold values are generated uniformly random. Weight values are between (0.30, 0.60) and threshold values are between (0.50, 0.75).

Table 3: Small-sized Test Instances

Instance	Network	V	E	WT	k	Instance	Graph	V	E	WT	k
1	BA	100	198	Deg	1	46	Celegansneural	239	1918	Deg	1
2	BA	100	198	Deg	2	47	Celegansneural	239	1918	Deg	2
3	BA	100	198	Deg	5	48	Celegansneural	239	1918	Deg	5
4	BA	100	198	Deg	10	49	Celegansneural	239	1918	Deg	10
5	BA	100	198	Deg	20	50	Celegansneural	239	1918	Deg	20
6	BA	100	198	Deg	30	51	Celegansneural	239	1918	Deg	30
7	BA	100	198	Hybrid	1	52	Celegansneural	239	1918	Deg	50
8	BA	100	198	Hybrid	2	53	Celegansneural	239	1918	Deg	100
9	BA	100	198	Hybrid	5	54	Celegansneural	239	1918	Hybrid	1
10	BA	100	198	Hybrid	10	55	Celegansneural	239	1918	Hybrid	2
11	BA	100	198	Hybrid	20	56	Celegansneural	239	1918	Hybrid	5
12	BA	100	198	Hybrid	30	57	Celegansneural	239	1918	Hybrid	10
13	WS	100	200	Deg	1	58	Celegansneural	239	1918	Hybrid	20
14	WS	100	200	Deg	2	59	Celegansneural	239	1918	Hybrid	30
15	WS	100	200	Deg	5	60	Celegansneural	239	1918	Hybrid	50
16	WS	100	200	Deg	10	61	Celegansneural	239	1918	Hybrid	100
17	WS	100	200	Deg	20	62	Celegansneural	239	1918	Rand	1
18	WS	100	200	Deg	30	63	Celegansneural	239	1918	Rand	2
19	WS	100	200	Hybrid	1	64	Celegansneural	239	1918	Rand	5
20	WS	100	200	Hybrid	2	65	Celegansneural	239	1918	Rand	10
21	WS	100	200	Hybrid	5	66	Celegansneural	239	1918	Rand	20
22	WS	100	200	Hybrid	10	67	Celegansneural	239	1918	Rand	30
23	WS	100	200	Hybrid	20	68	Celegansneural	239	1918	Rand	50
24	WS	100	200	Hybrid	30	69	Celegansneural	239	1918	Rand	100
25	Adjnoun	112	425	Deg	1	70	Netscience	379	914	Deg	1
26	Adjnoun	112	425	Deg	2	71	Netscience	379	914	Deg	2
27	Adjnoun	112	425	Deg	5	72	Netscience	379	914	Deg	5
28	Adjnoun	112	425	Deg	10	73	Netscience	379	914	Deg	10
29	Adjnoun	112	425	Deg	20	74	Netscience	379	914	Deg	20
30	Adjnoun	112	425	Deg	30	75	Netscience	379	914	Deg	30
31	Adjnoun	112	425	Deg	50	76	Netscience	379	914	Deg	50
32	Adjnoun	112	425	Hybrid	1	77	Netscience	379	914	Deg	100
33	Adjnoun	112	425	Hybrid	2	78	Netscience	379	914	Hybrid	1
34	Adjnoun	112	425	Hybrid	5	79	Netscience	379	914	Hybrid	2
35	Adjnoun	112	425	Hybrid	10	80	Netscience	379	914	Hybrid	5
36	Adjnoun	112	425	Hybrid	20	81	Netscience	379	914	Hybrid	10
37	Adjnoun	112	425	Hybrid	30	82	Netscience	379	914	Hybrid	20
38	Adjnoun	112	425	Hybrid	50	83	Netscience	379	914	Hybrid	30
39	Adjnoun	112	425	Rand	1	84	Netscience	379	914	Hybrid	50
40	Adjnoun	112	425	Rand	2	85	Netscience	379	914	Hybrid	100
41	Adjnoun	112	425	Rand	5	86	Netscience	379	914	Rand	1
42	Adjnoun	112	425	Rand	10	87	Netscience	379	914	Rand	2
43	Adjnoun	112	425	Rand	20	88	Netscience	379	914	Rand	5
44	Adjnoun	112	425	Rand	30	89	Netscience	379	914	Rand	10
45	Adjnoun	112	425	Rand	50	90	Netscience	379	914	Rand	20
						91	Netscience	379	914	Rand	30
						92	Netscience	379	914	Rand	50
						93	Netscience	379	914	Rand	100

Table 4: Medium-sized Test Instances

Instance	Network	V	E	WT	k	Instance	Graph	V	E	WT	k
94	BA	1000	1998	Deg	1	130	BA	2000	1999	Deg	1
95	BA	1000	1998	Deg	2	131	BA	2000	1999	Deg	2
96	BA	1000	1998	Deg	5	132	BA	2000	1999	Deg	5
97	BA	1000	1998	Deg	10	133	BA	2000	1999	Deg	10
98	BA	1000	1998	Deg	20	134	BA	2000	1999	Deg	20
99	BA	1000	1998	Deg	30	135	BA	2000	1999	Deg	30
100	BA	1000	1998	Deg	50	136	BA	2000	1999	Deg	50
101	BA	1000	1998	Deg	100	137	BA	2000	1999	Deg	100
102	BA	1000	1998	Deg	200	138	BA	2000	1998	Deg	200
103	BA	1000	1998	Hybrid	1	139	BA	2000	1999	Hybrid	1
104	BA	1000	1998	Hybrid	2	140	BA	2000	1999	Hybrid	2
105	BA	1000	1998	Hybrid	5	141	BA	2000	1999	Hybrid	5
106	BA	1000	1998	Hybrid	10	142	BA	2000	1999	Hybrid	10
107	BA	1000	1998	Hybrid	20	143	BA	2000	1999	Hybrid	20
108	BA	1000	1998	Hybrid	30	144	BA	2000	1999	Hybrid	30
109	BA	1000	1998	Hybrid	50	145	BA	2000	1999	Hybrid	50
110	BA	1000	1998	Hybrid	100	146	BA	2000	1999	Hybrid	100
111	BA	1000	1998	Hybrid	200	147	BA	2000	1998	Hybrid	200
112	WS	1000	2000	Deg	1	148	WS	2000	2000	Deg	1
113	WS	1000	2000	Deg	2	149	WS	2000	2000	Deg	2
114	WS	1000	2000	Deg	5	150	WS	2000	2000	Deg	5
115	WS	1000	2000	Deg	10	151	WS	2000	2000	Deg	10
116	WS	1000	2000	Deg	20	152	WS	2000	2000	Deg	20
117	WS	1000	2000	Deg	30	153	WS	2000	2000	Deg	30
118	WS	1000	2000	Deg	50	154	WS	2000	2000	Deg	50
119	WS	1000	2000	Deg	100	155	WS	2000	2000	Deg	100
120	WS	1000	2000	Deg	200	156	WS	2000	2000	Deg	200
121	WS	1000	2000	Hybrid	1	157	WS	2000	2000	Hybrid	1
122	WS	1000	2000	Hybrid	2	158	WS	2000	2000	Hybrid	2
123	WS	1000	2000	Hybrid	5	159	WS	2000	2000	Hybrid	5
124	WS	1000	2000	Hybrid	10	160	WS	2000	2000	Hybrid	10
125	WS	1000	2000	Hybrid	20	161	WS	2000	2000	Hybrid	20
126	WS	1000	2000	Hybrid	30	162	WS	2000	2000	Hybrid	30
127	WS	1000	2000	Hybrid	50	163	WS	2000	2000	Hybrid	50
128	WS	1000	2000	Hybrid	100	164	WS	2000	2000	Hybrid	100
129	WS	1000	2000	Hybrid	200	165	WS	2000	2000	Hybrid	200

Table 5: Large-sized Test Instances

Instance	Network	V	E	WT	k	Instance	Graph	V	E	WT	k
166	BA	5000	9998	Deg	1	210	HEP-TH	5835	13815	Deg	1
167	BA	5000	9998	Deg	2	211	HEP-TH	5835	13815	Deg	2
168	BA	5000	9998	Deg	5	212	HEP-TH	5835	13815	Deg	5
169	BA	5000	9998	Deg	10	213	HEP-TH	5835	13815	Deg	10
170	BA	5000	9998	Deg	20	214	HEP-TH	5835	13815	Deg	20
171	BA	5000	9998	Deg	30	215	HEP-TH	5835	13815	Deg	30
172	BA	5000	9998	Deg	50	216	HEP-TH	5835	13815	Deg	50
173	BA	5000	9998	Deg	100	217	HEP-TH	5835	13815	Deg	100
174	BA	5000	9998	Deg	200	218	HEP-TH	5835	13815	Deg	200
175	BA	5000	9998	Deg	500	219	HEP-TH	5835	13815	Deg	500
176	BA	5000	9998	Deg	1000	220	HEP-TH	5835	13815	Deg	1000
177	BA	5000	9998	Hybrid	1	221	HEP-TH	5835	13815	Hybrid	1
178	BA	5000	9998	Hybrid	2	222	HEP-TH	5835	13815	Hybrid	2
179	BA	5000	9998	Hybrid	5	223	HEP-TH	5835	13815	Hybrid	5
180	BA	5000	9998	Hybrid	10	224	HEP-TH	5835	13815	Hybrid	10
181	BA	5000	9998	Hybrid	20	225	HEP-TH	5835	13815	Hybrid	20
182	BA	5000	9998	Hybrid	30	226	HEP-TH	5835	13815	Hybrid	30
183	BA	5000	9998	Hybrid	50	227	HEP-TH	5835	13815	Hybrid	50
184	BA	5000	9998	Hybrid	100	228	HEP-TH	5835	13815	Hybrid	100
185	BA	5000	9998	Hybrid	200	229	HEP-TH	5835	13815	Hybrid	200
186	BA	5000	9998	Hybrid	500	230	HEP-TH	5835	13815	Hybrid	500
187	BA	5000	9998	Hybrid	1000	231	HEP-TH	5835	13815	Hybrid	1000
188	WS	5000	10000	Deg	1	232	HEP-TH	5835	13815	Rand	1
189	WS	5000	10000	Deg	2	233	HEP-TH	5835	13815	Rand	2
190	WS	5000	10000	Deg	5	234	HEP-TH	5835	13815	Rand	5
191	WS	5000	10000	Deg	10	235	HEP-TH	5835	13815	Rand	10
192	WS	5000	10000	Deg	20	236	HEP-TH	5835	13815	Rand	20
193	WS	5000	10000	Deg	30	237	HEP-TH	5835	13815	Rand	30
194	WS	5000	10000	Deg	50	238	HEP-TH	5835	13815	Rand	50
195	WS	5000	10000	Deg	100	239	HEP-TH	5835	13815	Rand	100
196	WS	5000	10000	Deg	200	240	HEP-TH	5835	13815	Rand	200
197	WS	5000	10000	Deg	500	241	HEP-TH	5835	13815	Rand	500
198	WS	5000	10000	Deg	1000	242	HEP-TH	5835	13815	Rand	1000
199	WS	5000	10000	Hybrid	1	243	BA	10000	10000	Deg	1
200	WS	5000	10000	Hybrid	2	244	BA	10000	10000	Deg	10
201	WS	5000	10000	Hybrid	5	245	BA	10000	10000	Deg	100
202	WS	5000	10000	Hybrid	10	246	BA	10000	10000	Deg	1000
203	WS	5000	10000	Hybrid	20	247	BA	10000	10000	Hybrid	1
204	WS	5000	10000	Hybrid	30	248	BA	10000	10000	Hybrid	10
205	WS	5000	10000	Hybrid	50	249	BA	10000	10000	Hybrid	100
206	WS	5000	10000	Hybrid	100	250	BA	10000	10000	Hybrid	1000
207	WS	5000	10000	Hybrid	200	251	WS	10000	10000	Deg	1
208	WS	5000	10000	Hybrid	500	252	WS	10000	10000	Deg	10
209	WS	5000	10000	Hybrid	1000	253	WS	10000	10000	Deg	100
						254	WS	10000	10000	Deg	1000
						255	WS	10000	10000	Hybrid	1
						256	WS	10000	10000	Hybrid	10
						257	WS	10000	10000	Hybrid	100
						258	WS	10000	10000	Hybrid	1000

5.3 Numerical Results

All computational experiments are done on a computer with 32GB RAM and 3.40 GHZ Intel Core i7-3770 CPU in Windows 8.1 operating system. The algorithms are implemented in Python programming language version 3.7. Solver CPLEX 22 is used and connected to python via Pyomo library.

Exact solutions are obtained by CPLEX. It is very efficient when the dataset is small. But most of the real-life datasets are big and takes time to be solved exactly. For smaller networks, CPLEX is used as a comparison method. In order to use CPLEX in Python, Pyomo library [25] [26] is used. With HEP-TH dataset, which has 5,835 nodes, the process of selecting seed set may reach almost 18 hours. In order to prevent this, a time limit of 1 hour is determined. The fitness function in Python uses Pyomo to calculate how many nodes will be active at the end of the time period when a seed set is given.

As mentioned above, our algorithm consists of two parts: ESS and local search. The initial seed set from ESS is used as an input in local search. ESS+LS represents when two algorithms are used together consecutively.

5.3.1 Comparison of All Algorithms

As explained in Section 4, before finalizing the solution method, we developed many algorithms. The earlier work starts with an empty set and step by step forms an initial seed set. They are a type of constructive heuristic and the abbreviated form is CH. In the tables, to be more obvious, they are numbered from 1 to 3 as CH-1, CH-2 and CH-3. CH-1 is the first version of the algorithm. CH-2 is an earlier version of ESS that only produces an initial set. CH-3 is very similar to local search algorithm. It takes an initial set and makes improvement on it. More specifically, it takes the output of CH-2 as an input and improves it to get better results. It calculates the total influence of each set using CPLEX. ESS+LS is the final version, it holds ESS

and local search together. Time spent of ESS+LS is the total time of both initial set of extended seed set and the local search. We compared the performances of all 4 algorithms in Table 6.

CH-2 is the best version among all constructive heuristics. It is very similar to ESS algorithm. The major difference between two algorithms is the metric used node selection. In ESS, we use cosine similarity where in Chapter 4 the number of mutual neighbors are used. In this metric when two nodes are compared to be in a set with a third node, the number of mutual neighbors is counted and the higher one is selected to be in the seed set.

The average time equals to 2178.0, 7.7, 1821.9, 75.6 seconds and the average influence spread equals to 161, 152, 152, 167 of algorithms CH-1, CH-2, CH-3 and ESS+LS respectively in Table 6. ESS+LS has the highest average influence spread among the solution methods. When we compare the time spent of them, we can see that the fastest one is CH-2. The main reason of it is the number of iterations is very low and the algorithm is not trying hard enough to improve the set. It can be useful on small instances but it may not be as beneficial in the larger networks. In most of the experiments, ESS+LS found a better solution than others and its average time is 24 and 29 times better than CH-1 and CH-3.

Table 6: Performance Comparison of Previous Algorithms

	Objective Value				Time (sec)			
Instance	CH-1	CH-2	CH-3	ESS+LS	CH-1	CH-2	CH-3	ESS+LS
25	112	112	112	112	47.5	3.8	29.2	3.9
26	112	112	112	112	79.4	3.6	49.6	4.4
27	112	112	112	112	240.9	3.7	184.3	4.7
28	112	112	112	112	625.1	3.7	543.1	4.5
29	112	112	112	112	894.0	4.2	636.9	4.2
30	112	112	112	112	1011.0	4.6	527.3	4.5
31	112	112	112	112	1122.5	5.2	19.8	4.1
46	29	29	29	29	54.7	7.2	42.1	87.2
47	51	43	43	45	106.0	7.9	87.5	82.9
48	69	69	69	68	400.4	8.2	356.8	85.2
49	102	120	120	150	1644.1	7.4	1578.9	105.8
50	201	210	210	212	4422.3	8.5	4213.4	123.5
51	210	215	215	215	7386.2	9.2	6978.1	156.6
52	215	218	218	216	9756.8	9.8	8776.0	229.1
53	225	230	230	227	4934.2	18.6	1890.0	682.9
70	165	64	64	165	56.2	8.1	39.9	0.4
71	166	166	166	166	110.8	8.7	93.8	0.6
72	237	166	166	298	375.1	8.8	344.7	0.8
73	304	256	256	304	1217.6	11.0	1177.5	0.9
74	304	304	304	304	3312.8	8.9	3200.4	0.9
76	325	317	317	324	7941.4	10.2	7489.9	1.0

5.3.2 Performance Comparison with CPLEX Solutions

We implemented our heuristic method (ESS+LS) and compared with CPLEX results in order to validate the performance of the heuristic. In this section we present the comparison of ESS, LS and CPLEX results before and after local search is applied. +LS shows the case when local search is applied to the result of ESS. Tables 7, 8 and 9 present the performance of the CPLEX, ESS algorithm and Local Search. Instance column refers to Tables 3, 4 and 5 for network, size and number of nodes in the seed set. Objective value is the number of nodes that are activated at the end of the period. Column GAP shows the difference between the CPLEX solution and +LS solution and calculated by Equation (16). σ represents the objective value found by respective method. Improvement column shows how much improvement local search algorithm makes over ESS seed set. It is calculated via Equation (17). In each table, at the average values of GAP, Improvement, time duration is for respective table at the last row.

$$GAP = \frac{\sigma_{CPLEX} - \sigma_{LS}}{\sigma_{CPLEX}} \tag{16}$$

$$Improvement = \frac{\sigma_{ESS} - \sigma_{LS}}{\sigma_{ESS}} \tag{17}$$

Table 7 shows the performance for small scale datasets, between instances 1-93. It can be seen that, there is only one case where ESS+LS outperforms CPLEX which is Instance 72 (Note that CPLEX results are limited to one hour). When small scale instances are taken into consideration, there is a gap of 14.6% between ESS+LS and CPLEX. On the other hand, our algorithm 9 times faster than CPLEX on average.

The performance of medium scale instances are given in Table 8. There are several cases where ESS+LS found a better solution than CPLEX. In ESS+LS, the sets with maximum set degree are stored and at the end the real values of the max sets are calculated. The best one gets selected and returned as solution. When k is larger,

calculating the real values takes too much time. It is explained in Section 4 that there is a time limit in iteration process and it is set to 1 hour just like in the CPLEX.

Table 9 shows the performance for large scale datasets, between instances 166-258. ESS+LS finds a better solution than CPLEX for some of instances. For example, the GAP value becomes negative for Instance 167-175. Negative GAP value means that the influence spread of LS is higher than the influence spread of CPLEX. When $k = 1000$, calculating the real influence spread takes too much time. So, another average value row is added to the end of the Table 9 to show the average values for instances where $k < 1000$.

In the tests with larger datasets, CPLEX has difficulties with finding optimal sets. We follow 2 different approaches to deal with these. First, as mentioned above, a time limit of one hour is given to CPLEX to prepare an optimal answer. It can finish before the time limit but cannot exceed it. In some cases, with larger networks, CPLEX may give inconsistent results while triggering a warning message which says that CPLEX aborted but containing a solution. For example, the influence spread can be lower when k gets higher. Thus, if the time that CPLEX spends is more than 1 hour, we write 3600 seconds on the performance comparison tables. In the second approach, CPLEX takes an input as the initial seed set from the previous solutions (when k was smaller) and builds its solution around that. For instance, when k is 10, it got confused and provided a solution which is smaller than the one it found when k was 5. So, we give the solution from when k was 5 as an initial solution and CPLEX used it while building a set for $k = 10$, for the Instances between 210-220. This was necessary, especially in the first 2 trials, CPLEX did not give a solution other than an empty set. We had to run it again to obtain a solution. Instance 213 and 216 performed poorly and get objective values of 954 and 995 were obtained respectively. Then, we fed the solution from the previous instance and made it build the solution on top of the given one. The objective values increased to 4789 and 4837, respectively.

Table 7: CPLEX, ESS and ESS+LS performance comparison on small-sized instances.

	Objective Value					Time (sec)		
Instance	CPLEX	ESS	+LS	GAP (%)	Imp (%)	CPLEX	ESS	+LS
1	1	1	1	0.0	0	0.7	2.1	32.0
2	3	2	2	33.3	0	10.6	2.1	35.9
3	40	5	40	0.0	700	137.7	2.5	39.8
4	59	56	56	5.1	0	3600.0	2.2	44.2
5	86	65	72	16.3	11	3600.0	2.3	81.4
6	98	72	82	16.3	14	908.9	2.4	140.7
7	5	5	5	0.0	0	0.3	2.0	30.7
8	9	6	8	11.1	33	0.2	1.8	27.2
9	18	12	14	22.2	17	0.2	2.1	38.6
10	29	25	26	10.3	4	0.2	1.8	56.7
11	49	44	46	6.1	5	0.2	1.8	88.3
12	60	60	60	0.0	0	0.2	1.7	53.0
13	1	1	1	0.0	0	0.5	2.0	33.8
14	3	2	3	0.0	50	6.0	2.0	35.2
15	10	6	6	40.0	0	125.9	2.1	38.3
16	21	11	11	47.6	0	1271.0	2.1	49.3
17	39	25	25	35.9	0	3600.0	2.1	68.0
18	59	38	38	35.6	0	3600.0	2.1	174.7
19	2	1	1	50.0	0	0.3	1.6	28.0
20	3	2	2	33.3	0	0.4	1.7	29.2
21	10	5	5	50.0	0	0.6	1.9	45.9
22	20	11	12	40.0	9	0.8	1.6	43.9
23	39	25	25	35.9	0	1.6	1.8	85.3
24	55	35	35	36.4	0	14.7	1.7	166.8
25	112	112	112	0.0	0	1.7	3.9	0.0
26	112	112	112	0.0	0	1.5	4.4	0.0
27	112	112	112	0.0	0	1.6	4.7	0.0
28	112	112	112	0.0	0	1.5	4.5	0.0
29	112	112	112	0.0	0	1.6	4.2	0.0
30	112	112	112	0.0	0	1.6	4.5	0.0
32	1	1	1	0.0	0	0.6	2.2	30.0
33	2	2	2	0.0	0	2.5	2.1	32.3
34	6	5	5	16.7	0	3600.0	2.1	44.7
35	23	20	20	13.0	0	3600.0	2.1	104.9
36	45	40	40	11.1	0	1764.8	2.2	82.1
37	65	54	54	16.9	0	2.3	2.3	116.8
39	69	69	69	0.0	0	0.7	2.4	32.7
40	85	49	85	0.0	1	0.8	2.3	33.7
41	97	91	91	6.2	0	0.9	2.2	48.3
42	105	98	98	6.7	0	0.5	2.3	84.7
43	112	101	101	9.8	0	0.6	2.2	114.3
44	112	103	103	8.0	0	0.5	2.4	161.1
45	112	108	108	3.6	0	0.6	2.4	504.6

	Objective Value					Time (sec)		
Instance	CPLEX	ESS	+LS	GAP (%)	Imp (%)	CPLEX	ESS	+LS
46	29	29	29	0.0	0	8.0	9.2	78.0
47	51	35	45	11.8	29	3600.0	8.4	74.5
48	71	66	68	4.2	3	3600.0	8.6	76.7
49	177	150	150	15.3	0	3600.0	9.0	96.8
50	226	210	212	6.2	1	3600.0	9.1	114.4
51	239	213	215	10.0	1	19.2	10.2	146.4
52	239	216	216	9.6	0	4.2	9.5	219.6
53	239	227	227	5.0	0	4.2	9.8	673.1
54	1	1	1	0.0	0	3.9	7.5	64.7
55	2	2	2	0.0	0	3600.0	7.6	65.3
56	5	5	5	0.0	0	3600.0	7.5	70.3
57	10	10	10	0.0	0	3600.0	8.0	77.4
58	20	20	20	0.0	0	3600.0	7.8	104.5
59	39	30	30	23.1	0	3600.0	7.9	143.1
60	200	109	109	45.5	0	3600.0	8.2	435.0
61	239	186	194	18.8	4	37.5	8.5	2214.4
62	1	1	1	0.0	0	3.8	8.1	60.0
63	239	239	239	0.0	0	5.8	8.5	0.0
64	239	239	239	0.0	0	4.4	8.2	0.0
65	239	239	239	0.0	0	4.5	7.8	0.0
66	239	239	239	0.0	0	3.6	8.4	0.0
67	239	239	239	0.0	0	3.8	8.2	0.0
68	239	239	239	0.0	0	3.7	8.6	0.0
69	239	239	239	0.0	0	3.8	8.3	0.0
70	165	64	165	0.0	158	6.2	8.4	64.8
71	219	166	166	24.2	0	3600.0	8.5	73.4
72	295	237	298	-1.0	26	3600.0	8.2	79.2
73	335	290	304	9.3	5	3600.0	9.0	85.2
74	360	304	304	15.6	0	3600.0	9.0	97.3
75	377	315	317	15.9	1	3600.0	8.5	136.6
76	379	317	324	14.5	2	8.2	9.1	168.5
77	379	333	335	11.6	1	3.9	9.6	370.3
78	1	1	1	0.0	0	1.5	7.4	61.9
79	2	2	2	0.0	0	8.0	7.5	65.3
80	8	5	5	37.5	0	26.9	7.1	72.0
81	16	10	10	37.5	0	1283.6	7.0	108.0
82	31	20	20	35.5	0	3600.0	6.8	120.3
83	46	34	34	26.1	0	3600.0	7.0	217.3
84	73	58	58	20.5	0	3600.0	6.9	386.2
85	136	117	117	14.0	0	3600.0	7.4	1283.1
86	17	7	13	23.5	1	1.1	6.8	52.4
87	26	18	20	23.1	0	63.8	7.0	61.3
88	59	35	37	37.3	0	3600.0	6.0	74.6
89	87	68	68	21.8	0	3600.0	6.8	100.0
90	131	95	99	24.4	0	3600.0	6.1	101.6
91	163	121	122	25.2	0	3600.0	6.2	116.7
92	208	151	160	23.1	0	3600.0	6.3	296.9
93	278	219	228	18.0	0	5.3	7.9	1826.3
			Average:	13.5	11.8	1250.4	5.3	149.3

Table 8: CPLEX, ESS and ESS+LS performance comparison on medium-sized instances.

	Objective Value					Time (sec)				Objective Value					Time (sec)		
Instance	CPLEX	ESS	+LS	GAP (%)	Imp (%)	CPLEX	ESS	+LS	Instance	CPLEX	ESS	+LS	GAP (%)	Imp (%)	CPLEX	ESS	+LS
94	1	1	1	0.0	0	3396.3	23.8	179.4	130	1	1	1	0.0	0	3600.0	56.7	364.0
95	108	103	103	4.6	0	3600.0	25.5	166.5	131	115	143	143	-24.3	0	3600.0	54.4	345.4
96	142	175	176	-23.9	1	3600.0	24.2	166.1	132	215	441	441	-105.1	0	3600.0	53.9	345.9
97	304	309	317	-4.3	3	3600.0	24.0	169.5	133	499	493	493	1.2	0	3600.0	54.4	348.7
98	362	352	369	-1.9	5	3600.0	24.4	185.7	134	527	544	582	-10.4	7	3600.0	65.9	365.6
99	439	393	428	2.5	9	3600.0	24.1	209.3	135	650	600	652	-0.3	9	3600.0	55.0	387.7
100	570	471	472	17.2	0	3600.0	24.9	253.5	136	810	716	734	9.4	3	3600.0	55.2	445.2
101	737	563	587	20.4	4	3600.0	24.8	875.5	137	1098	934	967	11.9	4	3600.0	69.2	797.5
102	927	647	735	20.7	14	3600.0	26.0	4033.3	138	1378	1142	1185	14.0	4	3600.0	58.1	3415.7
103	8	7	8	0.0	14	4.5	17.6	124.8	139	17	17	17	0.0	0	14.1	37.7	244.6
104	17	16	17	0.0	6	4.2	17.5	126.0	140	27	26	27	0.0	4	12.1	37.6	245.1
105	37	35	35	5.4	0	4.5	17.5	129.4	141	52	51	51	1.9	0	12.2	37.8	255.6
106	60	49	53	11.7	8	4.4	18.2	160.3	142	82	81	81	1.2	0	12.3	38.7	252.5
107	99	88	89	10.1	1	4.6	17.6	185.6	143	127	114	122	3.9	7	12.2	38.7	388.9
108	131	113	116	11.5	3	4.5	17.8	268.2	144	167	145	154	7.8	6	12.5	37.6	385.1
109	189	158	169	10.6	7	4.8	18.0	256.3	145	234	196	223	4.7	14	12.4	38.0	667.1
110	295	265	280	5.1	6	6.3	18.4	1265.3	146	382	329	357	6.5	9	12.5	38.5	2147.4
111	486	438	458	5.8	5	3600.0	18.9	2841.3	147	593	533	558	5.9	5	12.1	39.1	4255.4
112	1	1	1	0.0	0	74.0	18.1	126.5	148	1	1	1	0.0	0	90.1	41.0	262.5
113	4	2	2	50.0	0	3600.0	17.7	128.3	149	3	2	2	33.3	0	3600.0	40.6	269.1
114	11	5	6	45.5	20	3600.0	17.9	130.7	150	10	5	5	50.0	0	3600.0	41.9	287.2
115	19	12	12	36.8	0	3600.0	18.5	140.3	151	23	10	10	56.5	0	3600.0	41.9	277.8
116	42	24	24	42.9	0	3600.0	17.9	174.1	152	43	23	23	46.5	0	3600.0	42.0	294.0
117	57	35	35	38.6	0	3600.0	17.9	161.3	153	55	35	35	36.4	0	3600.0	42.0	346.3
118	102	55	55	46.1	0	3600.0	18.6	294.0	154	97	55	55	43.3	0	3600.0	42.4	434.8
119	187	109	109	41.7	0	3600.0	18.3	849.4	155	193	105	105	45.6	0	3600.0	41.1	702.7
120	353	219	219	38.0	0	3600.0	19.6	36540.3	156	355	211	211	40.6	0	3600.0	43.3	2657.4
121	2	1	2	0.0	100	3.8	22.4	128.5	157	2	1	2	0.0	100	52.2	37.7	243.3
122	6	3	3	50.0	0	7.8	17.9	130.4	158	5	3	3	40.0	0	32.6	37.4	244.6
123	13	7	9	30.8	29	22.2	17.6	133.5	159	12	6	8	33.3	33	313.8	37.6	250.2
124	24	14	15	37.5	7	44.2	17.5	177.3	160	23	11	13	43.5	18	3600.0	37.7	276.1
125	45	26	30	33.3	15	25.8	17.5	163.3	161	45	22	26	42.2	18	3600.0	37.9	338.9
126	65	40	45	30.8	13	20.4	17.9	254.7	162	65	35	38	41.5	9	456.6	37.7	383.7
127	105	69	69	34.3	0	18.8	17.8	519.2	163	105	60	62	41.0	3	509.1	37.9	520.8
128	202	131	131	35.1	0	3600.0	17.9	826.2	164	205	122	122	40.5	0	426.6	38.4	1408.7
129	375	248	248	33.9	0	3600.0	18.7	4086.6	165	388	238	238	38.7	0	3600.0	38.7	4097.7
												Average:	18.4	7.2	2028.4	31.9	1187.7

Table 9: CPLEX, ESS and ESS+LS performance comparison on large-sized instances.
*For k values smaller than 1000

	Objective Value					Time (sec)		
Instance	CPLEX	ESS	+LS	GAP (%)	Imp (%)	CPLEX	ESS	+LS
166	1	1	1	0.0	0	3600.0	202.9	935.1
167	123	136	136	-10.6	0	3600.0	183.9	1156.8
168	202	337	387	-91.6	15	3600.0	210.5	1276.4
169	716	828	829	-15.8	0	3600.0	185.6	1181.4
170	846	988	1070	-26.5	8	3600.0	211.4	1314.2
171	923	1147	1193	-29.3	4	3600.0	211.5	1334.3
172	1057	1458	1491	-41.1	2	3600.0	210.9	1369.2
173	1432	1771	1898	-32.5	7	3600.0	212.3	1493.6
174	1159	2325	2393	-106.5	3	3600.0	177.3	2865.9
175	2194	2956	3072	-40.0	4	3600.0	181.6	4809.9
176	4616	3356	3554	23.0	6	3600.0	208.1	16193.7
177	20	20	20	0.0	0	63.6	151.3	740.5
178	36	33	33	8.3	0	63.3	125.0	779.4
179	74	68	72	2.7	6	63.3	125.3	773.2
180	118	111	112	5.1	1	63.8	124.9	778.7
181	187	169	179	4.3	6	66.4	125.5	843.6
182	239	214	226	5.4	6	65.1	125.4	917.0
183	330	282	301	8.8	7	65.0	125.2	1084.5
184	528	441	474	10.2	7	64.8	125.9	2119.1
185	836	710	758	9.3	7	64.8	127.4	4938.9
186	1520	1377	1377	9.4	0	68.1	129.2	4496.1
187	2492	2220	2239	10.2	1	3600.0	131.9	14688.1
188	1	1	1	0.0	0	2049.3	132.9	811.2
189	3	2	2	33.3	0	3600.0	132.5	828.1
190	10	5	5	50.0	0	3600.0	132.4	817.2
191	20	10	10	50.0	0	3600.0	132.8	822.9
192	43	20	20	53.5	0	3600.0	132.4	828.4
193	52	30	30	42.3	0	3600.0	132.7	862.4
194	82	53	53	35.4	0	3600.0	133.0	905.3
195	194	112	112	42.3	0	3600.0	133.1	1242.0
196	346	220	220	36.4	0	3600.0	133.7	2588.2
197	832	533	533	35.9	0	3600.0	132.2	4432.9
198	1750	1099	1099	37.2	0	3600.0	143.5	26083.2
199	2	2	2	0.0	0	72.5	125.1	774.5
200	7	4	4	42.9	0	209.9	124.2	766.2
201	16	8	10	37.5	25	3600.0	123.8	778.1
202	29	14	16	44.8	14	3600.0	124.3	799.4
203	54	28	31	42.6	11	3600.0	124.6	876.6
204	78	40	44	43.6	10	3600.0	125.1	865.7
205	125	63	67	46.4	6	3600.0	124.5	1118.2
206	229	123	131	42.8	7	3600.0	124.8	2093.6
207	429	249	250	41.7	0	3600.0	125.7	4926.3
208	1009	600	600	40.5	0	3600.0	127.2	4461.4
209	1866	1219	1219	34.7	0	3600.0	130.0	14338.7
210	4573	4577	4577	-0.1	0	3600.0	335.6	1616.6
211	4575	4681	4704	-2.8	0	3600.0	255.4	1532.9
212	4579	4761	4762	-4.0	0	3600.0	278.5	1540.2
213	954	4787	4789	-402.0	0	3600.0	255.6	1558.4
214	4601	4810	4810	-4.5	0	3600.0	277.6	1596.5
215	4617	4822	4823	-4.5	0	3600.0	255.3	1717.3
216	995	4835	4837	-386.1	0	3600.0	257.0	1886.5
217	3836	4865	4865	-26.8	0	3600.0	257.3	2129.7
218	5150	4902	4902	4.8	0	3600.0	265.4	10203.4
219	5753	5035	5035	12.5	0	3600.0	269.8	19163.3
220	5835	5127	5127	12.1	0	1113.4	300.8	32430.6
221	2	1	1	50.0	0	118.6	190.8	1298.0
222	10	2	2	80.0	0	3600.0	219.2	1194.4
223	17	5	5	70.6	0	3600.0	169.3	1076.2
224	27	10	10	63.0	0	3600.0	169.4	1389.2
225	57	26	26	54.4	0	3600.0	229.4	1761.6
226	85	53	53	37.6	0	3600.0	232.0	1771.8
227	151	101	101	33.1	0	3600.0	236.4	1941.5
228	309	250	250	19.1	0	3600.0	232.6	4804.5
230	1066	872	872	18.2	0	3600.0	176.5	8446.2
231	1799	1475	1475	18.0	0	3600.0	184.0	30344.5
232	162	22	83	48.8	3	3600.0	176.6	1084.9
233	630	30	168	73.3	5	3600.0	173.0	1079.0
234	737	601	601	18.5	0	3600.0	164.6	1082.2
235	935	923	923	1.3	0	3600.0	177.3	1322.8
236	1291	1062	1062	17.7	0	3600.0	164.5	1195.9
237	1561	1147	1255	19.6	0	3600.0	164.6	1332.3
238	1954	1370	1497	23.4	0	3600.0	165.1	1957.2
239	2552	1886	1944	23.8	0	3600.0	176.6	4727.7
240	3223	2294	2367	26.6	0	3600.0	166.8	5757.9
241	4079	3015	3015	26.1	0	3600.0	171.2	8434.0
			Average:	6.1	2.3	2491.5	105.5	2591.8
			Average*:	4.7	2.4	2914.5	175.6	2339.3

Figures 4-27 show the influence spread and time graphs. The horizontal axis (x-axis) shows the number of nodes in the seed set which is k value. The vertical axis (y-axis) shows the influence spread and the time duration (in seconds). Blue, gray and red line represents CPLEX, ESS and LS respectively. The names of the networks are given after the real-life or synthetic network it is based on. There are two different weight/threshold generation methods used in Figures 4-27. If the weight and threshold are generated with Degree-based method the number 1 is added and if they are generated with Hybrid method then number 2 is added at the end of the name of the algorithm.

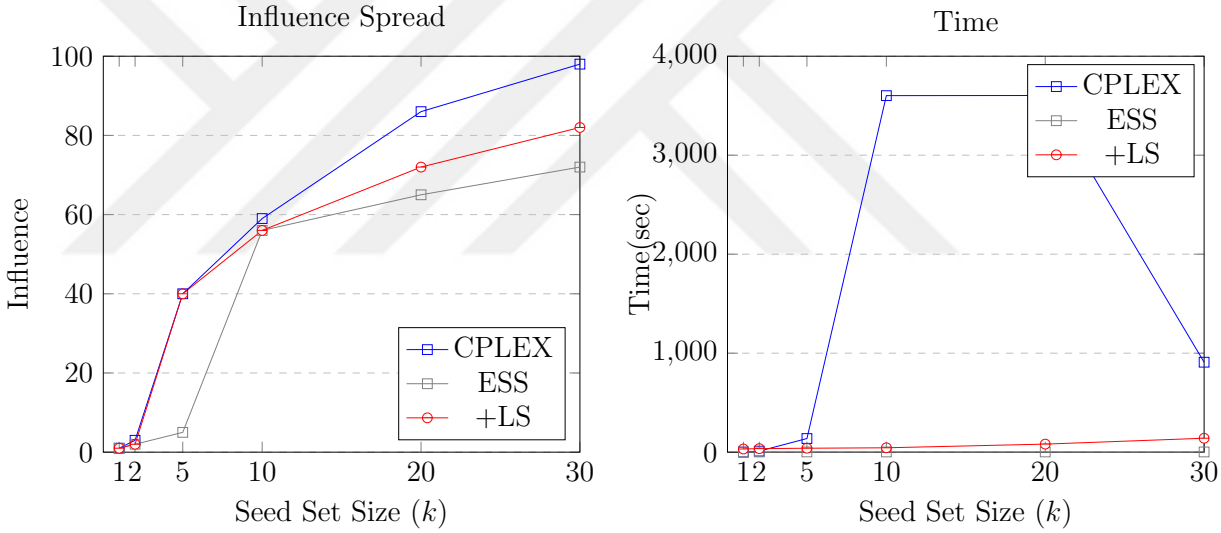


Figure 4: Influence and time graph of Barabasi-Small-1 dataset

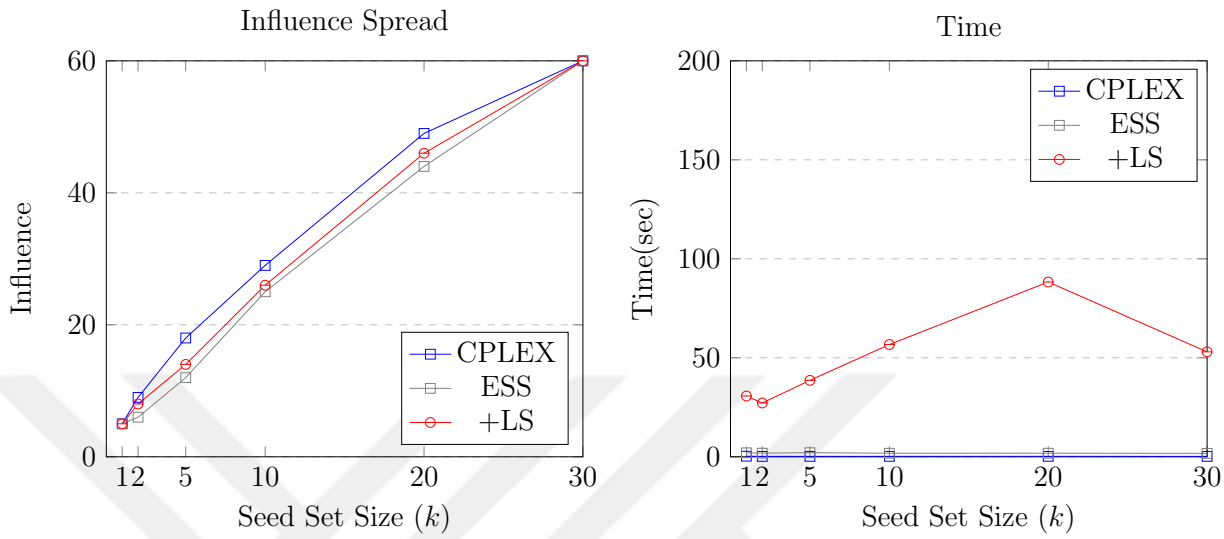


Figure 5: Influence and time graph of Barabasi-Small-2 dataset

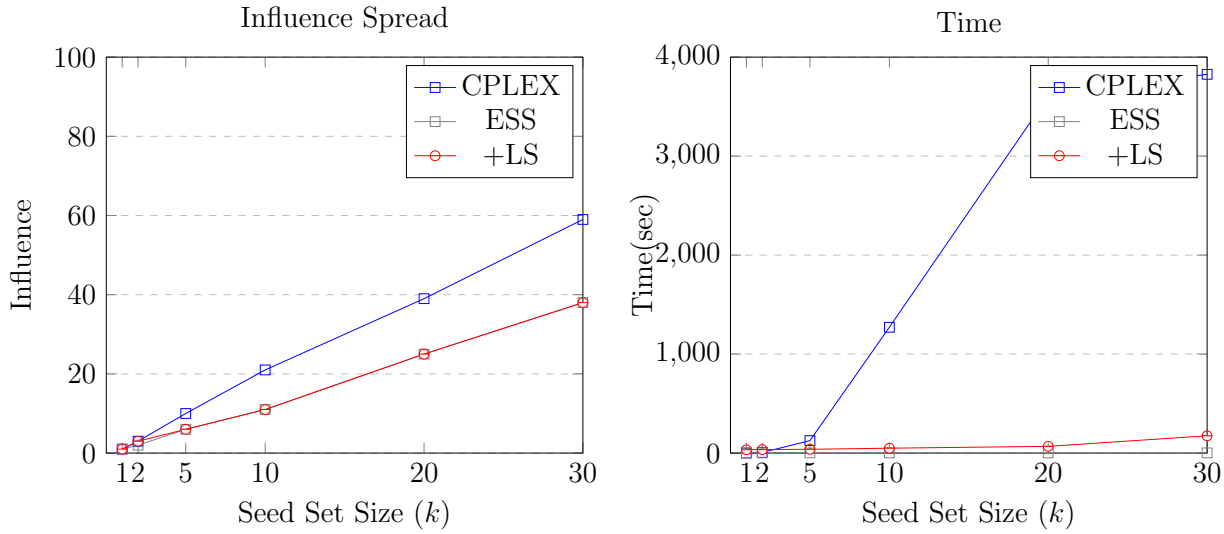


Figure 6: Influence and time graph of Watts-Small-1 dataset

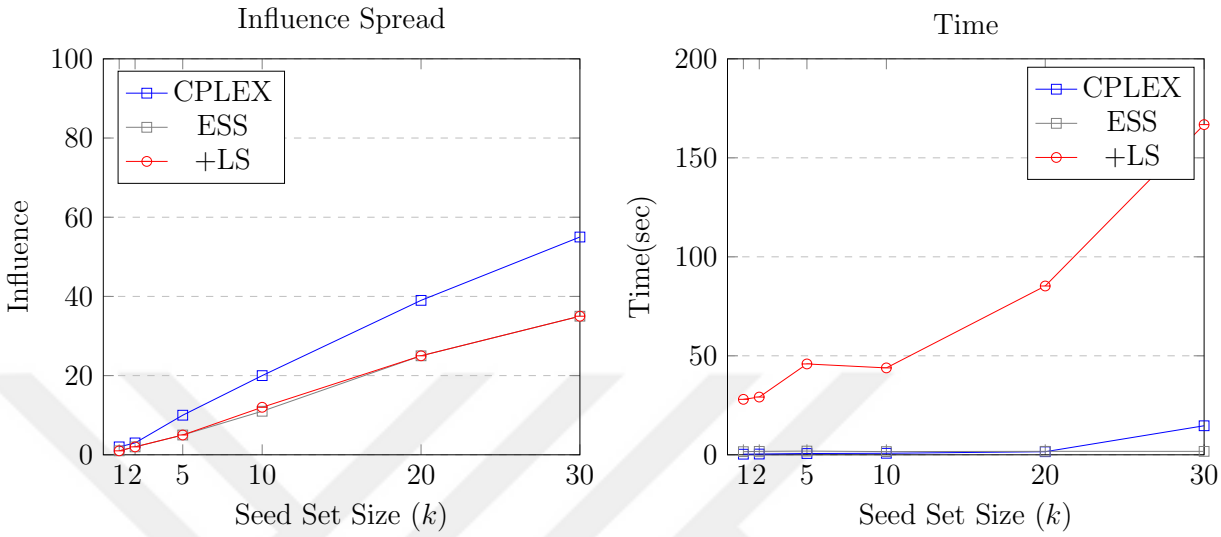


Figure 7: Influence and time graph of Watts-Small-2 dataset

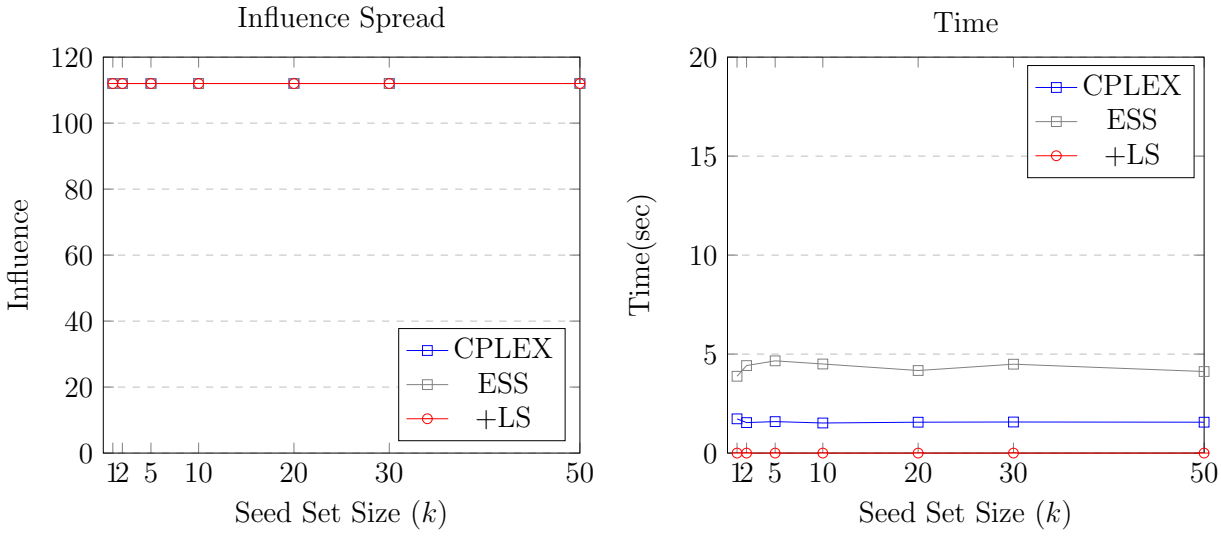


Figure 8: Influence and time graph of Adjnoun-1 dataset

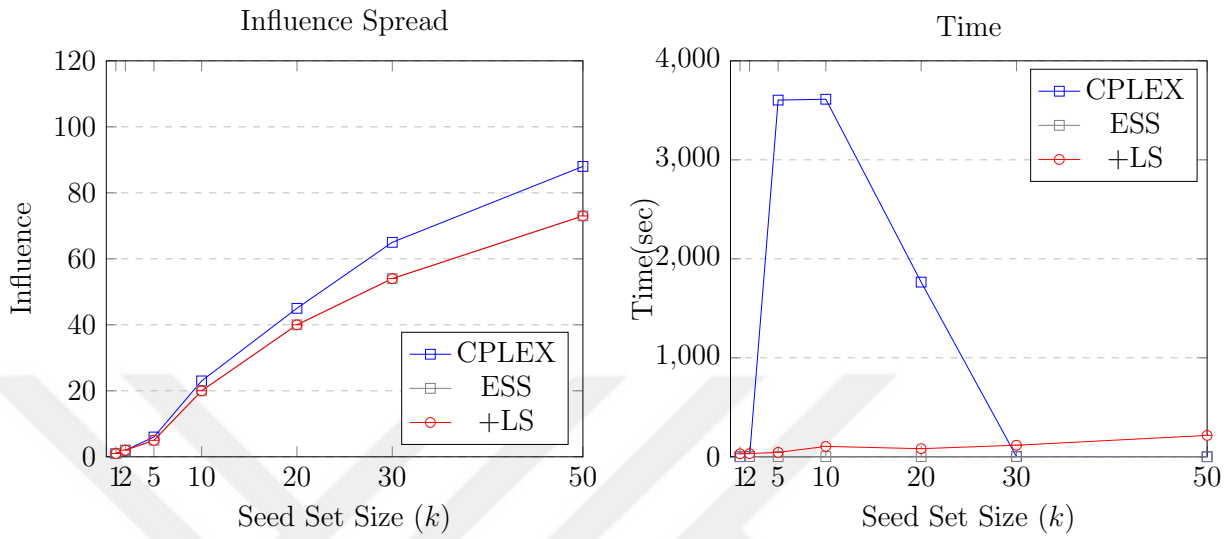


Figure 9: Influence and time graph of Adjnoun-2 dataset

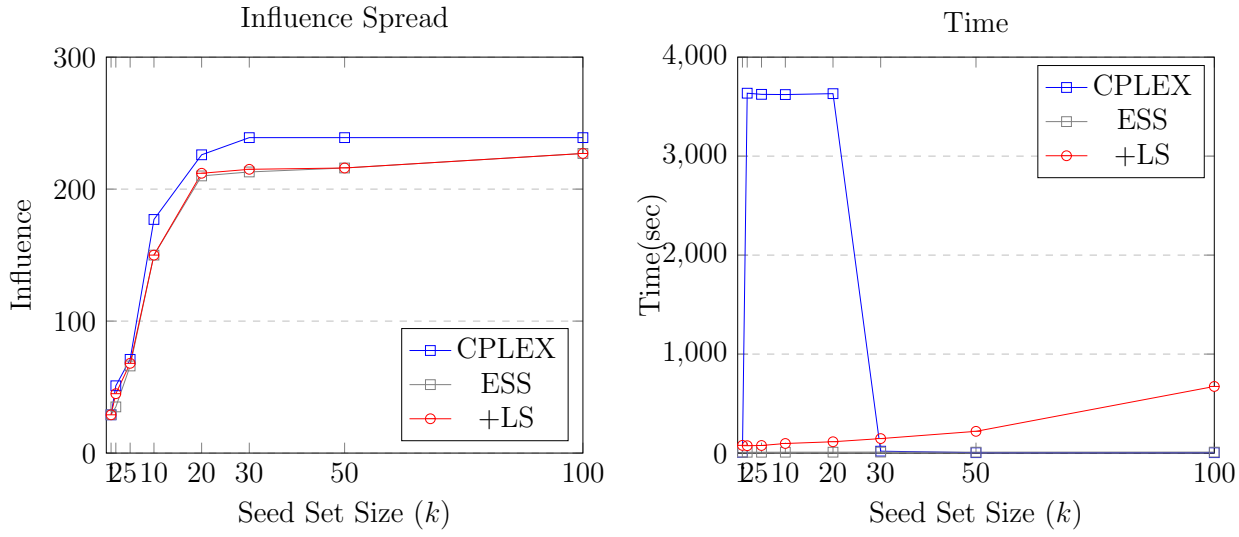


Figure 10: Influence and time graph of Celegansneural-1 dataset

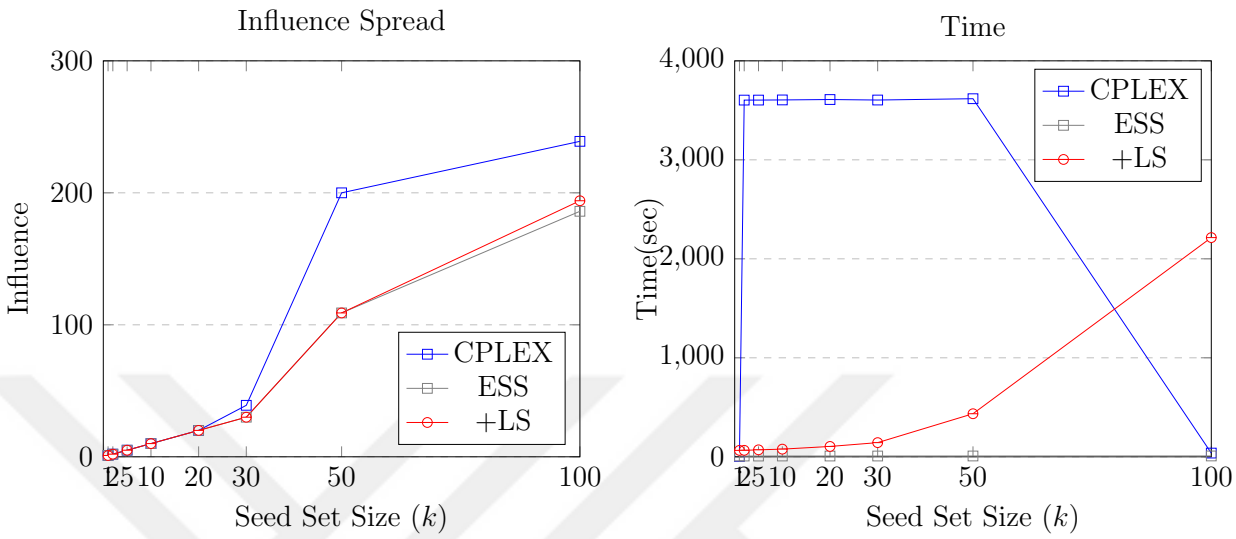


Figure 11: Influence and time graph of Celegansneural-2 dataset

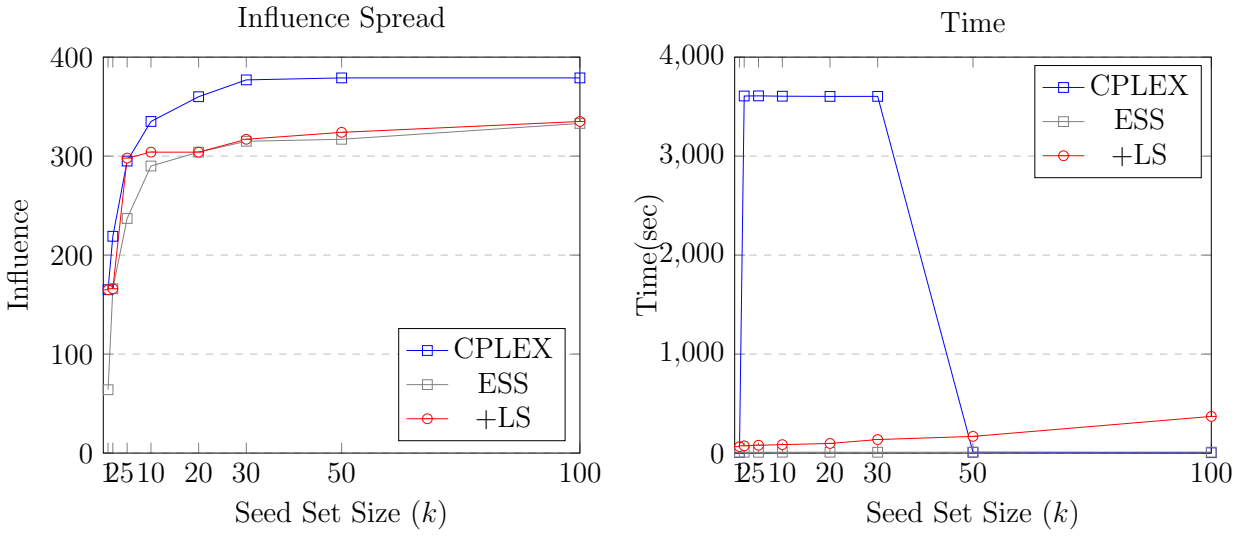


Figure 12: Influence and time graph of Netscience-1 dataset

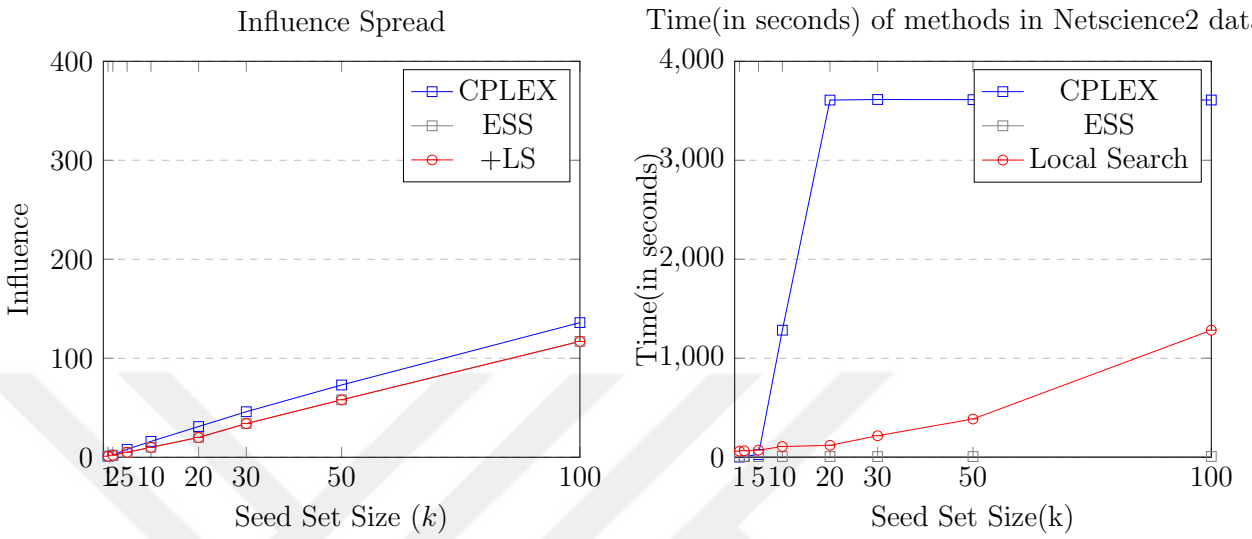


Figure 13: Influence and time graph of Netscience-2 dataset

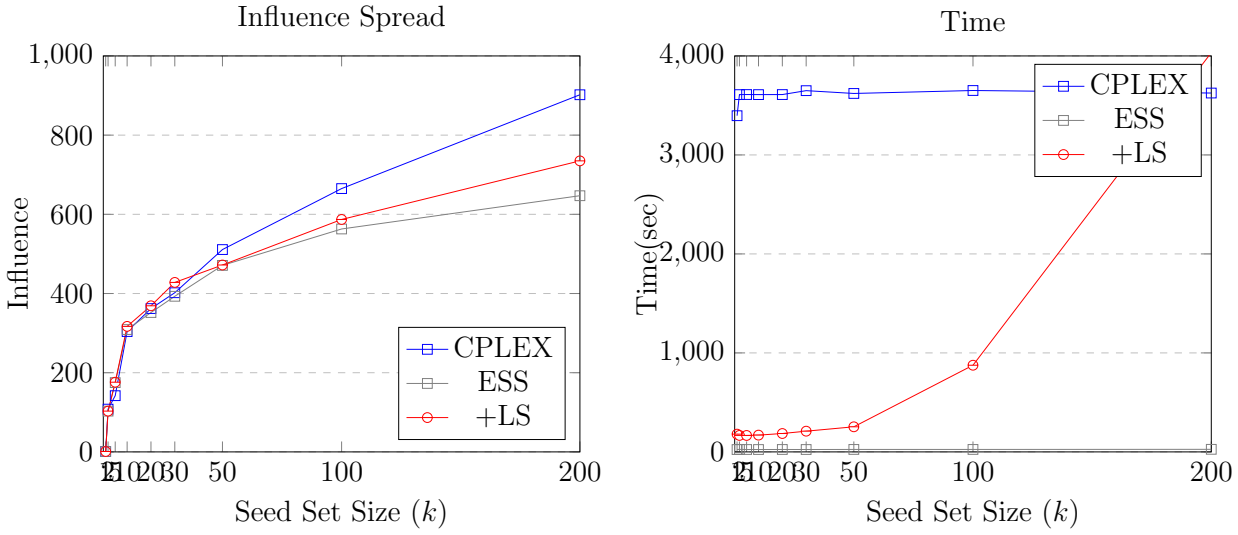


Figure 14: Influence and time graph of Barabasi-Medium-1 dataset

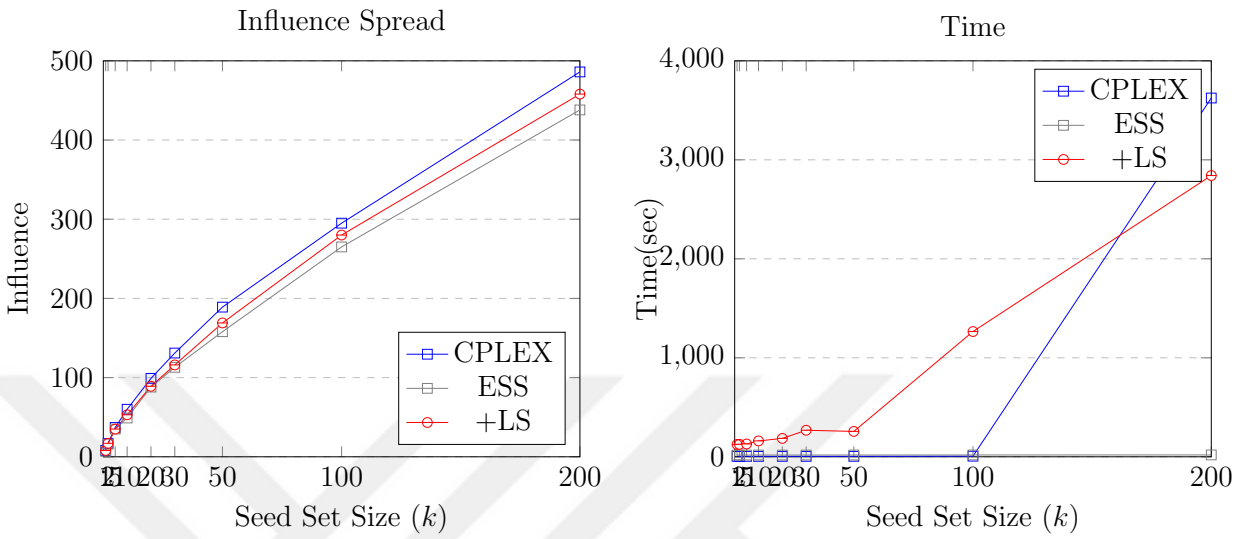


Figure 15: Influence and time graph of Barabasi-Medium-2 dataset

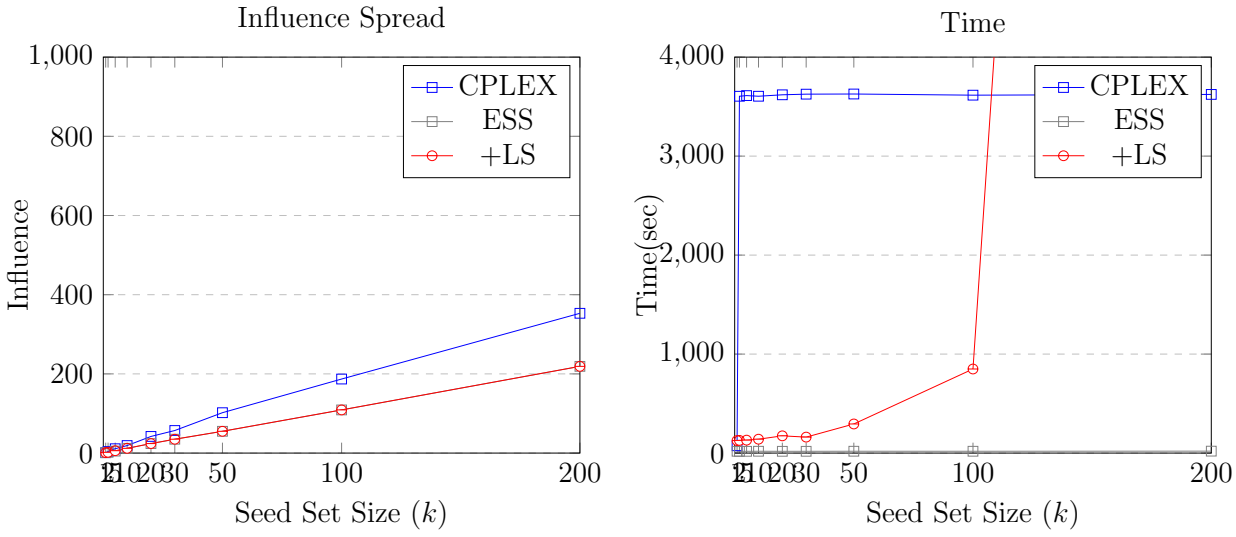


Figure 16: Influence and time graph of Watts-Medium-1 dataset

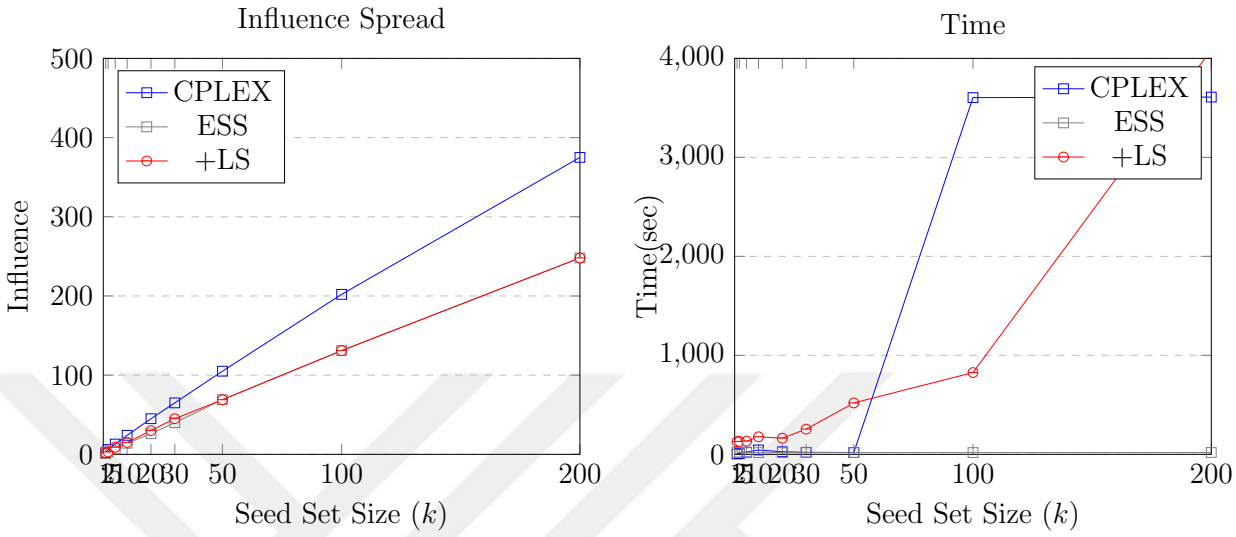


Figure 17: Influence and time graph of Watts-Medium-2 dataset

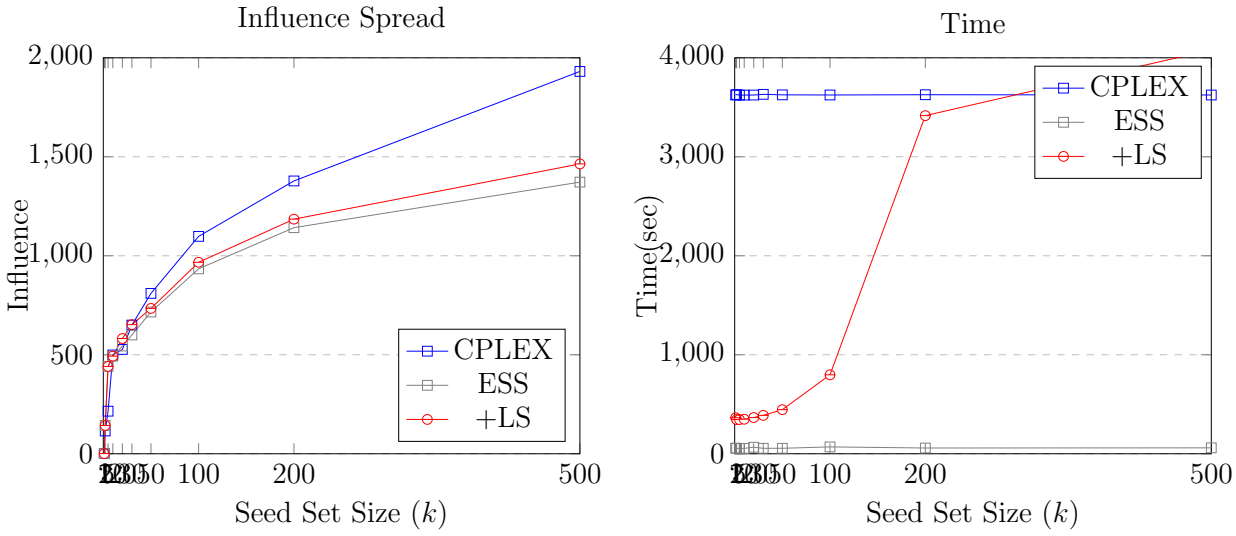


Figure 18: Influence and time graph of Barabasi-Medium(2000)-1 dataset

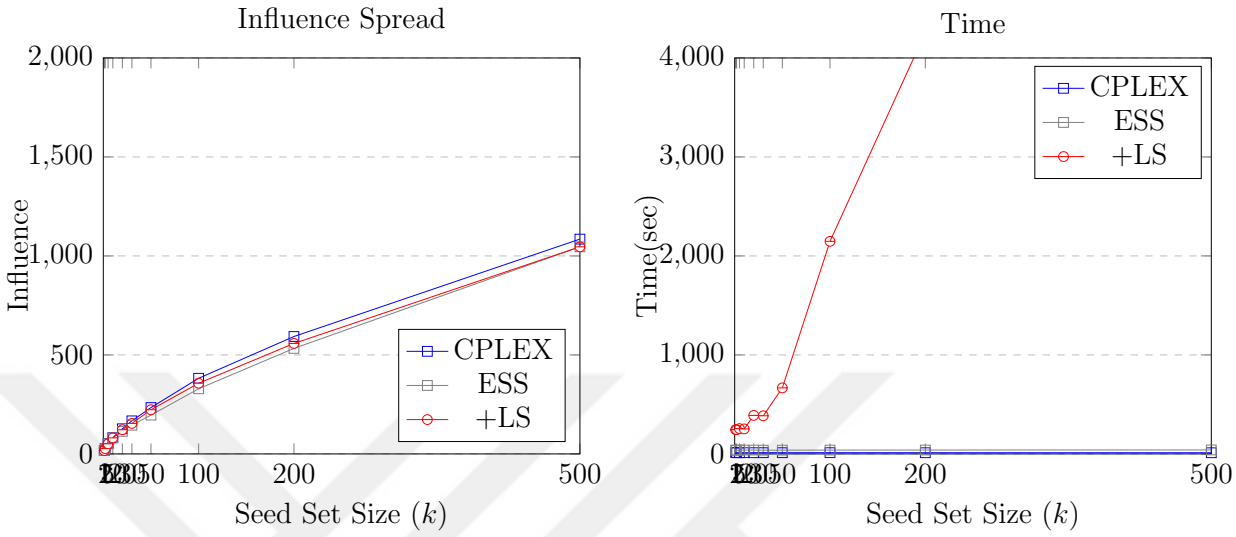


Figure 19: Influence and time graph of Barabasi-Medium(2000)-2 dataset

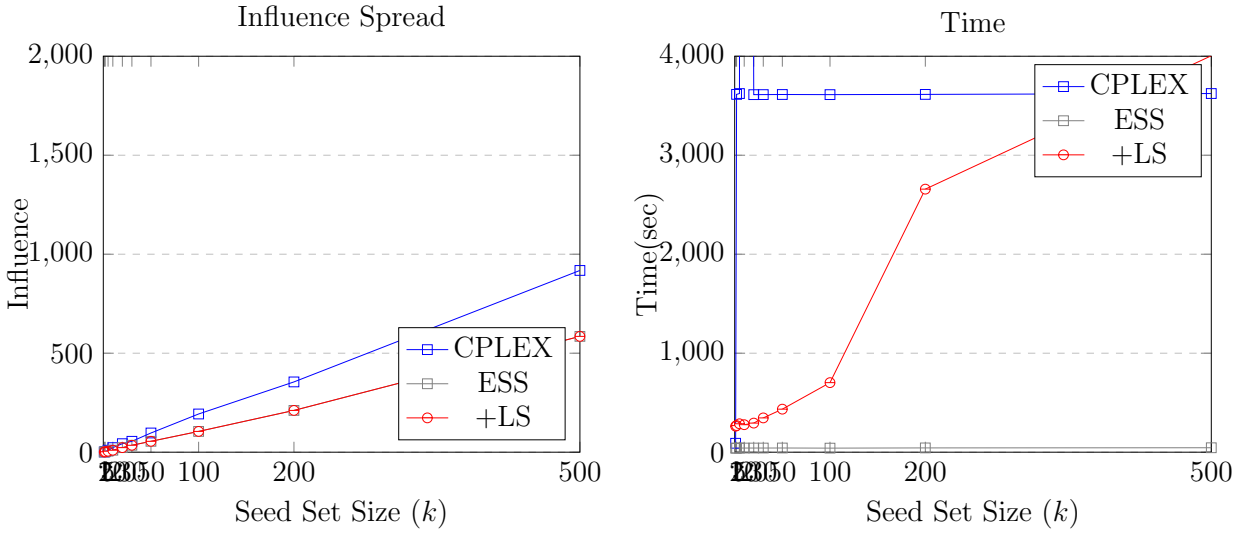


Figure 20: Influence and time graph of Watts-Medium(2000)-1 dataset

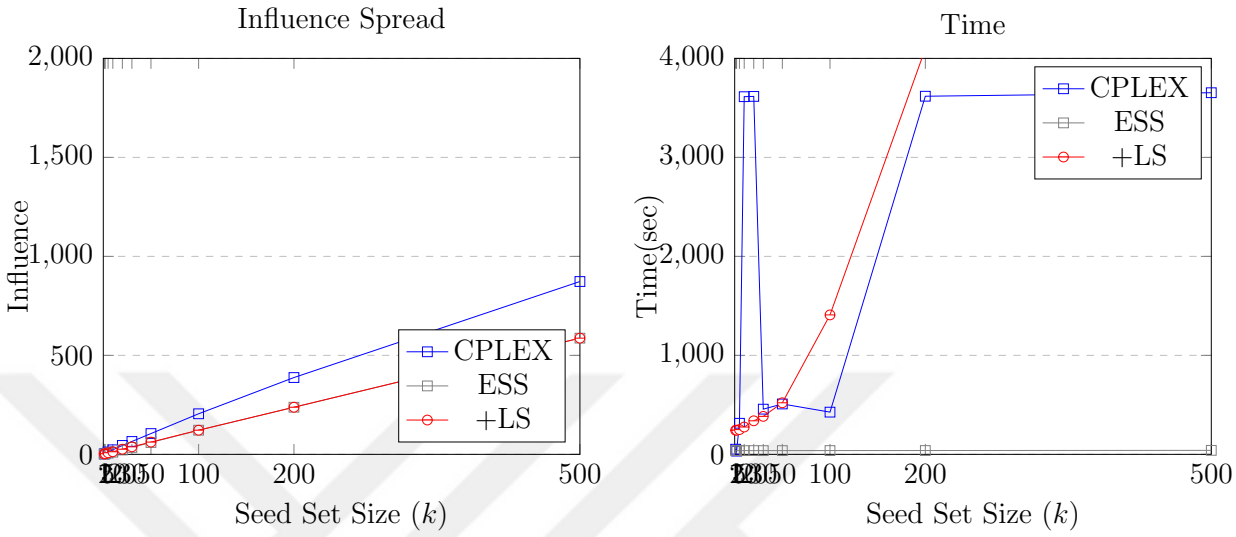


Figure 21: Influence and time graph of Watts-Medium(2000)-2 dataset

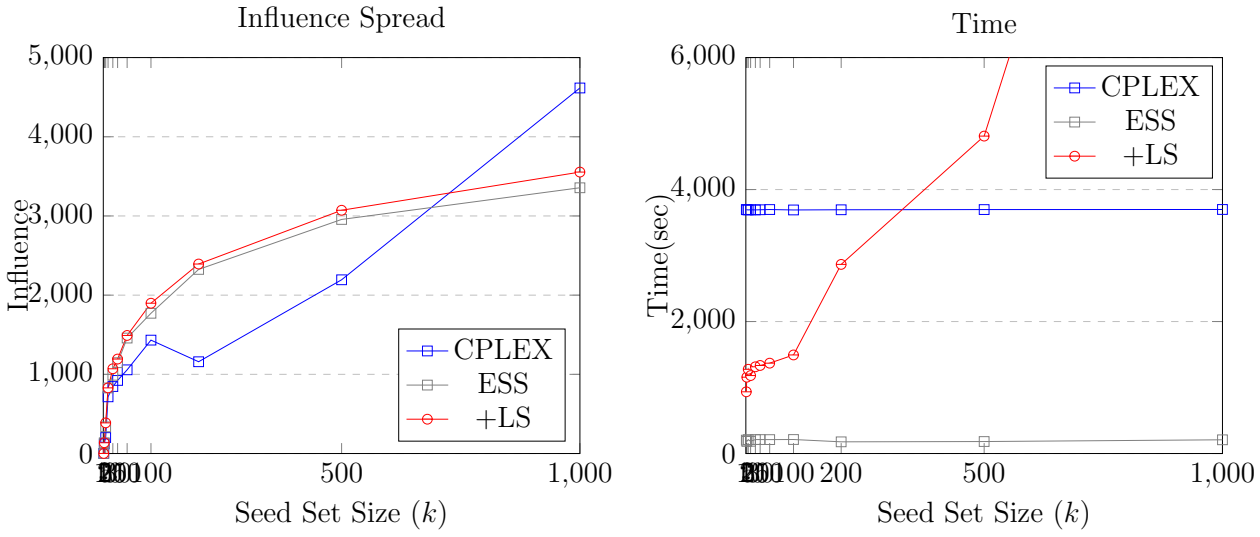


Figure 22: Influence and time graph of Barabasi-Large-1 dataset

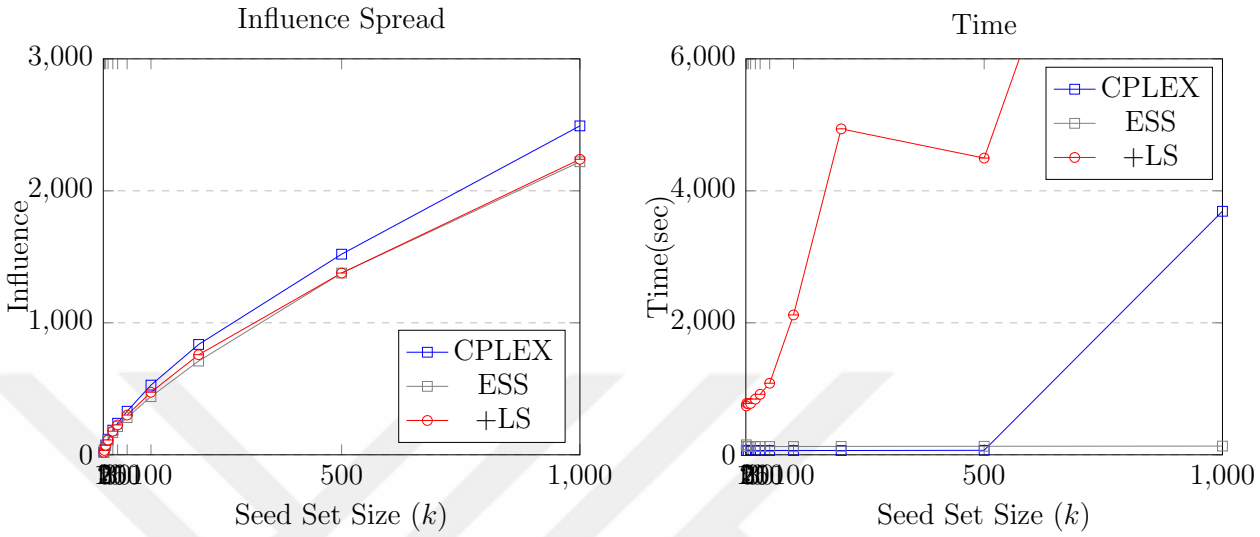


Figure 23: Influence and time graph of Barabasi-Large-2 dataset

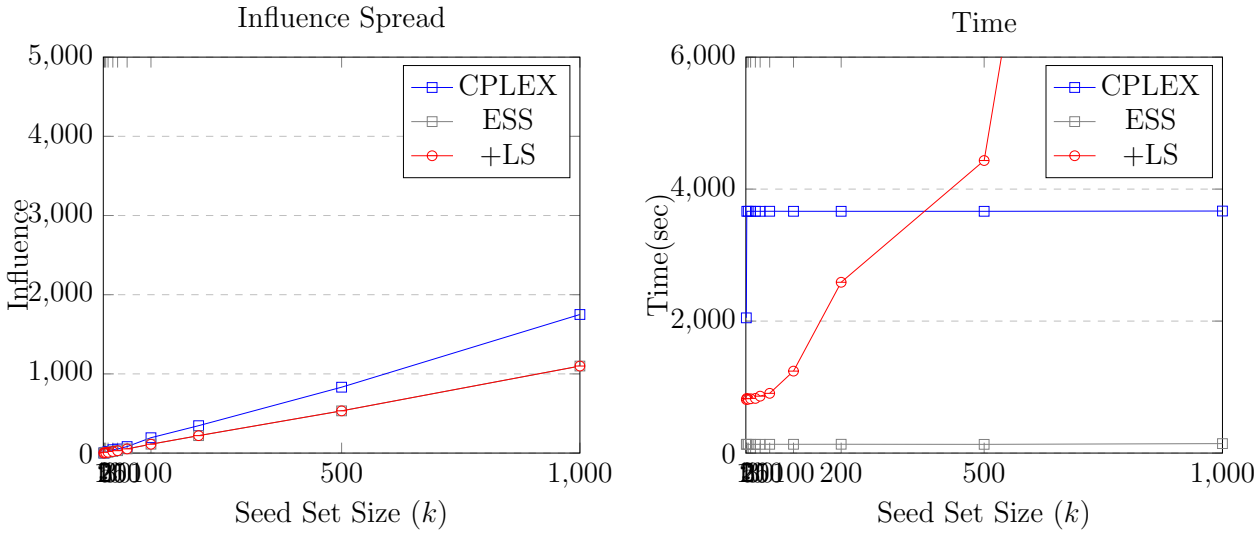


Figure 24: Influence and time graph of Watts-Large-1 dataset

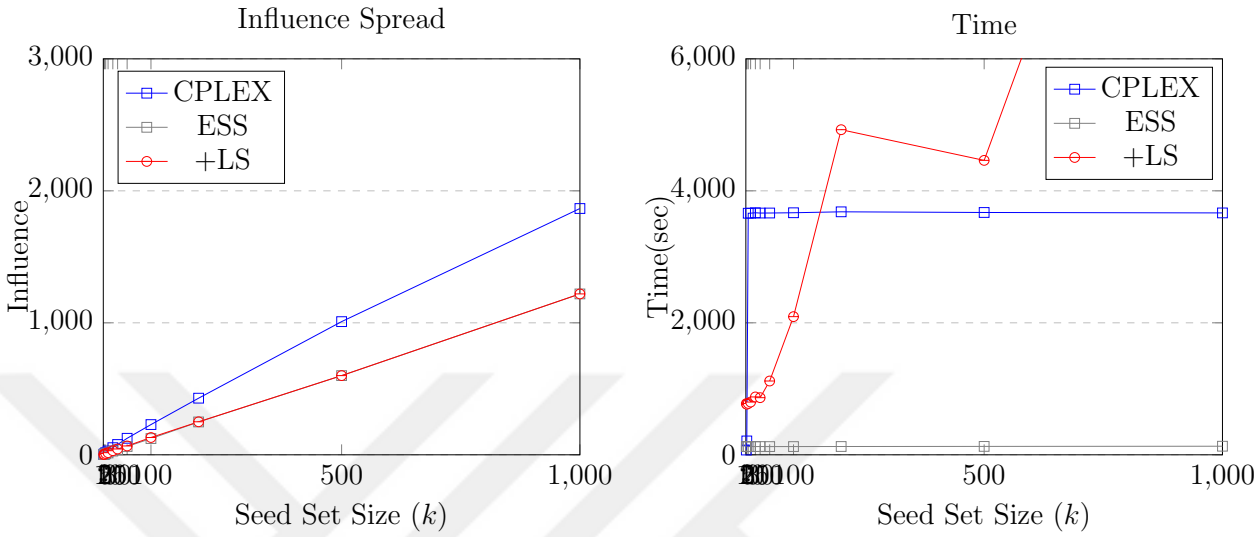


Figure 25: Influence and time graph of Watts-Large-2 dataset

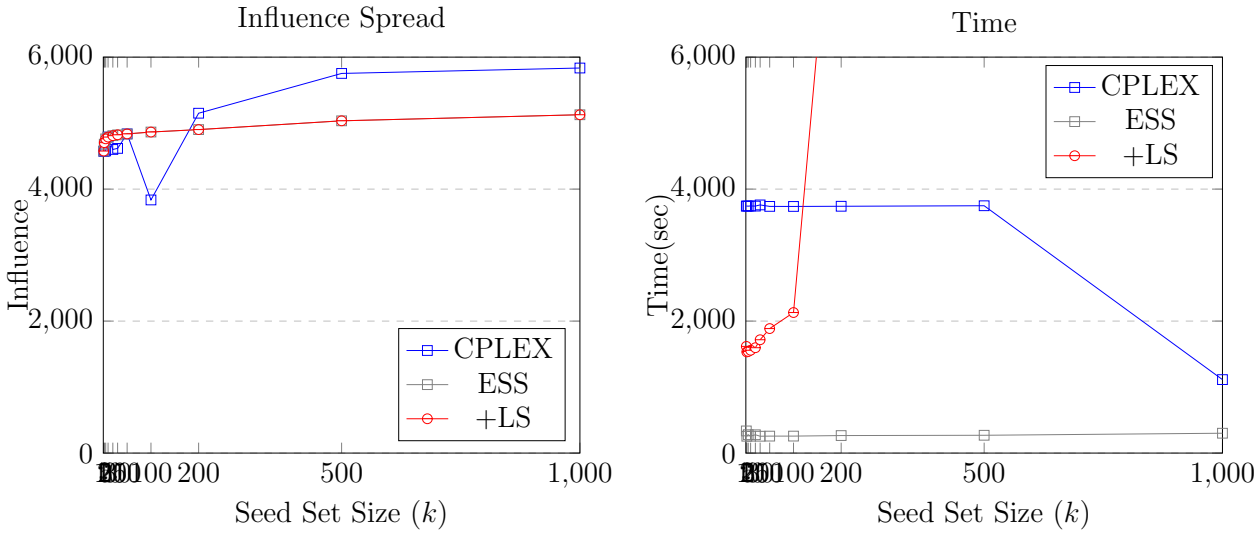


Figure 26: Influence and time graph of HEP-TH-1 dataset

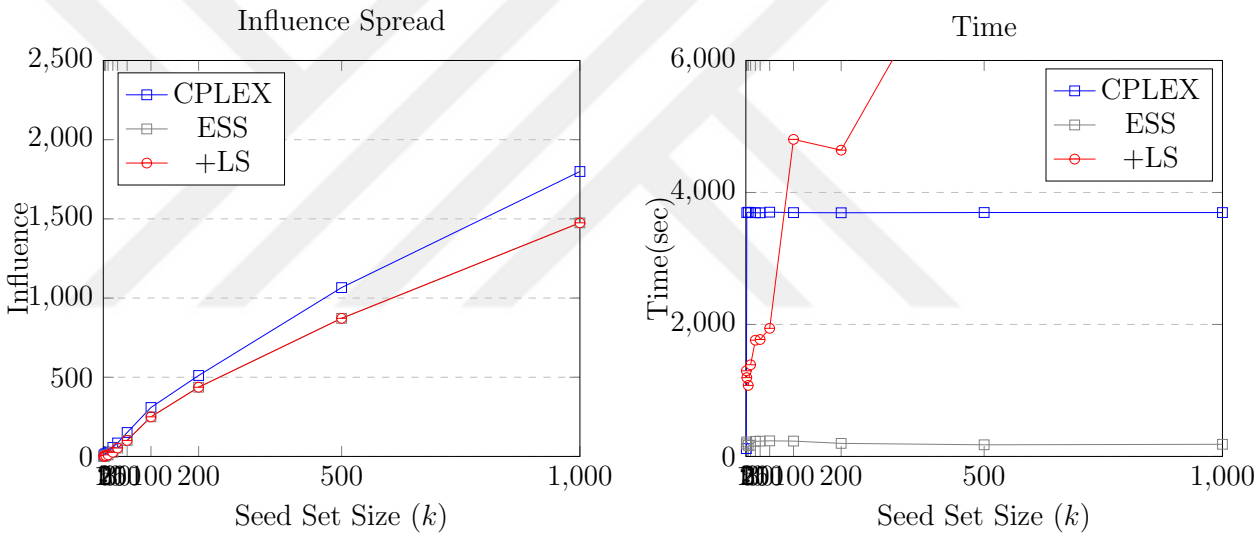


Figure 27: Influence and time graph of HEP-TH-2 dataset

5.3.3 Performance of the Local Search

In Tables 7, 8 and 9, we can see a column named Imp (Improvement) that show the percentage of improvement from Extended Seed Set to Local Search. Most of the results shows us there is an improvement from ESS to local search. When we compare it to CPLEX, the dynamic changes between influence and spent time according to size of the network. CPLEX gives the optimal solution when the size of the network is small, like real-life networks Celegansneural and Netscience. But in larger networks like HEP-TH, it spends more time and finds worse solutions than the heuristic method.

When the objective value of ESS is equal to the total number of in a set, there are no nodes left to be activated. For example, ESS obtained the exact solution, which is the whole network, for Instances 25-30. LS is unnecessary since there is no space to improve. Thus, ESS algorithm finds the exact solution and LS cannot find a better solution because there is not one. Unfortunately, LS cannot improve ESS for some instances even though it can be done. The average improvement percentage is 7% over all instances. Without considering 0 improvement, the average improvement is 17%.

We noticed that LS had difficulties when the hybrid model is used for weight and threshold generation. The stochastic threshold values could be inconsistent, which could be the cause of this. It would be preferable if a node's features were to influence weight and threshold generation more like in the Degree-based method.

CHAPTER VI

CONCLUSION

Ever since the social media became popular, many advertisers and marketers saw that as an opportunity to promote products. Influence Maximization Problem in social networks finds a set of nodes that maximizes the influence spread. In the IMP, people have influence on their neighbors. There is a factor in real-life that people can have influence on their neighbors' neighbors and their neighbors which are called 2-hop and 3-hop neighbors. In this research, we have worked on n-hop influence maximization problem under deterministic linear threshold model. We proposed a new method to calculate the influence transitivity from a node to its 2-hop and 3-hop neighbors.

We have introduced two new metrics to understand the characteristics of the nodes. We have found a method in order to estimate a set's value and proposed a degree calculation method called Unique Common Neighbor Degree for a set of nodes which can be used instead of calculating the real influence which calculates the number of unique common neighbors between a set of nodes. This is used in our proposed heuristic to be a comparison method between several solutions while replacing the nodes in the set.

We have proposed a heuristic solution to n-hop IMP named Extended Seed Set Algorithm. Our algorithm ESS first combines an initial seed set, then it improves it with local search. In smaller networks, CPLEX can be used but in larger ones CPLEX is not as beneficial as the proposed algorithm.

We tested the performance of our algorithm and used the solutions that are obtained from CPLEX under 1 hour limit using real-life and synthetic data. We have generated weight/threshold values according to three different methods which are the

Degree-based, the Hybrid and the Random method.

In the future work, the hopping number can be extended with larger numbers like 4-hop, 5-hop etc. Our proposed heuristic ESS+LS algorithm can be improved even more. A global search can be added to achieve higher solutions by swapping the nodes with further possible candidates.

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*, pp. 137–146, 2003.

[2] 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*, pp. 199–208, 2009.

[3] 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*, pp. 420–429, 2007.

[4] A. Goyal, W. Lu, and L. V. Lakshmanan, "Celf++ optimizing the greedy algorithm for influence maximization in social networks," in *Proceedings Of The 20th International Conference Companion On World Wide Web*, pp. 47–48, 2011.

[5] M. Gong, J. Yan, B. Shen, L. Ma, and Q. Cai, "Influence maximization in social networks based on discrete particle swarm optimization," *Information Sciences*, vol. 367, pp. 600–614, 2016.

[6] J. Tang, R. Zhang, P. Wang, Z. Zhao, L. Fan, and X. Liu, "A discrete shuffled frog-leaping algorithm to identify influential nodes for influence maximization in social networks," *Knowledge-Based Systems*, vol. 187, p. 104833, 2020.

[7] A. Jøsang and S. Pope, "Semantic constraints for trust transitivity," in *Proceedings Of The 2nd Asia-pacific Conference On Conceptual Modelling-volume 43*, pp. 59–68, 2005.

[8] C.-W. Hang, Y. Wang, and M. P. Singh, "Operators for propagating trust and their evaluation in social networks," tech. rep., North Carolina State University. Dept. Of Computer Science, 2008.

[9] W. Xu, Z. Lu, W. Wu, and Z. Chen, "A novel approach to online social influence maximization," *Social Network Analysis and Mining*, vol. 4, no. 1, p. 153, 2014.

[10] J. Tang, X. Tang, and J. Yuan, "Influence maximization meets efficiency and effectiveness: A hop-based approach," in *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 64–71, IEEE, 2017.

[11] M. E. J. Newman, "Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality," *Phys. Rev. E*, vol. 64, p. 016132, Jun 2001.

[12] W. Li, Y. Fan, J. Mo, W. Liu, C. Wang, M. Xin, and Q. Jin, "Three-hop velocity attenuation propagation model for influence maximization in social networks," *World Wide Web*, vol. 23, no. 2, pp. 1261–1273, 2020.

[13] A. Gulati and M. Eirinaki, "Influence propagation for social graph-based recommendations," in *IEEE International Conference on Big Data (Big Data)*, pp. 2180–2189, IEEE, 2018.

[14] D.-L. Nguyen, T.-H. Nguyen, T.-H. Do, and M. Yoo, "Probability-based multi-hop diffusion method for influence maximization in social networks," *Wireless Personal Communications*, vol. 93, no. 4, pp. 903–916, 2017.

[15] Y. Meng, Y. Yi, F. Xiong, and C. Pei, "T× one hop approach for dynamic influence maximization problem," *Physica A: Statistical Mechanics and its Applications*, vol. 515, pp. 575–586, 2019.

[16] M. E. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical Review E*, vol. 74, no. 3, p. 036104, 2006.

[17] J. G. White, E. Southgate, J. N. Thomson, S. Brenner, *et al.*, "The structure of the nervous system of the nematode caenorhabditis elegans," *Philos Trans R Soc Lond B Biol Sci*, vol. 314, no. 1165, pp. 1–340, 1986.

[18] M. E. Newman, "The structure of scientific collaboration networks," *Proceedings Of The National Academy Of Sciences*, vol. 98, no. 2, pp. 404–409, 2001.

[19] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews Of modern physics*, vol. 74, no. 1, p. 47, 2002.

[20] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[21] R. De Souza, D. R. Figueiredo, A. d. A. Rocha, and A. Ziviani, "Efficient network seeding under variable node cost and limited budget for social networks," *Information Sciences*, vol. 514, pp. 369–384, 2020.

[22] M. Duscu and D. Günneç, "A community-based solution approach for the influence maximization problem." working paper.

[23] F. Gursoy and D. Gunnec, "Influence maximization in social networks under deterministic linear threshold model," *Knowledge-Based Systems*, vol. 161, pp. 111–123, 2018.

[24] S. Hangal, D. MacLean, M. S. Lam, and J. Heer, "All friends are not equal: Using weights in social graphs to improve search," in *Workshop on Social Network Mining & Analysis, ACM KDD*, 2010.

[25] M. L. Bynum, G. A. Hackebeil, W. E. Hart, C. D. Laird, B. L. Nicholson, J. D. Siirola, J.-P. Watson, and D. L. Woodruff, *Pyomo–optimization modeling in python*, vol. 67. Springer Science & Business Media, third ed., 2021.

[26] W. E. Hart, J.-P. Watson, and D. L. Woodruff, "Pyomo: modeling and solving mathematical programs in python," *Mathematical Programming Computation*, vol. 3, no. 3, pp. 219–260, 2011.

VITA

Sena Odabaşı graduated from Haydarpaşa Anatolian High School in 2014 and obtained B.Sc. in Computer Science from Özyeğin University in February 2019 as top of her class. She has worked in IBM as project management officer for a year in 2019 and has been pursuing her M.Sc. degree in Data Science at Özyeğin University since September 2020 under the supervision of Asst. Prof. Dilek Günneç. She has been a teaching assistant in Programming for Operation Research and Applied Statistics courses. She currently works as a part-time data scientist at KoçDigital.