UC Irvine

UC Irvine Electronic Theses and Dissertations

Title

Diffusion Based Influence Maximization in GOLAP

Permalink

<https://escholarship.org/uc/item/1pj1h8x5>

Author

Kim, Mira

Publication Date

2020

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA,
IRVINE

Diffusion Based Influence Maximization in GOLAP

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Engineering

by

Mira Kim

Dissertation Committee:
Professor Phillip Sheu, Chair
Professor Jean-Luc Gaudiot
Professor Nader Bagherzadeh

2020

TABLE OF CONTENTS

	Page
LIST OF FIGURES	v
LIST OF TABLES	vii
ACKNOWLEDGMENTS	viii
CURRICULUM VITAE	ix
ABSTRACT OF THE DISSERTATION	x

Table of Contents

Chapter 1	Introduction.....	1
Chapter 2	Related work.....	3
2.1.	Information Diffusion Models.....	3
2.2.	Influence Maximization	4
2.3.	GOLAP	5
Chapter 3	Information Diffusion Model (IDM).....	7
3.1.	Preliminary Terms.....	7
3.2.	Definition of IDM.....	8
3.2.1.	<i>Threshold Value of Nodes</i>	8
3.2.2.	<i>Computing Combined Influence</i>	9
3.2.3.	<i>Example of IDM model</i>	10
3.3.	Influence Maximization based on IDM.....	12
Chapter 4	Colorblind IM-IDM	13
4.1.	Definition of ColorBlind IM-IDM.....	13
4.2.	Algorithm for ColorBlind IM- IDM	14
4.3.	Complexity of Algorithm	16
4.4.	Illustrating Example	17
Chapter 5	Colored IM-IDM	31
5.1.	Definition of Colored IM-IDM.....	31
5.2.	Algorithm for Colored IM-IDM.....	32

5.3.	Complexity of Algorithm	33
5.4.	Illustrating Example	34
Chapter 6	<i>Colored IM-IDM with Constraints</i>	40
6.1.	Definition of Colored IM-IDM with Constraint	40
6.2.	Algorithms for Colored IM-IDM with Constraint.....	40
6.2.1.	Alg-P1. findAnyColorSeedNode ( );	41
6.2.2.	Alg-P2. findColorMatchingSeedNode ( );	41
6.2.3.	Alg-P3. findColorNonMatchingSeedNode ( );	42
6.2.4.	Alg-1. IM-IDM with exactly t c -colored seed nodes.....	43
6.2.5.	Alg-2. IM-IDM with at least t c -colored seed nodes	45
6.2.6.	Alg-3. IM-IDM with at most t c -colored seed nodes	46
6.2.7.	Alg-4. Colored IM-IDM with $p\%$ c -colored seed nodes.....	47
6.2.8.	Alg-5. Colored IM-IDM with at least $p\%$ c -colored seed nodes.....	48
6.2.9.	Alg-6. Colored IM-IDM with at most $p\%$ c -colored seed nodes	48
6.3.	Complexity of Algorithms.....	49
6.4.	Illustrating Example	50
Chapter 7	<i>Lossy Colored IM-IDM</i>	58
7.1.	Definition of Lossy Colored IM- IDM	58
7.2.	Algorithm of Lossy Colored IM- IDM	58
Chapter 8	<i>Color-edged IM-IDM</i>	62
8.1.	Definition of Color-Edged IM-IDM	62
8.2.	Illustrating Example	62
8.3.	Significance of Threshold Values in Analytics	63
8.4.	Graph Representation of Gene Disease Relevance with Thresholds	64
8.5.	Semantics of Thresholds in Biomedical Domain	65
8.6.	Deriving Threshold Values of Gene Disease Relevance	67
8.7.	Transitive Relation in Gene-based network	68
8.8.	Algorithm for Gene-centric Network.....	69
8.8.1.	Graph Representation	69
8.8.2.	Defining Primitive Methods.....	71
8.8.3.	Defining Analytics Methods	72
Chapter 9	<i>Theorems and Proofs</i>	74

9.1.	Colorblind IM-IDM Theorems and Proofs.....	74
9.2.	Colored IM-IDM Theorems and Proofs	78
9.3.	Colored IM-IDM with Constraints Theorems and Proofs	79
Chapter 10	Experiments	81
10.1.	SNS Dataset for Experiments	81
10.2.	Experimenting with ColorBlind IM-IDM Algorithm	82
10.3.	Experimenting with Colored IM-IDM Algorithm.....	83
Chapter 11	Conclusions and Future Work	85

LIST OF FIGURES

	Page
Figure 1. Threshold Value in IDM Model	9
Figure 2. Combined Influence on Node E.....	9
Figure 3. Example of IDM Model.....	11
Figure 4. Individual vs. Combined Influences	11
Figure 5. Graph for ColorBlind IM-IDM	13
Figure 6. Directions of Edges/Relationships.....	14
Figure 7. Complexity Analysis of Algorithm	17
Figure 8. Example of ColorBlind IM-IDM.....	17
Figure 9. Node A Influencing Nodes B and C	18
Figure 10. Nodes influenced by Node A	19
Figure 11. Nodes influenced by Node B	19
Figure 12. Nodes influenced by Node C.....	20
Figure 13. Nodes influenced by Node D	20
Figure 14. Nodes influenced by Node E.....	21
Figure 15. Nodes influenced by Node F.....	21
Figure 16. Nodes influenced by Node G	22
Figure 17. Nodes influenced by Node H	22
Figure 18. Nodes influenced by Node I.....	23
Figure 19. Nodes influenced by Node J.....	23
Figure 20. Nodes influenced by Node K	24
Figure 21. Nodes influenced by Node L.....	24
Figure 22. Nodes influenced by Node M	25
Figure 23. Nodes influenced by Node N	25
Figure 24. Nodes influenced by Node D	27
Figure 25. Nodes influenced by Node G	27
Figure 26. Nodes influenced by Node I.....	28
Figure 27. Nodes influenced by Node J.....	28
Figure 28. Nodes influenced by Node L.....	28
Figure 29. Nodes influenced by Node M	29
Figure 30. Nodes influenced by the Seed Nodes {E, I}.....	30
Figure 31. Graph for Colored IM-IDM	31
Figure 32. Graph G with 14 Nodes and Three Colors.....	34
Figure 33. Node A Influencing Nodes B and C	35
Figure 34. Nodes Influenced by Node A	36
Figure 35. Nodes Influenced by Node D	36
Figure 36. Nodes Influenced by Node E	37
Figure 37. Nodes Influenced by Node K	37
Figure 38. Nodes Influenced by Node N	38
Figure 39. Nodes Influenced by Seed Nodes, E and D	39

Figure 40. Colored IM-IDM Graph with Constraint.....	50
Figure 41. Mode for Finding the First Seed Node	51
Figure 42. Nodes Influenced by Node A	52
Figure 43. Nodes Influenced by Seed Node $\{E\}$	53
Figure 44. Mode for Finding the Second Seed Node	54
Figure 45. Nodes Influenced by Seed Nodes $\{E, I\}$	55
Figure 46. Mode for Finding the Third Seed Node	56
Figure 47. Resulting Set of Seed Nodes	56
Figure 48. Nodes Influenced by Seed Nodes $\{E, I, D\}$	56
Figure 49. Reduced Graph G' from Original Graph G	59
Figure 50. Removal of Nodes with High Threshold.....	61
Figure 51. Effect of Combined Influence on Removed Node.....	61
Figure 52. Color-Edged Graph for Gene-Disease Network	62
Figure 53. Gene-Disease Network with more Edge Types.....	63
Figure 54. Influence Network.....	64
Figure 55. Graph Representation of Gene Disease Relevance with Thresholds.....	65
Figure 56. Effects of Threshold Values	66
Figure 57. Transitive Relations for SNS and Gene Disease Network	68
Figure 58. Example of Gene-centric Network.....	70
Figure 59. Dataset for Experiments	81
Figure 60. Accuracy for ColorBlind IM-IDM	82
Figure 61. Time Elapsed for ColorBlind IM-IDM	83
Figure 62. Accuracy for Colored IM-IDM	84
Figure 63. Time Elapsed for Colored IM-IDM	84

LIST OF TABLES

	Page
Table 1. IDM Notations for k-colored Graph	8
Table 2. Number of Nodes Influenced in Finding Seed Node #1	25
Table 3. Number of Nodes Influenced in Finding Seed Node #2	29
Table 4. Nodes Influenced by Each Candidate for Seed Node #1	38
Table 5. Nodes Influenced by Each Candidate for Seed Node #1	52
Table 6. Nodes Influenced by Each Candidate for Seed Node #2	54

ACKNOWLEDGMENTS

I would like to express my deep appreciation to my advisor, Professor Phillip Sheu. Professor Sheu has been a wonderful mentor for me. I would like to thank him for encouraging my research and for helping me to grow as a researcher.

I am thankful to the members in my Lab. Guigang Zhang helped me getting started with the Information Diffusion Model, provided me with ideas, and helped me develop examples. I would like to especially thank Jimmy Shih for helping me with proofs and theorems. I would also like extend my sincere thanks to Jenny Jin and Bryan Chou for helping me with the thesis and providing references.

As a cooperator of our Lab, I am also thankful to the supports from NEC Solution Innovators, Ltd., Japan. I am also grateful to Mr. Masahiro Hayakawa for him providing ideas.

For this dissertation I would also like to thank my committee members: Professor Jean-Luc Gaudiot and Professor Nader Bagherzadeh for their support and guidance.

I also thank my friends for providing support and friendship that I needed. I am deeply thankful to my family. Words cannot express how grateful I am to mom and dad for their guidance, invaluable advice, undying support and sacrifices.

The success would not have been possible without the constant support and caring nature of my beloved husband, Joseph Kim. I would also like to thank my best friend and amazing sister, Jenny Jin, for always being there for me.

Finally, I thank my God, for helping me through all the difficulties and hurdles.

Mira Kim
University of California, Irvine

2020

CURRICULUM VITAE

Mira Kim

2009	B.S. in Computer Science, University of Texas at Dallas
2011	M.S. in Computer Science, University of California - Los Angeles
2012-2015	Teaching Assistant, University of California, Irvine
2015	Ph.D. Intern, Samsung Research America
2015-now	Technical Product Manager, IBM
2020	Ph.D. in Computer Engineering, University of California, Irvine

FIELD OF STUDY

Influence Maximization in Information Diffusion Model

PUBLICATIONS

Mira Kim, et al. "Diffusion-Based Information Maximization Model." TBD, 2020.

ABSTRACT OF THE DISSERTATION

Diffusion based Influence Maximization in GOLAP

By

Mira Kim

Doctor of Philosophy in Computer Engineering

University of California, Irvine, 2020

Professor Phillip Sheu, Chair

Influence analysis is one of the most important research in social network. Specifically, more and more researchers and advertisers are interested in the area of influence maximization. The concept of influence among people or organizations has been the core basis for making business decisions as well as performing everyday social activities.

In this research, we begin by extending a new influence diffusion model (IDM). We incorporate colors and additional node constraints. By adding colors and constraints for different types of nodes in a graph, we are able to solve many practical applications as well as theoretical challenges. We would be able to answer complex queries on multi-dimensional graph such as ‘find at most 2 most important genes that are related to lung disease and heart disease’. More specifically, we discuss the following variations of IM-IDM: Colorblind IM-IDM, Colored IM-IDM, Colored IM-IDM with constraints, lossy IM-IDM as well as colored edge IM-IDM. We also present our experiment results to prove the effectiveness of our model and algorithms.

Chapter 1

Introduction

Nowadays studies on influence maximization problem is becoming an emerging research topic in graph database. Social influence research is a fundamental topic that has been studied for many years. Initially the topic was categorized as social sciences, and its main objective was to explore relationships of human groups [1]. Many social scientists and psychologists have begun to show interest in the topic of social influence. In their research, a person can be influenced by other people. When politicians make a speech, their purpose is to spread their influence to win an election or acquire support [2]. Businessmen construct their business network in order to sell their products to customers [3]. Different people have different influences; some people have very strong influence, thus being able to influence a vast number of people. Influence diffusion ranges also vary between people. Some people have strong influence and can influence a lot of people in a larger range and time period. However, some people have weaker influence and cannot spread their influence or have limited influence in a shorter time span.

With the development of internet, many social media applications have developed in recent years. People have constructed contact relationship through these social media apps or communication applications. Through email networks, we are able to contact whoever we want very easily. Through telephone networks, we can contact numerous people through their voice or text communication records. Purchase network, i.e., Amazon, can construct large purchase networks among products, buyers and sellers. Social network applications such as Twitter, Facebook, and Instagram can construct friends' networks for people all over the world [4][5]. LinkedIn constructed a very large contact network by using users' emails and constructed connections among the users. DBLP, a computer science bibliography website, set up a big

collaboration network for researchers all over the world. Online gaming web sites constructed big game networks for game players. As you can see, these large graph networks from various domains can construct a community network for diverse purposes. The topic of ways of analyzing these social networks, especially the dynamic social networks' influence among these networks' attendees, is becoming an emerging topic [6][7][8].

Our contributions can be summarized as the following:

- (1) We propose a new influence model: Influence Maximization - Information Diffusion Model (IM-IDM) model.
- (2) We propose algorithms to solve the Colored IM-IDM problem
- (3) We propose algorithms to solve the Colored IM-IDM with constraints problem and its variants
- (4) We propose algorithms to solve the Lossy Colored IM-IDM
- (5) We propose algorithms to solve the Colored edged IM-IDM

In Section 2, we review existing work related to graph influence, with a focus on influence maximization. We formally define influence maximization based on IDM model in Section 3. Section 4 investigates the information diffusion model (IDM). Section 5 discusses influence maximization problems based on IDM model and related algorithms. Section 6 explains problems related to the k-Colors IM-IDM problem and the associated reduction algorithms. Section 7 studies the lossy influence maximization based on IDM model. Section 8 discusses approximation methods for IM-IDM problem and the associated reduction algorithms. The thesis is concluded with Section 9 which talks about future directions.

Chapter 2

Related work

In this section, we summarize existing research that are related to influence maximization problems in graph.

2.1. Information Diffusion Models

In social networks, knowing how to find diffusion rules of information and mastering diffusion models for information are very important in assisting people learn how the information is being spread in social network [9][10].

Centola [11] investigates the effect of network structure on diffusion by studying the spread of health behavior through artificially structured online communities. Individual adoption is more likely when participants receive social reinforcement from multiple neighbors in the social network. The behavior spreads farther and faster across clustered-lattice networks than across corresponding random networks.

Cha [12] analyzes large scale traces of information dissemination on social network. They evaluate how popular pictures are distributed across social network using social links. Wang presents a community-based greedy algorithm for mining top-K influential nodes. It detects communities in social networks by taking into account information diffusion and a dynamic programming algorithm for selecting communities to find influential nodes.

With the development of social network, information spread models have developed from the two-class models to multi-class spread models [13][14][15][16].

There are numerous existing information diffusion models such as SIS, SIHR, SIR and SEIR. The existing influence diffusion models mainly include three types [17][18][19][20]. There are

Infectious disease model, which include the SIR(Susceptible , Infected , Recovered) and SIS(Susceptible , Infected , Susceptible). The second model is ICM(Independent Cascade Model) and LTM (Linear Threshold Model)is the third model.

Based on the IC and LT models, various other models have been proposed, such as triggering model, general cascade model and general threshold model. A lot of new information diffusion models have been proposed, such as continuous-time model, voter model, game-theoretic model, multi-item diffusion model and comparative influence diffusion model.

2.2. Influence Maximization

In many real-world applications, resources are very limited. We aim to find the initial diffusion seed nodes that can influence the whole network maximally. It will play a significant role in advertising new products and spreading new opinions.

Influence maximization has been proven to be an NP-hard problem [21-29]. Some researchers claim Influence Maximization problem is an optimization problem. That is to say, in a social network, if we find top-s most important nodes, these top-s nodes are not certain to have the maximum influence to the whole social network because the influence for many nodes may overlap together. The sum of these nodes' influence may not satisfy the influence maximization. In order to implement the influence maximization, we should compose a set of s seed nodes and get a maximum influence. Some studies discuss that IM problem is finding optimization algorithms. For example, CELF [30], CELF++ [31][32], NewGreedy[33], mixGreedy, and Degree Discount algorithms are used to compute IM. These algorithms work together with IC model or LT model, which are both NP-hard problems.

2.3. GOLAP

Chou [35] et al. thoroughly investigate graph online analytical processing (GOLAP) methods and research. They present various classes of queries, solutions for the challenges, and different definitions of the GOLAP operations.

The traditional OLAP includes three analytical operations: consolidation, drill-down, and slicing and dicing. Users can navigate the data in a multi-dimensional and multi-level view using these operations to obtain summarized information from among huge amounts of data. According to Chou, the challenges in GOLAP are speed and storage. Three solutions are available to solve these two challenges: speedup, graph data reduction, and approximation.

They also introduce color for nodes and edges as an additional dimension. For example, nodes of color ‘red’ could represent one political party whereas nodes of color ‘blue’ could represent another political party. The edges could also be colored to represent different classes of relationships.

Speedup is needed when dealing with large-network graphs that take exponential time to execute. Therefore, studies are now being conducted to apply graph data reduction to speed up processing. Data reduction algorithms can accelerate the performance significantly by reducing the size of the graph.

Another solution for solving the challenges in GOLAP is approximation. Approximation algorithms can provide an estimated query result in less execution time with a tolerable accuracy

and error rate. Moreover, they deploy heuristic algorithms to conquer the original NP-hard query problems that have exponential runtime.

Most of the IM models and algorithms proposed in the information diffusion space are very specific to a specific use case and are domain-specific such as the voter model and game-theoretic model. Additionally, they do not address complex queries. By adding colors as another dimension to either nodes or edges we can solve various types of queries involving multiple parameters and constraints. For example, a query of ‘find at least 2 most influential genes that are related to lung cancer and heart disease’ has nodes constraint such as ‘at least’ 2 seed nodes. It also has a constraint of having to be related to both ‘lung cancer’ and heart disease’. Thus, there is a need to design new efficient algorithms for colored graphs that can be widely applied to various industry segments and use cases.

It should be note that Jin [34] et al. present a comprehensive approach to managing influence networks using extended graph models. They discuss a method to find a Strongest Influence Path (SIP) from a source node to a target node. They extend the research by using colors as well for solving more complex problems and additionally applies their heuristic-based and graph reduction-based methods for reducing the time complexity.

Chapter 3

Information Diffusion Model (IDM)

3.1. Preliminary Terms

In this section, we first define terms used in graph theory and the influence maximization problem based on Information Diffusion Model.

Influence: In a social network, a person can influence another person. The probability of one node influencing another node is called the influence. The range is from 0 to 1. The influence is 0 if a person cannot influence another person. If a person can influence another person, the influence is 1. If the influence is 0.5, the probability of one person influencing another person is 50%.

ICM (Independent Cascade Model): When a node v becomes active, it has one chance of activating its inactive neighbor w . The activation attempt succeeds with probability P_{vw} .

LTM (Linear Threshold Model): A node v has a random threshold $\theta_v \in [0,1]$. A node v is influenced by each neighbor w according to a weight $b_{w,v}$ such that $\sum_{w \text{ neighbor of } v} b_{w,v} \leq 1$. A node v becomes active when at least (weighted) θ_v fraction of its neighbors are active.

$$\sum_{w \text{ active neighbor of } v} b_{w,v} \geq \theta_v$$

IM (Influence Maximization): We find s seed nodes from a graph G , which can influence the whole graph maximally based on the *IDM* (Influence Diffusion Model).

k-Colors Graph: Given a graph $G = \langle V, E \rangle$, there are k types of colors in this graph. Each color represents one kind of node. For example, the Democrats are blue nodes, the Republications are red nodes, and yellow nodes are 3rd party supporters.

IDM (Information Diffusion Model): IDM is a new model based on LTM. We add two types of colors constraints based on LTM. It includes confidence constraint between different colors and context sensitivity constraint among a node and its neighbors.

3.2. Definition of IDM

Given a k-Colors graph $G=<V,E>$, each node has a color and a threshold. Each edge can have an incoming influence, an outgoing influence or both. If the edge has an arrow, it represents that the edge only has one direction. If the edge has no arrow, it represents that the edge has two directions (bi-directional). **Error! Reference source not found.** shows the notations and their descriptions for IDM.

Table 1. IDM Notations for k-colored Graph

Notation	Description
th_A	The threshold of node A
$in_{A \rightarrow B}$	The influence from node A to node B
$in_{B \rightarrow A}$	The influence from node B to node A

3.2.1. Threshold Value of Nodes

We extend the conventional graph model with threshold values for nodes. Each node in a k-color graph G is associated with a threshold value, of which value range is $[0,1]$. Consider an example graph shown in **Error! Reference source not found.**.

Figure 1. Threshold Value in IDM Model

Note that the node E has a threshold of 0.3. If a neighbor node of E (such as B or J) has a combined influence value that is equal to or greater than 0.3 is allowed to exercise its influence through E . If not, the further influence by the neighbor node through E is prohibited.

3.2.2. Computing Combined Influence

Beyond the individual inference between two directly connected nodes, we consider the combinatorial influence of multiple nodes towards a target node. **Error! Reference source not found.** shows the target node E and its four inbound neighbor nodes, B , F , I , and J . The four neighbor nodes are marked with dashed border lines.

Figure 2. Combined Influence on Node E

The combined influence of the neighborhood nodes (B , F , I , J) towards the target node E is computed as the following;

$$\text{ComInfluence}(\{B, F, I, J\}, E) =$$

$$1 - \prod_{i=1}^n (1 - \text{influence}(i, E))$$

where n is the number of inbound neighbors of the target node E and $\text{influence}(i, E)$ denotes the influence value from the node i to node E .

Here we use Kempe [21] et al's formula for combined influence. Kempe use a cascade model, which is described in Section 3 of this thesis. Based on the cascade model, we are given a node v and a set of $S = \{u_1, \dots, u_k\}$ of its neighbors. They assume that the nodes in S try to activate v in the order of $u_1, \dots, u_k$, and they let $S_i = \{u_1, \dots, u_k\}$.

Then the probability that v is not activated by this process is, by definition, $\prod_{i=1}^k (1 - p_v(u_i, S_{i-1}))$. Kempe generalizes the cascade model to allow the probability that u succeeds in activating a neighbor v to depend on the set of v 's neighbors that have already tried. Therefore, an incremental function $p_v(u, S) \in [0,1]$ is defined where S and $\{u\}$ are disjoint subsets of v 's neighbor set. It succeeds with probability $p_v(u, S)$, where S is the set of neighbors that have already tried (and failed) to activate node v .

3.2.3. Example of IDM model

Consider an example of IDM model in **Error! Reference source not found.**. The k -color graph G consists of ten nodes and several edges between pairs of nodes. Each node is labelled with three attributes;

- Color as the node type
- Name of the node (such as 'A' and 'E')
- Threshold value of the node (between 0 and 1 inclusively)

Each edge is given an influence value between 0 and 1 inclusively, and it represents the degree of effectively applying an influence from one node to another node. The example in the figure assumes the influence is bidirectional.

Figure 3. Example of IDM Model

We illustrate how the influences are computed using seed nodes. Assume that a set of seed nodes consists of A , D , and G of the example. We can make the following observations;

The node A cannot influence the node B since the threshold value of B is 0.3 and the influence value from A to B is 0.2 which is lower than the threshold value.

The node D cannot influence the node H since the threshold value of H is 0.6 and the influence value from D to H is 0.3 which is lower than the threshold value.

The node G can influence the node J since the threshold value of J is 0.3 and the influence value from G to J is 0.4 which is higher than the threshold value.

Now, we consider a combined influence of multiple nodes on a target node. Consider only the three nodes A , B , and D in the same example.

Figure 4. Individual vs. Combined Influences

As shown in **Error! Reference source not found.(A)**, neither A nor D influences B since the threshold value of B is 0.3, which is higher than their influence value of 0.2. Then, we compute the combined influence of A and D towards B as the following.

$$\begin{aligned} ComInfluence(A, D) &= 1 - (1-(0.2)) * (1-(0.2)) \\ &= 1 - 0.8*0.8 = 1 - 0.64 = 0.36 \end{aligned}$$

The combined influence value, 0.36, is higher than the threshold value of 0.3. Hence, the node B gets influenced. This is shown in **Error! Reference source not found.(B)**, stating that the set of two nodes A and D influences the node B with the combined influence value of 0.36.

3.3. Influence Maximization based on IDM

Based on the IDM, we propose the influence maximization solutions. The IM-IDM problem can be defined as follows.

Given a weighted graph $G=\langle V, E \rangle$ whose nodes have k colors. Each color represents a type of nodes. We find s seed nodes from the whole graph satisfying all requirements, which these s seed nodes will have the maximal influence on the whole graph G .

Chapter 4

Colorblind IM-IDM

In this section, we present our approaches to influence maximization problem based on the colorblind IDM model.

4.1. Definition of ColorBlind IM-IDM

ColorBlind IM-IDM is represented as a graph, $G=\langle V, E \rangle$. Each node in a set V denotes a subject and an edge in the set E denotes a dependency relationship between two subjects. Figure 5 shows the graph representation of *ColorBlind IM-IDM*.

Figure 5. Graph for ColorBlind IM-IDM

The graph represents 10 nodes labelled as ‘A’ through ‘G’ and edges between pairs of nodes. In the SNS domain, a node represents a person and an edge represents an influence relationship between two nodes, i.e. a pair of persons. For example, consider an influence network of Tweeter SNS. A node represents a person using Tweeter SNS and an edge represents an influence relationship between two persons.

An edge can be either one-directional or bi-directional. Figure 6 shows both one-directional and bi-directional edges. The edge from node A to node D is considered a one-directional edge whereas the edge between node D and node E are considered a bi-directional edge.

Figure 6. Directions of Edges/Relationships

The ColorBlind IM-IDM problem is to find a set of s seed nodes for a given graph $G = \langle V, E \rangle$ that can influence the whole graph maximally based on the IDM. Here, the total number of nodes influenced by the solution set of the seed nodes is the greatest among all possible combinations of s seed nodes in the graph. Note that when choosing each candidate seed node, we consider the number of nodes that are influenced individually by that seed node. That is, we do not compute the accumulated influence value of the nodes influenced by a candidate seed node, which is another potential approach.

An example query with the ColorBlind IM-IDM for the Twitter SNS is “Find the set of 10 people who (as a whole) can influence the entire Twitter SNS.” This set of 10 is called a *seed node* set.

4.2. Algorithm for ColorBlind IM- IDM

We define the algorithm for finding the seed node set for a colorblind IM-IDM graph, $G = \langle V, E \rangle$.

The syntactic specification of the algorithm is the followings;

Signature: find_seednodes (s): setSeedNodes

where s is the number of seed nodes and setSeedNodes is the resulting seed node set.

Pre-condition:

$s \leq (\text{total number of nodes in } G)$

Post-Condition:

$S \subseteq V$ in $G(V, E)$ and S influences the whole G maximally.

Pseudocode:

```
1. setSeedNodes  $\leftarrow$  {} // The resulting set is initialized.
2. // Add all nodes in the candidate list.
3. list candiSeedNodes  $\leftarrow$  {node | node  $\in V$ }
4. // Repeat until all seed nodes are found
5. while ((size(setSeedNodes) < s) {
6.   array numNodes[size(candiSeedNodes)];
7.   // Find the number of nodes influenced for node.
8.   for i=1 to len(candiSeedNodes) {
9.     candiNode  $\leftarrow$  candiSeedNodes.get(i);
10.    numNodes[i]  $\leftarrow$  countNumNodes(candiNode);
11.  } // end of for loop
12. // Choose the node with the maximum number of
    nodes influenced and add it to setSeedNodes.
13. newSeedNode  $\leftarrow$  findNodeWithMax(numNodes[]);
14. setSeedNodes.add (newSeedNode);
15. // Find nodes influenced by the new seed node.
16. nodesInf  $\leftarrow$  findNodesInfluenced(newSeedNode);
17. // Update the candidate seed nodes;
18. candiSeedNodes  $\leftarrow$  (candiSeedNodes – nodesInf);
19.} // end of while loop
20.return setSeedNodes;
```

In line #1, the resulting set S is initialized as empty. In line #3, the list of candidate seed nodes initially contains all the nodes in the graph.

In lines #5 through #19, a while loops repeats until it finds the set of s seed nodes. In line #6, each element in the array *numNodes[]* is declared to contain the number of nodes influenced by each candidate seed node. Inside the while loop, a for loop runs lines from #8 to #11, finds the number of nodes influenced by each candidate node, and stores the result in the array *numNodes[]*. In lines #13 and #14, a new seed node is determined by choosing the node which has the largest

number of influenced nodes. In lines #16 and #18, the new list of candidate seed nodes is updated by discarding the nodes influenced by the new seed node.

Formally, the while loop is to find a new seed node $v \in V$ such that the value of $(\delta_{IDM}(S \cup \{v\}) - \delta_{IDM}(S))$ is maximized, while $\delta_{IDM}(S)$ denotes the number of nodes influenced each candidate node.

4.3. Complexity of Algorithm

We now estimate the complexity of the ColorBlind IM-IDM algorithm using big-O notation. Let n be the number of nodes in the graph $G(V, E)$ – that is, the size of V . We consider the loops and recursions in deriving the complexity. There are three loops in the given algorithm:

In line #5, the loop repeats n times at worst case. The complexity of this loop itself becomes $O(n)$.

In line #8, the loop repeats m times, which is the length of the list of candidate seed nodes. Hence the value of m becomes the value of n at worst case. The complexity of this loop with its outer loop becomes $O(n^2)$.

In line #10, the function *countNumNodes()* finds the number of nodes influenced by the given candidate seed node. This is done by looping through all the connected nodes for each candidate seed node. At worst case, the average number of connected nodes can be $n - 1$: that is, the candidate node has edges to all other nodes in G . Considering the two outer loops in this case, the complexity becomes $O(n^3)$.

Also in line #10, the average number of connected nodes at best case can be as few as 1; namely, when the candidate node has only one edge or neighborhood in G . Then, the complexity of “*countNumNodes(candiNode);*” in the line will be $O(1)$, the complexity of the “for” loop in line #8 become $O(n)$, and the complexity of “while” loop in line #5 becomes $O(n^2)$.

Hence, the overall complexity of the ColorBlind IM-IDM algorithm becomes $O(n^3)$. Figure 7 shows the algorithm and its associated complexity estimation in big-O.

Figure 7. Complexity Analysis of Algorithm

4.4. Illustrating Example

An example of *ColorBlind IM-IDM* is shown in Figure 8. The graph has 14 nodes labelled with A through N and a number of edges among pairs of nodes. A directed edge represents a directional influence relationship whereas an undirected edge represents a bi-directional influence relationship.

Figure 8. Example of ColorBlind IM-IDM

Note that a bi-directional edge has two influence values with a '/' between them. We use the convention that the first number is for the influence value from a left node to its right node or from an upper node and its lower node. If an edge is placed diagonally, the order of left-to-right is taken than upper-to-lower. In the figure, the influence value from the node A to the node B is 0.2 and the

value from the node B to the node A is 0.5. The influence value from the node A to the node D is 0.5 and the value from the node D to the node A is 0.6

Consider a query to find a set of 2 seed nodes having a maximal influence on the entire graph G. We define a function, $numNodes(x)$, which returns the number of nodes influenced by a node x .

Initially the resulting set S becomes empty as in line #1. We find the first seed node as the following;

Step 1. Finding the 1st Seed Node:

At the beginning of the search for seed nodes, the candidate list for the 1st node includes all the 14 nodes in G, {A, B, C, D, E, F, G, H, I, J, K, L, M, N}. We now compute the number of nodes that are influenced by each candidate seed node.

- Computing $numNodes(A)$

There are three nodes {B,C,D} connected to node A. Obviously A can influence A. Among the others, node A can influence node B because the influence from A to B is 0.2 and the threshold of node B is 0.2. Node A can influence node C because the influence from A to C is 0.3 and the threshold of node C is 0.3. Node A cannot influence node D because the influence from A to D is 0.5 and the threshold of node D is 0.6. The three nodes A, B, and C are highlighted in Figure 9.

Figure 9. Node A Influencing Nodes B and C

There are two new nodes {E,F} that connect to node B. B cannot influence node E, but can influence node F.

Figure 10. Nodes influenced by Node A

No new node connects with node C. So, we do not need to consider node C again.

There are two nodes {D,E} that connect to node F. F cannot influence node E because the influence is from E to F, and F cannot influence node D because the influence from node F to D is 0.2 that is lower than the threshold of D (0.8). Therefore we cannot expand the influence of node A further. In other words if we select node A as the first node, the influence to graph G is 4 ($|\{A,B,C,F\}|$).

- Computing numNodes(B)

If we select node B as the first node. The influence to the graph G is $4=|\{A,B,C,F\}|$, as shown in Figure 11.

Figure 11. Nodes influenced by Node B

- Computing numNodes(C)

If we select node C as the first node, the influence to the graph G is $2=|\{C,F\}|$, as shown in Figure 12.

Figure 12. Nodes influenced by Node C

- Computing numNodes(D)

If we select node D as the first node, the influence to the graph G is $7=|\{A,B,C,D,F,G,J\}|$, as shown in Figure 13.

Figure 13. Nodes influenced by Node D

- Computing numNodes(E)

If we select node E as the first node, the influence to the graph G is $8=|\{A,B,C,E,F,H,K,N\}|$, as shown in Figure 14.

Figure 16. Nodes influenced by Node G

- Computing numNodes(H)

If we select node H as the first node, the influence to the graph G is $3=\{H,K,N\}$, as shown in Figure 17.

Figure 17. Nodes influenced by Node H

- Computing numNodes(I)

If we select node I as the first node, the influence to the graph G is $3=\{I,L,N\}$, as shown in Figure 18.

Figure 18. Nodes influenced by Node I

- Computing numNodes(J)

If we select node J as the first node, the influence to the graph G is $1=\{J\}$, as shown in Figure 19.

Figure 19. Nodes influenced by Node J

- Computing numNodes(K)

If we select node K as the first node, the influence to the graph G is $3=\{H,K,N\}$, as shown in Figure

20.

Figure 20. Nodes influenced by Node K

- Computing numNodes(L)

If we select node L as the first node, the influence to the graph G is $2=\{L,N\}$, as shown in Figure

21.

Figure 21. Nodes influenced by Node L

- Computing numNodes(M)

If we select node M as the first node, the influence to the graph G is $3=\{J,M,N\}$, as shown in Figure

22.

Figure 22. Nodes influenced by Node M

- Computing numNodes(N)

If we select node N as the first node, the influence to the graph G is $1=\{N\}$, as shown in Figure 23.

Figure 23. Nodes influenced by Node N

We now summarize the number and list of nodes influenced by each candidate node for finding the seed node #1 in Table 2.

TABLE 2. NUMBER OF NODES INFLUENCED IN FINDING SEED NODE #1

Candidates \ Nodes	numNodes( )	List of Influenced Nodes
A	4	{A, B, C, F}
B	4	{A, B, C, F}

C	2	{C, F}
D	7	{A, B, C, D, F, G, J}
E	8	{A, B, C, E, F, H, K, N}
F	2	{C, F}
G	2	{G, J}
H	3	{H, K, N}
I	3	{I, L, N}
J	1	{J}
K	3	{H, K, N}
L	2	{L, N}
M	3	{J, M, N}
N	1	{N}

The table shows that the node E has the largest number of nodes influenced.

Step 2. Finding the 2nd Seed Node

Now, we find the second seed node by defining a new list of candidate seed nodes. This can be done by removing all the nodes influenced by the first seed node from the whole list of nodes in V. By this, the new list of candidate nodes becomes {D, G, I, J, L, M} and we find the second seed node using this list.

- Computing numNodes(D)

Now, we consider and test each node from the set {D, G, I, J, L, M}. First, we select node D as the second node. The combinational influence to the graph G is $8 + (3 = \{D, G, J\}) = 11$, as shown in Figure 24.

Figure 24. Nodes influenced by Node D

- Computing numNodes(G))

If we select node G as the second node, the combinational influence to the graph G is $8 + (2=\{G,J\})=10$, as shown in Figure 25.

Figure 25. Nodes influenced by Node G

- Computing numNodes(I)

If we select node I as the second node, the combinational influence to the graph G is $8 + (4=\{G,I,J,L\})=12$, as shown in Figure 26.

Figure 26. Nodes influenced by Node I

- Computing numNodes(J)

If we select node J as the second node, the combinational influence to the graph G is $8+(1=\{J\})=9$, as shown in Figure 27.

Figure 27. Nodes influenced by Node J

- Computing numNodes(L)

If we select node L as the second node, the combinational influence to the graph G is $8+(1=\{L\})=9$, as shown in Figure 28.

Figure 28. Nodes influenced by Node L

- Computing numNodes(M)

Last, if we select node M as the second node, the combinational influence to the graph G is $8 + (2 = \{J, M\}) = 10$, as shown in Figure 29.

Figure 29. Nodes influenced by Node M

We now summarize the number and list of nodes influenced by each candidate node for finding the seed node #2 in Table 3.

TABLE 3. NUMBER OF NODES INFLUENCED IN FINDING SEED NODE #2

Nodes Candidates	numNodes( )	List of Influenced Nodes
D	11	$8 + \{D, G, J\}$
G	10	$8 + \{G, J\}$
I	12	$8 + \{G, I, J, L\}$
J	9	$8 + \{J\}$
L	9	$8 + \{L\}$
M	10	$8 + \{J, M\}$

The table shows that the node I has the largest number of nodes influenced and hence it becomes the second seed node. The nodes influenced by the resulting seed nodes $\{E, I\}$ are shown in Figure 30.

Figure 30. Nodes influenced by the Seed Nodes {E, I}

Chapter 5

Colored IM-IDM

In this section, we present our approaches to influence maximization problem based on the colored IDM model.

5.1. Definition of Colored IM-IDM

The *Colored IM-IDM* is exemplified using a graph, $G = \langle V, E \rangle$. Each node in the set V denotes a subject, and an edge in the set E denotes a dependency relationship between two subjects.

Each node in V has an assigned color from among k colors, and each color denotes a specific type of node. For an influence network for a company, we can use colors to denote the departments of the company: for example, blue for Engineering, red for Marketing, and yellow for Accounting. Figure 31 shows the graph representation of the Colored IM-IDM, which has 14 nodes, each node has an assigned color, and edges among the nodes.

Figure 31. Graph for Colored IM-IDM

The Colored IM-IDM problem is to find a set of s seed nodes with a specified color c for a given graph $G = \langle V, E \rangle$ that can influence the whole graph maximally based on the IDM. Here, the total number of nodes influenced by the resulting set of c -colored seed nodes is the highest among all the combinations of s c -colored seed nodes.

An example query with the Colored IM-IDM for the Twitter SNS is “Find a set of 10 Marketing people who can influence the whole community maximally.” Here, “Marketing” is the staff type and is assigned to a specific color. Another example query on a graph of cosmetic product users is “Find those 100 female users who collectively have the maximal influence on the graph.” Here, “female” is considered as the node type and is assigned to a specific color.

5.2. Algorithm for Colored IM-IDM

The algorithm for the Colored IM-IDM is slightly different from that for the ColorBlind IM-IDM.

Signature: find_seednodes (s, c): setSeedNodes

where s is the number of seed nodes, c is the color of seed nodes, and setSeedNodes is the resulting seed node set.

Precondition:

$s \leq (\text{total number of } c\text{-colored nodes in } G).$

Post-condition:

$(S \subseteq V \text{ in } G(V, E)), (\text{color of } S == c), \text{ and } S \text{ influences the whole } G \text{ maximally.}$

Pseudocode:

```

1. setSeedNodes  $\leftarrow \{\}$  // The resulting set is initialized.
2. // Add all c-color nodes in the candidate list.
3. list candiSeedNodes  $\leftarrow$ 
   {node | node  $\in V$  and color(node) == 'c'}
4. // Repeat until all seed nodes are found.
5. while ((size(setSeedNodes) < s) {
6.   array numNodes[size(candiSeedNodes)];
7.   // Find the number of nodes influenced for node.
8.   for i=1 to len(candiSeedNodes) {
9.     candiNode  $\leftarrow$  candiSeedNodes.get(i);
10.    numNodes[i]  $\leftarrow$  countNumNodes(candiNode);
11.  } // end of for loop
12. // Choose the node with the maximum number of
   nodes influenced and add it to setSeedNodes.
13. newSeedNode  $\leftarrow$  findNodeWithMax(numNodes[]);
14. setSeedNodes.add (newSeedNode);
15. // Find nodes influenced by the new seed node.
16. nodesInf  $\leftarrow$  findNodesInfluenced(newSeedNode);
17. // Update the candidate seed nodes.
18. candiSeedNodes  $\leftarrow$  (candiSeedNodes – nodesInf);
19. removeNodeWithColor(candiSeedNodes, c);
20. } // end of while loop
21. return setSeedNodes;
```

In line #1, the resulting set S is initialized as empty. In line #3, the list of candidate seed nodes initially contains all the c -colored nodes in the graph. If we want to identify a set of two blue-colored seed nodes for the example graph in Figure 31, the list of candidates includes five blue nodes, (A, D, E, K, N), out of 14 nodes in total.

In lines #5 through #20, a while loop repeats until it finds the set of s seed nodes. In line #6, each element in the array $numNodes[]$ is declared to contain the number of nodes influenced by each candidate seed node. Inside the while loop, a for loop runs from lines #8 to #11, finds the number of nodes influenced by each candidate node, and stores the result in the array $numNodes[]$. In lines #13 and #14, the node with the greatest number of influenced nodes is chosen as a new seed node. In lines #16 and #18, the list of candidate seed nodes is updated by discarding those nodes influenced by the new seed node. In line #19, any non- c -colored node is discarded from the candidate list.

Formally, the while loop is used to find a new c -colored seed node $v \in V$ such that the value of $(\delta_{IDM}(S \cup \{v\}) - \delta_{IDM}(S))$ is maximized, where $\delta_{IDM}(S)$ denotes the number of nodes influenced by each candidate node.

5.3. Complexity of Algorithm

We now estimate the complexity of the Colored IM-IDM algorithm using big-O notation. Let n be the number of nodes in the graph $G(V, E)$ – that is, the size of V . We consider the loops and recursions in deriving the complexity. There are three loops in the given algorithm:

In line #5, the loop repeats n times at worst case. The worst case occurs when the value of s is same as n and all nodes in V are c -colored – essentially a one-colored problem. Hence, the complexity of this loop itself becomes $O(n)$.

In line #8, the loop repeats m times, which is the length of the list of candidate seed nodes. Initially, the list size is equal to the number of nodes, or V . Hence the value of m becomes the value of n . The complexity of this loop with its outer loop becomes $O(n^2)$.

In line #10, the function *countNumNodes()* finds the number of nodes influenced by the given candidate seed node. This is done by looping through all the connected nodes from the candidate's edges, and the number of connected nodes is much smaller than n . At worst case, however, that number can be $n - 1$: that is, the given node has edges to all other nodes in G . Hence, the complexity of the “while” loop in line #5 becomes $O(n^3)$, which is the overall complexity of the Colored IM-IDM algorithm.

5.4. Illustrating Example

An example of the Colored IM-IDM is shown in Figure 32. The graph has 14 nodes, three colors, and a number of edges connecting pairs of nodes. A directed edge represents a directional influence relationship, whereas an undirected edge represents a bidirectional influence relationship. The graph has five blue nodes, four red nodes, and five black nodes.

Figure 32. Graph G with 14 Nodes and Three Colors

Consider a query to find that set of two blue seed nodes that jointly have the maximal influence on the entire graph G . We define a function, $numNodes(x)$, that returns the number of nodes influenced by a node x .

Initially the resulting set S is empty, as in line #1. We find the first seed node using the following steps.

Step 1. Finding the First Blue Seed Node:

At the beginning of the search for seed nodes, the candidate list for the first node includes all the blue nodes in G : (A, D, E, K, N) . We now compute the number of nodes that are influenced by each candidate seed node.

- Computing $numNodes(A)$

We first select node A in the candidate list (A, D, E, K, N) . Figure 33 shows the intermediate step for finding nodes influenced by node A .

Figure 33. Node A Influencing Nodes B and C

There are three nodes $\{A, B, C\}$ that connect with node A . Node A can influence node B because the influence from A to B is 0.2 and the threshold of node B is 0.2. Node A can influence node C because the influence from A to C is 0.3 and the threshold of node C is 0.3. Node A cannot influence node D because the influence from node A to D is 0.6 and the threshold of node B is 0.8.

There are two new nodes, $\{E, F\}$, that connect to node B . B cannot influence node E , but it can influence node F .

Figure 34. Nodes Influenced by Node A

No new node can connect with node C . Thus, we do not consider node C again.

Node F connects to two nodes: $\{D, E\}$. Node F cannot influence node E . Node F also cannot influence node D because the influence from node F to D is 0.2 , and $0.2 < 0.8$. Thus, if we select node A as the first node, the influence on the graph G is $4 = \{A, B, C, F\}$.

- Computing $\text{numNodes}(D)$

Now let us consider the case in which we select node D , a blue node, as the first node.

Figure 35. Nodes Influenced by Node D

The influence of node D on the graph G is $7 = \{A, B, C, D, F, G, J\}$.

- Computing $\text{numNodes}(E)$

Then we test node E , another blue node, as the first node.

Figure 36. Nodes Influenced by Node E

The influence on the graph G of node E is $8 = \{A, B, C, E, F, H, K, N\}$.

- Computing $\text{numNodes}(K)$

Figure 37. Nodes Influenced by Node K

- Computing $\text{numNodes}(N)$

In similar fashion, we compute the influence of node K on the graph G to be $3 = \{H, K, N\}$ (Figure 36) and the influence of node N to be $1 = \{N\}$ (Figure 37).

Figure 38. Nodes Influenced by Node N

In Table 4, we summarize the number and list of nodes influenced by each candidate for seed node #1.

TABLE 4. NODES INFLUENCED BY EACH CANDIDATE FOR SEED NODE #1

Nodes Candidates	numNodes( )	List of Influenced Nodes
A	4	$\{A, B, C, F\}$
D	7	$\{A, B, C, D, F, G, J\}$
E	8	$\{A, B, C, E, F, H, K, N\}$
K	3	$\{H, K, N\}$
N	1	$\{N\}$

Table 4 shows that node E influences more nodes than any other candidate, and it thus becomes the first seed node.

Step 2. Finding the Second Blue Seed Node:

We find the second seed node by first defining a new list of candidates. This is done by removing from the list of c -colored nodes in V all those nodes influenced by the first seed node.

The list of nodes influenced by the first seed node is (A, B, C, E, F, H, K, N) . This list contains four blue nodes: (A, E, K, N) . Graph G has five blue nodes, (A, D, E, K, N) , and we get the new list of candidate seed nodes by $((A, D, E, K, N) - (A, E, K, N))$. The new list has only one candidate node, which is (D) . Thus D becomes the second blue seed node. Figure 39 shows the three nodes, (D, G, J) , influenced by the two blue seed nodes, (E, D) .

Figure 39. Nodes Influenced by Seed Nodes, E and D

Since node E has an influence value of 0.2 and node G has a threshold value of 0.2, node E influences node G . Similarly, because node G has an influence value of 0.3 and node J has a threshold value of 0.3, node G influences node J . Node G cannot influence node I , however, and node J cannot influence node M .

Hence, the list of nodes influenced by the candidate seed node D is $\{D, G, J\}$, and the full list of nodes influenced by the blue seed nodes, (E, D) , is $(A, B, C, D, E, F, G, H, J, K, N)$. This is shown in Figure 39.

Chapter 6

Colored IM-IDM with Constraints

In this section, we present the Colored IM-IDM under various constraints.

6.1. Definition of Colored IM-IDM with Constraint

The *Colored IM-IDM with Constraint* problem is to find a set of s seed nodes that has maximal influence on the whole graph and that includes t c -colored seed nodes. The applied constraint is the requirement for t c -colored nodes in the resulting set of seed nodes.

An example query for the Colored IM-IDM with constraint for the Twitter SNS is “Find a set of 10 people, including five Asians, who can maximally influence the whole community.” Here, the constraint is that the resulting set of people must include five Asians.

Another example query on a graph of cosmetic product users is “Find 100 users, 70% of whom are female, who can maximally influence the whole graph.” Here, the constraint is that the resulting set of users must be 70% females.

6.2. Algorithms for Colored IM-IDM with Constraint

The constraint for this model is that the resulting set of seed nodes must include the specified degree of c -colored nodes. We consider the following six types of constraints:

- Include exactly t c -colored seed nodes.
- Include at least t c -colored seed nodes.
- Include at most t c -colored seed nodes.
- Include exactly $p\%$ c -colored seed nodes.
- Include at least $p\%$ c -colored seed nodes.

- Include at most $p\%$ c -colored seed nodes.

Accordingly, there are six IM-IDM algorithms for these six types of constraints.

We first define three preliminary functions that are invoked by the six algorithms. The preliminary functions serve to find a new seed node with a given condition, as in the following:

- Function `findAnyColorSeedNode ( )`;

This function finds and returns a new seed node of any color.

- Function `findColorMatchingSeedNode ( )`;

This function finds and returns a new c -colored seed node.

- Function `findColorNonMatchingSeedNode ( )`;

This function finds and returns a new non- c -colored seed node.

6.2.1. Alg-P1. `findAnyColorSeedNode ( )`;

This method finds a seed node of any color.

1. // Find the # of nodes influenced for each node.
2. for $i=1$ to $\text{len}(\text{candiSeedNodes})$ {
3. $\text{candiNode} \leftarrow \text{candiSeedNodes.get}(i)$;
4. $\text{numNodes}[i] \leftarrow \text{countNumNodes}(\text{candiNode})$;
5. } // end of for loop
6. // Choose the node with the maximum # of nodes influenced and add it to *setSeedNodes*.
7. $\text{newSeedNode} \leftarrow \text{findNodeWithMax}(\text{numNodes}[])$;
8. return newSeedNode ;

The for loop beginning at line #2 finds the number of nodes influenced by each candidate node and stores the result in the array *numNodes[]*. Line #7 identifies the node that influences the greatest number of other nodes as the new seed node. Line #8 returns the new seed node.

6.2.2. Alg-P2. `findColorMatchingSeedNode ( )`;

This method finds a seed node of any color.

1. // Find the # of nodes influenced by each node.
2. for i=1 to len(candiSeedNodes) {
3.   candiNode ← candiSeedNodes.get(i);
4.   numNodes[i] ← countNumNodes(candiNode);
5. } // end of for loop
6. // Choose the node with the maximum # of nodes
    influenced and add it to *setSeedNodes*.
7. chooseColoredNodes(numNodes[], c);
8. newSeedNode ← findNodeWithMax(numNodes[]);
9. return newSeedNode;

The for loop beginning at line #2 finds the number of nodes influenced by each candidate node and stores the result in the array *numNodes[]*. In line #7, *chooseColoredNodes()* updates *numNodes[]* by choosing only *c*-colored nodes. This ensures that the seed node selected will be *c*-colored. In line #8, the node that influences the greatest number of other nodes is identified as the new seed node. Line #9 returns the new seed node.

6.2.3. Alg-P3. findColorNonMatchingSeedNode ( );

1. // Find the # of nodes influenced by each node.
2. for i=1 to len(candiSeedNodes) {
3.   candiNode ← candiSeedNodes.get(i);
4.   numNodes[i] ← countNumNodes(candiNode);
5. } // end of for loop
6. // Choose the node with the maximum # of nodes
    influenced and add it to *setSeedNodes*.
7. removeColoredNodes(numNodes[], c);
8. newSeedNode ← findNodeWithMax(numNodes[]);
9. return newSeedNode;

The for loop beginning at line #2 finds the number of nodes influenced by each candidate node and stores the result in the array *numNodes[]*. In line #7, *removeColoredNodes()* updates *numNodes[]* by removing all *c*-colored nodes. This ensures that a non-*c*-colored node will be selected. In line #8, the node that influences the greatest number of nodes is selected as the new seed node. Line #9 returns the new seed node.

Based on the preliminary functions, we now define the algorithms for the six types of constraints.

6.2.4. Alg-1. IM-IDM with exactly t c -colored seed nodes

This method finds a set of s seed nodes that includes exactly t c -colored nodes.

Signature: find_seednodes_exactly (s, t, c): setSeedNodes

where s is the number of seed nodes, t is the number of c -colored seed nodes, and setSeedNodes is the resulting seed node set.

Preconditions:

$s \leq (\text{total number of nodes in } G),$

$1 \leq t < s,$ and

$t \leq (\text{total number of } c\text{-colored nodes in } G).$

Post-conditions:

$(S \subseteq V \text{ in } G(V, E)),$

$(\# \text{ of } c\text{-colored seed nodes in } S) == t),$ and

S influences the whole G maximally.

Pseudocode:

```

1. setSeedNodes  $\leftarrow \{\}$  // The resulting set is initialized.
2. // Add all nodes to the candidate list.
3. list candiSeedNodes  $\leftarrow \{\text{node} \mid \text{node} \in V\}$ 
4. // Repeat until all seed nodes are found.
5. while ((size(setSeedNodes) < s) {
6.   array numNodes[size(candiSeedNodes)];
7.   // Mode 1. Find color-non-matching seed node.
8.   if (numColorMatchNodes(setSeedNodes) = t)
9.     newS = findColorNonMatchingSeedNode( );
10.  else
11.    // Mode 2. Find seed node of any color.
12.    if (s-len(setSeedNodes)) >
13.      (t-numColorMatchingNodes(setSeedNodes))
14.      newS = findAnyColorSeedNode( );
15.    else
16.      // Mode 3. Find color-matching seed node.
17.      newS = findColorMatchingSeedNode( );
18.  setSeedNodes.add (newS);
19.  // Find nodes influenced by the new seed node.
20.  nodesInf  $\leftarrow$  findNodesInfluenced(newS);
21.  // Update the candidate seed nodes.
22.  candiSeedNodes  $\leftarrow$  (candiSeedNodes – nodesInf);
23.} // end of while loop

```

23.return setSeedNodes;

In line #1, the resulting set S is initialized as empty. In line #3, the list of candidate seed nodes initially contains all the nodes in the graph.

In lines #5 through #22, a while loops repeats until it finds a set of s seed nodes that includes t c -colored seed nodes. Within the while loop, there are three modes of finding each seed node.

Mode 1: Finding color-non-matching seed node

The algorithm runs in this mode when t c -colored seed nodes have already been found: that is, the resulting set of seed nodes, *setSeedNodes*, already contains t c -colored seed nodes. Line #7 checks the condition for this mode, and line #8 finds the seed node in this mode: *findColorNonMatchingSeedNode()*.

Mode 2: Finding seed node of any color

The algorithm runs in this mode when t c -colored seed nodes have not yet been found and *setSeedNodes* has room to add a seed node of any color. Line #11-12 checks the condition for this mode, and line #13 finds the seed node in this mode: *findColorNonMatchingSeedNode()*.

Mode 3: Finding color-matching seed node

The algorithm runs in this mode when t c -colored seed nodes have not yet been found and *setSeedNodes* has room only for c -colored seed nodes.

Examples in which each of the three different modes would be applied are shown in Figure 40.

Consider the query *findSeedNodes* (4, 2, blue).

Figure 40. Example Cases for Three Modes

In mode 1, the set of seed nodes, *setSeedNodes*, already contains two blue nodes, and therefore the fourth seed node must not be a blue node.

In mode 2, the set of seed nodes, *setSeedNodes*, contains only one blue node, and there is room for adding two more seed nodes. Hence, the third seed node can be of any color.

In mode 3, the set of seed nodes, *setSeedNodes*, contains no blue node and there is room for adding only two more seed nodes. As two blue nodes are needed, the next seed node added must be blue.

6.2.5. Alg-2. IM-IDM with at least t c -colored seed nodes

This method finds a set of s seed nodes that includes at least t c -colored nodes.

Signature: find_seednodes_at_least (s, t, c): setSeedNodes
where s is the number of seed nodes, t is the number of c -colored seed nodes, and
setSeedNodes is the resulting seed node set.

Preconditions:

$s \leq (\text{total number of nodes in } G),$
 $1 \leq t < s,$ and
 $t \leq (\text{total number of } c\text{-colored nodes in } G).$

Post-conditions:

$(S \subseteq V \text{ in } G(V, E)),$
 $(\# \text{ of } c\text{-colored seed nodes in } S) \geq t),$ and
 S influences the whole G maximally.

Pseudocode:

1. *setSeedNodes* $\leftarrow \{\}$ // The resulting set is initialized.
2. // Add all nodes to the candidate list.
3. *list candiSeedNodes* $\leftarrow \{\text{node} \mid \text{node} \in V\}$
4. // Repeat until all seed nodes are found.
5. while $((\text{size}(\text{setSeedNodes}) < s) \{$
6.   *array numNodes* $[\text{size}(\text{candiSeedNodes})];$
7.   if $(s - \text{len}(\text{setSeedNodes})) \leq$
         $(t - \text{numColorMatchingNodes}(\text{setSeedNode}))$
8.   // Mode 3. Find color-matching seed node.
9.   *newS* = *findColorMatchingSeedNode*( );
10. else
- // Mode 2. Find any color seed node.
11.   *newS* = *findAnyColorMatchingSeedNode*( );

```

12. setSeedNodes.add (newS);
13. // Find nodes influenced by the new seed node.
14. nodesInf ← findNodesInfluenced(newS);
15. // Update the candidate seed nodes.
16. candiSeedNodes ← (candiSeedNodes – nodesInf);
17. } // end of while loop
18. return setSeedNodes;

```

The algorithm Alg-2 is similar to Alg-1 in its structure but has a different selection of modes. This algorithm requires only two modes: mode 2 for finding a seed node of any color and mode 3 for finding a color-matching seed node.

In line #7, if the remaining number of seed nodes in *setSeedNodes* is less than or equal to the number of *c*-colored seed nodes still needed, then the algorithm runs in mode 3 to add a color-matching seed node. Otherwise, the algorithm finds and adds any colored seed node, as shown in line #11.

6.2.6. Alg-3. IM-IDM with at most t *c*-colored seed nodes

This method finds a set of s seed nodes that includes at most t *c*-colored nodes.

Signature: find_seednodes_at_most (s, t, c): *setSeedNodes*
 where s is the number of seed nodes, t is the number of *c*-colored seed nodes, and *setSeedNodes* is the resulting seed node set.

Preconditions:

$s \leq (\text{total number of nodes in } G),$
 $1 \leq t < s,$ and
 $t \leq (\text{total number of } c\text{-colored nodes in } G).$

Post-conditions:

$(S \subseteq V \text{ in } G(V, E)),$
 $(\# \text{ of } c\text{-colored seed nodes in } S) \geq t),$ and
 $S \text{ influences the whole } G \text{ maximally.}$

Pseudocode:

```

1. setSeedNodes ← {} // The resulting set is initialized.
2. // Add all nodes to the candidate list.
3. list candiSeedNodes ← {node | node ∈ V}
4. // Repeat until all seed nodes are found.
5. while ((size(setSeedNodes) < s) {
6.   array numNodes[size(candiSeedNodes)];
7.   if numColorMatchingNodes(setSeedNode) == t

```

```

8.  // Mode 1. Find color-non-matching seed node.
9.    newS = findColorNonMatchingSeedNode( );
10. else
    // Mode 2. Find any color seed node.
11.    newS = findAnyColorMatchingSeedNode( );
12.  setSeedNodes.add (newS);
13.  // Find nodes influenced by the new seed node.
14.  nodesInf ← findNodesInfluenced(newS);
15.  // Update the candidate seed nodes.
16.  candiSeedNodes ← (candiSeedNodes – nodesInf);
17.} // end of while loop
18.return setSeedNodes;

```

The algorithm Alg-3 is identical to Alg-2 except for line #7. If the number of c -colored seed nodes in *setSeedNodes* is equal to t , then the algorithm runs in mode 1 to add a color-non-matching seed node. This is to avoid finding more than t c -colored seed nodes. Otherwise, the algorithm finds and adds any colored seed node, as shown in line #11.

6.2.7. Alg-4. Colored IM-IDM with $p\%$ c -colored seed nodes

This method finds a set of s seed nodes of which exactly $p\%$ are c -colored.

Signature: find_seednodes_exactly_p (s, p, c): setSeedNodes
*where s is the number of seed nodes, p is the percentage of c -colored seed nodes, and *setSeedNodes* is the resulting seed node set.*

Pseudocode:

```

1.  $t \leftarrow \text{round}(s * p / 100)$ ;
2. find_seednodes_exactly (s, t, c);

```

Algorithm Alg-4 requires preprocessing to compute the value of t from $p\%$. It then invokes Alg-1 with this value for t . In line #1, we compute the value of t using s and p and apply a round() operation to return an integer.

An example query is find_seednodes_exactly_p (4, 50%, Blue). The computed value of t is $(4*50/100)$, which equals 1. The algorithm then invokes Alg-1 with find_seednodes_exactly (4, 2, Blue).

6.2.8. Alg-5. Colored IM-IDM with at least $p\%$ c -colored seed nodes

This method finds a set of s seed nodes at least $p\%$ of which are c -colored.

Signature: find_seednodes_at_least_p (s, p, c): setSeedNodes
where s is the number of seed nodes, p is the percentage of c -colored seed nodes, and setSeedNodes is the resulting seed node set.

Pseudocode:

1. $t \leftarrow \text{round}(s * p / 100)$;
2. find_seednodes_at_least (s, t, c);

Algorithm Alg-5 requires preprocessing to compute the value of t from $p\%$. It then invokes Alg-2 with this value for t . In line #1, we compute the value of t using s and p and apply a round() operation to return an integer.

An example query is find_seednodes_at_least_p (4, 50%, Blue). The computed value of t is $(4*50/100)$, or 1. The algorithm then invokes Alg-2 with find_seednodes_at_least (4, 2, Blue).

6.2.9. Alg-6. Colored IM-IDM with at most $p\%$ c -colored seed nodes

This method finds a set of s seed nodes of which at most $p\%$ are c -colored.

Signature: find_seednodes_at_most_p (s, p, c): setSeedNodes
where s is the number of seed nodes, p is the percentage of c -colored seed nodes, and setSeedNodes is the resulting seed node set.

Pseudocode:

1. $t \leftarrow \text{round}(s * p / 100)$;
2. find_seednodes_at_most (s, t, c);

As with Alg-4 and Alg-5, algorithm Alg-6 requires preprocessing to compute the value of t from $p\%$. It then invokes Alg-3 with this value for t . In line #1, we compute the value of t using s and p and apply a round() operation to return an integer.

An example query is find_seednodes_at_most_p (4, 50%, Blue). The computed value of t is $(4*50/100)$, or 1. The algorithm then invokes Alg-3 with find_seednodes_at_most (4, 2, Blue).

6.3. Complexity of Algorithms

The complexity of the Colored IM-IDM algorithm with constraint is same as the complexity of the Colored IM-IDM. This is because the structures of the algorithms are essentially identical.

The three preliminary algorithms are invoked by the six algorithms with constraints. The preliminary algorithms have essentially identical structures:

- Alg-P1 finds any colored seed node.
- Alg-P2 finds c -colored seed nodes.
- Alg-P3 finds seed nodes that are not c -colored.

However, the three algorithms have the same structure and the difference in the color constraint does not change the complexity. The preliminary algorithms all have the same level of complexity: $O(n)$. This value is used in computing the complexity of the six main algorithms.

The six algorithms have essentially identical structures; they differ only in the constraint.

- Alg-1 finds exactly t c -colored nodes.
- Alg-2 finds at least t c -colored nodes.
- Alg-3 finds at most t c -colored nodes.
- Alg-4 finds a set of nodes of which exactly $p\%$ are c -colored.
- Alg-5 finds a set of nodes at least $p\%$ of which are c -colored.
- Alg-6 finds a set of nodes of which at most $p\%$ are c -colored.

As shown in the algorithms in Section 7(B), Alg-1, Alg-2, and Alg-3, modify only the conditional branch inside the while loop. The portion of the algorithm that determines which mode to run in for each iteration runs in $O(1)$, since it does not include a loop.

The other three algorithms, Alg-4, Alg-5, and Alg-6, only transform the value of $p\%$ into the integer value t . These also run in $O(1)$, since they do not have a loop.

Therefore, the difference in the constraint applied does not change the complexity of the Colored IM-IDM algorithms; they all have the complexity of $O(n^3)$.

6.4. Illustrating Example

The six algorithms with constraints are essentially the same in their control logic and vary only in their constraints. Hence, here we give an example for Alg-1, which becomes the basis for deriving the other five algorithms.

Given a graph for Colored IM-IDM in Figure 41, we want to find a set of three seed nodes that includes exactly two blue-colored seed nodes. That is, we run this query: *find_seednodes_exactly* (3, 2, “Blue”).

Figure 41. Colored IM-IDM Graph with Constraint

We utilize the function, $numNodes(x)$, which returns the number of nodes influenced by a node x , as presented in Section 5(D). Initially the resulting set S is empty, as in line #1, and we find the first seed node in the following manner.

Step 1. Finding the First Seed Node:

At the beginning of the search, the candidate list for the first seed node includes all the nodes in G . We now compute the number of nodes that are influenced by each candidate seed node.

- Computing $\text{numNodes}(A)$

We first decide what mode the algorithm must run in by evaluating the Boolean condition of the if statement in line #12:

$$(s - \text{len}(\text{setSeedNodes})) > (t - \text{numColorMatchingNodes}(\text{setSeedNode})).$$

From the given example query, $\text{find\_seednodes\_exactly}(3, 2, \text{"Blue"})$, we find that the value of s is 3, and the value of t is 2. Initially, setSeedNodes is null, and hence $\text{len}(\text{setSeedNodes})$ is 0. Moreover, the number of color-matching nodes in setSeedNodes must be 0, since setSeedNodes is initially null. Now, we have the computation as

$$((3-0) > (2-0)) \rightarrow (3 > 2) \rightarrow \text{True}.$$

Since the condition of the if statement is evaluated to *true*, we apply mode 2 as in line #13:

$$\text{newS} = \text{findAnyColorSeedNode}().$$

The content of the set of seed nodes and the current mode are show in Figure 42.

Figure 42. Mode for Finding the First Seed Node

We first select node A in the candidate list as shown in Figure 46. There are three nodes, $\{A, B, C\}$, that connect with node A . Node A can influence node B because the influence from A to B is 0.2 and the threshold of node B is 0.2. Node A can influence node C because the influence from A to C

is 0.3 and the threshold of node C is 0.3. Node A cannot influence node D , however, because the influence from node A to D is 0.6 and the threshold of node B is 0.8.

There are two new nodes, $\{E, F\}$, that connect to node B . B cannot influence node E , but it can influence node F .

Figure 43. Nodes Influenced by Node A

No new node can connect with node C . Thus, we do not consider node C again.

There are two nodes $\{D, E\}$ that connect to node F . Node F cannot influence node E , and it cannot influence node D because the influence from node F to I is 0.2, and $0.2 < 0.8$.

Thus, if we select node A as the first node, the influence on the graph G is $4 = \{A, B, C, F\}$.

We step through this process of finding how many nodes each candidate node influences; the intermediate steps are same as those given in Section 5(D). The results are shown in Table 5.

TABLE 5. NODES INFLUENCED BY EACH CANDIDATE FOR SEED NODE #1

Candidates	Nodes	
	numNodes( )	List of Influenced Nodes
A	4	$\{A, B, C, F\}$
B	4	$\{A, B, C, F\}$
C	2	$\{C, F\}$

D	7	$\{A, B, C, D, F, G, J\}$
E	8	$\{A, B, C, E, F, H, K, N\}$
F	2	$\{C, F\}$
G	2	$\{G, J\}$
H	3	$\{H, K, N\}$
I	3	$\{I, L, N\}$
J	1	$\{J\}$
K	3	$\{H, K, N\}$
L	2	$\{L, N\}$
M	3	$\{J, M, N\}$
N	1	$\{N\}$

The table shows that node E influences the greatest number of nodes among all candidates, and it thus becomes the first seed node. The set of nodes influenced by the current seed node $\{E\}$ are shown in Figure 47.

Figure 44. Nodes Influenced by Seed Node $\{E\}$

Step 2. Finding the Second Seed Node:

Now, we find the second seed node by defining a new list of candidates. This can be done by removing all the nodes influenced by the first seed node. The list of nodes influenced by the first

seed node $\{E\}$ is (A, B, C, E, F, H, K, N) , and hence the candidate list for the second seed node becomes $\{D, G, I, J, L, M\}$.

We again decide which mode the algorithm must run in by evaluating the Boolean condition of the if statement in line #12:

$(s-len(setSeedNodes)) >$

$(t-numColorMatchingNodes(setSeedNode))$,

$((3-1) > (2-1)) \rightarrow (2 > 1) \rightarrow \text{True}$.

Since the condition of the if statement is evaluated to true, we apply mode 2 as in line #13. The contents of the set of seed nodes and the current mode are illustrated in Figure 45.

Figure 45. Mode for Finding the Second Seed Node

We complete the process of finding how many nodes each candidate node influences. We apply the same intermediate steps used in Section 5(D). The results are listed in Table 6.

TABLE 6. NODES INFLUENCED BY EACH CANDIDATE FOR SEED NODE #2

Candidates \ Nodes	numNodes( )	List of Influenced Nodes
D	11	$8 + \{D, G, J\}$
G	10	$8 + \{G, J\}$
I	12	$8 + \{G, I, J, L\}$

J	9	$8 + \{J\}$
L	9	$8 + \{L\}$
M	10	$8 + \{J, M\}$

Table 6 shows that node I influences more nodes than any other candidate, and hence it becomes the second seed node. The nodes influenced by the resulting seed nodes, $\{E, I\}$, are shown in Figure 46.

Figure 46. Nodes Influenced by Seed Nodes $\{E, I\}$

Step 2. Finding the Third Seed Node:

Now, we find the third seed node by defining a new list of candidate seed nodes. This can be done by removing all the nodes already influenced by the first and second seed nodes; the new list of candidate seed nodes thus becomes $\{D, M\}$.

We again decide which mode the algorithm must run in by evaluating the Boolean condition of the if statement in line #12:

$(s-len(setSeedNodes)) >$

$(t-numColorMatchingNodes(setSeedNode)) ,$

$((3-2) > (2-1)) \rightarrow (1 > 1) \rightarrow \text{False} .$

Since the condition of the if statement is evaluated to false, we apply mode 3 as in line #13. The contents of the set of seed nodes and the current mode are illustrated in Figure 47.

Figure 47. Mode for Finding the Third Seed Node

We undertake the process of finding the number of nodes that are influenced by each node in the candidate set $\{D, M\}$. However, because the color of node M is not blue, it is discarded from the candidate set, leaving D as the only candidate. It becomes the third seed node.

- Computing $\text{numNodes}(D)$

The nodes directly connected to D – A , F , and G – are already influenced by the earlier seed nodes, and hence node D can influence only itself. The value for $\text{numNodes}(D)$ becomes $8 + 4 + 1$, which is 13. The resulting set of seed nodes for the given query becomes $\{E, I, D\}$, as shown in Figure 48.

Figure 48. Resulting Set of Seed Nodes

The set of nodes influenced by these three seed nodes are shown in Figure 49.

Figure 49. Nodes Influenced by Seed Nodes $\{E, I, D\}$

Note that the number of nodes influenced by the set of seed nodes is only 13, and node M is not influenced.

Chapter 7

Lossy Colored IM-IDM

7.1. Definition of Lossy Colored IM- IDM

Wang [36] et al. discuss graph data reduction and discuss the importance of storing and processing such data efficiently. They present the concept of “lossy” graph reduction for applications that may tolerate approximate results with small but bounded errors in exchange for further data reduction. They focus their study on a well-known graph problem which is the shortest path problem, design the lossy graph reduction algorithms, and implement their algorithms based on real world datasets.

The motivation of applying the lossy version on IM-IDM comes from the observation that ID-IDM models are often too complex to deliver the desired solutions with a reasonable time efficiency. In some application domains, reasonably good solutions close to the most optimal solution could be accepted in favor of the increased time efficiency.

Lossy Colored IM-IDM is to reduce the search space, i.e. the size of the graph G , with the hope of increasing runtime efficiency while losing some degree of accuracy. That is, the solution delivered by the lossy model may not be the most optimal once delivered by Brute Force algorithms. The lossy model has the following two features;

- Delivering the solution faster with increased time efficiency
- The solution is close enough to the most optimal solution

7.2. Algorithm of Lossy Colored IM- IDM

The algorithm for this problem consists of two stages. The first stage is to reduce the given graph $G = \langle V, E \rangle$, with a reference threshold value r . Here, all the nodes having a threshold value greater

than r are removed from the graph. Accordingly, all the edges associated with the removed nodes are also discarded from G . The resulting graph $G' = \langle V', E' \rangle$ becomes a subset of G , and contains nodes with relatively low threshold values.

An example of reduction from an original graph with a reference threshold value of 0.5 is shown in Figure 50.

Figure 50. Reduced Graph G' from Original Graph G

The graph G consists of 14 nodes as in Figure 50(A) and the node 'D' has a threshold value of 0.8, which is higher than the reference threshold value of 0.5. Hence, the node and its 3 edges are removed, resulting in a reduced graph G' as shown in Figure 50(B).

The second stage is to apply the same set of algorithms presented earlier in sections V, VI, and VII for finding the set of seed nodes on the reduced graph G' . Finding the solution on G' should be considerably faster than working on G .

Signature: find_seednodes_lossy (s, c, r): setSeedNodes

where s is the number of seed nodes, c is the color of seed nodes, r is the reference threshold value, and setSeedNodes is the resulting seed node set.

Pre-condition:

$s \leq (\text{total number of 'c' colored nodes in } G)$ and $0 < r < 1$

Post-Condition:

*(G' is a reduced graph of G with r) and
($S \subseteq V$ in $G'(V, E)$) and (color of $S == c$) and
 S influences the whole G' maximally.*

Pseudocode:

1. // Stage 1. Acquire a reduced graph G' .
2. $G' \leftarrow G$ // Copy the original graph G to G' .
3. for $i = 1$ to n { // 'n' is the # of nodes in G' .
4.   if ($\text{threshold}(\text{node } i) > r$) then {
5.   remove node i from G'
6.   remove all edges of node i from G'
7.   }
8. } // End of 'for' loop
9. // Stage 2. Invoke the target algorithm.
10. // Any of the foundation algorithms can be invoked.

Once the stage 1 is completed, the state 2 may invoke any of the algorithms presented earlier;

- ColorBlind IM-IDM, $\text{find\_seednodes}(s)$
- Colored IM-IDM, $\text{find\_seednodes}(s, c)$
- ColorBlind IM-IDM with Constraints;
   $\text{find\_seednodes\_exactly}(s, t, c)$,
   $\text{find\_seednodes\_at\_least}(s, t, c)$,
   $\text{find\_seednodes\_at\_most}(s, t, c)$,
   $\text{find\_seednodes\_exactly\_p}(s, p, c)$,
   $\text{find\_seednodes\_atleast\_p}(s, p, c)$, or
   $\text{find\_seednodes\_atmost\_p}(s, p, c)$

Theorem: For an original graph G , a reference threshold value r , and an accuracy error rate of ϵ , the pre-processing algorithm produces a reduced set $G' \subseteq G$ such that the accuracy function $\Phi(G) = 1$ and $\Phi(G') = \Phi(G) - \epsilon$.

Proof: Let SetNodes be the set of all nodes in G . For a node $N_i \in \text{SetNodes}$, if $\text{threshold}(N_i) > r$, the node is removed from G . Otherwise, it stays in G . The removal of nodes with r is illustrated in Figure 51.

Figure 51. Removal of Nodes with High Threshold

Now, consider combined influences of removed nodes. Let N_j be a neighbor node of a removed node RN_i . If the $ComInfluence(N_j \cup \text{any node combination}, RN_i) > r$, then $\Phi(G)$ is decreased by an accuracy loss $\phi(RN_i)$. This effect of combined influences on removed nodes is illustrated in Figure 52.

Figure 52. Effect of Combined Influence on Removed Node

As shown in Figure 52, there exists a case of losing the accuracy with removed nodes. With this, the summation of all accuracy errors $\sum(RN_i)$ becomes ϵ , and now this holds $\Phi(G') = \Phi(G) - \epsilon$. The proof on the second stage is already given in the previous sections.

Chapter 8

Color-edged IM-IDM

8.1. Definition of Color-Edged IM-IDM

In this section, we presented an extended version of IM-IDM with colors assigned to edges. In some applications, it becomes useful to assign types, i.e. colors, for edges.

- **Node:** A node specifies a gene.
- **Edge:** An edge between two genes represents a dependency occurrence for a disease. That is, both genes are relevant to the disease. Each edge is given a value of dependency strength between 0 and 1.
- **Edge Color:** The color of an edge represents a particular type of disease.

8.2. Illustrating Example

Consider a IM-IDM graph for Gene-Disease Network. In the graph, the node represents a type of a gene and the edge between two nodes represents a dependency occurrence for a particular disease. The particular disease is tagged with an edge type, i.e. edge color. Figure 53 shows a Color-Edged graph for gene disease network.

Figure 53. Color-Edged Graph for Gene-Disease Network

In the figure, two genes G1 and G2 are represented as node. Two of diseases, D_{red} and D_{blue} , are represented as the types of edges, using different colors. The two directional dependencies for a disease D_{red} are 0.3 and 0.5, and those for another disease D_{blue} are 0.2 and 0.7 in the graph.

The interpretation of the dependence is given here;

- For the given disease D_{red} , the probability of G3 being relevant to the disease is 0.3 when G1 occurs.
- For the given disease D_{red} , the probability of G1 being relevant to the disease is 0.5 when G3 occurs.
- For the given disease D_{blue} , the probability of G3 being relevant to the disease is 0.2 when G1 occurs.
- For the given disease D_{blue} , the probability of G1 being relevant to the disease is 0.7 when G3 occurs.

A more complete example of Gene-Disease network is shown in Figure 54.

Figure 54. Gene-Disease Network with more Edge Types

The figure shows another example with more genes and dependencies over three diseases; D_{red} , D_{blue} , and D_{green} . Alternative way of representing the gene disease networks is to define nodes for diseases and edges for the degree of common genes regarding pairs of diseases.

8.3. Significance of Threshold Values in Analytics

In dictionaries and general applications, a threshold value is a minimum or maximum value which serves as a benchmark for comparison. It specifies a level at which something would happen or

would take effect for a specific target activity, attribute or a characteristic. Examples of utilizing threshold values can be found in controlling hardware devices such as motors and lights, propagating epidemic diseases, and influencing people in SNS.

Thresholds have directionality; i.e. a threshold is applied, i.e. occurs in one direction. To observe the directionality of thresholds, consider influence relationships of a SNS as in Figure 55.

Figure 55. Influence Network

John has an influence level of 0.7 *onto* Susan, and Susan has a threshold value of 0.8 onto John. Hence, the influence from John to Susan does not take place, even if John has a higher influence strength than Susan. Susan has an influence level of 0.6, and John has a threshold value of 0.5. Hence, the influence from Susan to John do take place, even if Susan has a lower influence strength than John.

From the example, we can make two observations. An observation is that threshold values reveal additional significant information beyond the influence strengths themselves. Both the threshold value on edges and the threshold values must be considered in the occurrence of thresholds. The other observation is that threshold values are always given in one direction.

8.4. Graph Representation of Gene Disease Relevance with Thresholds

The datasets of gene disease relevance with threshold values can well be represented as an extended graph model. A node in the graph model represents a disease, and denoted as D_i and D_j . Each disease has a threshold value and the threshold value of a disease D_i is denoted as $T(D_i)$ as in Figure 56.

Figure 56. Graph Representation of Gene Disease Relevance with Thresholds

The weight of edges in the model represents the commonality of genes for the given pair of diseases, and it is denoted as a $C(D_i, D_j)$, as in Figure 56. The commonality, $C(D_i, D_j)$, is expressed as a value between 0 and 1. It can be computed from the available datasets. We first find the maximum value of $C(D_x, D_y)$ for all possible pairs of directly connected diseases D_x and D_y . This is done by computing the commonality values for all the pairs. Let the maximum commonality value be *maxComm*. Now, we compute the commonality value for each pair of directly connected diseases, D_i and D_j . Let the commonality value of each pair be the *oneComm*. Then, the value of $C(D_x, D_y)$ becomes (*oneComm* / *maxComm*).

8.5. Semantics of Thresholds in Biomedical Domain

Consider a disease, D_j , its threshold value $T(D_j)$ and the commonality value of $C(D_i, D_j)$. Then, the semantics of threshold values in the gene disease relevance datasets is defined as;

IF $C(D_i, D_j) \geq T(D_j)$ **THEN** the commonality takes place, i.e. occurring,

ELSE the commonality does not take place, i.e. being ignored.

The threshold value plays the role of safe guard in accepting the occurrence of the commonality. If the commonality only takes place when the value is higher or equal to the threshold value. In other words, the threshold value specifies a numerical level which determines the effectiveness of a commonality of genes between two diseases. Further elaboration of the semantics is given as;

- When $C(D_i, D_j) = 1$, all genes that are relevant to D_i are also relevant to D_j , regardless of $T(D_j)$.

- When $C(D_i, D_j) = 0$, no gene that is relevant to D_i is relevant to D_j .
- When $C(D_i, D_j) = 0.5$, about 50% of genes that are relevant to D_i are relevant to D_j , if $T(D_j)$ is 0.5 or higher.
- When $C(D_i, D_j)$ is near 1, many genes that are relevant to D_i will also be relevant to D_j with a less effect of $T(D_j)$.
- When $C(D_i, D_j)$ is near 0, only a few genes that are relevant to D_i will also be relevant to D_j , with a high effect of $T(D_j)$.

Another way of giving the semantics is to observe the level of threshold values. Consider a threshold with a high value. Then, we can make the following observation.

- A higher threshold value would limit the effectiveness of the commonality, i.e. ignoring.
- When $T(D_j)$ is near 1, many genes that are relevant to D_i will not be relevant to D_j , and the value of $C(D_i, D_j)$ is insignificantly treated/reflected.

Consider the example in 57(a). The threshold value of *Liver_Fat*, $T(\text{Liver\_Fat})$, is given the value of 0.95, and the commonality of genes between them, $C(\text{Stomach\_Cancer}, \text{Liver\_Fat})$, is given the value of 0.8. Since the threshold value is higher, the commonality does not take place even if the commonality of 0.8 is considered as high. The semantics interpretation of this case is that the genes relevant to *Stomach_Cancer* would not be relevant to *Liver_Fat*, due to the high threshold value.

Figure 57. Effects of Threshold Values

Consider a threshold with a low value. Then, we can make the following observation.

- A lower threshold value would promote the effectiveness of the commonality, i.e. reflecting.
- When $T(D_j)$ is near 0, most genes that are relevant to D_i will be relevant to D_j , and the value of $C(D_i, D_j)$ is significantly treated/reflected.

Consider the example in Figure 57(b). The threshold value of *Stomach_Cancer*, $T(\text{Liver\_Fat})$, is given the value of 0.1 and the commonality of genes between them, $C(\text{Liver\_Fat}, \text{Stomach\_Cancer})$, is given the value of 0.3. Since the threshold value is low enough, the commonality takes place even if the commonality of 0.3 is considered as low. The semantics interpretation of this case is that the genes relevant to *Liver_Fat* would not be relevant to *Stomach_Cancer*, due to the low threshold value.

8.6. Deriving Threshold Values of Gene Disease Relevance

The sources for deriving threshold values vary, and we identify common ways of acquiring the values in gene disease relevance datasets.

- Physician's Experience of Treatments
- Physician's Consensus on Medical Knowledge
- Statistically Driven from Medical Treatment Database
- Results from Clinical Tests, typically on Animals and then move onto People

Threshold values in gene disease relevance datasets are often refined over time for reasons. Initially, the threshold values come from some sources or they can be given with acceptable hypothesis by medical experts. Then, the values can be calibrated or refined over time using observation values on further clinical tests and medical treatments. Regression methods or machine learning algorithms such as Support-Vector Machine (SVM) can be effectively utilized in generating the refinement values.

8.7. Transitive Relation in Gene-based network

A transitive relation for SNS domain specifies that if person A informs person B, and person B informs person C, then person A informs person C indirectly. The transitive relation for SNS is shown in Figure 58(a) and the transitive relation for gene disease network is shown in Figure 58(b).

Figure 58. Transitive Relations for SNS and Gene Disease Network

The transitive relation can be specified as $(A \rightarrow B) \cap (B \rightarrow C) \rightarrow (A \rightarrow C)$, however, the computation of the transitive relation values varies for different application domains. The transitive relation from a disease D_i to a disease D_k , i.e. $C(D_i, D_k)$ in Figure 58(b) can be interpreted and derived in three different use cases.

Use Case as Correlation: In this use case, if gene A has correlation with gene B, and gene B has correlation with gene C, then gene A does not necessarily have correlation with gene C indirectly. Because the edge just represents the frequency of the 2 genes co-occurred in a same thesis or abstract, and it does not care about the relationship between the two. Correlation between A and B and the correlation between B and C are not necessarily the same.

Use Case as Promotion/Suppression: In this use case, if gene A promotes gene B, and gene B suppresses gene C, then what would gene A do against gene C indirectly? This is unknown since

the semantics of edges are not the same, we cannot determine the outcome. Thus, we cannot use

Transitive Relation: $(A \rightarrow B) \cap (B \rightarrow C) \rightarrow (A \rightarrow C)$.

Use Case as Micro Array: In this use case, if gene A is similar to gene B, and gene B is similar to gene C, then gene A is similar to gene C indirectly. This holds true because the semantics of similarity can be interpreted in various ways.

The appropriate interpretation, i.e. use case, of the transitive relation should be chosen by considering the dataset semantics of the target gene disease network application.

8.8. Algorithm for Gene-centric Network

In this section, we define analytics methods for the two graph representations of the gene disease networks.

8.8.1. Graph Representation

In gene-centric network model, the node of the graph represents a gene and the edge represents a dependency occurrence for a particular disease.

- Node) A node specifies a gene, and the number of nodes in this model can potentially be extremely high.
- Edge) An edge between two genes represents a dependency occurrence for a disease. That is, both genes are relevant to the disease. Each edge is given a value of dependency strength between 0 and 1.
- Type of Edge) Each edge is of a certain edge type which represents a particular disease. The type of edge can be assigned with a color for visualization.

Examples of gene-centric network are shown in Figure 599.

Figure 59. Example of Gene-centric Network

In Figure 599(a), two genes G1 and G2 are represented as node. Two of diseases, D_{red} and D_{blue} , are represented as the types of edges, using different colors. The two directional dependencies for a disease D_{red} are 0.3 and 0.5, and those for another disease D_{blue} are 0.2 and 0.7 in the graph. The interpretation of the dependences are given here;

- For the given disease D_{red} , the probability of G3 being relevant to the disease is 0.3 when G1 occurs.
- For the given disease D_{red} , the probability of G1 being relevant to the disease is 0.5 when G3 occurs.
- For the given disease D_{blue} , the probability of G3 being relevant to the disease is 0.2 when G1 occurs.
- For the given disease D_{blue} , the probability of G1 being relevant to the disease is 0.7 when G3 occurs.

Figure 599(a) shows another example with more genes and dependencies over three diseases; D_{red} , D_{blue} , and D_{green} . Alternative way of representing the gene disease networks is to define nodes for diseases and edges for the degree of common genes regarding pairs of diseases.

8.8.2. Defining Primitive Methods

We specify the set of analytics methods that can be applied to gene-centric network model. There are two categories of methods; *primary methods* which are commonly applied to various analytics problems and *application-level analytics methods* which provide the results of applying analytics methods to users. Here are the primitive methods;

- *readAllDiseases()* to read all the diseases in the dataset.
- *readAllGenes()* to read all the genes in the dataset.
- *readAllRelations()* is to retrieve all the relevance occurrences in the dataset.
- *findRelations(D_i)* is to return the set of relations which are relevant to the given disease D_i .
- *findCommonRelations(Disease_Set)* is to return the set of relations of which each relation is relevant to the specified set of diseases such as (D_2 , D_5 , D_6). This can be done by invoking *findRelations(D_i)* method for each disease of interest.
- *findCommonGenes(Disease_Set)* is to return the set of genes which appear in any relation. It can be done by invoking *findCommonRelations()*.
- *computeAveRelevance(G_i)* is to return the average relevance values for a given gene G_i to the specified set of diseases of interest. This can be done by invoking *findCommonGenes(setTargetDiseases)*. For example, invoking *findCommonRelations(D_2 , D_5 , D_6)* produces a list of common genes {346, 473, 136, 300}, and

computeAveRelevance(G_{346}) finds the average relevance values of the gene to the set of three diseases.

- *getRelevances(D_i)* is to return the set of all relevance instances related to a disease D_i , i.e. the set of edges for D_i . For example, invoking *getRelatedGenes($D_{stomach\_cancer}$)* returns the relevance instances of $\{(G1, G2), (G1, G3), (G1, G4), (G2, G3), (G2, G5), (G4, G6)\}$.
- *getMostRelevantGenes(x)* is to return the set of 'n' genes having highest average relevance to the set of target diseases. This can be done by choosing the x genes having the highest *averageRelevance()* values.

8.8.3. Defining Analytics Methods

There can be a large number of analytics query types, and we define some of common and essential methods.

- Method to Find Genes relevant to a Disease

This method is to find 'x' number of genes that are the most closely related to a single disease, and its interface is defined as;

- *findRelatedGenes_SD(x, disease)* where 'x' is the number of genes to find and the 'disease' is the target disease of interest.

This methods invokes two underlying methods;

- *findRelatedGenes_MD(x, diseaseSet)* to find genes for a disease set
- *findRelatedGenes(x)* to find genes that are most closely related

The pseudocode for this method is given here;

- // Step 1.

Select all the relations for the given disease, D_i .

Compute $\text{getRelatedGenes}(D_i)$ for each disease D_i in setDiseases

// Step 2

Select all the genes participating in the selected relations.

Compute $\text{commonGenes}(\text{diseaseSet})$ where $\text{commonGenes}(D_1, D_2, D_3)$ be the common set of $\text{getGenes}(D_1)$, $\text{getGenes}(D_2)$, and $\text{getGenes}(D_3)$

// Step 3

Compute Average Relevance Value of each gene in the set of selected genes.

Sort them by the computed values.

Compute $\text{aveStrength}(G_i, D_j)$ for each $G_i \in \text{commonGenes}(D_1, D_2, \dots, D_n)$ and each given disease D_i .

// Step 4

Choose the 'x' genes having highest values from the sorted list.

Chapter 9 Theorems and Proofs

9.1. Colorblind IM-IDM

Theorem 1: $\delta_{IDM}(S)$ is monotonic. i.e.

$$\delta_{IDM}(S \cup \{j\}) \geq \delta_{IDM}(S)$$

Proof: To prove $\delta_{IDM}(S \cup \{j\}) \geq \delta_{IDM}(S)$, let us suppose the statement is false. In some cases, $\delta_{IDM}(S \cup \{j\}) < \delta_{IDM}(S)$. Let $R = \delta_{IDM}(S) - \delta_{IDM}(S \cup \{j\})$. R is the difference between sets $\delta_{IDM}(S)$ and $\delta_{IDM}(S \cup \{j\})$.

Now if there is such a node $n \in R$ that is reachable from a subset of S , $r \subseteq S$. In such case, r should also be a subset of $S \cup \{j\}$ for $S \subseteq S \cup \{j\}$.

All the nodes that activate r belong to set $S \cup \{j\}$, so $n \in \delta_{IDM}(S \cup \{j\})$.

By the definition of difference of sets, n cannot belong to R , $n \notin R$, which contradicts the definition of n .

Thus, our assumption is incorrect, and $\delta_{IDM}(S \cup \{j\}) \geq \delta_{IDM}(S)$.

This completes our proof.

■

Theorem 2: $\delta_{IDM}(S)$ is sub modular.

Proof: We first prove that there is an intrinsic linear threshold model (LTM) activation function in our model. **Using the activation function of IM-IDM, we have**

$$1 - \prod_{i \in N_A} (1 - \inf(i, A)) \geq T_A$$

$$\prod_{i \in N_A} (1 - \inf(i, A)) \leq 1 - T_A$$

$$\prod_{i \in N_A} \log(1 - \inf(i, A)) \leq \log(1 - T_A)$$

$$\sum_{i \in N_A} \log(1 - \inf(i, A))^{-1} \geq \log(1 - T_A)^{-1} .$$

Then, we find the maximum C of $\sum_{i \in N_A} \log(1 - \inf(i, A))^{-1}$ and $\log(1 - T_A)^{-1}$.

We then divide both sides by C , ensuring the range will be $[0, 1]$. According to this inequality, we can find the corresponding LTM because we have proved that there is an intrinsic LTM activation function in our model.

$$\sum_{i \in N_A} \frac{\log(1 - \inf(i, A))^{-1}}{C} \geq \frac{\log(1 - T_A)^{-1}}{C}$$

Then,

$$\sum_{i \in N_A} \inf(i, A) \geq T_A .$$

This completes the proof.

■

Theorem 3: The error of ColorBlind IM-IDM is bounded.

The error bound of a greedy algorithm is defined as the degree of the gap between the solution produced by the greedy algorithm and the solution provided by the optimal version of the algorithm: that is, the difference between an approximation solution and the optimal solution.

Let $A_i = \{v_1, v_2, \dots, v_i\}$ be the solution returned by the greedy algorithm composed of colorblind nodes.

Let $A^* = \{v_1^*, v_2^*, \dots, v_k^*\}$ be the solution returned by the optimal algorithm composed of colorblind nodes.

Let f be an IM-IDM monotone submodular function. Then, $OPT = f(A^*)$, the value of the optimal solution.

Proof:

We want to prove that

$f(A_k) \geq (1-1/e) OPT$, where k is the number of seed nodes in set A^* .

For all $i \leq k$, we have

$$\begin{aligned}
f(A^*) &\leq f(A^* \cup A_i), \text{ based on monotonicity} \\
&= f(A_i) + \sum_{j=1}^k \Delta(v_j^* | A_i \cup \{v_1^*, v_2^*, \dots, v_{j-1}^*\}) \\
&\leq f(A_i) + \sum_{z \in A^*} \Delta(z | A_i), \text{ based on submodularity} \\
&\leq f(A_i) + \sum_{z \in A^*} \Delta(v_{i+1} | A_i), \text{ based on greedy properties} \\
&= f(A_i) + k \Delta(v_{i+1} | A_i).
\end{aligned}$$

Rearranging the terms, we have proved that

$$\Delta(v_{i+1} | A_i) \geq \frac{1}{k} (OPT - f(A_i)).$$

Now we define $\delta_i - \delta_{i+1} = f(A_{i+1} - A_i)$. This implies

$$\delta_i - \delta_{i+1} = f(A_{i+1} - A_i) = \Delta(v_{i+1} | A_i).$$

We now plug this into our previous equation of $\Delta(v_{i+1}|A_i) \geq \frac{1}{k} (\text{OPT} - f(A_i))$ and obtain the following.

$$\delta_i - \delta_{i+1} \geq \frac{1}{k}(\delta_i)$$

This means the element added at iteration $i + 1$ by the greedy algorithm reduces the gap to the optimal solution by at least $\frac{1}{k}(\text{OPT} - f(A_i))$. We rearrange this and obtain the following.

$$\delta_{i+1} \leq (1 - \frac{1}{k})(\delta_i)$$

Recursively applying this definition, we obtain the following.

$$\delta_k \leq (1 - \frac{1}{k})^k(\delta_0)$$

Now, $\delta_0 = \text{OPT} - f(0) \leq \text{OPT}$. We use the well-known bound of $1 - x \leq e^{-x}$ and obtain the following.

$$\delta_k \leq \left(1 - \frac{1}{k}\right)^k \text{OPT} \leq \frac{1}{e} \text{OPT}$$

We now plug this definition of δ_k back and obtain the following.

$$\text{OPT} - f(A_k) \leq \frac{1}{e} \text{OPT}$$

Equivalently, we obtain the following, which proves the error bound.

$$f(A_k) \geq \left(1 - \frac{1}{e}\right) \text{OPT}$$

■

9.2. Colored IM-IDM Theorems and Proofs

Theorem 4: $\delta_{IDM}(S)$ is monotonic. i.e.

$$\delta_{IDM}(S \cup \{j\}) \geq \delta_{IDM}(S)$$

Proof: Proofs for ColorBlind monotonicity (Theorem 1) and Colored monotonicity are the same, because adding colors to a graph does not change our underlying activation function and algorithm.

■

Theorem 5: $\delta_{IDM}(S)$ is sub modular.

Proof: Proofs for ColorBlind submodularity (Theorem 2) and Colored submodularity do not differ, since adding colors to a graph does not change our underlying activation function and algorithm.

We can find the corresponding LTM for both ColorBlind and Colored IM-IDMs because we have proved that there is an intrinsic LTM activation function in our model.

■

Theorem 6: The error of Colored IM-IDM is bounded.

The error bounds are the same for the ColorBlind IM-IDM (Theorem 3) and the Colored IM-IDM. We follow Theorem 3 for the Colored IM-IDM error bound proof, but the initial definition changes as follows:

Let $A_i = \{v_1, v_2, \dots, v_i\}$ be the solution returned by the greedy algorithm composed of i nodes in color c .

Let $A^* = \{v_1^*, v_2^*, \dots, v_k^*\}$ be the solution returned by the optimal algorithm composed of k nodes in color c .

■

9.3. Colored IM-IDM with Constraints Theorems and Proofs

Theorem 7: The error of Colored IM-IDM with constraints is bounded.

We have multiple Colored IM-IDM with constraints algorithms, as we stated in Section 6. This error bound proof is for the “exactly k nodes” algorithm. More specifically, the query would be “Find a set of k nodes that includes n c -colored nodes.”

Let $A_{k-n, \text{others}}$ be the solution returned by the greedy algorithm composed of $k - n$ nodes in any color other than c .

Let $A_{n, c}$ be the solution returned by the greedy algorithm composed of n nodes in color c .

Let $A^* = \{v_1^*, v_2^*, \dots, v_k^*\}$ be the solution returned by the optimal algorithm composed of k nodes in color c .

$$f(A_{k-n,\text{others}}) \geq \left(1 - \frac{1}{e}\right) (\text{OPT}_{k-n,\text{others}})$$

$$f(A_{n,c}) \geq \left(1 - \frac{1}{e}\right) (\text{OPT}_{n,c})$$

$$f(A_{k-n,\text{others}} \cup A_{n,c}) \geq \left(1 - \frac{1}{e}\right) \max(\text{OPT}_{k-n,\text{others}}, \text{OPT}_{n,c})$$

$$\text{OPT}_{k-n,\text{others}} = f(A_{k-n,\text{others}}^*)$$

$$\text{OPT}_{n,c} = f(A_{n,c}^*)$$

$$f(A_{k-n,\text{others}}^* \cup A_{n,c}^*) \leq 2 * \max(\text{OPT}_{k-n,\text{others}}, \text{OPT}_{n,c})$$

Finally, we obtain the error bound of

$$f(A_{k-n,\text{others}} \cup A_{n,c}) \geq \left(1 - \frac{1}{e}\right) * \frac{1}{2} f(A_{k-n,\text{others}}^* \cup A_{n,c}^*).$$

■

Chapter 10 Experiments

10.1. SNS Dataset for Experiments

We used a Twitter social circles dataset that was crawled from public sources. The dataset includes node features (profiles), circles, and ego networks. The size of the datasets for our experiments are 835 nodes and 9,000 edges. The original dataset has 81,306 nodes and 1,768,149 edges. We reduced the dataset size for experiments.

Figure 59 shows how the dataset for our experiments was created. Threshold values were randomly assigned with parameters. We used a mean value of 0.3304 and a standard deviation of 0.1001. Influence was also assigned randomly with parameters: a mean value of 0.6983 and a standard deviation of 0.1015.

■ Node

Node ID (Generated by MySQL)
Node Name (from Twitter)
UI Coordinates (Assigned by Program)
IDM ID (Assigned by Program)
Person ID (Assigned by Program)
Node Type (Assigned by Program)
Threshold (Assigned by Program)

id	name	SCN_x	SCN_y	graph_id	person_id	nodetype_id	threshold
119493	17885184	-450	-450	16	41027	31	0
119494	47015938	-420	-450	16	61258	31	0.0185788
119495	14551043	-390	-450	16	5370	31	0
119496	14809096	-360	-450	16	84146	32	0
119497	8067082	-330	-450	16	48592	32	0
119498	75939851	-300	-450	16	34866	32	0.345917
119499	7434252	-270	-450	16	62511	31	0.0432454
119500	15814666	-240	-450	16	57525	32	0.270294
119501	14	-210	-450	16	4060	31	0.0299661

■ Edge

Source Node (from Twitter)
Target Node (from Twitter)

source_node	target_node	influence	graph_id
119493	119650	0.786235	16
119496	119510	0.521737	16
119496	119558	0.796402	16
119496	119589	0.616828	16
119496	119717	0.960502	16
119496	119758	0.647046	16
119496	119967	0.706458	16
119496	120096	0.733863	16

Figure 60. Dataset for Experiments

10.2. Experimenting with ColorBlind IM-IDM Algorithm

We experimented with the ColorBlind IM-IDM with varying values for s ($s = 1, 2, 3, 5$, and 10), the number of seed nodes. To measure the efficiency of our heuristic algorithm, we ran both the brute-force and heuristic algorithms to compare accuracy and elapsed time.

As can be seen in Figure 61, the accuracy for our heuristic algorithm was 89.1% at $s = 2$ and declined to 75.2% when $s = 10$. However, we observe that at $s = 10$, the time savings from using our heuristic algorithm was significant, as illustrated in Figure 62. Our heuristic algorithm took 6.428 s to find 10 seed nodes, whereas the brute-force algorithm required 87,022 s.

Figure 61. Accuracy for ColorBlind IM-IDM

Figure 62. Time Elapsed for ColorBlind IM-IDM

10.3. Experimenting with Colored IM-IDM Algorithm

We also experimented with the Colored IM-IDM with varying values for s ($s = 1, 2, 3, 5$, and 10), the number of seed nodes. Once again, we ran both the brute-force and heuristic algorithms to compare the accuracy and the elapsed time to measure the efficiency of our heuristic algorithm.

As is shown in Figure 63, the accuracy for our heuristic algorithm was 89.7% for $s = 2$ but declined to 77.3% when $s = 10$. As before, however, we observed that at $s = 10$ the time savings from our heuristic algorithm was significant, as illustrated in Figure 64. Our heuristic algorithm found 10 seed nodes in just 6.334 s, whereas the brute-force approach needed 93,480 s.

Figure 63. Accuracy for Colored IM-IDM

Figure 64. Time Elapsed for Colored IM-IDM

Chapter 11 Conclusions and Future Work

Influence analysis is one of the most trending and important topics in social network research. Growing numbers of researchers and advertisers are interested in maximizing their influence in such networks. Additionally, research on influence maximization (IM) problems is an emerging area. In a social network, knowing how to find diffusion rules for information and mastering models for information diffusion are essential for understanding how information spreads in the network.

In this thesis, we address influence maximization problems based on the IDM in k -color graphs and propose a new influence diffusion model. Our motivation is to solve problems in the Influence Maximization in Information Diffusion space that current algorithms and models lack the accuracy while yielding high time complexity.

We discuss the basic IM problem based on the IDM and propose new methods for solving such problems with colors. We also research and present method for solving such problems with node constraints in the k -color graph. Finally, in addition to providing formal proofs on the proposed algorithms, we present our experiment results comparing accuracy and time elapsed for Brute Force and our heuristic algorithms proving that our algorithms and models yield high accuracy while reducing the time complexities.

We have developed a lossy version of IM-IDM from the observation that IM-IDM models are often too complex to deliver the desired solutions with a reasonable time efficiency. Our lossy IM-IDM algorithm consists of two stages where the first stage is to reduce the given graph G with a reference threshold value r . In the second stage, we apply the same set of algorithms presented in this paper for finding the set of seed nodes on the reduced graph G' . We have also extended the IM-

IDM with different colors assigned to edges. We have created a gene-disease network with Pubmed data and assigned colors to edges so edges can represent a particular type of disease and a dependency occurrence for a disease between two genes.

In the future, we will add another constraint, i.e., budget to our IM-IDM. This would enable us to answer not only simple budget IM-IDM queries such as ‘find 3 red seed nodes that would maximize the graph under the budget of 100’, but could also be branched out for addressing various other budget-related use cases. Using our IM-IDM, budget constraints can be incorporated into the algorithms we have presented in this dissertation.

Bibliography

1. R. Dunbar. Neocortex size as a constraint on group size in primates. *Human Evolution*, 1992, 20: 469–493.
2. R. M. Bond, C. J. Fariss, J. J. Jones, A. D. I. Kramer, C. Marlow, J. E. Settle and J. H. Fowler. A 61-million-person experiment in social influence and political mobilization. *Nature*, 489:295-298, 2012.
3. P. Domingos and M. Richardson. Mining the network value of customers. In *KDD’01*, pages 57–66, 2001.
4. R. Wilson, S. Gosling and L. Graham. A Review of Facebook Research in the Social Sciences. 2012.
5. Why I Deleted My Klout Profile, by Pam Moore, at Social Media Today, originally published November 19, 2011; retrieved November 26 2011
6. J.H. Fowler and N.A. Christakis. The Dynamic Spread of Happiness in a Large Social Network: Longitudinal Analysis Over 20 Years in the Framingham Heart Study. *British Medical Journal* 2008; 337: a2338
7. S. Aral, L. Muchnik, and A. Sundararajan. Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks. *PNAS*, 106 (51):21544-21549, 2009.
8. J. Scripps, P.-N. Tan, and A.-H. Esfahanian. Measuring the effects of preprocessing decisions and network forces in dynamic network analysis. In *KDD’09*, pages 747–756, 2009.
9. E. Bakshy, B. Karrer, and L. A. Adamic. Social influence and the diffusion of user-created content. In *EC ’09*, pages 325–334, New York, NY, USA, 2009. ACM.
10. M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring Networks of Diffusion and Influence. In *KDD’10*, pages 1019–1028, 2010.
11. Centola D. The Spread of Behavior in an Online Social Network Experiment. *Science*, 2010, 329(5996): 1194-1197.
12. Cha M, Mislove A, Gummadi K P. A Measurement-driven Analysis of Information Propagation in the Flickr Social Network. *Proceedings of the 18th international conference on World wide web*. ACM 2009. 721-730.

13. Katz E. The two-step flow of communication: An up-to-date report on an hypothesis. *Public Opinion Quarterly*, 1957, 21(1):61–78.
14. Nisbet M C, Kotcher J E. A two-step flow of influence? opinion-leader campaigns on climate change. *Science Communication*, 2009.
15. Webb D, Jamison T F. Continuous flow multi-step organic synthesis. *Chemical Science*, 2010,1(6):675–680.
16. Seidl T, Kriegel H P. Optimal multi-step k-nearest neighbor search. *ACM SIGMOD Record*, volume 27. ACM, 1998. 154–165.
17. Huang J, Jin X. Preventing rumor spreading on small-world networks. *Journal of Systems Science and Complexity*, 2011, 24(3):449–456.
18. Zhao L, Wang J, Chen Y, et al. Sihr rumor spreading model in social networks. *Physica A: Statistical Mechanics and its Applications*, 2012, 391(7):2444–2453.
19. Yanchao Zhang, Yun Liu and Haifeng Zhang. Information Diffusion Model based on Online Social Network. *Physics Journal of China*, 2011, 60(5):50501–050501.
20. Gao J, Ghasemiefteh G, Schoenebeck G and Yu F. General Threshold Model for Social Cascades: Analysis and Simulations. 2016.
21. Kempe D, Kleinberg J and Tardos E. Maximizing the spread of influence through a social network, 2003. SIGKDD.
22. Anderson R M, May R M. Infectious diseases of humans, volume 1. Oxford university press Oxford, 1991.
23. Chen W, Wang Y, Yang S. Efficient influence maximization in social networks. In: *Proc. of the 15th ACM SIGKDD Int’l Conf. on Knowledge Discovery and Data Mining (KDD 2009)*. New York: ACM Press, 2009. 199–208.
24. Jung K, Heo W, Chen W. Irie: Scalable and robust influence maximization in social networks. In: *Proc. of the 12th Int’l Conf. on Data Mining (ICDM 2012)*. New York: ACM Press, 2012. 918–923.
25. Goyal A, Bonchi F, Lakshmanan LVS. A data-based approach to social influence maximization. *Proc. of the VLDB Endowment*, 2011, 5(1):73–84.
26. Tong G, Wu W, Tang S, Du D. Adaptive influence maximization in dynamic social networks. *arXiv*, 1506.06294. 2015.

27. Zhuang H, Sun Y, Tang J, Zhang J, Sun X. Influence maximization in dynamic social networks. In: Proc. of the 13th Int'l Conf. on Data Mining (ICDM 2013). New York: ACM Press, 2013. 1313–1318.
28. Liu B, Cong G, Zeng Y, Sun X. Influence spreading path and its application to the time constrained social influence maximization problem and beyond. IEEE Trans. on Knowledge and Data Engineering, 2014,26(8):1904–1917.
29. Bharathi S, Kempe D, Salek M. Competitive influence maximization in Social Networks. In Proceedings of WINE 2007, San Diego, CA.
30. Chen W, Wang Y, Yang S. Efficient influence maximization in social networks. Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2009. 199- 208.
31. Chen W, Yuan Y, Zhang L. Scalable influence maximization in social networks under linear threshold model. Data Mining (ICDM), 2010 IEEE 10th International Conference on IEEE, 2010. 88-97.
32. Leskovec J, Krause A, Guestrin C, et al. Cost-effective outbreak detection in networks. Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2007. 420–429.
33. Goyal A, Lu W, Lakshmanan L V. Celf++: optimizing the greedy algorithm for influence maximization in social networks. Proceedings of the 20th international conference companion on World wide web. ACM. 2011. 47-48.
34. J. Jin, G. Zhang, P. C-Y Sheu, M. Hiyakawa, A. Kitazawa, "Influence Maximization in Graph-based OLAP (GOLAP)," Journal of Social Network Analysis and Mining, Springer, 2019 9: 54
35. B. Chou and M. Kim, Graph Online Analytical Processing. Encyclopedia with Semantic Computing. 2018
36. S. Wang, G. Zhang, P. C-Y Sheu, M. Hayakawa, H. Shigematsu and Atsushi Kitazawa, "Lossy Graph Reduction," International Journal of Semantic Computing, Vol. 12, No. 3 (2018) 425–456