

Virginia Commonwealth University
VCU Scholars Compass

Theses and Dissertations | Graduate School

2018

Big Networks: Analysis and Optimal Control

Hung The Nguyen
Virginia Commonwealth University

Follow this and additional works at: https://scholarscompass.vcu.edu/etd

 Part of the Artificial Intelligence and Robotics Commons, Information Security Commons, Numerical Analysis and Scientific Computing Commons, and the Theory and Algorithms Commons

© The Author

Downloaded from
https://scholarscompass.vcu.edu/etd/5514

This Dissertation is brought to you for free and open access by the Graduate School at VCU Scholars Compass. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of VCU Scholars Compass. For more information, please contact libcompass@vcu.edu.

©Hung T. Nguyen, June 2018

All Rights Reserved.

BIG NETWORKS: ANALYSIS AND OPTIMAL CONTROL

A Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Virginia Commonwealth University.

by

HUNG T. NGUYEN

Bachelor of Science, Vietnam National University, Vietnam - 2014

Director: Thang N. Dinh, PhD,

Assistant Professor, Department of Computer Science

Virginia Commonwealth University

Richmond, Virginia

June, 2018

Acknowledgements

I am deeply indebted to my advisor, Dr. Thang N. Dinh, for all his immeasurable time and patience in advising me through the whole four years of my PhD study. Looking back in time, I appreciate every rewarding discussion that could last for the whole day. I have learned a wealth from his research expertise and through the mistakes that I made and got confronted in those discussions.

I sincerely thank my committee members, Dr. Tom Arodz, Dr. Preetam Ghosh, Dr. Siqian Shen and Dr. Tam Vu, for all their insightful and worthwhile advices in both specific research topics and my future researcher career.

I would like to express my gratitude to all the faculty members, staffs and fellow graduate students in the Department of Computer Science, School of Engineering, Virginia Commonwealth University for creating an excellent environment for learning and collaborating in research.

Of course, I am so thankful for having my family and friends with me all the time and their constant support at every single step on my journey.

TABLE OF CONTENTS

Chapter Page

Acknowledgements i

Table of Contents ii

List of Tables viii

List of Figures x

Abstract xiv

1 Introduction 1

1.1 Research Scopes, Objectives, and Motivations of the Dissertation 1
1.1.1 Information Diffusion 2
1.1.1.1 Influence Estimation (IE) 2
1.1.1.2 Influence Maximization (IM) 3
1.1.1.3 Tracing the Sources of Misinformation Cascades 5
1.1.2 Community Detection 6
1.1.3 Security and Privacy: Targeted Cyber-attacks 6
1.2 Contributions of the Dissertation 7
1.2.1 Information Diffusion 8
1.2.1.1 Influence Estimation (IE) 8
1.2.1.2 Influence Maximization (IM) 10
1.2.1.3 Tracing the Sources of Misinformation Cascades 13
1.2.2 Community Detection 14
1.2.2.1 Community Detection in Multiplex Social Networks . . . 14
1.2.2.2 Community Detection in Multi-attributed Networks 15
1.2.3 Security and Privacy: Targeted Cyber-attacks 16
1.3 Organization of the Dissertation 17

2 Influence Estimation 18

2.1 Basic Concepts and Definitions 18
2.1.1 Probabilistic Graph 18
2.1.2 Diffusion Models 18

2.1.3 Monotone Submodular Functions 19
2.2 High-confident Influence Estimation 20
2.2.1 Definitions and Properties 21
2.2.1.1 Graph Samples and Probabilistic Space 22
2.2.1.2 Outward Influence under the IC model 24
2.2.1.3 Hardness of Computation 25
2.2.2 Outward Influence Estimation via Importance Sampling 27
2.2.2.1 Importance IC Polling 28
2.2.2.2 FPRAS for Outward Influence Estimation 31
2.2.3 Efficient Mean Estimation for Bounded Random Variables 32
2.2.3.1 Generalized Stopping Rule Algorithm 32
2.2.3.2 High-confident Sampling Algorithm 37
2.2.4 Influence Estimation at Scale 39
2.2.4.1 Outward Influence Estimation 40
2.2.4.2 Influence Spread Estimation 41
2.2.4.3 Influence Spread under other Models 44
2.2.4.4 Parallel Estimation Algorithms 44
2.2.5 Experiments 45
2.2.5.1 Experimental Settings 45
2.2.5.2 Outward Influence Estimation 48
2.2.5.3 Influence Spread Estimation 49
2.2.5.4 Scalability Test 51
2.2.5.5 Influence Estimation under LT Model 54
2.2.6 Related work 54
2.2.7 Conclusion 55
2.2.8 Appendix 56
2.2.8.1 Proof of Lemma 1 56
2.2.8.2 Proof of Lemma 4 58
2.2.8.3 Proof of Theorem 2 59
2.2.8.4 Proof of Theorem 3 63
2.2.8.5 Proof of Lemma 6 65
2.3 Importance Sketching for Influence Estimation in Billion-scale Networks 66
2.3.1 Preliminaries 67
2.3.2 Influence Estimation/Maximization Problems 67
2.3.3 Sketch-based Methods for IE/IM 68
2.3.3.1 Reverse Influence Sketch (RIS) 68
2.3.3.2 Combined Reachability Sketch (SKIM) 69
2.3.4 Importance Sketching 71

2.3.4.1 Importance Influence Sampling (ROIS) . . . 72
2.3.5 Influence Oracle via IIS Sketch (SKIS) . . . 75
2.3.6 SKIS-based IM Algorithms . . . 79
2.3.6.1 Greedy Algorithm on SKIS Sketches . . . 80
2.3.6.2 Sufficient Size of SKIS Sketch for IM . . . 81
2.3.7 Extensions to other diffusion models . . . 84
2.3.8 Experiments . . . 86
2.3.8.1 Experimental Settings . . . 86
2.3.8.2 Influence Estimation . . . 89
2.3.8.3 Influence Maximization . . . 92
2.3.9 Conclusion . . . 94
2.3.10Omitted Proofs of Lemmas and Theorems . . . 95
2.3.10.1 Proof of Lemma 9 . . . 95
2.3.10.2 Proof of Lemma 10 . . . 97
2.3.10.3 Proof of Lemma 11 . . . 97
2.3.10.4 Proof of Lemma 12 . . . 98
2.3.10.5 Proof of Lemma 14 . . . 99

3 Influence Maximization . . . 101

3.1 Optimal Sampling Algorithms for Influence Maximization . . . 101
3.1.1 Unified RIS framework . . . 103
3.1.1.1 Preliminaries . . . 103
3.1.1.2 RIS Framework and Thresholds . . . 106
3.1.1.3 Two Types of Minimum Thresholds . . . 108
3.1.2 Stop-and-Stare Algorithm (SSA) . . . 110
3.1.2.1 SSA Algorithm . . . 111
3.1.2.2 Parameter Settings for SSA . . . 114
3.1.3 SSA Theoretical Analysis . . . 115
3.1.3.1 Approximation Guarantee . . . 115
3.1.3.2 Achieving Type-1 Minimum Threshold . . . 116
3.1.4 Dynamic Stop-and-Stare Algorithm . . . 119
3.1.4.1 Theoretical Guarantees Analysis . . . 121
3.1.5 Experiments . . . 123
3.1.5.1 Experimental Settings . . . 125
3.1.5.2 Experiments with IM problem . . . 126
3.1.5.3 Experiments with TVM problem . . . 131
3.1.6 Conclusion . . . 133
3.2 Cost-aware Targeted Viral Marketing . . . 151

3.2.1 Models and Problem Definitions . . . 153
3.2.1.1 Model and Problem Definition . . . 153
3.2.1.2 Summary of the RIS Approach . . . 154
3.2.2 Benefit-aware Reverse Influence Sampling . . . 158
3.2.2.1 Summary of the RIS method . . . 158
3.2.2.2 Efficient Sampling Strategy for CTVM . . . 161
3.2.3 BCT Approximation Algorithm . . . 163
3.2.3.1 Efficient Benefit Sampling Algorithm - BSA . . . 163
3.2.3.2 Solving Budgeted Max-Coverage Problem . . . 165
3.2.3.3 BCT - The Main Algorithm . . . 166
3.2.4 Approximation and Complexity Analysis . . . 166
3.2.4.1 Approximation Guarantee for uniform cost CTVM . . . 167
3.2.4.2 Time Complexity . . . 169
3.2.4.3 Sample Complexity and Comparison to IMM . . . 169
3.2.4.4 Approximation Algorithm for Arbitrary Cost CTVM . . . 170
3.2.4.5 Extension to IC model . . . 171
3.2.5 Experiments . . . 172
3.2.5.1 Experimental Settings . . . 172
3.2.5.2 Experimental results . . . 175
3.2.5.3 Comparison of solution quality on CTVM . . . 176
3.2.5.4 Comparison of solution quality on IM . . . 176
3.2.5.5 Comparison of running time . . . 177
3.2.5.6 Robustness Testing . . . 177
3.2.6 Twitter: A billion-scale social network . . . 178
3.2.6.1 Compare BCT against IMM and TIM+ . . . 179
3.2.6.2 A Case Study on Twitter network. . . . 179
3.2.7 Martingale View on Benefit Estimation . . . 181
3.2.8 Proofs of lemmas and theorems . . . 183
3.2.8.1 Proof of Lemma 51 . . . 183
3.2.8.2 Proof of Lemma 37 . . . 184
3.2.8.3 Proof of Lemma 39 . . . 185
3.3 Social Influence Spectrum with Guarantees . . . 186
3.3.1 Problem Definition . . . 186
3.3.1.1 IS through Greedy approach. . . . 187
3.3.2 Simultaneous High-confident Estimation of Influence Spectrum . . . 188
3.3.2.1 Number of Samples (RR sets) . . . 190
3.3.2.2 Efficient Influence Spectrum Estimation . . . 192
3.3.3 LISA Approximation Algorithm for Identify Multiple Seed Sets . 194

3.3.3.1 Approximation Guarantees . . . 196
3.3.3.2 Complexity Analysis . . . 203
3.3.3.3 IM . . . 206
3.3.3.4 Extension to IC model . . . 206
3.3.4 Experiments . . . 207
3.3.4.1 Experimental Settings . . . 207
3.3.4.2 Results . . . 210
3.3.5 Conclusion . . . 213

4 Tracing the Sources of Misinformation Cascades . . . 214

4.1 Related works . . . 215
4.2 Models and Problem definition . . . 217
4.2.1 Infection Model . . . 218
4.2.2 Problem Formulation . . . 219
4.2.3 Hardness and Inapproximability . . . 221
4.3 Sampling-based SISI algorithm . . . 225
4.3.1 Truncated Reverse Infection Sampling . . . 226
4.3.1.1 Generating RR Sets under SI model. . . . 226
4.3.1.2 Chance of Being Infection Sources . . . 230
4.3.2 Submodular-cost Covering . . . 234
4.3.3 SISI Approximation Algorithm . . . 238
4.4 Algorithm Analysis . . . 239
4.4.1 Approximation Guarantee . . . 240
4.4.2 Time Complexity . . . 245
4.4.2.1 Submodular-cost covering algorithm . . . 245
4.4.2.2 Generating RR sets . . . 245
4.5 Experiments . . . 248
4.5.1 Experimental Settings . . . 248
4.5.1.1 Algorithms compared . . . 248
4.5.1.2 Quality measures . . . 249
4.5.1.3 Datasets . . . 250
4.5.1.4 Testing Environments . . . 250
4.5.2 Experiments on real network and SI model . . . 251
4.5.3 Experiments on the IC model . . . 252
4.6 Discussion and Conclusion . . . 253

5 Community Detection . . . 255

5.1 Non-negative Matrix Factorization (NMF) . . . 255

5.2 Community Detection in Multiplex Social Networks 256
5.2.1 Problem formulation 256
5.2.2 Unifying approach 258
5.2.2.1 Network aggregation 258
5.2.2.2 NMF-based algorithm on the original networks 259
5.2.3 Coupling approach 264
5.2.3.1 Coupling techniques 264
5.2.3.2 Directed networks 265
5.2.3.3 Undirected networks 266
5.2.4 Experiments 267
5.2.4.1 Extend LFR bechmark 267
5.2.4.2 Dataset and Settings 268
5.2.4.3 Experimental results 268
5.2.5 Conclusions 271
5.3 Community Detection in Multi-attributed Networks 272
5.3.1 Models and problem formulation 272
5.3.2 Methods 275
5.3.2.1 3NCD Algorithm 275
5.3.2.2 Proof of convergence to a KKT stationary point 278
5.3.2.3 Complexity analysis 281
5.3.3 Experiments 281
5.3.3.1 Datasets. 281
5.3.3.2 Measurement metrics. 282
5.3.3.3 Experimental Results 283
5.3.4 Conclusion 286

6 Security and Privacy 287

6.1 Target Reconnaissance Strategy via Social Networks 287
6.1.1 Problem Formulation and NP-hardness 288
6.1.1.1 Model and Problem Formulation 288
6.1.1.2 NP-hardness 291
6.1.2 Adaptive Greedy Policy and Guarantee 294
6.1.2.1 Adaptive Greedy Policy 294
6.1.2.2 Approximation Guarantee 295
6.1.3 Simulation 298
6.1.3.1 Simulation Settings 299
6.1.3.2 Simulation Results 299
6.1.4 Conclusion 302

7 Conclusion and Future Works 303
7.1 Conclusion 303
7.2 Future works 303
Appendix A List of Publications by the Author 304
References 308
Vita 320

LIST OF TABLES

Table		Page
1	Table of notations	26
2	Datasets' Statistics	46
3	Comparing performance of algorithms in estimating outward influences	46
4	Comparing performance of algorithms in estimating influence spread in WC Model (seed set size $\|S\| = 1$)	50
5	Comparing performance of algorithms in estimating influence spread in WC Model (seed set size $\|S\| = 5\%\|V\|$)	50
6	Comparing performance of algorithms in estimating influence spread in LT model (seed set size $\|S\| = 1$)	53
7	Table of notations	67
8	Datasets' Statistics	86
9	Average relative differences (dnf: "did not finish" within 24h). SKIS almost always returns the lowest errors.	89
10	Performance of IM algorithms with $k = 100$ (dnf: "did not finish" within 6h, mem: "out of memory").	90
11	Sketch construction time and index memory of algorithms on different edge models. SKIS and RIS uses roughly the same time and memory and less than those of SKIM.	91
12	Table of notations	102
13	Datasets' Statistics	124
14	Performance of SSA, D-SSA and IMM on various datasets under LT model.	131
15	Topics, related keywords	132

16 Main results of related methods (k is the number of selected seed nodes, n is the number of nodes in the graph, m is the number of edges) 152

17 Table of Notations 155

18 Datasets' Statistical Summary 173

19 Comparison between different methods on IM problem and various datasets (with $\epsilon = 0.1, k = 50, \delta = \frac{1}{n}$). 175

20 Robustness results (% to true value). 178

21 Topics, related keywords 180

22 Table of Notations 188

23 Datasets' Statistics 207

24 Table of Notations 217

25 Comparison on Symmetric Difference and Jaccard-based Distance of different methods. 248

26 True sources detected (%) with $|V_I| = 1000$. 251

27 Comparison under the IC model. 253

28 Abbreviation Table 273

29 Datasets summary. 282

30 Accuracy of all the methods in terms of F1 score (Notions: T - Topology only, T&A - Topology + Attributes). 283

31 Accuracy of the methods in terms of Jaccard similarity. 284

32 Datasets' statistical summary 298

LIST OF FIGURES

Figure Page

1 Left: the influence of nodes under IC model. The influence of all nodes are roughly the same, despite that w is much less influential than u and v. Right: Outward influence is better at reflecting the relative influence of the nodes. w has the least outward influence, 0, while v's is nearly twice as that of u. 8

2 Infection sources detection on a 60×60 grid graph 13

3 Neighbors of nodes in S 28

4 Error distributions (histogram) of the approximation errors of SOIEA, $\mathsf{MC}_{10\mathsf{K}}$, $\mathsf{MC}_{100\mathsf{K}}$ on NetHEP 47

5 Running time of SIEA on synthetic networks 52

6 Comparing SIEA, $\mathsf{MC}_{10\mathsf{K}}$ and $\mathsf{MC}_{\epsilon,\delta}$ on Twitter 52

7 Distribution of reversed cascade sizes on various real-world networks with two popular different edge weight models: Trivalency (TRI) and Weighted Cascade (WC). The majority of the cascades are singular. 70

8 Relative difference on Epinions under a) TRI model and b), c), d) WC model with $|S| = 1$. SKIS are the closest to the 'ground truth' among the three sketches. 87

9 Efficiency of SKIS and RIS sketches in finding the maximum seed sets. SKIS sketch is up to 80% more efficient. 88

10 Running time of algorithms under the IC model. 92

11 Running time of algorithms under the LT model. 92

12 An example of generating random RR sets under the LT model. Three random RR sets R_1, R_2 and R_3 are generated. Node a has the highest influence and is also the most frequent element across the RR sets. 103

13 Expected Influence under LT model. 124

14 Expected Influence under IC model. 125

15 Running time under LT model 127

16 Running time under IC model 128

17 Memory usage under LT model 129

18 Memory usage under IC model 130

19 Running time on Twitter network 133

20 Comparisons on CTVM problem. The whole column indicates influence of the selected seeds while the darker colored portion reflects the benefit gained from that set. 174

21 Comparison on IM problem under the LT model. 174

22 Comparison on IM problem under the IC model. 175

23 BCT, IMM and TIM++ on Twitter 179

24 Case Study on real-world Twitter 180

25 Spread of Influence under the LT model (the higher the better) 208

26 Running time of the algorithms under the LT model (see Fig. 25 for legends) . 209

27 Memory usage of the algorithms under the LT model (see Fig. 25 for legends) . 210

28 Running time of the best algorithms on the billion-scale Twitter network . . . 212

29 Memory usage of the best algorithms on the billion-scale Twitter network . . . 212

30 Illustration of symmetric difference. 219

31 Reduction from Red-Blue Set Cover to ISI in which infected nodes are blue-colored and uninfected nodes are in red. 222

32 Conversion to Submodular-cost covering. 235

33 F1-measure scores of different algorithms. Higher is better. 247

34 Runtime of the tested algorithms 247

35 A toy example of 7 users participating in three OSNs, namely Facebook, Twitter and LinkedIn. If we analyze each layer separately, node 3 can be grouped with node 1 and 2 or with 4 and 5 in Facebook network. However, with the information from Twitter, we can surely assign node 3 to the same community with nodes 1, 2. From LinkedIn, we obtain one more structural information that nodes 3, 4, 5, 6, 7 should be in the same group. 257

36 NMI scores on network with $p = 100\%$. 269

37 NMI scores on network with $p = 20\%$. 270

38 Quality of detection with different mixing parameters ($p = 20\%$) 271

39 Quality of detection with different matching fractions ($\mu = 0.3$) 271

40 Combination model of topology and node attributes with CS: **(1)** the original graph with 7 nodes $V = \{v_1, ..., v_7\}$, two communities $\{\mathbf{C}_1, \mathbf{C}_2\}$ and three node attributes $T = \{a_1, a_2, a_3\}$, **(2)** the tripartite graph describing relationships between nodes, communities and attributes, **(3)** generative formula for A_{34} (the expected number of edge between nodes 3 and 4) and P_{31} (the likelihood of node 3 having attribute a_1). 274

41 Running time of the methods on three dataset collections. No runtime for Infomap on Google+ dataset due to running out of 24 hours. 285

42 Accuracy when removing some portion of edges on Facebook, Twitter and Google+ datasets, respectively from left to right. 286

43 Reduction from Maximum Coverage to ATCM 292

44 Comparison between policies on various networks. 300

45 Greedy policy on two communities (Lines indicate the utility percentage w.r.t. the total and bars describe the percentage of selected nodes inside and outside targeted set). 301

Abstract

BIG NETWORKS: ANALYSIS AND OPTIMAL CONTROL

By Hung T. Nguyen

A Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Virginia Commonwealth University.

Virginia Commonwealth University, 2018.

Director: Thang N. Dinh, PhD,

Assistant Professor, Department of Computer Science

The study of networks has seen a tremendous breed of researches due to the explosive spectrum of practical problems that involve networks as the access point. Those problems widely range from detecting functionally correlated proteins in biology to finding people to give discounts and gain maximum popularity of a product in economics. Thus, understanding and further being able to manipulate/control the development and evolution of the networks become critical tasks for network scientists. Despite the vast research effort putting towards these studies, the present state-of-the-arts largely either lack of high quality solutions or require excessive amount of time in real-world 'Big Data' requirement.

This research aims at affirmatively boosting the modern algorithmic efficiency to approach practical requirements. That is developing a ground-breaking class of algorithms that provide simultaneously both *provably good solution qualities* and *low time and space complexities*. Specifically, I target the important yet challenging problems in the three main areas:

- *Information Diffusion*: Analyzing and maximizing the influence in networks and

extending results for different variations of the problems.

- *Community Detection*: Finding communities from multiple sources of information.
- *Security and Privacy*: Assessing organization vulnerability under targeted-cyber attacks via social networks.

CHAPTER 1

INTRODUCTION

The study of networks has seen an incredible surge in both depth and breadth dimensions due to the ubiquitous role of network representation for real-world problems. As now one of the fastest-growing platforms for marketing, political campaigns or even negative propagandas, Online Social Networks (OSNs) with billion of users and connections have disclosed an open-ended line of questions with broad applications, e.g., Information Diffusion which focuses on propagations of influence, rumors or viruses over a probabilistic network has found numerous practices in viral marketing, controlling/containing the epidemics/propaganda. Thus, understanding and further being able to control the dynamical development or evolution of networks become vital tasks for network scientists. As a result, such a rich body of research effort from various disciplines have been devoted to studying the intrinsic properties of the networks and controlling the dynamic processes modeling the diffusion of information or viruses. However, many of the fundamental questions on networks have not been answered satisfactorily due to their intractability nature plus with the unprecedented sizes of their real-world instances, e.g. networks with millions or billions of nodes and connections.

1.1 Research Scopes, Objectives, and Motivations of the Dissertation

This research aims towards affirmatively answering a wide variety of fundamental problems that have not been solved rigorously. These tasks range from analyzing the network structure, nodal properties to initiating, steering and stopping the dynamic processes of information diffusion on networks. In particular, the problems of interest in this disser-

tation fall into three important yet challenging areas: Information Diffusion, Community Detection, Security and Privacy on Online Social Networks (OSNs).

1.1.1 Information Diffusion

Information diffusion studies the cascade or propagation of information, innovations, rumors or viruses, which are generalized and termed *Influence*, over networks and have found applications in viral marketing, outbreak detection and finding news leaders, trend-setters, etc. We focus on three primary tasks:

- **Influence Estimation (IE)**: Estimate the influence spread, i.e. cascade size, if the propagation starts from a node.
- **Influence Maximization (IM)**: Find a set of k nodes in the network to maximize the influence spread.
- **Tracing the Sources of Misinformation Cascades (TMC)**: Given an aftermath of a propaganda, find a set of source nodes (*unknown how many nodes*) that best explains the misinformation cascade.

Various empirical extensions of the IM, i.e., Cost-aware Targeted Viral Marketing (CTVM) and Influence Spectrum (IS), are also investigated as described in the following.

1.1.1.1 Influence Estimation (IE)

A fundamental task in analyzing the cascades of influence is to estimate the cascade size, also known as *influence spread* in social networks. This task is the foundation of the solutions for many applications including viral marketing [55, 106, 105, 84], estimating users' influence [32, 77], optimal vaccine allocation [98], identifying critical nodes in the network [29], and many others. Yet this task becomes computationally challenging in the face of the nowadays social networks that may consist of billions of nodes and edges.

Most of the existing work in network cascades uses stochastic diffusion models and estimates the influence spread through sampling [55, 25, 29, 105, 77, 92]. The common practice is to use a fixed number of samples, e.g. 10K or 20K [55, 105, 25, 92], to estimate the expected size of the cascade, aka *influence spread*. Not only is there no single sample size that works well for all networks of different sizes and topologies, but those approaches also do not provide any accuracy guarantees. Recently, Lucier et al. [77] introduced INFEST, the first estimation method that comes with accuracy guarantees. Unfortunately, our experiments suggest that INFEST does not perform well in practice, taking hours on networks with only few thousand nodes. *Will there be a rigorous method to estimate the cascade size in billion-scale networks?*

1.1.1.2 Influence Maximization (IM)

Given a network and a budget k, Influence Maximization (IM) asks for k influential users who can spread the influence widely into the network. Kempe et al. [55] were the first to formulate IM as a combinatorial optimization problem on the two pioneering diffusion models, namely, *Independent Cascade* (IC) and *Linear Threshold* (LT). They prove IM to be NP-hard and provide a natural greedy algorithm that yields $(1 - 1/e - \epsilon)$-approximate solutions for any $\epsilon > 0$. This celebrated work has motivated a vast amount of work on IM in the past decade [70, 19, 47, 46, 25, 93, 106]. However, most of the existing methods either too slow for billion-scale networks [55, 70, 47, 46, 25, 93] or ad-hoc heuristics without performance guarantees [20, 19, 52, 111].

The most scalable methods with performance guarantee for IM are TIM/TIM+[106] and latter IMM[105]. They utilize a novel RIS sampling technique introduced by Borgs et al. in [11]. All these methods attempt to generate a $(1 - 1/e - \epsilon)$ approximate solution with minimal numbers of RIS samples. They use highly sophisticated estimating methods to make the number of RIS samples close to some theoretical thresholds θ [106, 105].

However, they all share two shortcomings: 1) the number of generated samples can be arbitrarily larger than θ, and 2) the thresholds θ are not shown to be the minimum among their kinds.

Furthermore, the formulation of viral marketing as the IM problem encloses two impractical assumptions: 1) any seed user can be acquired with the same cost and 2) the same benefit obtained when influencing one user. The first assumption implies that incentivizing high-profile individuals costs the same as incentivizing common users. This often leads to impractical solutions with unaffordable seed nodes, e.g., the solutions in Twitter often include celebrities like Katy Perry or President Obama. The second assumption can mislead the company to influence "wrong audience" who are neither interested nor potentially profitable. In practice, companies often target not all users but specific sets of potential customers, decided by the factors like age and gender. Moreover, the targeted users can bring different amount of benefit to the company. Thus, simply counting the number of influenced users, as in the case of IM, does not measure the true impact of the campaign and lead to the choosing of wrong seed set. A few recent works attempt to address the above two issues *separately*. In [90] the authors study the *Budgeted Influence Maximization* (BIM) that considers an arbitrary cost for selecting a node and propose an $(1-1/\sqrt{e}-\epsilon)$ approximation algorithm for the problem. However, their algorithm is not scalable enough for billion-scale networks. Recently, there is a serial works in [9, 7, 18] investigating the *Targeted Viral Marketing* (TVM) problem, in which they attempt to influence a subset of users in the network. Unfortunately, all of these methods rely on heuristics strategy and provide no performance guarantees.

On top of the challenges in solving the IM in huge networks, we often need to find seed sets for multiple sizes k to make informed choices regarding budget and cost-effectiveness. For example, a viral campaign marketing might go through multiple phases. The planning of the expenses for each phase cannot be done properly without knowing the influence for

multiple ranges of the number of seeds. Towards finding the optimal choices for multiple budgets, the authors in [70] optimize the size-influence ratio (the expected number of influenced individuals per seed node) or finding the min-seed set that can influence a large fraction of networks [76]. However, these approaches still give only one solution, that may not suite the multi-objective nature of decision making processes.

1.1.1.3 Tracing the Sources of Misinformation Cascades

The explosion of online social networks with billion of users such as Facebook or Twitter have fundamentally changed the landscapes of information sharing, nowadays. Unfortunately, the same channels can be exploited to spread rumors and misinformation that cause devastating effects such as widespread panic in the general public [1], diplomatic tensions [2], and witch hunts towards innocent people [3].

Given a snapshot of the network with a set V_I of *infected nodes* who posted the rumors, identifying the set of nodes who initially spread the rumors is a challenging, yet important question, whether for forensic use or insights to prevent future epidemics. Other applications of infection source detection can be found in finding first computing devices that get infected with a virus or source(s) of contamination in water networks.

Despite recent interest towards this problem, termed Infection Sources Identifications (ISI), most of existing works either limit to single source detection [78, 75] or simple network topologies, e.g. trees or grids, with ad hoc extensions to general graphs [67, 102, 101, 78]. A recent work in [97] provides an MDL-based method, called NETSLEUTH, to detect both the number of infection sources and the sources themselves. However the proposed heuristics seems to only work well for grid networks and cannot detect any true infection source. Thus there is lack of a rigorous and accurate method to detect multiple infection sources in general graphs.

1.1.2 Community Detection

Online social networks (OSNs) have become ubiquitous in everyday settings for decades. Many popular OSNs now have millions of users such as Twitter, Google+ or even billions of users as in the case of Facebook [4]. Despite their distinct natures, social networks exhibit several common topological properties, such as small-world [112], scale-free phenomenon [95] and the crucial feature known as *community structure* (CS) [82].

Communities can be defined intuitively as groups of nodes that are more densely connected to each other than to the rest of the network. For example, a community in Facebook may correspond to a group of users who share a common interest, such as cooking, fashion, music, etc. The goal of community detection, consequently, is to partition meaningfully networks into groups of nodes. Thus, it lends itself into a wealth of applications, such as forwarding and routing strategies in communication networks [30, 91]. Such structures give us insight into how the network function and topology affect each other. A large number of methods has been proposed for community detection (see [66] and the reference therein).

Despite the vast amount of work on the problem, even the state-of-the art methods perform poorly in recent benchmarks on real networks with known ground-truth communities [50]. This dissertation focuses on two directions to improve the accuracy: finding communities across multiple networks and combining network topology with nodes' attributes.

1.1.3 Security and Privacy: Targeted Cyber-attacks

As a double-edged sword, the massive explosion of Online Social Networks (OSNs) all over the world has brought both opportunities and deadly dangers. On one side, OSNs help increase social ties, e.g., bringing people closer no matter how distant they are, sharing emotion/sympathy with others, or create business opportunities, e.g., cheap/targeted

advertising. On the other side, they are irresistible places for the *intelligent attackers* who have patience and skills to visit for *target reconnaissance*. That is because people share/post on OSNs a wealth of valuable data including personal information, daily activities or even work processes which are extremely useful for the attackers. The leak of this information becomes severely devastating for companies/organizations when their employees' data reach to the bad hands who use these to financially attack the organization. Therefore, studying the attackers' methods to find countermeasures is utterly important.

There have been a large number of studies on the methods and preventions/detections of the attackers gathering target's information with their own weaknesses. Web crawling studied in [23, 15, 35, 16] is possibly the most traditional method with multiple variations, e.g., focused crawling [16], crawling relevant websites [35]. This class of methods can only collect public information on the organization's web pages which are usually well-inspected by the administrator and thus the crawled data are much less informative for attackers. Another typical method is crawling online social networks (OSNs) in [17, 64] which also admit the similar weakness as crawling websites. That is the attackers can only retrieve the public profile of the users due to privacy setting feature. The privacy in OSNs determines who can see your profile, e.g., only your mutual friends, everybody (public), and is configured by the account owner. Only recently, a stream of work on socialbots [33, 38, 96, 94] is emerged and able to crawl private information by friending the targets in OSNs. However, the bots exhibit abnormal behavior and easily get detected by network monitoring. Moreover, there was very little understanding of how effective the method is compared to the best possible one.

1.2 Contributions of the Dissertation

Under each category, we propose theoretically rigorous algorithms to find the solutions and run in large-scale real-world networks.

1.2.1 Information Diffusion

1.2.1.1 Influence Estimation (IE)

Outward Influence and High-confident Influence Estimation Algorithm:

We investigate efficient estimation methods for nodes' influence under stochastic cascade models [27, 55, 32]. First, we introduce a new influence measure, called *outward influence* and defined as $\mathbb{I}_{out}(S) = \mathbb{I}(S) - |S|$, where $\mathbb{I}(S)$ denotes the influence spread. The new measure excludes the self-influence artifact in influence spread, making it *more effective in comparing relative influence of nodes*. As shown in Fig. 1, the influence spread of the nodes are roughly the same, 1. In contrast, the outward influence of nodes u, v and w are $0.12, 0.20$, and 0.00, respectively. Those values correctly reflect the intuition that w is the least influential nodes and v is nearly twice as influential as u.

S	Influence $\mathbb{I}(S)$	Outward Influence $\mathbb{I}_{out}(S)$
$\{u\}$	$1 + p + 2p^2 = 1.12$	$p + 2p^2 = 0.12$
$\{v\}$	$1 + 2p = 1.20$	$2p = 0.20$
$\{w\}$	1.00	0.00

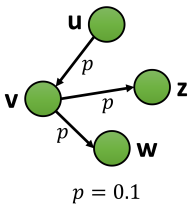


Fig. 1.: Left: the influence of nodes under IC model. The influence of all nodes are roughly the same, despite that w is much less influential than u and v. Right: Outward influence is better at reflecting the relative influence of the nodes. w has the least outward influence, 0, while v's is nearly twice as that of u.

More importantly, the outward influence measure inspires novel methods, termed SIEA/SOIEA, to estimate influence spread/outward influence *at scale and with rigorous theoretical guarantees*. Both SOIEA and SIEA guarantee *arbitrary small relative error with high probability* within an $O(n)$ observed influence. The proposed methods are built on two novel components 1) IICP an important sampling method for outward influence;

and 2) RSA, a high-confident mean estimation method that minimize the number of samples through analyzing variance and range of random variables. IICP focuses only on *non-trivial cascades* in which at least one node outside the seed set must be activated. As each IICP generates cascades of size at least two and outward influence of at least one, it leads to smaller variance and much faster convergence to the mean value. Under the well-known independent cascade model [55], SOIEA is $\Omega(\log^4 n)$ *times faster* than the state-of-the-art INFEST [77] in theory and is *four to five orders of magnitude faster* than both INFEST and the naive Monte-Carlo sampling. For other stochastic models, such as continuous-time diffusion model [32], LT model [55], SI, SIR, and variations [27], RSA can be applied directly to estimate the influence spread, given a Monte-Carlo sampling procedure, or, better, with an extension of IICP to the model.

Importance Sampling for Accurate Influence Estimation at Scale: We propose a new importance sketching technique, termed SKIS, that consists of *non-singular* reverse influence cascades, or simply non-singular cascades. Each non-singular cascade simulates the reverse diffusion process from a source node and must spread beyond the source to one or more nodes. Thus, our sketch, specifically, suppresses *singular* cascades that die prematurely at the source. Those singular cascades, consisting of 30%-80% portion in the previous sketches [25, 11], not only waste the memory space and processing time but also reduce estimation efficiency of the sketches. Consequently, SKIS contains samples of smaller variances providing estimations of high concentration with less memory and running time. Our new sketch also powers a new principle and scalable influence maximization class of methods, that inherits the algorithmic designs of existing algorithms on top of SKIS sketch. Particularly, SKIS-based IM methods are the only provably good and efficient enough that can scale to networks of billions of edges across different settings.

1.2.1.2 Influence Maximization (IM)

Optimal Sampling Algorithms for IM: To address the weaknesses of previous studies, we 1) unify the approaches in [11, 106, 105] to characterize the necessary number of RIS samples to achieve $(1 - 1/e - \epsilon)$-approximation guarantee; 2) design two novel sampling algorithms SSA and D-SSA aiming towards achieving minimum number of RIS samples. In the first part, we begin with defining classes of *RIS thresholds* on the sufficient numbers of RIS samples, generalizing θ thresholds in [106, 105]. The minimum threshold in each class is then termed *type-1 minimum threshold*, and the minimum among all type-1 minimum thresholds is termed *type-2 minimum threshold*.

In the second part, we develop the *Stop-and-Stare Algorithm* (SSA) and its dynamic version D-SSA that guarantee to achieve, within constant factors, the two minimum thresholds, respectively. In short, the algorithms keep generating samples and *stop* at *exponential check points* to verify (*stare*) if there is adequate statistical evidence on the solution quality for termination. This strategy will be shown to address both of the shortcomings in [106, 105]: 1) guarantee to be close to the theoretical thresholds and 2) the thresholds are minimal by definitions. The dynamic algorithm, D-SSA, improves over SSA by automatically and dynamically selecting the best parameters for the RIS framework. We note that the Stop-and-Stare strategy enables SSA and D-SSA to meet the minimum thresholds without explicitly computing/looking for these thresholds.

Our experiments show that both SSA and D-SSA outperform the best existing methods up to several orders of magnitudes w.r.t running time while returning comparable seed set quality. More specifically, on Friendster network with roughly 65.6 million nodes and 1.8 billion edges, SSA and D-SSA, taking 3.5 seconds when $k = 500$, are up to 1200 times faster than IMM. We also run CELF++ (the fastest greedy algorithm for IM with guarantees) on Twitter network with $k = 1000$ and observe that D-SSA is $2 \cdot 10^9$ times

faster.

Extension to Cost-aware Targeted Viral Marketing: We introduce the *Cost-aware Targeted Viral Marketing* (CTVM) problem which takes into account both arbitrary cost for selecting a node and arbitrary benefit for influencing a node. Given a social network abstracted by a graph $G = (V, E)$, each node u represents a user with a *cost* $c(u)$ to select into the seed set and a *benefit* $b(u)$ obtained when u is influenced. Given a budget B, the goal is to find a seed set S with total cost at most B that maximizes the expected total benefit over the influenced nodes. CTVM is more relevant in practice as it generalizes other viral marketing problems including TVM, BIM and the fundamental IM. However, the problem is much more challenging with heterogeneous costs and benefits. As we show in Section 3, extending the state-of-the-art method for IM in [106] may increase the running time by a factor $|V|$, making the method unbearable for large networks.

We develop BCT, an efficient approximation algorithm for CTVM for billion-scale networks. Given arbitrarily small $\epsilon > 0$, our algorithm guarantees a $(1 - 1/\sqrt{e} - \epsilon)$-approximate solution in general case and a $(1 - 1/e - \epsilon)$-approximate solution when nodes have uniform costs. BCT also dramatically outperforms the existing state-of-the-art methods for IM, e.i., IMM, TIM/TIM+, when nodes have uniform costs and benefits. In particular, BCT only takes several minutes to process a network with 41.7 million nodes and 1.5 billion edges.

Extension to Influence Spectrum: We propose the computation of *Influence Spectrum* (IS), the maximum influences (and the corresponding seed sets) for all possible seed sizes from $k = k_{lower}$ up to k_{upper}. The influence spectrum gives better insights for decision making and resource planning in viral marketing campaigns. Given the influence spectrum, we can find the solutions for not only IM but also cost-effective seed set [70] and min-seed set selection [76] problems (with the best approximation guarantees). As useful as it is, no one has ever considered computing IS due to the perception that it seems extremely com-

putationally expensive. The fact is computing IS implies solving of $(k_{upper} - k_{lower} + 1)$ IM instances with seed size as large as n. Unfortunately, existing IM methods either do not scale well with large seed sets [55, 70, 47, 46] or resort to heuristics [19], i.e., obtained results that could be arbitrarily worse than the optimal ones.

One might look into adapting some greedy methods for IM for the task, e.g., solving IM with $k = k_{upper}$ and output the solutions for all intermediate values of $k = k_{lower}, \ldots, k_{upper}$. However, the original greedy approaches [55, 70] has a prohibitive cost and provide little or no guarantees on each individual seed size. Direct usage of the state-of-the-art methods for IM in [106, 105] for each seed size IS also results in an unbearable running time for large ranges of $k_{lower} \leq k \leq k_{upper}$. Not to mention that the extension only guarantees approximation quality for each seed set individually in contrast to the whole range of seed set sizes.

We introduce LISA, an efficient approximation algorithm to compute IS in billion-size networks. Given arbitrarily small $\epsilon, \delta > 0$, our algorithm has an expected running time $O((m+n)(k^* \log(n) + \log(k_{upper} - k_{lower} + 1))\epsilon^{-2})$[1] and output $(1 - \frac{1}{e} - \epsilon)$-approximate IS with probability of $(1 - \delta)$. Also, LISA requires only an additional $O(n)$ space. The proposed algorithm has the best theoretical guarantees and outperforms the state-of-the-art methods for IM in practice. In particular, when $\epsilon = 0.1$ and $\delta = 1/n$, it takes about 15 minutes on a network with 41.7 million nodes and 1.5 billion edges under the LT model. In comparison, it is up to 100 times faster than IMM [105], the fastest known method with approximation guarantee for IM, when $k = 1000$ and is several magnitudes of order faster than TIM and TIM+ for larger k while providing similar solution quality.

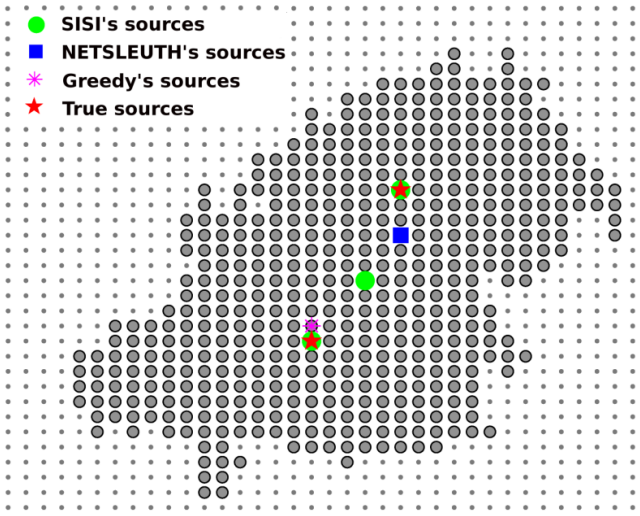


Fig. 2.: Infection sources detection on a 60×60 grid graph

1.2.1.3 Tracing the Sources of Misinformation Cascades

We present a new approach to identify *multiple* infection sources that looks into both infected and uninfected nodes. This contradicts to existing methods [67, 97] which limit the attention to the subgraph induced by the infected nodes. Given a snapshot of network $G = (V, E)$ and a set of infected nodes V_I, we identify the sources by searching for a set $\hat{S}$ that minimize the *symmetric difference* between the cascade from S and V_I. While our objective, the symmetric difference, is similar to the one used in k-effector [67], our novel formulation does not require the knowledge of the number of infection sources k. In deriving optimization method for this new approach, we face *strong challenges* in developing efficient solution:

- The *exponential* number, up to $2^{\theta(n)}$ for large V_I, of possible solutions. This makes the exhaustive search for the case of single source [78, 75, 36] intractable.
- The *non-submodular* objective. Thus, it is inefficient to solve the problem through simple greedy methods.

[1] k^* is the seed size that results in the longest running time among $k_{lower} \leq k \leq k_{upper}$

- The *stochastic* nature of the infection process requires considering exponentially many possible cascades.

To tackle ISI, we propose SISI, an algorithm that can accurately detect infection sources. We employ in SISI two novel techniques: a *Truncated Reverse Infection Sampling* (TRIS) method to generate random reachability RR sets that encode the infection landscape and a primal-dual algorithm for the *Submodular-cost Covering* [61]. SISI, to our best knowledge, is the *first algorithm with provable guarantee* for multiple infection sources detection in general graphs. It returns an $\frac{2}{(1-\epsilon)^2}\Delta$-approximate solution with a high probability, where Δ denotes the maximum number of nodes in V_I that may infect the same node in the network. Experiments on real-world networks show the huge leap of SISI in detecting true infection sources, boosting the true source discovery rates from merely few percents, for the state-of-the-art NETSLEUTH, to more than 70%. Thus SISI has both high empirical performance and theoretical guarantees.

The advantages of SISI over other methods are illustrated through a cascade on a 60×60 grid in Fig. 2. SISI is the only one which can *detect the true infection sources*. To avoid false negative, which is more serious than false positive, SISI often output slightly more infection sources than other methods (SISI: 3, NETSLEUTH: 1, Greedy:1, Ground-truth: 2). However, it maintains a reasonable F1-score of over 50%.

1.2.2 Community Detection

1.2.2.1 Community Detection in Multiplex Social Networks

In multiplex social networks, the participant of users across multiple networks requires us to analyze all the networks (also referred to as the layers) simultaneously. The connections in a layer may reveal latent relationship in other layers and, therefore, provide additional information to unveil the underlying structure of those networks.

Community detection in multiplex OSNs exposes several challenges. First, multiplex OSNs are often heterogeneous, i.e., they can be directed vs. undirected, weighted vs. unweighted or have different degree densities. Moreover, the diverse topological wirings of networks make the problem very complicated. Despite a large amount of research on CS detection, CS in multiplex OSNs remains unaddressed at large. The closest works are the ones on CS detection in *multi-relational* networks [73], however, these methods cannot be applied directly for multiplex social networks. The reason for that lies in an unique feature that multiplex OSNs has only one entity type, i.e., user, each entity is present in several layers and existing approaches ignore this important phenomenon.

We proposed and compare two classes of approaches. The first class, named unifying approach, finds a consistent CS in the networks by aggregating multiple accounts of the same users. The second class finds *mostly* consistent CSs in the network using coupling techniques. We also develop specialized NMF-based method for each class.

1.2.2.2 Community Detection in Multi-attributed Networks

In many cases, the formation of group connections is strongly influenced by users' attributes/characters such as geographical location, occupations, gender, and so on. Moreover, there is a lack of efficient techniques to cope with overlapping communities, which occur abundantly in real networks.

Thus, it is essential to design new methods that combine information from both network topology and node attributes to uncover overlapping communities with higher order of accuracy. Since topology and nodal attributes are two different aspects of data, they can complement each other in finding more suitable community structure. For example, attributes might tell us to which community a node with very few links belongs to. Conversely, two nodes having no common attributes but still belong to the same community due to their strong topological similarity. The most direct work, in this direction, is from

[114] where they obtain the best results over all previous methods. However, the proposed method suffers heavily from an over-complex model, thus, scale poorly and has the potential of overfitting.

We propose a new generative model that describes the formation of topological edges and attribute values in relation with CS. We, consequently, define finding overlapping communities as an optimization problem using NMF framework and develop 3NCD algorithm. We prove the convergence and provide efficient update procedure in order to speed up the computational process by a factor of n compared to the straightforward implementation of the update rules.

1.2.3 Security and Privacy: Targeted Cyber-attacks

We study the problem of adaptive targeted crawling in which an intelligent attacker desires to gain benefit of users from a targeted organization. The attacker not only maximizes the crawling performance but also avoids being detected by mimicking the normal behavior. That is, he approaches the target step by step: at each step, he sends a friend request to one user and waits for the response. After receiving the response (accepted or rejected indicated by no reply some period of time), he will select the next user to friend. To maximize the crawled information, the attacker needs to specify which node to send request given the results of all the previous requests. The node specification strategy is called a *policy* π.

We model the online social network, where there is a set of users from the targeted organization, as a stochastic graph of nodes and weighted edges, i.e., nodes correspond to users and edge weight reflects the probability of two users being connected. Here the edge weights can be learned by link prediction [40, 39] from the public information. The attacker is also a user in the network and has a probability of successfully friending each node in the network. These probabilities can also be learned from the similarities between the attacker

and public information of the users. Based on privacy setting, we quantitatively define two types of benefit: 1) information benefit obtained from a user if that user is a friend of attacker's friends and 2) friending benefit obtained when a user accepts the attacker's friend request.

In our model, we define the crawling problem as an adaptive targeted maximization problem that we later prove to be NP-hard. Thus, the problem is unsolvable in polynomial time unless P=NP. Based on the recent advances in maximizing adaptive submodular functions, we propose an approximation greedy policy π that is at least $(1 - 1/e)$ as good as the optimal policy π^*. The superiority of the proposed greedy policy is demonstrated in our simulations compared with several naive node-ranking policy.

1.3 Organization of the Dissertation

In the following, each chapter will present in detail our proposed solutions for each of the studied problems. Specifically, Chapter 2 introduces a high-confident influence estimation and importance sketching technique that provide better accuracy and scalability than existing algorithms. Chapter 3 presents our optimal sampling algorithms for IM and their extensions to various real-world scenarios. We propose an approximation algorithm for infection sources identification in Chapter 4. Chapters 5 gives details on our community detections algorithms on multiplex and multi-attributed networks. Finally, our analysis of targeted attack reconnaissance on social networks is demonstrated in Chapter 6.

CHAPTER 2

INFLUENCE ESTIMATION

We propose high-confident, accurate and scalable algorithms for estimating the influences of sets of nodes in a network. We investigate sampling algorithms that incorporate the traditional Monte-Carlo estimation with our proposed state-of-the-art importance sampling technique to maximize the utility of a sample intrinsically and algorithmically.

2.1 Basic Concepts and Definitions

2.1.1 Probabilistic Graph

We abstract a network using a probabilistic graph (weighted graph) $G = (V, E, w)$ with $|V| = n$ nodes and $|E| = m$ directed edges. For example, in a social network, V and E correspond to the set of users and their social relationships, respectively. Each edge $(u, v) \in E$ is associated with a weight $w(u, v) \in [0, 1]$ which indicates the probability that u influences v.

2.1.2 Diffusion Models

Let's consider a graph $\mathcal{G} = (V, E, w)$. Assume that there is a cascade starting from a subset of nodes $S \subseteq V$, called *seed set*. How the cascade progress is described by a diffusion model (aka cascade model) $\mathcal{M}$ that dictates how nodes gets activated/influenced. In a stochastic diffusion model, the cascade is dictated by a random vector θ in a sample space Ω_θ. Describing the diffusion model is then equivalent to specifying the distribution P of θ.

Let $r_S(\theta)$ be the size of the cascade, the number of activated nodes in the end. The

influence spread of S, denoted by $\mathbb{I}(S)$, under diffusion model $\mathcal{M}$ is the expected size of the cascade, i.e.,

$$\mathbb{I}(S) = \begin{cases} \sum_{\theta \in \Omega_\theta} r_\theta(S) \Pr[\theta] & \text{for discrete } \Omega_\theta, \\ \int_{\theta \in \Omega_\theta} r_\theta(S) dP(\theta) & \text{for continuous } \Omega_\theta \end{cases} \tag{2.1}$$

For example, we describe below the unknown vector θ and their distribution for the most popular diffusion models.

- Information diffusion models, e.g. Independent Cascade (IC), Linear Threshold (LT), the general triggering model [55]: $\theta \in \{0,1\}^{|E|}$, and $\forall (u,v) \in E, \theta_{(u,v)}$ is a Bernouli random variable that indicates whether u activates/influences v. That is for given $w(u,v) \in (0,1)$, $\theta(u,v) = 1$ if u activates v with a probability $w(u,v)$ and 0, otherwise.

- Epidemic cascading models, e.g., Susceptible-Infected (SI) [27, 86] and its variations: $\theta \in \mathbb{N}^{|E|}$, and $\forall (u,v) \in E, \theta_{(u,v)}$ is a random variable following a geometric distribution. $\theta_{(u,v)}$ indicates how long it takes u to activates v after u is activated.

- Continuous-time models [32]: $\theta \in \mathbb{R}^{|E|}$, and $\theta_{(u,v)}$ is a continuous random variable with density function $\pi_{u,v}(t)$. $\theta_{(u,v)}$ also indicates the transmission times (time until u activates v) like that in the SI model, however, the transmissions time on different edges follow different distributions.

2.1.3 Monotone Submodular Functions

Given a finite set Ω, a submodular function is a set function $f : 2^\Omega \to \mathbb{R}$, where 2^Ω denotes the power set of Ω, which satisfies one of the following properties,

- For every $X, Y \subseteq \Omega$ with $X \subseteq Y$ and every $x \in \Omega \backslash X$, we have that,

$$f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y) \tag{2.2}$$

- For every $X, Y \subseteq \Omega$, we have that,

$$f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y) \tag{2.3}$$

The set function $f : 2^{\Omega} \rightarrow \mathbb{R}$ is called monotone if for every $X, Y \subseteq \Omega$ and $X \subseteq Y$, we have that,

$$f(X) \leq f(Y) \tag{2.4}$$

The set function $f : 2^{\Omega} \rightarrow \mathbb{R}$ is monotone submodular if it is simultaneously monotone and submodular.

2.2 High-confident Influence Estimation

Summary of contributions:

- We introduce a new influence measure, called *Outward Influence* which is more effective in differentiating nodes' influence. We investigate the characteristics of this new measure including non-monotonicity, submodularity, and #P-hardness of computation.

- Two fully polynomial time randomized approximation schemes (FPRAS) SIEA and SOIEA to provide (ϵ, δ)-approximate for influence spread and outward influence with only an $O(n)$ observed influence in total. Particularly, SOIEA, our algorithm to estimate influence spread, is $\Omega(\log^4 n)$ *times faster* than the state-of-the-art INFEST [77] in theory and is *four to five orders of magnitude faster* than both INFEST and the naive Monte-Carlo sampling.

- The high-confident mean estimation algorithm, termed RSA, a building block of SIEA, can be used to estimate influence spread under *other stochastic diffusion models*, or, in general, mean of bounded random variables of unknown distribution. RSA will be our favorite statistical algorithm moving forwards.

- We perform comprehensive experiments on both real-world and synthesis networks with size up to 65 million nodes and *1.8 billion edges*. Our experiments indicate the superior of our algorithms in terms of both accuracy and running time in comparison to the naive Monte-Carlo and the state-of-the-art methods. The results also give *evidence against the practice of using a fixed number of samples* to estimate the cascade size. For example, using 10000 samples to estimate the influence will deviate up to 240% from the ground truth in a Twitter subnetwork. In contrast, our algorithm can provide (pseudo) *ground truth* with guaranteed small (relative) error (e.g. 0.5%). Thus it is a more concrete benchmark tool for research on network cascades.

2.2.1 Definitions and Properties

Outward Influence. We introduce the notion of Outward Influence which captures the influence of a subset of nodes towards the rest of the network. Outward influence excludes the self-influence of the seed nodes from the measure.

Definition 1 (Outward Influence)**.** *Given a graph $\mathcal{G} = (V, E)$, a set $S \subseteq V$ and a diffusion model $\mathcal{M}$, the Outward Influence of S, denoted by $\mathbb{I}_{out}(S)$, is*

$$\mathbb{I}_{out}(S) = \mathbb{I}(S) - |S| \tag{2.5}$$

Thus, influence and outward influence of a seed set S differ exactly by the number of nodes in S.

Influence Spread/Outward Influence Estimations. A fundemental task in network

science is to estimate the influence of a given seed set S. Since the exact computation is #P-hard (Subsection 2.2), we aim for estimation with bounded error.

Definition 2 (Influence Spread Estimation)**.** *Given a graph $\mathcal{G}$ and a set $S \subseteq V$, the problem asks for an (ϵ, δ)-estimate $\hat{\mathbb{I}}(S)$ of influence spread $\mathbb{I}(S)$, i.e.,*

$$\Pr[(1-\epsilon)\mathbb{I}(S) \leq \hat{\mathbb{I}}(S) \leq (1+\epsilon)\mathbb{I}(S)] \geq 1-\delta. \tag{2.6}$$

The outward influence estimation problem is stated similarly:

Definition 3 (Outward Influence Estimation)**.** *Given a graph $\mathcal{G}$ and a set $S \subseteq V$, the problem asks for an (ϵ, δ)-estimate $\hat{\mathbb{I}}_{out}(S)$ of influence spread $\mathbb{I}_{out}(S)$, i.e.,*

$$\Pr[(1-\epsilon)\mathbb{I}_{out}(S) \leq \hat{\mathbb{I}}_{out}(S) \leq (1+\epsilon)\mathbb{I}_{out}(S)] \geq 1-\delta. \tag{2.7}$$

A common approach for estimation is through generating independent Monte-Carlo samples and taking the average. However, one faces two major challenges:

- How to achieve a minimum number samples to get an (ϵ, δ)-approximate?
- How to effectively generate samples with small variance, and, thus, reduce the number of samples?

For simplicity, we focus on the well-known *Independent Cascade* (IC) model and provide the extension of our approaches to other cascade models in Subsection 3.2.4.5.

2.2.1.1 Graph Samples and Probabilistic Space

Given a probabilistic graph $\mathcal{G} = (V, E)$ in which each edge $(u, v) \in E$ is associated with a number $w(u, v) \in (0, 1)$. $w(u, v)$ indicates the probability that node u will successfully activate v once u is activated. In practice, the probability $w(u, v)$ can be mined from interaction frequency [55, 106] or learned from action logs [43].

Cascading Process. The cascade starts from a subset of nodes $S \subseteq V$, called seed set. The cascade happens in discrete rounds $t = 0, 1, ...|V|$. At round 0, only nodes in S are active and the others are inactive. When a node u becomes active, it has a single chance to activate (aka influence) each neighbor v of u with probability $w(u, v)$. An active node remains active till the end of the cascade process. It stops when no more nodes get activated.

Sample Graph. Associate with each edge $(u, v) \in E$ a biased coin that lands heads with probability $w(u, v)$ and tails with probability $1 - w(u, v)$. Deciding the outcome when u attempts to activate v is then equivalent to the outcome of flipping the coin. If the coin landed heads, the activation attemp succeeds and we call (u, v) a *live-edge*. Since all the activation on the edges are independent in the IC model, it does not matter when we flip the coin. That is we can flip all the coins associated with the edges (u, v) at the same time instead of waiting until node u becomes active. We call the graph g that contains the nodes V and all the live-edges a *sample graph* of $\mathcal{G}$.

Note that the model parameter θ for the IC is a random vector indicating the states of the edges, i.e. *live-edge* or not. In other words, Ω_θ corresponds to the space of all possible sample graphs of $\mathcal{G}$, denoted by $\Omega_\mathcal{G}$.

Probabilistic Space. The graph $\mathcal{G}$ can be seen as a generative model. The set of all sample graphs generated from $\mathcal{G}$ together with their probabilities define a probabilistic space $\Omega_\mathcal{G}$. Recall that each sample graph $g \in \Omega_\mathcal{G}$ can be generated by flipping coins on all the edges to determine whether or not the edge is live or appears in g. Each edge (u, v) will be present in the a sample graph with probability $w(u, v)$. Thus, the probability that a sample graph $g = (V, E' \subseteq E)$ is generated from $\mathcal{G}$ is

$$\Pr[g \sim \mathcal{G}] = \prod_{(u,v)\in E'} w(u, v) \prod_{(u,v)\in E\setminus E'} (1 - w(u, v)). \tag{2.8}$$

Influence Spread and Outward Influence. In a sample graph $g \in \Omega_{\mathcal{G}}$, let $r_g(S)$ be the set of nodes reachable from S. The *influence spread* in Eq. 2.1 is rewritten,

$$\mathbb{I}(S) = \sum_{g \in \Omega_{\mathcal{G}}} |r_g(S)| \Pr[g \sim \mathcal{G}], \tag{2.9}$$

and the outward influence is defined accordingly to Eq. 2.5,

$$\mathbb{I}_{out}(S) = \mathbb{I}(S) - |S| \tag{2.10}$$

2.2.1.2 Outward Influence under the IC model

We show the properties of outward influence under the IC model.

Better Influence Discrepancy. As illustrated through Fig. 1, the elimination of the nominal constant $|S|$ helps to differentiate the "actual influence" of the seed nodes to the other nodes in the network. In the extreme case when $p = o(1)$, the ratio between the influence spread of u and v is $\frac{1+p+2p^2}{1+p+2p} \approx 1$, suggesting u and v have the same influence. However, outward influence can capture the fact that v can influence roughly twice the number of nodes than u, since s $\frac{\mathbb{I}_{out}(u)}{\mathbb{I}_{out}(v)} = \frac{p+2p^2}{2p} \approx 1/2$.

Non-monotonicity. Outward influence as a function of seed set S is non-monotone. This is different from the influence spread. In Figure 1, $\mathbb{I}_{out}(\{u\}) = 0.12 < \mathbb{I}_{out}(\{u, v\}) = 0.2$, however, $\mathbb{I}_{out}(\{u\}) = 0.12 > \mathbb{I}_{out}(\{u, w\}) = 0.11$. That is adding nodes to the seed set may increase or decrease the outward influence.

Submodularity. A submodular function expresses the diminishing returns behavior of set functions and are suitable for many applications, including approximation algorithms and machine learning. If Ω is a finite set, a submodular function is a set function $f : 2^{\Omega} \leftarrow \mathbb{R}$, where 2^{Ω} denotes the power set of Ω, which satisfies that for every $X, Y \subseteq \Omega$ with

$X \subseteq Y$ and every $x \in \Omega \setminus Y$, we have,

$$f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y). \tag{2.11}$$

Similar to influence spread, outward influence, as a function of the seed set S, is also submodular.

Lemma 1. *Given a network $G = (V, E, w)$, the outward influence function $\mathbb{I}_{out}(S)$ for $S \in 2^{|V|}$, is a submodular function.*

2.2.1.3 Hardness of Computation

If we can compute outward influence of S, the influence spread of S can be obtained by adding $|S|$ to it. Since computing influence spread is #P-hard [19], it is no surprise that computing outward influence is also #P-hard.

Lemma 2. *Given a probabilistic graph $G = (V, E, w)$ and a seed set $S \subseteq V$, it is #P-hard to compute $\mathbb{I}_{out}(S)$.*

However, while influence spread is lower-bounded by one, the outward influence of any set S can be arbitrarily small (or even zero). Take an example in Figure 1, node u has influence of $\mathbb{I}(\{u\}) = 1 + p + 2p^2 \geq 1$ for any value of p. However, u's outward influence $\mathbb{I}_{out}(\{u\}) = p + 2p^2$ can be exponentially small if $p = \frac{1}{2^n}$. This makes estimating outward influence challenging, as the number of samples needed to estimate the mean of random variables is inversely proportional to the mean.

Monte-Carlo estimation. A typical approach to obtain an (ϵ, δ)-approximaion of a random variable is through Monte-Carlo estimation: taking the average over many samples of that random variable. Through the Bernstein's inequality [26], we have the lemma:

Lemma 3. *Given a set $X_1, X_2, \ldots$ of i.i.d. random variables having a common mean μ_X, there exists a Monte-Carlo estimation which gives an (ϵ, δ)-approximate of the mean*

μ_X and uses $T = O(\frac{1}{\epsilon^2}\ln(\frac{2}{\delta})\frac{b}{\mu_X})$ random variables where b is an upper-bound of X_i, i.e. $X_i \leq b$.

To estimate the influence spread $\mathbb{I}(S)$, existing work often simulates the cascade process using a BFS-like procedure and takes the average of the cascades' sizes as the influence spread. The number of samples needed to obtain an (ϵ, δ)-approximation is $O(\frac{1}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\frac{n}{\mathbb{I}(S)})$ samples. Since $\mathbb{I}(S) \geq 1$, in the worst-case, we need only a polynomial number of samples, $O(\frac{1}{\epsilon^2}\log\left(\frac{1}{\delta}\right)n)$.

Unfortunately, the same argument does not apply for the case of $\mathbb{I}_{out}(S)$, since $\mathbb{I}_{out}(S)$ can be arbitrarily close to zero. For the same reason, the recent advances in influence estimation in [11, 77] cannot be adapted to obtain a polynomial-time algorithm to compute an (ϵ, δ)-approximation (aka FPRAS) for outward influence. We shall address this challenging task in the next section.

We summarize the frequently used notations in Table 24.

Table 1.: Table of notations

Notations	**Descriptions**
n, m	#nodes, #edges of graph $\mathcal{G} = (V, E, w)$.
$\mathbb{I}(S)$	Influence Spread of seed set $S \subseteq V$.
$\mathbb{I}_{out}(S)$	Outward Influence of seed set $S \subseteq V$.
$N^{out}(u)$	The set of out-neighbors of u: $N^{out}(u) = \{v \in V \vert (u, v) \in E\}$.
N_S^{out}	$N_S^{out} = \bigcup_{u \in S} N^{out}(u) \backslash S$.
A_i	The event that v_i is active and $v_1, \ldots, v_{i-1}$ are not active after round 1.
β_0	$\beta_0 = \sum_{i=1}^{l} \Pr[A_i] = 1 - \Pr[A_{l+1}]$.
$c(\epsilon, \delta)$	$c(\epsilon, \delta) = (2 + \frac{2}{3}\epsilon)\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}$.
ϵ'	$\epsilon' = \epsilon\Big(1 - \frac{\epsilon b}{(2+\frac{2}{3}\epsilon)\ln(\frac{2}{\delta})(b-a)}\Big) \approx \epsilon(1 - O(\frac{1}{\ln n}))$ for $\delta = \frac{1}{n}$.
Υ	$\Upsilon = (1 + \epsilon)c(\epsilon', \delta)(b - a)$.

2.2.2 Outward Influence Estimation via Importance Sampling

We propose a Fully Polynomial Randomized Approximation Scheme (FPRAS) to estimate the outward influence of a given set S. Given two precision parameters $\epsilon, \delta \in (0, 1)$, our FPRAS algorithm guarantees to return an (ϵ, δ)-approximate $\hat{\mathbb{I}}_{out}(S)$ of the outward influence $\mathbb{I}_{out}(S)$,

$$\Pr[(1-\epsilon)\mathbb{I}_{out}(S) \leq \hat{\mathbb{I}}_{out}(S) \leq (1+\epsilon)\mathbb{I}_{out}(S)] \geq 1-\delta. \tag{2.12}$$

General idea. Our starting point is an observation that the cascade triggered by the seed set with small influence spread often stops right at round 0. The probability of such cascades, termed *trivial cascades*, can be computed exactly. Thus if we can sample only the *non-trivial cascades*, we will obtain a better sampling method to estimate the outward influence. The reason is that the "outward influence" associated with non-trivial cascade is also lower-bounded by one. Thus, we again can apply the argument in the previous section on the polynomial number of samples.

Given a graph $\mathcal{G}$ and a seed set S, we introduce our *importance sampling* strategy to generate such non-trivial cascades. It consists of two stages:

1. Guarantee that at least one neighbor of S will be activated through a biased selection towards the cascades with at least one node outside of S and,

2. Continue to simulate the cascade using the standard procedure following the diffusion model.

This importance sampling strategy is general for different diffusion models. In the following, we illustrate our importance sampling under the focused IC model.

2.2.2.1 Importance IC Polling

We propose *Importance IC Polling* (IICP) to sample non-trivial cascades in Algorithm 1.

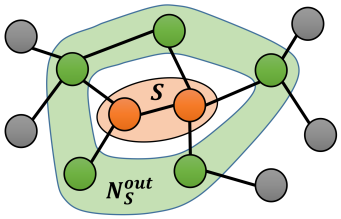


Fig. 3.: Neighbors of nodes in S

First, we "merge" all the nodes in S and define a "unified neighborhood" of S. Specifically, let $N^{out}(u) = \{v|(u,v) \in E\}$ the set of out-neighbors of u and $N_S^{out} = \bigcup_{u \in S} N_u^{out} \backslash S$ the set of out-neighbors of S *excluding* S. For each $v \in N_S^{out}$,

$$P_{S,v} = 1 - \prod_{u \in S}(1 - w(u,v)), \tag{2.13}$$

the probability that v is activated directly by one (or more) node(s) in S. Without loss of generality, assume that $P_{S,v} < 1$ (otherwise, we simply add v into S).

Assume an order on the neighborhood of S, that is

$$N_S^{out} = \{v_1, v_2, \ldots, v_l\},$$

where $l = |N_S^{out}|$. For each $i = 1..l$, let A_i be the event that v_i be the "first" node that gets activated directly by S:

$$A_i = \{v_1, \ldots, v_{i-1} \text{ are not active and } v_i \text{ is active after round } 1\}.$$

The probability of A_i is

$$\Pr[A_i] = P_{S,v_i} \prod_{j=1}^{i-1}(1 - P_{S,v_j}). \tag{2.14}$$

For consistency, we also denote A_{l+1} the event that none of the neighbors are activated, i.e.,

$$\Pr[A_{l+1}] = 1 - \sum_{i=1}^{l} \Pr[A_i]. \tag{2.15}$$

Note that A_{l+1} is also the event that the cascade stops right at round 0. Such a cascade is termed a *trivial cascade*. As we can compute exactly the probability of trivial cascades, we do not need to sample those cascades but focus only on the non-trivial ones.

Denote by β_0 the probability of having at least one nodes among $v_1, \ldots, v_l$ activated by S, i.e.,

$$\beta_0 = \sum_{i=1}^{l} \Pr[A_i] = 1 - \Pr[A_{l+1}]. \tag{2.16}$$

We now explain the details in the Importance IC Polling Algorithm (IICP), summarized in Alg. 1. The algorithm outputs the size of the cascade minus the seed set size. We term the output of IICP the *outer size* of the cascade. The algorithm consists of two stages.

Stage 1. By definition, the events $A_i, A_2, ..., A_l, A_{l+1}$ are disjoint and form a partition of the sample space. To generate a non-trivial cascade, we first select in the first round $v_i, i = 1, \ldots, l$ with a probability $\frac{\Pr[A_i]}{\beta_0}, i = 1, \ldots, l$ (excluding A_{l+1}). This will guarantee that at least one of the neighbors of S will be activated. Let v_i be the selected node, after the first round v_i becomes active and $v_1, \ldots, v_{i-1}$ remains inactive. The nodes v_j among $v_{i+1}, \ldots, v_l$ are then activated independently with probability P_{S,v_j} (Eq. 2.13).

Stage 2. After the first stage of sampling neighbors of S, we get a non-trivial set of nodes directly influenced from S. For each of those nodes and later influenced nodes, we will sample a set of its neighbors by the naive BFS-like IC polling scheme [55]. Assume sampling neighbors of a newly influenced node u, each neighbor $v_j \in N^{out}(u)$ is influenced by u with probability $w(u, v_j)$. The neighbors of those influenced nodes are next to be sampled in the same fashion.

Algorithm 1: IICP - Importance IC Polling

Input: A graph $\mathcal{G} = (V, E, w)$ and a seed set S

Output: $Y^{(S)}$ - size of a random outward cascade from S

Stage 1 `// Sample non-trivial neighbors of set S`

Precompute $\Pr[A_i], i = 1, \ldots, l+1$ using Eq. 2.14 and Eq. 2.15

Select one neighbor v_i among $v_1, \ldots, v_l$ with probability of selecting v_i being $\frac{\Pr[A_i]}{\beta_0}$

Queue $R \leftarrow \{v_i\}$; $Y^{(S)} = 1$; Mark v_i and all nodes in S visited

for $j = i + 1 : l$ **do**

With a probability P_{S,v_j} do

Add v_j into R; $Y^{(S)} \leftarrow Y^{(S)} + 1$; Mark v_j visited.

end

Stage 2 `// Sample from newly influenced nodes`

while R *is non-empty* **do**

$u \leftarrow R$.pop()

foreach *unvisited neighbor* v *of* u **do**

With a probability $w(u, v)$

Add v to R; $Y^{(S)} \leftarrow Y^{(S)} + 1$; Mark v visited.

end

end

return $Y^{(S)}$;

In addition, we keep track of the newly influenced nodes using a queue R and the number of active nodes outside S using $Y^{(S)}$.

The following lemma shows how to estimate the (expected) cascade size through the (expected) outer size of non-trivial cascades.

Lemma 4. *Given a seed set $S \subseteq V$, let $Y^{(S)}$ be the random variable associated with the output of the IICP algorithm. The following properties hold,*

- $1 \leq Y^{(S)} \leq n - |S|$,
- $\mathbb{I}_{out}(S) = \mathcal{E}[Y^{(S)}] \cdot \beta_0$.

Further, let Ω_W be the probability space of non-trivial cascades and Ω_Y the probability space for the outer size of non-trivial cascades, i.e, $Y^{(S)}$. The probability of $Y^{(S)} \in [1, n - |S|]$ is given by,

$$\Pr[Y^{(S)} \in \Omega_Y] = \sum_{W^{(S)} \in \Omega_W, |W^{(S)}| = Y^{(S)}} \Pr[W^{(S)} \in \Omega_W].$$

2.2.2.2 FPRAS for Outward Influence Estimation

From Lemma 4, we can obtain an estimate $\hat{\mathbb{I}}_{out}(S)$ of $\mathbb{I}_{out}(S)$ through getting an estimate $\hat{\mathcal{E}}[Y^{(S)}]$ of $\mathcal{E}[Y^{(S)}]$ by,

$$\begin{aligned}
&\Pr\left[(1-\epsilon)\mathcal{E}[Y^{(S)}] \leq \hat{\mathcal{E}}[Y^{(S)}] \leq (1+\epsilon)\mathcal{E}[Y^{(S)}]\right] \\
&= \Pr\left[(1-\epsilon)\mathcal{E}[Y^{(S)}]\beta_0 \leq \hat{\mathcal{E}}[Y^{(S)}]\beta_0 \leq (1+\epsilon)\mathcal{E}[Y^{(S)}]\beta_0\right] \\
&= \Pr\left[(1-\epsilon)\mathbb{I}_{out}(S) \leq \hat{\mathbb{I}}_{out}(S) \leq (1+\epsilon)\mathbb{I}_{out}(S)\right], \qquad (2.17)
\end{aligned}$$

where the estimate $\hat{\mathbb{I}}_{out}(S) = \hat{\mathcal{E}}[Y^{(S)}] \cdot \beta_0$. Thus, finding an (ϵ, δ)-approximation of $\mathbb{I}_{out}(S)$ is then equivalent to finding an (ϵ, δ)-approximate $\hat{\mathcal{E}}[Y^{(S)}]$ of $\mathcal{E}[Y^{(S)}]$.

The advantage of this approach is that estimating $\mathcal{E}[Y^{(S)}]$, in which the random variable $Y^{(S)}$ has value of at least 1, requires only a polynomial number of samples. Here the same argument on the number of samples to estimate influence spread in subsection 2.2.1.3 can be applied. Let $Y_1^{(S)}, Y_2^{(S)}, \ldots$ be the random variables denoting the output of IICP. We can apply Lemma 3 on the set of random variables $Y_1^{(S)}, Y_2^{(S)}, \ldots$ satisfying $1 \leq Y_i^{(S)} \leq |V| - |S|$. Since each random variable $Y_i^{(S)}$ is at least 1 and hence, $\mu_Y = \mathcal{E}[Y^{(S)}] \geq 1$, we need at most a polynomial $T = O(\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}(n - |S|))$ random variables for the Monte-Carlo estimation. Since, IICP has a worst-case time complexity

$O(m+n)$, the Monte-Carlo using IICP is an FPRAS for estimating outward influence.

Theorem 1. *Given arbitrary $0 \leq \epsilon, \delta \leq 1$ and a set S, the Monte-Carlo estimation using IICP returns an (ϵ, δ)-approximation of $\mathbb{I}_{out}(S)$ using $O(\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}(n - |S|))$ samples.*

In Section 2.2.4, we will show that both outward influence and influence spread can be estimated by a powerful algorithm saving a factor of more than $\frac{1}{\epsilon}$ random variables compared to this FPRAS estimation. The algorithm is built upon our mean estimation algorithms for bounded random variables proposed in the following.

2.2.3 Efficient Mean Estimation for Bounded Random Variables

In this section, we propose an efficient mean estimation algorithm for bounded random variables. This is the core of our algorithms for accurately and efficiently estimating the outward influence and influence spread in Section 2.2.4.

We first propose an 'intermediate' algorithm: *Generalized Stopping Rule Estimation* (GSRA) which relies on a simple stopping rule and returns an (ϵ, δ)-approximate of the mean of lower-bounded random variables. The GSRA simultaneously generalizes and fixes the error of the Stopping Rule Algorithm [26] which only aims to estimate the mean of $[0, 1]$ random variables and has a technical error in its proof.

The main mean estimation algorithm, namely Robust Sampling Algorithm (RSA) presented in Alg. 3, effectively takes into account both mean and variance of the random variables. It uses GSRA as a subroutine to estimate the mean value and variance at different granularity levels.

2.2.3.1 Generalized Stopping Rule Algorithm

We aim at obtaining an (ϵ, δ)-approximate of the mean of random variables $X_1, X_2, \ldots$. Specifically, the random variables are required to satisfy the following conditions:

- $a \leq X_i \leq b, \forall i = 1, 2, \ldots$
- $\mathcal{E}[X_{i+1}|X_1, X_2, ..., X_i] = \mu_X, \forall i = 1, 2, \ldots$

where $0 \leq a < b$ are fixed constants and (unknown) μ_X.

Our algorithm generalizes the stopping rule estimation in [26] that provides (ϵ, δ) estimation of the mean of i.i.d. random variables $X_1, X_2, ... \in [0, 1]$. The notable differences are the following:

- We discover and amend an error in the stopping algorithm in [26]: the number of samples drawn by that algorithm may not be sufficient to guarantee the (ϵ, δ)-approximation.
- We allow estimating the mean of random variables that are *possibly dependent* and/or with *different distributions*. Our algorithm works as long as the random variables have the same means. In contrast, the algorithm in [26] can only be applied for i.i.d random variables.
- Our proposed algorithm obtains an unbiased estimator of the mean, i.e. $\mathcal{E}[\hat{\mu}_X] = \mu_X$ while [26] returns a biased one.
- Our algorithm is faster than the one in [26] whenever the lower-bound for random variables $a > 0$.

Our Generalized Stopping Rule Algorithm (GSRA) is described in details in Alg. 2. Denote $c(\epsilon, \delta) = (2 + \frac{2}{3}\epsilon) \ln(\frac{2}{\delta}) \frac{1}{\epsilon^2}$.

The algorithm contains two main steps: 1) Compute the stopping threshold Υ (Line 2) which relies on the value of ϵ' computed from the given precision parameters ϵ, δ and the range $[a, b]$ of the random variables; 2) Consecutively acquire the random variables until the sum of their outcomes exceeds Υ (Line 4-5). Finally, it returns the average of the outcomes,

Algorithm 2: Generalized Stopping Rule Alg. (GSRA)

Input: Random variables $X_1, X_2, \ldots$ and $0 < \epsilon, \delta < 1$

Output: An (ϵ, δ)-approximate of $\mu_X = E[X_i]$

If $b - a < \epsilon b$, **return** $\mu_X = a$.

Compute: $\epsilon' = \epsilon\Big(1 - \frac{\epsilon b}{(2+\frac{2}{3}\epsilon)\ln(\frac{2}{\delta})(b-a)}\Big)$; $\Upsilon = (1+\epsilon)c(\epsilon', \delta)(b-a)$;

Initialize $h = 0, T = 0$;

while $h < \Upsilon$ **do**
| $h \leftarrow h + X_T, T \leftarrow T + 1$;

end

return $\hat{\mu}_X = h/T$;

$\hat{\mu}_X = h/T$ (Line 6), as an estimate for the mean, μ_X. Notice that Υ in GSRA depends on $(b - a)$ and thus, getting tighter bounds on the range of random variables holds a key for the efficiency of GSRA in application perspectives.

The approximation guarantee and number of necessary samples are stated in the following theorem.

Theorem 2. *The Generalized Stopping Rule Algorithm (GSRA) returns an (ϵ, δ)-approximate $\hat{\mu}_X$ of μ_X, i.e.,*

$$\Pr[(1-\epsilon)\mu_X \leq \hat{\mu}_X \leq (1+\epsilon)\mu_X] > 1 - \delta, \tag{2.18}$$

and, the number of samples T satisfies,

$$\Pr[T \leq (1+\epsilon)\Upsilon/\mu_X] > 1 - \delta/2. \tag{2.19}$$

The hole in the Stopping Rule Algorithm in [26]. The estimation algorithm in [26] for estimating the mean of random variables in range $[0, 1]$ also bases on a main stopping

rule condition as our GSRA. It computes a threshold

$$\Upsilon_1 = 1 + (1+\epsilon)4(e-2)\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}, \tag{2.20}$$

where e is the base of natural logarithm, and generates samples X_j until $\sum_{j=1}^{T} X_j \geq \Upsilon_1$. The algorithm returns $\hat{\mu}_X = \frac{\Upsilon_1}{T}$ as a biased estimate of μ_X.

Unfortunately, the threshold Υ_1 to determine the stopping time does not completely account for the fact that the necessary number of samples should go over the expected one in order to provide high solution guarantees. This actually causes a flaw in their later proof of the correctness.

To amend the algorithm, we slightly strengthen the stopping condition by replacing the ϵ in the formula of Υ with an $\epsilon' = \epsilon\Big(1 - \frac{\epsilon b}{(2+\frac{2}{3}\epsilon)\ln(\frac{2}{\delta})(b-a)}\Big)$ (Line 2, Alg. 2). Since $\epsilon b < b - a$ (else the algorithm returns $\mu_X = a$) and assume w.l.o.g. that $\delta < 1/2$, it follows that $\epsilon' \geq 0.729\epsilon$. Thus the number of samples, in comparison to those in the stopping rule algorithm in [26] increases by at most a constant factor.

Benefit of considering the lower-bound a. By dividing the random variables by b, one can apply the stopping rule algorithm in [26] on the normalized random variables. The corresponding value of Υ is then

$$\Upsilon = 1 + (1+\epsilon)(2+\frac{2}{3}\epsilon)\ln(\frac{2}{\delta})\frac{1}{\epsilon'^2}b \tag{2.21}$$

Υ in our proposed algorithm is however smaller by a multiplicative factor of $\frac{b-a}{b}$. Thus it is faster than the algorithm in [26] by a factor of $\frac{b-a}{b}$ on average. Note that in case of estimating the influence, we have $a = 1, b = n - |S|$. Compared to algorithm applied [26] directly, our GSRA algorithm saves the generated samples by a factor of $\frac{b-a}{b} = \frac{n-|S|-1}{n} = 1 - \frac{|S|+1}{n} < 1$.

Martingale theory to cope with weakly-dependent random variables. To prove Theorem 2, we need a stronger Chernoff-like bound to deal with the general random vari-

ables $X_1, X_2, \ldots$ in range $[a, b]$ presented in the following.

Let define random variables $Y_i = \sum_{j=1}^{i}(X_j - \mu_X), \forall i \geq 1$. Hence, the random variables $Y_1, Y_2, \ldots$ form a Martingale [79] due to the following,

$$\mathcal{E}[Y_i|Y_1, \ldots, Y_{i-1}] = \mathcal{E}[Y_{i-1}] + \mathcal{E}[X_i - \mu_X] = \mathcal{E}[Y_{i-1}].$$

Then, we can apply the following lemma from [24] stating,

Lemma 5. *Let $Y_1, \ldots, Y_i, \ldots$ be a martingale, such that $|Y_1| \leq \alpha$, $|Y_j - Y_{j-1}| \leq \alpha$ for all $j = [2, i]$, and*

$$\mathrm{Var}[Y_1] + \sum_{j=2}^{i} \mathrm{Var}[Y_j|Y_1, \ldots, Y_{j-1}] \leq \beta. \tag{2.22}$$

Then, for any $\lambda \geq 0$,

$$\Pr[Y_i - \mathcal{E}[Y_i] \geq \lambda] \leq \exp(-\frac{\lambda^2}{2/3 \cdot \alpha \cdot \lambda + 2 \cdot \beta}) \tag{2.23}$$

In our case, we have $|Y_1| = |X_1 - \mu_X| \leq b - a$, $|Y_j - Y_{j-1}| = |X_i - \mu_X| \leq b - a$, $\mathrm{Var}[Y_1] = \mathrm{Var}[X_1 - \mu_X] = \mathrm{Var}[X]$ and $\mathrm{Var}[Y_j|Y_1, \ldots, Y_j] = \mathrm{Var}[X_j - \mu_X] = \mathrm{Var}[X]$. Apply Lemma 2 with $i = T$ and $\lambda = \epsilon T \mu_X$, we have,

$$\Pr\Big[\sum_{j=1}^{T} X_j \geq (1+\epsilon)\mu_X T\Big] \leq \exp\big(\frac{-\epsilon^2 T^2 \mu_X^2}{\frac{2}{3}(b-a)\epsilon\mu_X T + 2\mathrm{Var}[X]T}\big) \tag{2.24}$$

Then, since $\mathrm{Var}[X] \leq \mu_X(b - \mu_X) \leq \mu_X(b - a)$ (since Bernoulli random variables with the same mean μ_X have the maximum variance), we also obtain,

$$\Pr\Big[\sum_{j=1}^{T} X_j \geq (1+\epsilon)\mu_X T\Big] \leq \exp\big(\frac{-\epsilon^2 T \mu_X}{(2 + \frac{2}{3}\epsilon)(b-a)}\big). \tag{2.25}$$

Similarly, $-Y_1, \ldots, -Y_i, \ldots$ also form a Martingale and applying Lemma 5 gives the

following probabilistic inequality,

$$\Pr\Big[\sum_{j=1}^{T} X_j \leq (1-\epsilon)\mu_X T\Big] \leq \exp\big(-\frac{\epsilon^2 T\mu_X}{2(b-a)}\big). \tag{2.26}$$

Algorithm 3: Robust Sampling Algorithm (RSA)

Input: Two streams of i.i.d. random variables, $X_1, X_2, \ldots$ and $X'_1, X'_2, \ldots$ and $0 < \epsilon, \delta < 1$

Output: An (ϵ, δ)-approximate $\hat{\mu}_X$ of μ_X

Step 1 // Obtain a rough estimate $\hat{\mu}'_X$ of μ_X

if $\epsilon \geq 1/4$ **then**
  return $\hat{\mu}_X \leftarrow$ GSRA$(< X_1, X_2, \ldots >, \epsilon, \delta)$
end

$\hat{\mu}'_X \leftarrow$ GSRA$(< X_1, X_2, \ldots >, \sqrt{\epsilon}, \delta/3)$

Step 2 // Estimate the variance $\hat{\sigma}^2_X$

$\Upsilon_2 = 2\frac{1+\sqrt{\epsilon}}{1-\sqrt{\epsilon}}(1 + \ln(\frac{3}{2})/\ln(\frac{2}{\delta})) \cdot \Upsilon; N_\sigma = \Upsilon_2 \cdot \epsilon/\hat{\mu}'_X; \Delta = 0;$ // Υ is defined the same as in Alg. 2

for $i = 1 : N_\sigma$ **do**
  $\Delta \leftarrow \Delta + (X'_{2i} - X'_{2i+1})^2/2;$
end

$\hat{\rho}_X = \max\{\hat{\sigma}^2_X = \Delta/N_\sigma, \epsilon\hat{\mu}'_X(b-a)\};$

Step 3 // Estimate μ_X

Set $T = \Upsilon_2 \cdot \hat{\rho}_X/(\hat{\mu}'^2_X(b-a)), S \leftarrow 0;$

for $i = 1 : T$ **do**
  $S \leftarrow S + X_i;$
end

return $\hat{\mu}_X = S/T;$

2.2.3.2 High-confident Sampling Algorithm

Our previously proposed GSRA algorithm may have problem in estimating means of random variables with small variances. An important tool that we rely on to prove the approximation guarantee in GSRA is the Chernoff-like bound in Eq. 2.25 and Eq. 2.26. However, from the inequality in Eq. 2.24, we can also derive the following stronger in-

equality,

$$\Pr\Big[\sum_{j=1}^{T} X_j \geq (1+\epsilon)\mu_X T\Big] \leq \exp\Big(\frac{-\epsilon^2 T^2 \mu_X^2}{\frac{2}{3}(b-a)\epsilon\mu_X T + 2\mathrm{Var}[X]T}\Big)$$
$$\leq \exp\Big(\frac{-\epsilon^2 T \mu_X^2}{(2+\frac{2}{3})\max\{\epsilon\mu_X(b-a), \mathrm{Var}[X]\}}\Big). \tag{2.27}$$

In many cases, random variables have small variances and hence $\max\{\epsilon\mu_X(b-a), \mathrm{Var}[X]\} = \epsilon\mu_X(b-a)$. Thus, Eq. 2.27 is much stronger than Eq. 2.25 and can save a factor of $\frac{1}{\epsilon}$ in terms of required observed influences translating into the sample requirement. However, both the mean and variance are not available.

To achieve a robust sampling algorithm in terms of sample complexity, we adopt and improve the $\mathcal{AA}$ algorithm in [26] for general cases of $[a, b]$ random variables. The robust sampling algorithms (RSA) subsequently will estimate both the mean and variance in three steps: 1) roughly estimate the mean value with larger error ($\sqrt{\epsilon}$ or a constant); 2) use the estimated mean value to compute the number of samples necessary for estimating the variance; 3) use both the estimated mean and variance to refine the required samples to estimate mean value with desired error (ϵ, δ).

Let $X_1, X_2, \ldots$ and $X'_1, X'_2, \ldots$ are two streams of i.i.d random variables. Our robust sampling algorithm (RSA) is described in Alg. 3. It consists of three main steps:

1) If $\epsilon \geq 1/4$, run GSRA with parameter ϵ, δ and return the result (Line 1-2). Otherwise, assume $\epsilon < 1/4$ and use the Generalized Stopping Rule Algorithm (Alg. 2) to obtain an rough estimate $\hat{\mu}'_X$ using parameters of $\epsilon' = \sqrt{\epsilon} < 1/2, \delta' = \delta/3$ (Line 3).

2) Use the estimated $\hat{\mu}'_X$ in step 1 to compute the necessary number of samples, N_σ, to estimate the variance of X_i, $\hat{\sigma}^2_X$. Note that this estimation uses the second set of samples, $X'_1, X'_2, \ldots$

3) Use both $\hat{\mu}'_X$ in step 1 and $\hat{\sigma}^2_X$ in step 2 to compute the actual necessary number of

samples, T, to approximate the mean μ_X. Note that this uses the same set of samples $X_1, X_2, \ldots$ as in the first step.

The numbers of samples used in the first two steps are always less than a constant times $\Upsilon \cdot \epsilon/\mu_X$ which is the minimum samples that we can achieve using the variance. This is because the first takes the error parameter $\sqrt{\epsilon}$ which is higher than ϵ and the second step uses $N_\sigma = \Upsilon_2 \cdot \epsilon/\hat{\mu}'_X$ samples.

At the end, the algorithm returns the influence estimate $\hat{\mu}_X$ which is the average over T samples, $\hat{\mu}_X = S/T$. The estimation guarantees are stated in the following theorem.

Theorem 3. *Let X be the probability distribution that $X_1, X_2, \ldots$ and $X'_1, X'_2, \ldots$ are drawn from. Let $\hat{\mu}_X$ be the estimate of $\mathcal{E}[X]$ returned by Alg. 3 and T be the number of drawn samples in Alg. 3 w.r.t. ϵ, δ. We have,*

(1) $\Pr[\mu_X(1-\epsilon) \leq \hat{\mu}_X \leq (1+\epsilon)\mu_X] \geq 1-\delta$,

(2) There is a universal constant c' such that

$$\Pr[T > c'\Upsilon\rho_X/(\mu_X^2(b-a))] \leq \delta \tag{2.28}$$

where $\rho_Z = \max\{\epsilon\mu_X(b-a), \mathrm{Var}[X]\}$.

Compared to the $\mathcal{AA}$ algorithm in [26], first of all, we replace their stopping rule algorithm with GSRA and also, we change the computation of Υ_2 which is always smaller than that of [26] by a factor of $1 + \sqrt{\epsilon} - 2\epsilon \geq 1$ when $\epsilon \leq 1/4$.

2.2.4 Influence Estimation at Scale

This section applies our RSA algorithm to estimate both the outward influence and the traditional influence spread.

2.2.4.1 Outward Influence Estimation

We directly apply RSA algorithm on two streams of i.i.d. random variables $Y_1^{(S)}, Y_2^{(S)}, \ldots$ and $Y_1'^{(S)}, Y_2'^{(S)}, \ldots$, which are generated by IICP sampling algorithm, with the precision parameters ϵ, δ.

The algorithm is called *Scalable Outward Influence Estimation Algorithm* (SOIEA) and presented in Alg. 4 which generates two streams of random variables $Y_1^{(S)}, Y_2^{(S)}, \ldots$ and $Y_1'^{(S)}, Y_2'^{(S)}, \ldots$ (Line 1) and applies RSA algorithm on these two streams (Line 2). Note that outward influence estimate is achieved by scaling down μ_Y by β_0 (Lemma 4).

Algorithm 4: SOIEA Alg. to estimate outward influence

Input: A probabilistic graph $\mathcal{G}$, a set S and ϵ, δ

Output: $\hat{\mathbb{I}}(S)$ - an (ϵ, δ)-estimate of $\mathbb{I}(S)$

Generate two streams of i.i.d. random variables $Y_1^{(S)}, Y_2^{(S)}, \ldots$ and $Y_1'^{(S)}, Y_2'^{(S)}, \ldots$ by IICP algorithm.

return $\hat{\mathbb{I}}_{out}(S) \leftarrow \beta_0 \cdot \mathsf{RSA}(< Y_1^{(S)}, \cdots >, < Y_1'^{(S)}, \cdots >, \epsilon, \delta)$

We obtain the following theoretical results incorporated from Theorem 3 of RSA and IICP samples.

Theorem 4. *The SOIEA algorithm gives an (ϵ, δ) outward influence estimation. The observed outward influences (sum of $Y^{(S)}$) and the number of generated random variables are in $O(\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}\frac{\rho_Y}{\mathbb{I}_{out}(S)/\beta_0})$ and $O(\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}\frac{\rho_Y}{\mathbb{I}^2_{out}(S)/\beta_0^2})$ respectively, where $\rho_Y = \max\{\epsilon\mathbb{I}_{out}(S)(n-|S|-1)/\beta_0, \mathrm{Var}[Y_i^{(S)}]\}$.*

Note that $\mathcal{E}[Y^{(S)}] = \mathbb{I}_{out}(S)/\beta_0 \geq 1$.

2.2.4.2 Influence Spread Estimation

Not only is the concept of *outward influence* helpful in discriminating the relative influence of nodes but also its sampling technique, IICP, can help scale up the estimation of influence spread (IE) to billion-scale networks.

Naive approach. A naive approach is to 1) obtain an (ϵ, δ)-approximation $\hat{\mathbb{I}}_{out}(S)$ of $\mathbb{I}_{out}(S)$ using Monte-Carlo estimation 2) return $\hat{\mathbb{I}}_{out}(S) + |S|$. It is easy to show that this approach return an (ϵ, δ)-approximation for $\mathbb{I}(S)$. This approach will require $O(\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}n)$ IICP random samples.

However, the naive approach is not optimized to estimate influence due to several reasons: 1) a loose bound $\mu_Y = \mathcal{E}[Y^{(S)}] \geq 1$ is applied to estimate outward influence; 2) casting from (ϵ, δ)-approximation of outward influence to (ϵ, δ)-approximation of influence introduces a gap that can be used to improve the estimation guarantees. We next propose more efficient algorithms based on Importance IC Sampling to achieve an (ϵ, δ)-approximate of both outward influence and influence spread. Our methods are based on two effective mean estimation algorithms.

Our approach. Based on the observations that

- $1 \leq Y^{(S)} \leq n - |S|$, i.e., we know better bounds for $Y^{(S)}$ in comparison to the cascade size which is in the range $[1, n]$.
- As we want to have an (ϵ, δ)-approximation for $Y^{(S)} + |S|$, the fixed add-on $|S|$ can be leveraged to reduce the number of samples.

We combine the effective RSA algorithm with our Importance IC Polling (IICP) for estimating the influence spread of a set S. For influence spread estimation, we will analyze random variables based on samples generated by our Importance IC Polling scheme and use those to devise an influence estimation algorithm.

Since outward influence and influence spread differ by an additive factor of $|S|$, for each outward sample $Y^{(S)}$ generated by IICP, let define a corresponding variable $Z^{(S)}$,

$$Z^{(S)} = Y^{(S)} \cdot \beta_0 + |S|, \tag{2.29}$$

where β_0 is defined in Eq. 2.16. We obtain,

- $|S| + \beta_0 \leq Z^{(S)} \leq |S| + \beta_0(n - |S|)$,
- $\mathcal{E}[Z^{(S)}] = \mathcal{E}[Y^{(S)}] \cdot \beta_0 + |S| = \mathbb{I}_{out}(S) + |S| = \mathbb{I}(S)$,

and thus we can to approximate $\mathbb{I}(S)$ by estimating $\mathcal{E}[Z^{(S)}]$.

Recall that to estimate the influence $\mathbb{I}(S)$ of a seed set S, all the previous works [55, 70, 19] resort to simulating many influence cascades from S and take the average size of those generated cascades. Let call $M^{(S)}$ the random variable representing the size of such a influence cascade. Then, we have $\mathcal{E}[M^{(S)}] = \mathbb{I}(S)$. Although both $Z^{(S)}$ and $M^{(S)}$ can be used to estimate the influence, they have different variances that lead to difference in convergence speed when estimating their means. The relation between variances of $Z^{(S)}$ and $M^{(S)}$ is stated as follows.

Lemma 6. *Let $Z^{(S)}$ defined in Eq. 2.29 and $M^{(S)}$ be random variable for the size of a influence cascade, the variances of $Z^{(S)}$ and $M^{(S)}$ satisfy,*

$$\mathrm{Var}[Z^{(S)}] = \beta_0 \cdot \mathrm{Var}[M^{(S)}] - (1 - \beta_0)\mathbb{I}^2_{out}(S) \tag{2.30}$$

Note that $0 \leq \beta_0 \leq 1$ and $\mathbb{I}(S) \geq |S|$. Thus, the variance of $Z^{(S)}$ is much smaller than $M^{(S)}$. Our proposed RSA on random variables X_i makes use of the variances of random variables and thus, benefits from the small variance of $Z^{(S)}$ compared to the same algorithm on the previously known random variables $M^{(S)}$.

Thus, we apply the RSA on random variables generated by IICP to develop Scalable Influence Estimation Algorithm (SIEA). SIEA is described in Alg. 5 which consists of two

Algorithm 5: SIEA Algorithm to estimate influence spread

Input: A probabilistic graph $\mathcal{G}$, a set S and ϵ, δ

Output: $\hat{\mathbb{I}}(S)$ - an (ϵ, δ)-estimate of $\mathbb{I}(S)$

Generate two streams of i.i.d. random variables $Y_1^{(S)}, Y_2^{(S)}, \dots$ and $Y_1'^{(S)}, Y_2'^{(S)}, \dots$ by IICP algorithm.

Compose two streams $Z_1^{(S)}, Z_2^{(S)}, \dots$ and $Z_1'^{(S)}, Z_2'^{(S)}, \dots$ from $Y_1^{(S)}, Y_2^{(S)}, \dots$ and $Y_1'^{(S)}, Y_2'^{(S)}, \dots$ using Eq. 2.29.

return $\hat{\mathbb{I}}(S) \leftarrow$RSA$(< Z_1^{(S)}, \cdots >, < Z_1'^{(S)}, \cdots >, \epsilon, \delta)$

main steps: 1) generate i.i.d. random variables by IICP and 2) convert those variables to be used in RSA to estimate influence of S. The results are stated as follows,

Theorem 5. *The SIEA algorithm gives an (ϵ, δ) influence spread estimation. The observed influences (sum of random variables $Z^{(S)}$) and the number of generated random variables are in $O(\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}\frac{\rho_Z}{\mathbb{I}(S)})$ and $O(\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}\frac{\rho_Z}{\mathbb{I}^2(S)})$, where $\rho_Z = \max\{\epsilon\mathbb{I}(S)\beta_0(n - |S| - 1), \mathrm{Var}[Z_i^{(S)}]\}$.*

Comparison to INFEST [77]. Compared to the most recent state-of-the-art influence estimation in [77] that requires $O(\frac{n\log^5(n)}{\epsilon^2})$ observed influences, the SIEA algorithm incorporating IICP sampling with RSA saves at least a factor of $\log^4(n)$. That is because the necessary observed influences in SIEA is bounded by $O(\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}\frac{\beta_0\rho_Z}{\mathbb{I}(S)})$. Since $\mathrm{Var}[Z_i^{(S)}] \leq \mathbb{I}(S)(|S| + \beta_0(n - |S|) - \mathbb{I}(S)) \leq \mathbb{I}(S)(n - |S| - 1)$ and hence, $\rho_Z \leq \mathbb{I}(S)(n - |S| - 1)$, when $\delta = \frac{1}{n}$ as in [77], the observed influences is then,

$$O(\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}\frac{\rho_Z}{\mathbb{I}(S)}) \leq O(\frac{n\log(2/\delta)}{\epsilon^2}) \leq O(\frac{n\log(n)}{\epsilon^2}) \tag{2.31}$$

Consider ϵ, δ as constants, the observed influences is $O(n)$.

2.2.4.3 Influence Spread under other Models

We can easily apply the RSA estimation algorithm to obtain an (ϵ, δ)-estimate of the influence spread under other cascade models as long as there is a Monte-Carlo sampling procedure to generate sizes of random cascades. For most stochastic diffusion models, including both discrete-time models, e.g. the popular LT with a naive sample generator described in [55], SI and SIR [27] or their variants with deadlines [86], and continuous-time models [32], designing such a Monte-Carlo sampling procedure is straightforward. Since the influence cascade sizes are at least the seed size, we always needs at most $O(n)$ samples.

To obtain more efficient sampling procedures, we can extend the idea of sampling non-trivial cascade in IICP to other models. Such sampling procedures in general will result in random variables with smaller variances and tighter bounds on the ranges. In turns, RSA, that benefits from smaller variance and range, will requires fewer samples for estimation.

2.2.4.4 Parallel Estimation Algorithms

We develop the parallel versions of our algorithms to speed up the computation and demonstrate the easy-to-parallelize property of our methods. Our main idea is that the random variable generation by IICP can be run in parallel. In particular, random variables used in each step of the core RSA algorithm can be generated simultaneously. Recall that IICP only needs to store a queue of newly active nodes, an array to mark the active nodes and a single variable $Y^{(S)}$. In total, each thread requires space in order of the number of active nodes in that simulation, $O(Y^{(S)})$, which is at most linear with size of the graph $O(n)$. In fact due to the stopping condition of linear number of observed influences, the total size of all the threads is bounded by $O(n)$ assumed the number of threads is relatively small compared to n.

Moreover, our algorithms can be implemented efficiently in terms of communication cost in distributed environments. This is because the output of IICP algorithm is just a single number $Y^{(S)}$ and thus, worker nodes in a distributed environment only communicate that single number back to the node running the estimation task. Here each IICP node holds a copy of the graph. However, the programming model needs to be considered carefully. For instance, as pointed out in many studies that the famous MapReduce is not a good fit for iterative graph processing algorithms [48, 72].

2.2.5 Experiments

We will experimentally show that Outward Influence Estimation (SOIEA) and Outward-Based Influence Estimation (SIEA) are not only several orders of magnitudes faster than existing state-of-the-art methods but also consistently return much smaller errors. We present empirical validation of our methods on both real world and synthetic networks.

2.2.5.1 Experimental Settings

Algorithms. We compare performance of SOIEA and SIEA with the following algorithms:

- INFEST [77]: A recent influence estimation algorithm by Lucier et al. [77] in KDD'15 that provides approximation guarantees. We reimplement the algorithm in C++ accordingly to the description in [77][1].

- $\mathsf{MC}_{10\mathsf{K}}$, $\mathsf{MC}_{100\mathsf{K}}$: Variants of Monte-Carlo method that generates the traditional influence cascades [55, 70] to estimate (outward) influence spread.

- $\mathsf{MC}_{\epsilon,\delta}$: The Monte-Carlo method that uses the traditional influence cascades and

[1]Through communication with the authors of [77], the released code has some problem and is not ready for testing.

guarantees (ϵ, δ)-estimation. Following [77], $\mathsf{MC}_{\epsilon,\delta}$ is only for measuring the running time of the normal Monte-Carlo to provide the same (ϵ, δ)-approximation guarantee. In particular, we obtain running time of $\mathsf{MC}_{\epsilon,\delta}$ by interpolating from that from $\mathsf{MC}_{10\mathsf{K}}$, i.e. $\frac{1}{\epsilon^2}\ln(\frac{1}{\delta})n\frac{\mathsf{Time}(\mathsf{MC}_{10\mathsf{K}})}{10000}$.

Table 2.: Datasets' Statistics

Dataset	**#Nodes**	**#Edges**	**Avg. Degree**
NetHEP[2]	15K	59K	4.1
NetPHY[2]	37K	181K	13.4
Epinions[2]	75K	841K	13.4
DBLP[2]	655K	2M	6.1
Orkut[2]	3M	117M	78.0
Twitter [65]	41.7M	1.5G	70.5
Friendster[2]	65.6M	1.8G	54.8

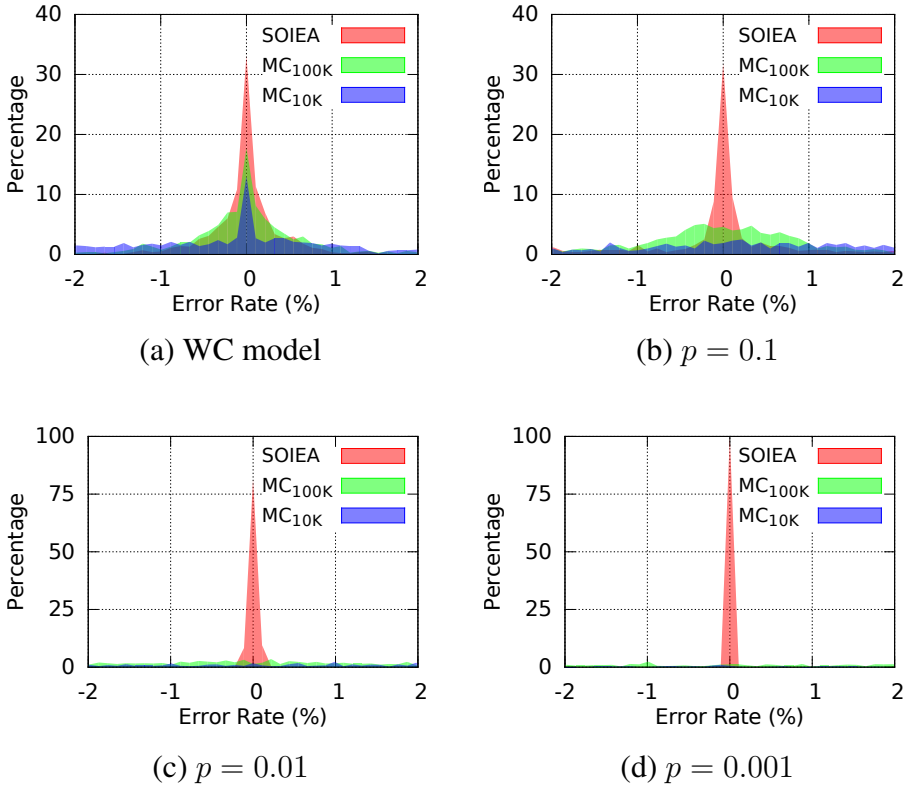
[2]From http://snap.stanford.edu

Table 3.: Comparing performance of algorithms in estimating outward influences

		Avg. Rel. Error (%)			**Max. Rel. Error (%)**			**Running time (sec)**			
Dataset	**Edge Models**	SOIEA	$\mathsf{MC}_{10\mathsf{K}}$	$\mathsf{MC}_{100\mathsf{K}}$	SOIEA	$\mathsf{MC}_{10\mathsf{K}}$	$\mathsf{MC}_{100\mathsf{K}}$	SOIEA	$\mathsf{MC}_{10\mathsf{K}}$	$\mathsf{MC}_{100\mathsf{K}}$	$\mathsf{MC}_{\epsilon,\delta}$
NetHEP	wc	0.3	1.9	0.6	2.3	25.0	8.9	0.1	0.1	0.1	12.3
	$p = 0.1$	1.0	3.7	1.2	9.7	63.0	17.2	0.2	0.1	1.0	149.5
	$p = 0.01$	0.0	4.5	1.6	0.2	20.2	9.2	0.2	0.1	0.1	8.8
	$p = 0.001$	0.0	19.2	4.6	0.1	100.0	26.4	0.2	0.1	0.1	8.5
NetPHY	wc	0.1	1.4	0.4	1.5	32.8	6.2	0.4	0.1	0.1	34.7
	$p = 0.1$	0.5	4.0	1.3	6.6	46.3	18.5	0.5	0.1	0.5	203.0
	$p = 0.01$	0.0	5.5	1.7	0.2	30.4	10.7	0.6	0.1	0.1	25.0
	$p = 0.001$	0.0	19.1	5.1	0.0	80.0	28.1	0.7	0.1	0.1	24.0

Datasets. We use both real-world networks and synthetic networks generated by GTgraph [8]. For real world networks, we choose a set of 7 datasets with sizes from tens of thousands to 65.6 millions. Table 18 gives a summary. GTgraph generates synthetic graphs with varying number of nodes and edges.

Metrics. We compare the performance of the algorithms in terms of solution quality and running time. To compare the solution quality, we adopt the relative error which



(a) WC model (b) $p = 0.1$

(c) $p = 0.01$ (d) $p = 0.001$

Fig. 4.: Error distributions (histogram) of the approximation errors of SOIEA, MC_{10K}, MC_{100K} on NetHEP

shows how far the estimated number from the "ground truth". The relative error of outward influence is computed as follows:

$$|\frac{\hat{\mathbb{I}}_{out}(S)}{\mathbb{I}_{out}(S)} - 1| \cdot 100\% \tag{2.32}$$

where $\hat{\mathbb{I}}_{out}(S)$ is estimated outward influence of seed set S by the algorithm, $\mathbb{I}_{out}(S)$ is "ground truth" for S.

Similarly, relative error of influence spread is,

$$|\frac{\hat{\mathbb{I}}(S)}{\mathbb{I}(S)} - 1| \cdot 100\% \tag{2.33}$$

We test the algorithms on estimating different seed set sizes. For each size, we generate a set of 1000 random seed sets. We will report the average relative error (Avg. Rel. Error) and maximum relative error (Max. Rel. Error).

Ground-truth computation. We use estimates of influence and outward influence with a very small error corresponding to the setting $\epsilon = 0.005, \delta = 1/n$. We note that previous researches [77, 105] compute the "ground truth" by running Monte-Carlo with 10,000 samples which is not sufficient as we will show later in our experiments.

Parameter Settings. For each of the datasets, we consider two common edge weighting models:

- Weighted Cascade (WC): The weight of edge (u, v) is calculated as $w(u, v) = \frac{1}{d_{in}(v)}$ where $d_{in}(v)$ denotes the in-degree of node v, as in [19, 106, 25, 105, 84].
- Constant model: All the edges has the same constant probability p as in [55, 19, 25]. We consider three different values of p, i.e. $0.1, 0.01, 0.001$.

We set $\epsilon = 0.1$, $\delta = 1/n$ for SOIEA and SIEA by default or explicitly stated otherwise.

Environment. All algorithms are implemented in C++ and compiled using GCC 4.8.5. We conduct all experiments on a CentOS 7 workstation with two Intel Xeon 2.30GHz CPUs adding up to 20 physical cores and 250GB RAM.

2.2.5.2 Outward Influence Estimation

We compare SOIEA against $\mathsf{MC}_{10\mathsf{K}}$ and $\mathsf{MC}_{100\mathsf{K}}$ in four different edge models on NetHEP and NetPHY dataset. The results are presented in Table 3 and Figure 4.

Solution Quality: Table 3 illustrates that the outward influences computed by SOIEA consistently have much smaller errors in both average and maximum cases than $\mathsf{MC}_{10\mathsf{K}}$ and $\mathsf{MC}_{100\mathsf{K}}$ in all edge models. In particular, on NetHEP with $p = 0.001$ edge model, SOIEA

has average relative error close to 0% while it is 19.2% and 4.6% for MC_{10K}, MC_{100K} respectively; the maximum relative errors of MC_{10K}, MC_{100K} in this case are 100%, 26.4% which are much higher than SOIEA of 0.1%. Apparently, MC_{100K} has smaller error rate than MC_{10K} since it uses 10 times more samples.

Figure 4 shows error distributions of SOIEA, MC_{10K}, and MC_{100K} on NetHEP. In all considered edge models, SOIEA's error highly concentrates around 0% while errors of MC_{10K} and MC_{100K} wildly spread out to a very large spectrum. In particular, SOIEA has a huge spike at the 0 error while both MC_{10K} and MC_{100K} contain two heavy tails in two sides of their error distributions. Moreover, when p gets smaller, the tails get larger as more and more empty influence simulations are generated in the traditional method.

Running Time: From Table 3, the running time of MC_{10K} and MC_{100K} is close to that of SOIEA while $\mathsf{MC}_{\epsilon,\delta}$ takes up to 700 times slower than the others. Thus, in order to achieve the same approximation guarantee as SOIEA, the naive Monte-Carlo will need 700 more time than SOIEA.

Overall, SOIEA achieves significantly better solution quality and runs substantially faster than Monte-Carlo method. With larger number of samples, Monte-Carlo method can improve the quality but the running time severely suffers.

2.2.5.3 Influence Spread Estimation

This experiment evaluates SIEA by comparing its performance with the most recent state-of-the-art INFEST and naive Monte-Carlo influence estimation. Here, we use WC model to assign probabilities for the edges. We set the ϵ parameter for INFEST to 0.4 since we cannot run with smaller value of ϵ for this algorithm. Note that INFEST guarantees an error of $(1 + 8\epsilon)$, *which is equivalent to a maximum relative error of 320%.* For a fair comparison, we also run SIEA with $\epsilon = 0.4$. We use the gold-standard 10000 samples for the Monte-Carlo method (MC_{10K}). We set a time limit of 6 hours for all algorithms.

Table 4.: Comparing performance of algorithms in estimating influence spread in WC Model (seed set size $|S| = 1$)

	Avg. Rel. Error (%)			Max. Rel. Error (%)			Running time (sec.)				
Dataset	SIEA	MC_{10K}	INFEST	SIEA	MC_{10K}	INFEST	SIEA	SIEA (16 cores)	MC_{10K}	$MC_{\epsilon,\delta}$	INFEST
NetHEP	0.2	1.2	17.7	1.5	6.6	82.7	0.1	0.1	0.0	0.8	3417.6
NetPHY	0.1	0.4	22.9	0.6	5.3	43.0	0.1	0.1	0.0	2.6	8517.7
Epinions	0.9	5.3	n/a	5.2	19.7	n/a	0.2	0.1	0.0	21.9	n/a
DBLP	0.3	1.2	n/a	1.9	8.7	n/a	2.8	1.3	0.1	770.4	n/a
Orkut	0.5	3.0	n/a	3.2	16.0	n/a	54.2	4.76	2.9	$8.2 \cdot 10^4$	n/a
Twitter	1.0	37.1	n/a	3.1	**240.8**	n/a	1272.3	106.2	7.9	$3.5 \cdot 10^6$	n/a
Friendster	0.1	3.1	n/a	0.6	23.6	n/a	1510.1	**165.1**	2.8	$2.1 \cdot 10^6$	n/a

Table 5.: Comparing performance of algorithms in estimating influence spread in WC Model (seed set size $|S| = 5\%|V|$)

	Avg. Rel. Error (%)			Max. Rel. Error (%)			Running time (sec.)				
Dataset	SIEA	MC_{10K}	INFEST	SIEA	MC_{10K}	INFEST	SIEA	SIEA (16 cores)	MC_{10K}	$MC_{\epsilon,\delta}$	INFEST
NetHEP	0.1	0.0	11.1	0.4	0.2	14.1	0.1	0.1	2.1	191.7	600.5
NetPHY	0.1	0.0	24.4	0.2	0.1	26.3	0.1	0.1	5.3	1297.1	3326.4
Epinions	0.2	0.1	20.2	0.4	0.2	23.8	0.3	0.1	20.1	$1.1 \cdot 10^4$	9325.6
DBLP	0.0	1.8	n/a	0.2	1.9	n/a	3.5	0.3	184.9	$1.0 \cdot 10^6$	n/a
Orkut	0.1	0.0	n/a	0.7	0.1	n/a	51.6	4.6	5322.8	$1.5 \cdot 10^8$	n/a
Twitter	0.2	n/a	n/a	0.5	n/a	n/a	1061.6	93.5	n/a	n/a	n/a
Friendster	0.1	n/a	n/a	0.2	n/a	n/a	2068.8	**183.1**	n/a	n/a	n/a

Solution Quality: Table 4 presents the solution quality of the algorithms in estimating size 1 seed sets, i.e. $|S| = 1$. It shows that SIEA consistently achieves substantially higher quality solution than both INFEST and MC_{10K}. Note that INFEST can only run on NetHEP and NetPHY under time limit. The average relative error of INFEST is 88 to 229 times higher than SIEA while its maximum relative error is up to 82% compared to the ground truth. The large relative error of INFEST is explained by its loose guaranteed relative error (320%). Whereas, the average relative error of MC_{10K} is up to 37 times higher than SIEA. The maximum relative error of MC_{10K} is up to 240% higher than the ground truth on Twitter dataset that demonstrates the insufficiency of using 10000 traditional influence samples to get the ground truth.

Differ from Table 4, Table 5 shows the results in estimating influences of seed sets of

size 5% the total number of nodes. Under 6 hour limit, INFEST can only run on NetHEP, NetPHY, and Epinions while $\mathsf{MC}_{10\mathsf{K}}$ could not handle the large Twitter and Friendster graph. INFEST still has a very high error compared to the other two while SIEA and $\mathsf{MC}_{10\mathsf{K}}$ returns the similar quality solutions. This is because 5% of the nodes is an enormous number, i.e. > 1000000 for Friendster, and thus, the influence is huge and very few samples are needed regardless of using the traditional method or IICP.

Running Time: In both cases of two seed set sizes, SIEA vastly outperforms $\mathsf{MC}_{\epsilon,\delta}$ and INFEST by several orders of magnitudes. INFEST is up to 10^5 times slower than SIEA and can only run on small networks, i.e. NetHEP, NetPHY and Epinions. Compared with $\mathsf{MC}_{\epsilon,\delta}$, the speedup factor is around 10^4, thus, $\mathsf{MC}_{10\mathsf{K}}$ cannot run for the two largest networks, Twitter and Friendster in case $|S| = 5\%|V|$.

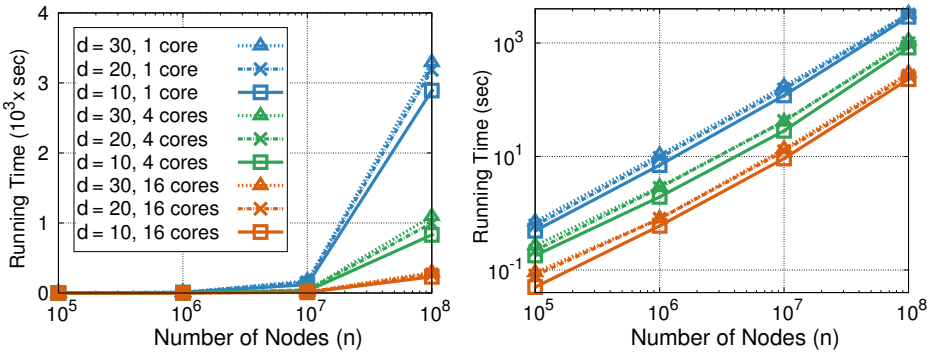
We also test the parallel version of SIEA. With 16 cores, SIEA runs about 12 times faster than that on a single core in large networks achieving an effective factor of around 75%.

Overall, SIEA consistently achieves much better solution quality and run significantly fastest than INFEST and the naive MC method. Surprisingly, under time limit of 6 hours, INFEST can only handle small networks and has very high error. The MC method achieves better accuracy for large seed sets, however, its running time increases dramatically resulting in failing to run on large datasets.

2.2.5.4 Scalability Test

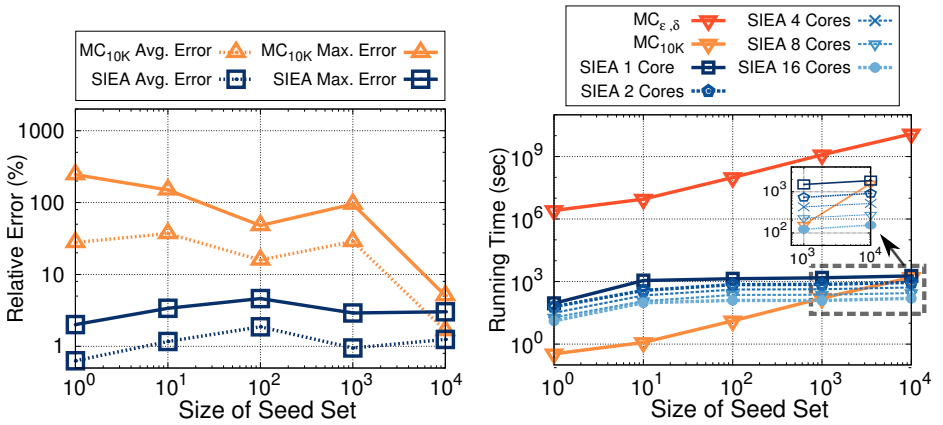
We test the scalability of the single core and parallel versions of our method on synthetic networks generated by the well-known GTgraph with various network sizes. We also carry the same tests on the real-world Twitter network in comparison with the MC.

On Synthetic Datasets: We generate synthetic graphs using GTgraph[8], a standard graph generator used widely in large scale experiments on graph algorithms [49, 5, 12].



(a) Running time in linear (b) Running time in log scale

Fig. 5.: Running time of SIEA on synthetic networks



(a) Relative Error (b) Running Time

Fig. 6.: Comparing SIEA, MC_{10K} and $MC_{\epsilon,\delta}$ on Twitter

We generate graphs with number of nodes $n \in \{10^5, 10^6, 10^7, 10^8\}$. For each size n, we generate 3 different graphs with average degree $d \in \{10, 20, 30\}$. We use the WC model to assign edge weights. We run SIEA with different number of cores $C = \{1, 4, 16\}$

Figure 5 reports the time SIEA spent to estimate influence spread of seed set of size 1. With the same number of nodes, we see that the running time of SIEA does not significantly increase as the average degree increases. Figure 5b views Figure 5a in logarithmic scale to show the linear increase of running time with respect to the increases of nodes. As

Table 6.: Comparing performance of algorithms in estimating influence spread in LT model (seed set size $|S| = 1$)

	Avg. Rel. Error (%)			Max. Rel. Error (%)			Running time (sec.)				
Dataset	SIEA$_{LT}$	MC$_{10K}$	MC$_{100K}$	SIEA$_{LT}$	MC$_{10K}$	MC$_{100K}$	SIEA$_{LT}$	SIEA$_{LT}$ (16 cores)	MC$_{10K}$	MC$_{100K}$	MC$_{\epsilon,\delta}$
NetHEP	1.6	1.6	0.6	8.4	7.9	2.5	0.0	0.0	0.0	0.1	1.0
NetPHY	1.2	0.5	0.3	12.7	4.4	1.4	0.0	0.0	0.0	0.1	2.9
Epinions	1.5	4.3	2.2	7.0	17.4	7.4	0.7	0.4	0.0	0.4	24.5
DBLP	0.4	1.0	0.5	5.7	11.4	2.2	2.4	0.4	0.3	2.5	1530.4
Orkut	0.5	3.3	1.1	1.9	22.1	5.9	249.4	25.0	8.5	84.2	$4.6 \cdot 10^4$
Twitter	2.4	36.1	20.7	7.1	97.5	85.6	6820.0	548.6	32.2	287.6	$1.4 \cdot 10^7$
Friendster	0.2	3.1	1.4	2.4	16.5	9.0	6183.9	701.8	20.4	137.8	$9.3 \cdot 10^6$

expected, SIEA speeds up proportionally to number of cores used. As a result, SIEA with 16 cores is able to estimate influence spread of a random node on a synthetic graph of 100 million nodes and 1.5 billion of edges in just 5 minutes.

On Twitter Dataset: Figure 6 evaluates the performance of SIEA in comparison with MC$_{10K}$ on various seed set sizes $|S| = \{1, 10, 100, 1k, 10k\}$ on Twitter dataset. On all the sizes of seed sets, SIEA consistently has average and maximum relative errors smaller than 10% (Figure 6a). The maximum relative error of MC$_{10K}$ goes up to 244% with seed set size $|S| = 1$. As observed in experiments with large size seed sets, both SIEA and MC$_{10K}$ have similar error rate with seed set size $|S| = 10000$.

In terms of running time, as the seed set size increases in powers of ten, SIEA's running time increases in much lower pace, e.g. few hundreds of seconds, while MC$_{\epsilon,\delta}$ consumes proportionally more time (Figure 6b). Figure 6b also evaluates parallel implementation of SIEA by varying number of CPU cores $C = \{1, 2, 4, 8, 16\}$. The running time of SIEA reduces almost two times every time the number of cores doubles confirming the almost linear speedup.

Altogether, the parallel implementation of SIEA shows a linear speedup behavior with respect to the number of cores used. On the same network with size of seed sets linearly grows, SIEA requires slightly more time to estimate influence spread while Monte-Carlo

shows a linear runtime requirement. Throughout the experiments, SIEA always guarantees small error rate within ϵ.

2.2.5.5 Influence Estimation under LT Model

We illustrate the generality of our algorithms in various diffusion model by adapting SIEA for the LT model by only replacing IICP with the sampling algorithm for the LT [55]. The algorithm is then named SIEA_{LT}. The setting is similar to the case of IC. We present the results of SIEA_{LT} compared with MC_{10K}, MC_{100K}, $\mathsf{MC}_{\epsilon,\delta}$ in Table 6. INFEST is initially proposed for the IC model, thus, we results for INFEST under the LT model are not available.

The results are mostly consistent with those observed under the IC model. SIEA_{LT} obtains significantly smaller errors and runs in order of magnitudes faster than the counterparts. The results again confirm that the estimation quality of MC using $10K$ samples is not good enough to be considered as gold-standard quality benchmark.

2.2.6 Related work

In a seminal paper [55], Kempe et al. formulated and generalized two important influence diffusion models, i.e. Independent Cascade (IC) and Linear Threshold (LT). This work has motivated a large number of follow-up researches on information diffusion [54, 19, 11, 77, 25, 92] and applications in multiple disciplines [70, 51, 62]. Kempe et al. [55] proved the monotonicity and submodularity properties of influence as a function of sets of nodes. Later, Chen et al. [19] proved that computing influence under these diffusion models is #P-hard.

Most existing works uses the naive influence cascade simulations to estimate influences [55, 70, 19, 77]. Most recently, Lucier et al. [77] proposed an estimation algorithm with rigorous quality guarantee for a single seed set. The main idea is guessing a small in-

terval of size $(1+\epsilon)$ that the true influence falls in and verifying whether the guess is right with high probability. However, their approach is not scalable due to a main drawback that the guessed intervals are very small, thus, the number of guesses as well as verifications made is huge. As a result, the method in [77] can only run for small dataset and still takes hours to estimate a single seed set. They also developed a distributed version on MapReduce however, graph algorithms on MapReduce have various serious issues [48, 72].

Influence estimation oracles are developed in [25, 92] which take advantage of sketching the influence to preprocess the graph for fast queries. Cohen et al. [25] use the novel bottom-k min-hash sketch to build combined reachability sketches while Ohsaka et al. in [92] adopt the reverse influence sketches. [92] also introduces the reachability-true-based technique to deal with dynamic changes in the graphs. However, these methods require days for preprocessing in order to achieve fast responses for multiple queries.

There have also been increasing interests in many related problems. [14, 43] focus on designing data mining or machine learning algorithms to extract influence cascade model parameters from real datasets, e.g. action logs. Influence Maximization, which finds a seed set of certain size with the maximum influence among those in the same size, found many real-world applications and has attracted a lot of research work [55, 70, 19, 83, 11, 105, 80, 84].

2.2.7 Conclusion

This paper investigates a new measure, called Outward Influence, for nodes' influence in social networks. Outward influence inspires new statiscal algorithms, namely Importance IC Polling (IICP) and Robust Mean Estimation (RSA) to estimate influence of nodes under various stochastic diffusion models. Under the popular IC model, the IICP leads to an FPRAS for estimating outward influence and SIEA to estimate influence spread. SIEA is $\Omega(\log^4(n))$ times faster than the most recent state-of-the-art and experimentally

outperform the other methods by several orders of magnitudes. As previous approaches to compute ground truth influence can result in high error and long computational time, our algorithms provides concrete and scalable tools to estimate ground-truth influence for research on network cascade and social influence.

2.2.8 Appendix

2.2.8.1 Proof of Lemma 1

Recall that on a sampled graph $g \sim \mathcal{G}$, for a set $S \subseteq V$, we denote $r_g^{(o)}(S)$ to be the set of nodes, excluding the ones in S, that are reachable from S through live edges in g, i.e. $r_g^{(o)}(S) = r_g(S) \backslash S$. Alternatively, $r_g^{(o)}(S)$ is called the outward influence cascade of S on sample graph g and, consequently, we have,

$$\mathbb{I}_{out}(S) = \sum_{g \sim \mathcal{G}} |r_g^{(o)}(S)| \Pr[g \sim \mathcal{G}]. \tag{2.34}$$

It is sufficient to show that $|r_g^{(o)}(S)|$ is submodular, as $\mathbb{I}_{out}(S)$ is a linear combination of submodular functions. Consider a sample graph $g \sim \mathcal{G}$, two sets S, T such that $S \subseteq T \subseteq V$ and $v \in V \backslash T$. We have three possible cases:

- **Case** $v \in r_g^{(o)}(S)$: then $v \in r_g^{(o)}(T)$ since $S \subseteq T$ and $v \notin T$. Thus, we have the following,

$$r_g^{(o)}(S \cup \{v\}) - r_g^{(o)}(S) = r_g^{(o)}(T \cup \{v\}) - r_g^{(o)}(T) = -1. \tag{2.35}$$

- **Case** $v \notin r_g^{(o)}(S)$ **but** $v \in r_g^{(o)}(T)$: We have that,

$$r_g^{(o)}(S \cup \{v\}) - r_g^{(o)}(S) = |r_g^{(o)}(\{v\}) \backslash (r_g^{(o)}(S) \cup S)| \geq 0, \tag{2.36}$$

while $r_g^{(o)}(T \cup \{v\}) - r_g^{(o)}(T) = -1$. Thus,

$$r_g^{(o)}(S \cup \{v\}) - r_g^{(o)}(S) > r_g^{(o)}(T \cup \{v\}) - r_g^{(o)}(T). \tag{2.37}$$

- **Case** $v \notin r_g^{(o)}(T)$: Since $\forall u \in r_g^{(o)}(S) \cup S$, we have either $u \in r_g^{(o)}(T)$ or $u \in T$ or $r_g^{(o)}(S) \cup S \subseteq r_g^{(o)}(T) \cup T$, and thus,

$$\begin{aligned} r_g^{(o)}(S \cup \{v\}) - r_g^{(o)}(S) &= |r_g^{(o)}(\{v\}) \backslash (r_g^{(o)}(S) \cup S)| \\ &\geq |r_g^{(o)}(\{v\}) \backslash (r_g^{(o)}(T) \cup T)| = r_g^{(o)}(T \cup \{v\}) - r_g^{(o)}(T). \end{aligned} \tag{2.38}$$

In all three cases, we have,

$$r_g^{(o)}(S \cup \{v\}) - r_g^{(o)}(S) \geq r_g^{(o)}(T \cup \{v\}) - r_g^{(o)}(T). \tag{2.39}$$

Applying Eq. 2.39 on all possible $g \sim \mathcal{G}$ and taking the sum over all of these inequalities give

$$\begin{aligned} &\sum_{g \sim \mathcal{G}} (r_g^{(o)}(S \cup \{v\}) - r_g^{(o)}(S)) \Pr[g \sim \mathcal{G}] \\ &\qquad \geq \sum_{g \sim \mathcal{G}} (r_g^{(o)}(T \cup \{v\}) - r_g^{(o)}(T)) \Pr[g \sim \mathcal{G}], \end{aligned}$$

or,

$$\mathbb{I}_{out}(S \cup \{v\}) - \mathbb{I}_{out}(S) \geq \mathbb{I}_{out}(T \cup \{v\}) - \mathbb{I}_{out}(T). \tag{2.40}$$

That completes the proof.

2.2.8.2 Proof of Lemma 4

Let Ω_W^+ be the probability space of all possible cascades from S. For any cascade $W^{(S)} \supseteq S$, the probability of that cascade in Ω_W^+ is given by

$$\Pr[W^{(S)} \in \Omega_W^+] = \sum_{g \in \Omega_\mathcal{G}, g \rightsquigarrow W^{(S)}} \Pr[g \in \Omega_\mathcal{G}],$$

where $g \rightsquigarrow W^{(S)}$ means that $W^{(S)}$ is the set of reachable nodes from S in g.

Let Ω_W be the probability space of non-trivial cascades. According to the Stage 1 in IICP, the probability of the trivial cascade is:

$$\Pr[S \in \Omega_W] = 0.$$

Comparing to the mass of cascades in Ω_W^+, the probability mass of the trivial cascade S in Ω_W is redistributed proportionally to other cascades in Ω_W. Specifically, according to line 2 in IICP, the probability mass of all the non-trivial cascades in Ω_W is multiplied by a factor $1/\beta_0$. Thus,

$$\Pr[W^{(S)} \in \Omega_W^+] = \Pr[W^{(S)} \in \Omega_W] \cdot \beta_0 \;\; \forall W^{(S)} \neq S.$$

It follows that

$$\mathbb{I}_{out}(S) = \sum_{W^{(S)} \in \Omega_W^+} |W^{(S)} \setminus S| \cdot \Pr[W^{(S)} \in \Omega_W^+] \tag{2.41}$$

$$= \sum_{W^{(S)} \in \Omega_W} |W^{(S)} \setminus S| \cdot \Pr[W^{(S)} \in \Omega^W] \beta_0 \tag{2.42}$$

$$= \mathcal{E}[|W^{(S)}|] \cdot \beta_0 = \mathcal{E}[Y^{(S)}] \cdot \beta_0. \tag{2.43}$$

We note that for $W^{(S)} = S$, $|W^{(S)} \setminus S| = 0$. Thus the difference in the probability masses between the two probabilistic spaces does not affect the 2nd step.

2.2.8.3 Proof of Theorem 2

We will equivalently prove two probabilistic inequalities:

$$\Pr[\hat{\mu}_X < (1-\epsilon)\mu_X] \leq \frac{\delta}{2}, \tag{2.44}$$

and

$$\Pr[\hat{\mu}_X > (1+\epsilon)\mu_X] \leq \frac{\delta}{2}. \tag{2.45}$$

Prove Eq. 2.44. We first realize that at termination point of Alg. 2, due to the stopping condition $h = \sum_{j=1}^{T} X_j \geq \Upsilon$ and $X_j \leq b, \forall j$, the following inequalities hold,

$$\Upsilon \leq \sum_{j=1}^{T} X_j \leq \Upsilon + b. \tag{2.46}$$

The left hand side of Eq. 2.44 is rewritten as follows,

$$\Pr[\hat{\mu}_X < (1-\epsilon)\mu_X] = \Pr\left[\frac{\sum_{j=1}^{T} X_j}{T} < (1-\epsilon)\mu_X\right] \tag{2.47}$$

$$= \Pr\left[\sum_{j=1}^{T} X_j < (1-\epsilon)\mu_X T\right] \tag{2.48}$$

$$\leq \Pr[\Upsilon < (1-\epsilon)\mu_X T]. \tag{2.49}$$

The last inequality is due to our realization in Eq. 2.46. Assume that $\epsilon < 1$ and $\mu_X > 0$, let denote $L_1 = \lceil \frac{\Upsilon}{(1-\epsilon)\mu_X} \rceil$. We then have,

$$L_1 \geq \frac{\Upsilon}{(1-\epsilon)\mu_X} \Rightarrow \frac{\Upsilon}{L_1} \leq (1-\epsilon)\mu_X, \tag{2.50}$$

and

$$L_1 > \frac{\Upsilon}{\mu_X} > (2 + \frac{2}{3}\epsilon)\ln(\frac{2}{\delta})\frac{1}{\epsilon'^2\mu_X}(b-a). \tag{2.51}$$

Thus, from Eq. 2.49, we obtain,

$$\Pr[\hat{\mu}_X < (1-\epsilon)\mu_X] \leq \Pr[L_1 \leq T] = \Pr\Big[\sum_{j=1}^{L_1} X_j \leq \sum_{j=1}^{T} X_j\Big]$$

$$\leq \Pr\Big[\sum_{j=1}^{L_1} X_j \leq \Upsilon + b\Big] \tag{2.52}$$

$$\leq \Pr\Big[\frac{\sum_{j=1}^{L_1} X_j}{L_1} \leq \frac{\Upsilon + b}{L_1}\Big], \tag{2.53}$$

where the second inequality is due to Eq. 2.46. Note that $\frac{\sum_{j=1}^{L_1} X_j}{L_1}$ is an estimate of μ_X using the first L_1 random variables $X_1, \ldots, X_{L_1}$. Furthermore, from Eq. 2.50 that $\frac{\Upsilon}{L_1} \leq (1-\epsilon)\mu_X$, we have,

$$\frac{\Upsilon + b}{L_1} \leq (1-\epsilon)\mu_X + \frac{b}{L_1} = (1 - \epsilon + \frac{b}{L_1\mu_X})\mu_X. \tag{2.54}$$

Since $L_1 > (2 + \frac{2}{3}\epsilon)\ln(\frac{2}{\delta})\frac{1}{\epsilon'^2\mu_X}(b-a)$ from Eq. 2.51,

$$\frac{\Upsilon + b}{L_1} \leq \Big(1 - \epsilon + \frac{\epsilon^2 b}{(2 + \frac{2}{3}\epsilon)\ln(\frac{2}{\delta})(b-a)}\Big)\mu_X = (1-\epsilon')\mu_X. \tag{2.55}$$

Plugging these into Eq. 2.53, we obtain,

$$\Pr[\hat{\mu}_X < (1-\epsilon)\mu_X] \leq \Pr\Big[\sum_{j=1}^{L_1} X_j \leq (1-\epsilon')\mu_X L_1\Big]. \tag{2.56}$$

Now, apply the Chernoff-like bound in Eq. 2.26 with $T = L_1$ and note that $L_1 > (2 + \frac{2}{3}\epsilon)\ln(\frac{2}{\delta})\frac{1}{\epsilon'^2\mu_X}(b-a) > 2\ln(\frac{2}{\delta})\frac{1}{\epsilon'^2\mu_X}(b-a)$, we achieve,

$$\Pr[\hat{\mu}_X < (1-\epsilon)\mu_X] \leq \exp\Big(-\frac{\epsilon'^2 L_1 \mu_X}{2(b-a)}\Big) \tag{2.57}$$

$$\leq \exp\Big(-\frac{\epsilon'^2 2\ln(\frac{2}{\delta})\frac{1}{\epsilon'^2\mu_X}(b-a)}{2(b-a)}\Big) = \frac{\delta}{2}. \tag{2.58}$$

That completes the proof of Eq. 2.44.

Prove Eq. 2.45. The left hand side of Eq. 2.45 is rewritten as follows,

$$\Pr[\hat{\mu}_X > (1+\epsilon)\mu_X] = \Pr\Big[\sum_{j=1}^{T} X_j > (1+\epsilon)\mu_X T\Big] \tag{2.59}$$

$$\leq \Pr[\Upsilon + b > (1+\epsilon)\mu_X T], \tag{2.60}$$

where the last inequality is due to our observation that $\sum_{j=1}^{T} X_j \leq \Upsilon + b$. Under the same assumption that $0 < \mu_X \leq \frac{b}{1+\epsilon}$, we denote $L_2 = \lfloor \frac{\Upsilon+b}{(1+\epsilon)\mu_X} \rfloor$. We then have,

$$L_2 \geq \frac{\Upsilon}{(1+\epsilon)\mu_X} = (2+\frac{2}{3}\epsilon)\ln(\frac{2}{\delta})\frac{1}{\epsilon'^2\mu_X}(b-a), \tag{2.61}$$

and

$$L_2 \leq \frac{\Upsilon+b}{(1+\epsilon)\mu_X} \Rightarrow \frac{\Upsilon+b}{L_2} \geq (1+\epsilon)\mu_X \tag{2.62}$$

$$\Rightarrow \frac{\Upsilon}{L_2} \geq (1+\epsilon)\mu_X - \frac{b}{L_2} = (1+\epsilon-\frac{b}{L_2\mu_X})\mu_X \tag{2.63}$$

$$\Rightarrow \frac{\Upsilon}{L_2} \geq \Big(1+\epsilon-\frac{\epsilon^2 b}{(2+\frac{2}{3}\epsilon)\ln(\frac{2}{\delta})(b-a)}\Big)\mu_X = (1+\epsilon')\mu_X \tag{2.64}$$

Thus, from Eq. 2.60, we obtain,

$$\Pr[\hat{\mu}_X > (1+\epsilon)\mu_X] \leq \Pr[L_2 \geq T] = \Pr\Big[\sum_{j=1}^{L_2} X_j \geq \sum_{j=1}^{T} X_j\Big]$$

$$\leq \Pr\Big[\sum_{j=1}^{L_2} X_j \geq \Upsilon\Big] = \Pr\Big[\frac{\sum_{j=1}^{L_2} X_j}{L_2} \geq \frac{\Upsilon}{L_2}\Big] \tag{2.65}$$

$$\leq \Pr\Big[\frac{\sum_{j=1}^{L_2} X_j}{L_2} \geq (1+\epsilon')\mu_X\Big] \tag{2.66}$$

where the last inequality follows from Eq. 2.64. By applying another Chenoff-like bound from Eq. 2.25 combined with the lower bound on L_2 in Eq. 2.61, we achieve,

$$\Pr[\hat{\mu}_X > (1+\epsilon)\mu_X] \leq \exp\big(-\frac{\epsilon'^2 L_2 \mu_X}{(2+\frac{2}{3}\epsilon)(b-a)}\big) = \frac{\delta}{2}, \tag{2.67}$$

which completes the proof of Eq. 2.45.

Follow the same procedure as in the proof of Eq. 2.45, we obtain the second statement in the theorem that,

$$\Pr[T \leq (1+\epsilon)\Upsilon/\mu_X] > 1 - \delta/2, \tag{2.68}$$

which completes the proof of the whole theorem.

More elaborations on the hold in [26]. The stopping rule algorithm in [26] is described in Alg. 6.

Algorithm 6: Stopping Rule Algorithm [26]

Input: Random variables $X_1, X_2, \ldots$ and $0 < \epsilon, \delta < 1$
Output: An (ϵ, δ)-approximate of $\mu_X = E[X_i]$
Compute: $\Upsilon_1 = 1 + (1+\epsilon)4(e-2)\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}$;
Initialize $h = 0, T = 0$;
while $h < \Upsilon_1$ **do**
  $h \leftarrow h + X_T, T \leftarrow T + 1$;
end
return $\hat{\mu}_X = \Upsilon_1/T$;

The algorithm first computes Υ_1 and then, generates samples X_j until the sum of their outcomes exceed Υ_1. Afterwards, it returns Υ_1/T as the estimate. Apparently, Υ_1/T is a biased estimate of μ_X since $\sum_{j=1}^{T} X_j \geq \Upsilon_1$.

An important realization for this algorithm from our proof of Theorem 2 is that $\Upsilon_1 \leq \sum_{j=1}^{T} X_j \leq \Upsilon_1 + b$ with $b = 1$ for $[0, 1]$ random variables. In section 5 of [26], following the proof of $\Pr[\hat{\mu}_X > (1+\epsilon)\mu_X] \leq \delta/2$ to prove $\Pr[\hat{\mu}_X < (1-\epsilon)\mu_X] \leq \delta/2$, there is step that derives as follows: $\Pr[L_1 \leq T] = \Pr\left[\sum_{j=1}^{L_1} X_j \leq \sum_{j=1}^{T} X_j\right] = \Pr\left[\sum_{j=1}^{L_1} X_j \leq \Upsilon_1\right]$ where L_1 is a predefined number, i.e. $L_1 = \lfloor \frac{\Upsilon_1}{(1-\epsilon)\mu_X} \rfloor$. However, since $\Upsilon_1 \leq \sum_{j=1}^{T} X_j \leq \Upsilon_1 + b$, the last equality does not hold. This is based on Eq. 2.52 with the correct expression being $\Pr\left[\sum_{j=1}^{L_1} X_j \leq \Upsilon + b\right]$ instead of $\Pr\left[\sum_{j=1}^{L_1} X_j \leq \Upsilon\right]$.

2.2.8.4 Proof of Theorem 3

[Proof of Part (1)] If $\epsilon \geq 1/4$, then RSA only runs GSRA and hence, from Theorem 2, the returned solution satisfies the precision requirement. Otherwise, since the first steps is literally applying GSRA with $\sqrt{\epsilon} < 1/2, \delta/3$, we have,

$$\Pr[\mu_X(1-\sqrt{\epsilon}) \leq \hat{\mu}'_X \leq \mu_X(1+\sqrt{\epsilon})] \geq 1-\delta/3 \tag{2.69}$$

We prove that in step 2, $\hat{\rho}_X \geq \rho_X/2$. Let define the random variables $\xi_i = (X'_{2i-1} - X'_{2i})^2/2, i = 1, 2, \ldots$ and thus, $\mathcal{E}[\xi_i] = \mathrm{Var}[X]$. Consider the following two cases.

1. **If** $\mathrm{Var}[X] \geq \epsilon\mu_X(b-a)$, consider two sub-cases:

   (a) If $\mathrm{Var}[X] \geq 2(1-\sqrt{\epsilon})\epsilon\mu_X(b-a)$, then since $N_\sigma = \Upsilon_2\epsilon/\hat{\mu}'_X \geq \frac{2}{1-\sqrt{\epsilon}}(1 + \ln(\frac{3}{2})/\ln(\frac{2}{\delta}))\Upsilon\epsilon/\mu_X$, applying the Chernoff-like bound in Eq. 2.24 gives,

   $$\Pr[\mathrm{Var}[X]/2 \leq \Delta/N_\sigma] \geq 1-\delta/3 \tag{2.70}$$

   Thus, $\hat{\rho}_X \geq \mathrm{Var}[X]/2 = \rho_X/2$ with a probability of at least $1-\delta/3$.

   (b) If $\mathrm{Var}[X] \leq 2(1-\sqrt{\epsilon})\epsilon\mu_X(b-a)$, then $\epsilon\mu_X(b-a) \geq \mathrm{Var}[X]/(2(1-\sqrt{\epsilon}))$ and therefore, $\hat{\rho}_X \geq \epsilon\hat{\mu}'_X(b-a) \geq (1-\sqrt{\epsilon})\epsilon\mu_X(b-a) \geq \mathrm{Var}_X/2 = \rho_X/2$.

2. **If** $\mathrm{Var}[X] \leq \epsilon\mu_X(b-a)$, it follows that $\hat{\rho}_X \geq \epsilon\hat{\mu}_X \geq \rho_X(1-\min\{\sqrt{\epsilon}, 1/2\})$ with probability at least $1-\delta/3$.

Thus, after steps 1 and 2, $2\frac{1+\sqrt{\epsilon}}{1-\sqrt{\epsilon}}\hat{\rho}_X/\hat{\mu}'^2_X \geq \rho_X/\mu_Z^2$ with probability at least $1-\delta/3$. In step 3, since $T = \Upsilon_2\hat{\rho}_X/(\hat{\mu}'^2_X(b-a)) \geq (1+\ln(\frac{3}{2})/\ln(\frac{2}{\delta}))\Upsilon\rho_X/(\mu_X^2(b-a))$ and hence, applying the Chernoff-like bound in Eq. 2.27 again gives,

$$\Pr[\mu_X(1-\epsilon) \leq \hat{\mu}_X \leq \mu_X(1+\epsilon)] \geq 1-2\delta/3. \tag{2.71}$$

Accumulating the probabilities, we finally obtain,

$$\Pr[\mu_X(1-\epsilon) \leq \hat{\mu}_X \leq \mu_X(1+\epsilon)] \geq 1-\delta, \tag{2.72}$$

This completes the proof of part (1).

[Proof of Part (2)] The RSA algorithm may fail to terminate after using $\mathcal{O}(\Upsilon\rho_X/(\mu_X^2(b-a)))$ samples if either:

1. The GSRA algorithm fails to return an $(\sqrt{\epsilon}, \delta/3)$-approximate $\hat{\mu}'_X$ with probability at most $\delta/2$, or,

2. In step 2, for $\mathrm{Var}[X] \leq 2(1-\sqrt{\epsilon})\epsilon\mu_X(b-a)$, $\hat{\rho}_X$ is not $\mathcal{O}(\epsilon\mu_X(b-a))$ with probability at most $\delta/2$.

From Theorem 2, with $T = (1+\epsilon)\Upsilon/\mu_X = \mathcal{O}(\Upsilon\rho_X/(\mu_X^2(b-a)))$, the first case happens with probability at most $\delta/2$. In addition, we can show similarly to Theorem 2 that if $\mathrm{Var}[X] \leq 2\epsilon\mu_X(b-a)$, then,

$$\Pr[\Delta/T \geq 4\epsilon\mu_X(b-a)] \leq \exp(-T\epsilon\mu_X(b-a)/2). \tag{2.73}$$

Thus, for $T \geq 2\Upsilon\epsilon/\mu_X$, we have $\Pr[\Delta/T \geq 4\epsilon\mu_X] \leq \delta/2$.

2.2.8.5 Proof of Lemma 6

We start with the computation of $\mathrm{Var}[Z^{(S)}]$ with a note that $\mathcal{E}[Z^{(S)}] = \mathbb{I}(S)$,

$$\begin{aligned}
\mathrm{Var}[Z^{(S)}] &= \sum_{z=\beta_0+|S|}^{(n-|S|)\beta_0+|S|} (z - \mathcal{E}[Z^{(S)}])^2 \Pr[Z^{(S)} = z] \\
&= \sum_{y=1}^{n-|S|} (y\beta_0 + |S| - \mathbb{I}(S))^2 \Pr[Y^{(S)} = y] \\
&= \sum_{y=1}^{n-|S|} (y\beta_0 - \mathbb{I}_{out}(S)\beta_0 + \mathbb{I}_{out}(S)\beta_0 + |S| - \mathbb{I}(S))^2 \Pr[Y^{(S)} = y] \\
&= \beta_0^2 \sum_{y=1}^{n-|S|} (y - \mathbb{I}_{out}(S))^2 \Pr[Y^{(S)} = y] \\
&\quad + \sum_{y=1}^{n-|S|} (\mathbb{I}_{out}(S)\beta_0 + |S| - \mathbb{I}(S))^2 \Pr[Y^{(S)} = y] \\
&\quad + 2\beta_0 \sum_{y=1}^{n-|S|} (y - \mathbb{I}_{out}(S))(\mathbb{I}_{out}(S)\beta_0 + |S| - \mathbb{I}(S)) \Pr[Y^{(S)} = y]
\end{aligned}$$

Since $Y^{(S)} \geq 1$ and $\Pr[Y^{(S)} = y] = \frac{\Pr[M^{(S)}=y+|S|]}{\beta_0}$, we have,

$$\begin{aligned}
\sum_{y=1}^{n-|S|} (y - \mathbb{I}_{out}(S))^2 \Pr[Y^{(S)} = y] &= \frac{1}{\beta_0} \sum_{m=1+|S|}^{n} (m - \mathcal{E}[M^{(S)}])^2 \Pr[M^{(S)} = m] \\
&= \frac{1}{\beta_0} \sum_{m=|S|}^{n} (m - \mathcal{E}[M^{(S)}])^2 \Pr[M^{(S)} = m] - \frac{1}{\beta_0}\mathbb{I}_{out}^2(S)(1-\beta_0) \\
&= \frac{1}{\beta_0}(\mathrm{Var}[M^{(S)}] - \mathbb{I}_{out}^2(S)(1-\beta_0)), \qquad (2.74)
\end{aligned}$$

and,

$$\sum_{y=1}^{n-|S|} \beta_0(y - \mathbb{I}_{out}(S))(\mathbb{I}_{out}(S)\beta_0 + |S| - \mathbb{I}(S)) \Pr[Y^{(S)} = y]$$
$$= (\mathbb{I}_{out}(S)\beta_0 + |S| - \mathbb{I}(S)) \sum_{y=1}^{n-|S|} (y - \mathbb{I}_{out}(S)) \Pr[Y^{(S)} = y]$$
$$= (\mathbb{I}_{out}(S)\beta_0 + |S| - \mathbb{I}(S))\mathbb{I}_{out}(S)(1/\beta_0 - 1). \tag{2.75}$$

Plug these back in the $\mathrm{Var}[Z^{(S)}]$, we obtain,

$$\begin{aligned}\mathrm{Var}[Z^{(S)}] &= \beta_0(\mathrm{Var}[M^{(S)}] - \mathbb{I}^2_{out}(S)(1-\beta_0)) + (\mathbb{I}_{out}(S)\beta_0 + |S| - \mathbb{I}(S))^2 \\ &\quad + 2\beta_0(\mathbb{I}_{out}(S)\beta_0 + |S| - \mathbb{I}(S))\mathbb{I}_{out}(S)(1/\beta_0 - 1) \\ &= \beta_0 \cdot \mathrm{Var}[M^{(S)}] - (1-\beta_0)\mathbb{I}^2_{out}(S) \end{aligned} \tag{2.76}$$

That completes the computation.

2.3 Importance Sketching for Influence Estimation in Billion-scale Networks

Summary of contribution:

- At the central of our sketch is an importance sampling algorithm to sample non-singular cascades (Alg. 7). For simplicity, we first present the sketch and its sampling algorithm using the popular *independent cascade* model [55], and later extend them to other diffusion models.
- We provide general frameworks to apply SKIS for existing algorithms for the influence estimation and influence maximization problems. We provide theoretical analysis to show that using SKIS leads to improved influence estimation oracle due to smaller sample variances and better concentration bounds; and that the state-of-the-art methods for influence maximization like D-SSA [84], IMM [105], and, TIM+/TIM++[106] can also immediately benefit from our new sketch.

- We conduct comprehensive empirical experiments to demonstrate the effectiveness of our sketch in terms of quality, memory and computational time. Using SKIS, we can design high-quality influence oracle for seed set with average estimation error up to 10x times smaller than those using RIS and 6x times those using SKIM. In addition, our influence maximization using SKIS substantially improves the quality of solutions for greedy algorithms. It achieves up to 10x times speed-up and 4x memory reduction for the fastest RIS-based D-SSA algorithm, while maintaining the same theoretical guarantees.

2.3.1 Preliminaries

Frequently used notations are summarized in Table 24.

Table 7.: Table of notations

Notation	Description
n, m	#nodes, #edges of graph $\mathcal{G} = (V, E, w)$.
$\mathbb{I}(S), \hat{\mathbb{I}}(S)$	Expected Influence of $S \subseteq V$ and an estimate.
$N^{in}(S)$	Set of in-neighbor nodes of S.
γ_v, Γ	$\gamma_v = 1 - \Pi_{u \in N^{in}(v)}(1 - w(u, v)); \Gamma = \sum_{v \in V} \gamma_v$.
γ_0	$\gamma_0 = \sum_{v \in V} \gamma_v / n$.
$R_j, \mathcal{R}$	A random ROIS sample and a SKIS sketch.
$\text{Cov}_{\mathcal{R}}(S)$	$\text{Cov}_{\mathcal{R}}(S) = \lvert R_j \in \mathcal{R} \rvert \mathcal{R}_j \cap S \neq \emptyset \rvert$.

2.3.2 Influence Estimation/Maximization Problems

We describe the tasks of Influence Estimation and Maximization which are used to evaluate sketches' efficiency.

Definition 4 (Influence Estimation (IE)). *Given a probabilistic graph $\mathcal{G}$ and a seed set of nodes $S \subseteq V$, the IE problem asks for a close estimation $\hat{\mathbb{I}}(S)$ of the influence spread $\mathbb{I}(S)$.*

Definition 5 (Influence Maximization (IM) [55])**.** *Given a probabilistic graph $\mathcal{G}$, a budget k, the IM problem asks to identify a subset $S_k \subset V$ with the maximum influence among all subsets of size at most k,*

$$S_k = \arg\max_{S \subseteq V, |S| \leq k} \mathbb{I}(S). \tag{2.77}$$

2.3.3 Sketch-based Methods for IE/IM

We summarize and analyze below the two existing sketch-based approaches for IE/IM.

2.3.3.1 Reverse Influence Sketch (RIS)

Essentially, a random RIS sample, denoted by R_j, contains a *random* set of nodes, following a diffusion model, that can influence a *randomly selected* source node, denoted by $\mathsf{src}(R_j)$. A RIS sample is generated in three steps:

1) Select a random node $v \in V$ which serves as $\mathsf{src}(R_j)$.
2) Generate a sample graph $g \sim \mathcal{G}$.
3) Return the set R_j of nodes that can reach v in g.

Thus, the probability of generating a particular RIS sample R_j can be computed based on the source selection and the sample graphs that has R_j as the set of nodes that reach $\mathsf{src}(R_j)$ in g. Let denote such set of nodes that can reach to a node v in sample graph g by $\eta_g^-(v)$. We have,

$$\Pr[R_j] = \frac{1}{n} \sum_{g, \eta_g^-(\mathsf{src}(R_j)) = R_j} \Pr[g]. \tag{2.78}$$

The key property of RIS samples for influence estimation/maximization is stated in the following lemma.

Lemma 7 ([11])**.** *Given a random RIS sample R_j generated from $\mathcal{G} = (V, E, w)$, for a set*

$S \subseteq V$ of nodes, we have,

$$\mathbb{I}(S) = n \cdot \Pr[R_j \cap S \neq \emptyset]. \tag{2.79}$$

Thus, estimating/maximizing $\mathbb{I}(S)$ is equivalent to estimating/maximizing the probability $\Pr[R_j \cap S \neq \emptyset]$.

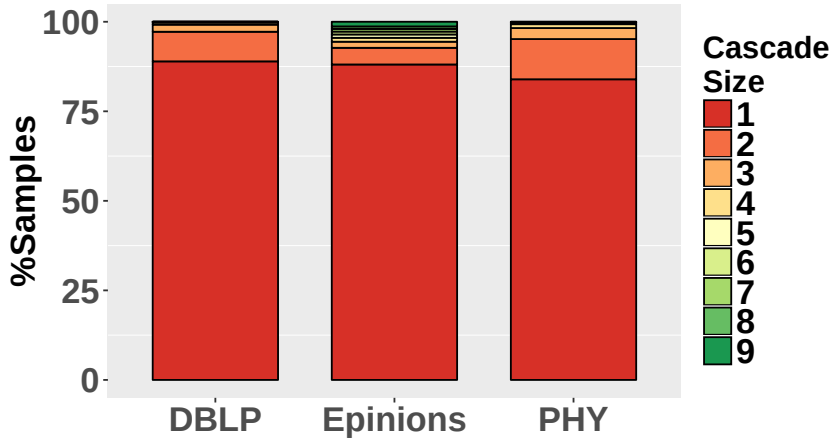
Using RIS samples for IE/IM. Thanks to Lemma 7, a general strategy for IE/IM is generating a set of RIS samples, then returning an empirical estimate of $\Pr[R_j \cap S \neq \emptyset]$ on generated samples for IE or the set $\hat{S}_k$ that intersects with most samples for IM. The strong advantage of RIS is the reuse of samples to estimate influence of any seed set $S \subseteq V$. Ohsaka et al. [92] build a query system to answer influence queries. [**Nguyen173**, 11, 106, 105, 84] recently use RIS samples in solving Influence Maximization problem with great successes, i.e. handling large networks with tens of millions of nodes and billions of edges.

2.3.3.2 Combined Reachability Sketch (SKIM)

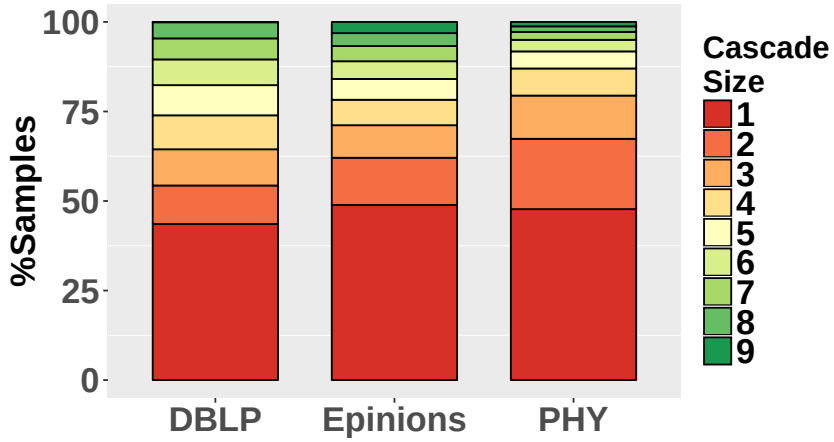
Cohen et al. [25] proposed the combined reachability sketch which can be used to estimate influences of multiple seed sets. Each node u in the network is assigned a combined reachability sketch which is a bottom-k min-hash sketch of the set of nodes reachable from u in l sample graphs. [25] generates l sample graphs g of $\mathcal{G}$, i.e. $l = 64$ by default, and build a combined reachability sketch of size k for each node.

The influence estimate of a seed set S is computed by taking the bottom-k sketch of the union over all the sketches of nodes in S and applying the cardinality estimator [25]. Using the sketches, the solution for IM is found by following the greedy algorithm which repeatedly adds a node with highest marginal influence into the solution. Here, the marginal influences are similarly estimated from node sketches.

Common Shortcomings. According to recent benchmarks [**Arora17**] and our own empirical evaluations (details in Section 2.3.8), both RIS and SKIM yield significant in-



(a) Trivalency (TRI)



(b) Weighted Cascade (WC)

Fig. 7.: Distribution of reversed cascade sizes on various real-world networks with two popular different edge weight models: Trivalency (TRI) and Weighted Cascade (WC). The majority of the cascades are singular.

fluence estimation errors. For RIS, it is due to the fact that the majority of RIS samples contain only their sources as demonstrated in Figure 7 with up to 86% of such RIS samples overall. These samples, termed *singular*, harm the performance in two ways: 1) they do not contribute to the influence computation of other seed sets than the ones that contain the sources, however, the contribution is known in advance, i.e. number of seed nodes; 2)

these samples magnify the variance of the RIS-based random variables used in estimation causing high errors.

This motivates our Importance Sketching techniques to generate only *non-singular* samples (containing more nodes than just the source) that are useful for influence estimations of many seed sets (not just ones with the sources).

2.3.4 Importance Sketching

This section introduces our core construction of *Importance Sketching* algorithm to generate random *non-singular* samples with probabilities proportional to those in the original sample space of reverse influence samples and normalized by the probability of generating a non-singular one.

Algorithm 7: Importance Influence Sampling (ROIS) Alg.

Input: Graph $\mathcal{G} = (V, E, w)$
Output: R_j - A random ROIS sample
Pick a node $v \in V$ as the source with probability in Eq. 2.82;
Select an in-neighbor u_i of v, $u_i \in N^{in}(v)$, with probability of selecting u_i given in Eq. 2.83;
Initialize a queue $Q = \{u_i\}$ and $R_j = \{v, u_i\}$;
foreach $u_t \in N^{in}(v), t > i$ **do**
    With probability $w(u_t, v)$:
        Q.push(u_t); $R_j \leftarrow R_j \cup \{u_t\}$;
end
while Q *is not empty* **do**
    $v = Q$.pop$()$
    foreach $u \in N^{in}(v) \backslash R_j$ **do**
        With probability $w(u, v)$:
            Q.push(u); $R_j \leftarrow R_j \cup \{u\}$;
    end
end
return R_j;

2.3.4.1 Importance Influence Sampling (ROIS)

Sample Spaces and Desired Property. Let Ω_{RIS} be the sampling space of reverse influence samples (RIS) with probability $\Pr[R_j \in \Omega_{\mathsf{RIS}}]$ of generating sample R_j. Let Ω_{SKIS} be a subspace of Ω_{RIS} and corresponds to the space of *only* non-singular reverse influence samples in Ω_{RIS}. Since Ω_{SKIS} is a subspace of Ω_{RIS}, the probability $\Pr[R_j \in \Omega_{\mathsf{SKIS}}]$ of generating a non-singular sample from Ω_{SKIS} is larger than that from Ω_{RIS}. Specifically, for a node $v \in V$, let γ_v be the probability of generating a non-singular sample if v is selected as the source and $\Gamma = \sum_{v \in V} \gamma_v$. Then, since the sample sources are selected randomly, the ratio of generating a non-singular sample to generating any sample in Ω_{RIS} is $\frac{\Gamma}{n}$ and thus, the probability $\Pr[R_j \in \Omega_{\mathsf{SKIS}}]$ is as follows,

$$\Pr[R_j \in \Omega_{\mathsf{SKIS}}] = \frac{n}{\Gamma} \Pr[R_j \in \Omega_{\mathsf{RIS}}]. \tag{2.80}$$

Our upcoming ROIS algorithm aims to achieve this desired property of sampling non-singular samples from Ω_{SKIS}.

Sampling Algorithm. Our Importance Influence Sampling (ROIS) scheme involves three core components:

1) *Probability of having a non-singular sample.* For a node $v \in V$, a sample with source v is singular if no in-neighbor of v is selected, that happens with probability $\Pi_{u \in N^{in}(v)}(1 - w(u, v))$. Hence, the probability of having a non-singular sample from a node v is the complement:

$$\gamma_v = 1 - \Pi_{u \in N^{in}(v)} \left(1 - w(u, v)\right). \tag{2.81}$$

2) *Source Sampling Rate.* Note that the set of non-singular samples is just a subset of all possible samples and we want to generate uniformly random samples from that subset. Moreover, each node v has a probability γ_v of generating a non-singular

sample from it. Thus, in order to generate a random sample, we select v as the source with probability $\Pr[\mathsf{src}(R_j) = v]$ computed as follows,

$$\Pr[\mathsf{src}(R_j) = v] = \frac{\gamma_v}{\sum_{u \in V} \gamma_u} = \frac{\gamma_v}{\Gamma}, \tag{2.82}$$

where $\Gamma = \sum_{u \in V} \gamma_u$, and then generate a uniformly random non-singular sample from the specific source v as described in the next component.

3) *Sample a non-singular sample from a source.* From the $\mathsf{src}(R_j) = v$, we generate a non-singular sample R_j from v uniformly at random. Let $N^{in}(v) = \{u_1, u_2, \ldots, u_l\}$ be a fixed-order set of in-neighbors of v. We divide the all possible non-singular samples from v into l buckets: bucket $B_i, 1 \leq i \leq l$ contains those samples that have the first node from $N^{in}(v)$ being u_i. That means all the nodes $u_1, \ldots, u_{i-1}$ are not in the sample but u_i is in for certain. The other nodes from u_{i+1} to u_l may appear and will be sampled following the normal RIS sampling. Now we select the bucket that R_j belongs to with the probability of selecting B_i being as follows,

$$\Pr[\text{select } B_i] = \frac{w(u_i, v) \prod_{t=1}^{i-1} (1 - w(u_t, v))}{\gamma_v}. \tag{2.83}$$

For $i = 1$, we have $\Pr[\text{select } B_1] = w(u_1, v)$. Note that $\sum_{i=1}^{l} \Pr[\text{select } B_i] = 1$. Assume bucket B_i is selected and, thus, node u_i is added as the second node besides the source into R_j. For each other node $u_t, t \neq i$, u_t is selected into R_j with probability $w(u_t, v)$ following the ordinary RIS for the IC model.

These three components guarantee a non-singular sample. The detailed description of ROIS sampling is in Alg. 7. The first step selects the source of the ROIS sample among V. Then, the first incoming node to the source v is picked (Line 2) following the above description of the component 3). Each of the other incoming neighbors also tries to influence the source (Lines 4-6). The rest performs similarly as in RIS [11]. That is

for each newly selected node, its incoming neighbors are randomly added into the sample with probabilities equal to their edge weights. It continues until no newly selected node is observed. Note that Line 3 only adds the selected neighbors u_i of v into Q but adds both v and u_i to R_j. The loop from Lines 7-11 mimics the BFS-like sampling procedure of RIS.

Let $\Pr[R_j]$ be the probability of generating a non-singular sample R_j using ROIS algorithm. We have

$$\begin{aligned}
\Pr[R_j] &= \sum_{v \in V} \Pr[\mathsf{src}(R_j) = v] \Pr[\text{generate } R_j \text{ from } v] \\
&= \sum_{v \in V} \frac{\gamma_v}{\Gamma} \frac{\Pr[R_j \in \Omega_{\mathsf{RIS}} \text{ and } \mathsf{src}(R_j) = v]}{\gamma_v} \\
&= \frac{n}{\Gamma} \sum_{v \in V} \frac{1}{n} \Pr[R_j \in \Omega_{\mathsf{RIS}} \text{ and } \mathsf{src}(R_j) = v] \\
&= \frac{n}{\Gamma} \Pr[R_j \in \Omega_{\mathsf{RIS}}] = \Pr[R_j \in \Omega_{\mathsf{SKIS}}],
\end{aligned}$$

where $\Pr[\text{generate } R_j \text{ from } v] = \frac{\Pr[R_j \in \Omega_{\mathsf{RIS}} \text{ and } \mathsf{src}(R_j) = v]}{\gamma_v}$ due to the selection of the bucket that R_j belongs to in ROIS. Thus, the output R_j of ROIS is an random sample from non-singular space Ω_{SKIS} and we obtain the following lemma.

Lemma 8. *Recall that Ω_{SKIS} is the sample space of non-singular reverse influence samples. ROIS algorithm generates a random non-singular sample from sample space Ω_{SKIS}.*

Connection between IIS Samples and Influences. We establish the following key lemma that connects our ROIS samples with the influence of any seed set S.

Lemma 9. *Given a random ROIS sample R_j generated by Alg. 7 from the graph $\mathcal{G} = (V, E, w)$, for any set $S \subseteq V$, we have,*

$$\mathbb{I}(S) = \Pr[R_j \cap S \neq \emptyset] \cdot \Gamma + \sum_{v \in S} (1 - \gamma_v), \tag{2.84}$$

where γ_v and Γ are defined in Eqs. 2.81 and 2.82.

The proof is presented in our extended version [**extended**]. The influence $\mathbb{I}(S)$ of any set S comprises of two parts: 1) $\Pr[R_j \cap S \neq \emptyset] \cdot \Gamma$ depends on the randomness of R_j and 2) the fixed amount $\sum_{v \in S}(1 - \gamma_v)$ that is inherent to set S and accounts for the contribution of singular samples in Ω_{RIS} to the influence $\mathbb{I}(S)$. Lemma 9 states that instead of computing or estimating the influence $\mathbb{I}(S)$ directly, we can equivalently compute or estimate $\Pr[R_j \cap S \neq \emptyset] \cdot \Gamma + \sum_{v \in S}(1 - \gamma_v)$ using ROIS samples.

Remark: Notice that we can further generate samples of larger sizes and reduce the variance as shown later, however, the computation would increase significantly.

2.3.5 Influence Oracle via IIS Sketch (SKIS)

We use ROIS sampling to generate a sketch for answering influence estimation queries of different node sets. We show that the random variables associated with our samples have much smaller variances than that of RIS, and hence, lead to better concentration or faster estimation with much fewer samples required to achieve the same or better quality.

SKIS-based Influence Oracle. An SKIS sketch $\mathcal{R}$ is a collection of ROIS samples generated by Alg. 7, i.e. $\mathcal{R} = \{R_1, \ldots, R_T\}$. As shown in Lemma 9, the influence $\mathbb{I}(S)$ can be estimated through estimating the probability $\Pr[R_j \cap S \neq \emptyset]$. Thus, from a SKIS sketch $\mathcal{R} = \{R_1, \ldots, R_T\}$, we can obtain an estimate $\hat{\mathbb{I}}(S)$ of $\mathbb{I}(S)$ for any set S by,

$$\hat{\mathbb{I}}_{\mathcal{R}}(S) = \frac{\mathrm{Cov}_{\mathcal{R}}(S)}{|\mathcal{R}|} \cdot \Gamma + \sum_{v \in S} (1 - \gamma_v)\,, \tag{2.85}$$

where $\mathrm{Cov}_{\mathcal{R}}(S)$ is coverage of S on $\mathcal{R}$, i.e.,

$$\mathrm{Cov}_{\mathcal{R}}(S) = |\{R_j \in \mathcal{R} | R_j \cap S \neq \emptyset\}|. \tag{2.86}$$

We build an SKIS-based oracle for influence queries by generating a set $\mathcal{R}$ of T ROIS samples in a preprocessing step and then answer influence estimation query $\hat{\mathbb{I}}_{\mathcal{R}}(S)$ for any requested set S (Alg. 8). In the following, we show the better estimation quality of our

Algorithm 8: SKIS-based Influence Oracle

Input: Graph $\mathcal{G} = (V, E, w)$

Preprocessing: Generate a SKIS sketch $\mathcal{R} = \{R_1, \ldots, R_T\}$ of ROIS samples using Alg. 7.

For any influence query for any set S: return $\hat{\mathbb{I}}_{\mathcal{R}}(S)$ (Eq. 4.34).

sketch through analyzing the variances and estimating concentration properties.

SKIS Random Variables for Estimations. For a random ROIS sample R_j and a set S, we define random variables:

$$X_j(S) = \begin{cases} 1 & \text{if } R_j \cap S \neq \emptyset \\ 0 & \text{otherwise.} \end{cases}, \text{and} \tag{2.87}$$

$$Z_j(S) = \frac{X_j(S) \cdot \Gamma + \sum_{v \in S}(1 - \gamma_v)}{n}. \tag{2.88}$$

Then, the means of $X_j(S)$ and $Z_j(S)$ are as follows,

$$\mathcal{E}[X_j(S)] = \Pr[R_j \cap S \neq \emptyset] = \frac{\mathbb{I}(S) - \sum_{v \in S}(1 - \gamma_v)}{\Gamma} \tag{2.89}$$

$$\mathcal{E}[Z_j(S)] = \mathcal{E}[X_j(S)] \cdot \frac{\Gamma}{n} + \frac{\sum_{v \in S}(1 - \gamma_v)}{n} = \frac{\mathbb{I}(S)}{n}. \tag{2.90}$$

Hence, we can construct a corresponding set of random variables $Z_1(S), Z_2(S), \ldots, Z_T(S)$ by Eqs. 2.87 and 2.88. Then, $\hat{\mathbb{I}}_{\mathcal{R}}(S) = \frac{n}{T} \sum_{j=1}^{T} Z_j(S)$ is an empirical estimate of $\mathbb{I}(S)$ based on the SKIS sketch $\mathcal{R}$.

For comparison purposes, let $Y_j(S)$ be the random variable associated with RIS sample Q_j in a RIS sketch $\mathcal{Q}$,

$$Y_j(S) = \begin{cases} 1 & \text{if } Q_j \cap S \neq \emptyset \\ 0 & \text{otherwise.} \end{cases} \tag{2.91}$$

From Lemma 7, the mean value of $Y_j(S)$ is then,

$$\mathcal{E}[Y_j(S)] = \frac{\mathbb{I}(S)}{n}. \tag{2.92}$$

Variance Reduction Analysis. We show that the variance of $Z_j(S)$ for SKIS is much smaller than that of $Y_j(S)$ for RIS. The variance of $Z_j(S)$ is stated in the following.

Lemma 10. *The random variable $Z_j(S)$ (Eq. 2.88) has*

$$\begin{aligned}\mathrm{Var}[Z_j(S)] = &\frac{\mathbb{I}(S)}{n}\frac{\Gamma}{n} - \frac{\mathbb{I}^2(S)}{n^2} \\ &- \frac{\sum_{v\in S}(1-\gamma_v)}{n^2}(\Gamma + \sum_{v\in S}(1-\gamma_v) - 2\mathbb{I}(S)).\end{aligned} \tag{2.93}$$

Since the random variables $Y_j(S)$ for RIS samples are Bernoulli and $\mathcal{E}[Y_j(S)] = \frac{\mathbb{I}(S)}{n}$, we have $\mathrm{Var}[Y_j(S)] = \frac{\mathbb{I}(S)}{n}(1 - \frac{\mathbb{I}(S)}{n})$. Compared with $\mathrm{Var}[Z_j(S)]$, we observe that since $\frac{\Gamma}{n} \leq 1$, $\frac{\mathbb{I}(S)}{n}\frac{\Gamma}{n} - \frac{\mathbb{I}^2(S)}{n^2} \leq \frac{\mathbb{I}(S)}{n} - \frac{\mathbb{I}^2(S)}{n^2} = \mathrm{Var}[Y_j(S)]$,

$$\begin{aligned}\mathrm{Var}[Z_j(S)] \leq &\mathrm{Var}[Y_j(S)] \\ &- \frac{\sum_{v\in S}(1-\gamma_v)}{n^2}(\Gamma + \sum_{v\in S}(1-\gamma_v) - 2\mathbb{I}(S)).\end{aligned}$$

In practice, most of seed sets have small influences, i.e. $\mathbb{I}(S) \ll \frac{\Gamma}{2}$, thus, $\Gamma + \sum_{v\in S}(1-\gamma_v) - 2\mathbb{I}(S) \gg 0$. Hence, $\mathrm{Var}[Z_j(S)] < \mathrm{Var}[Y_j(S)]$ holds for most seed sets S.

Better Concentrations of SKIS Random Variables. Observe that $Z_j(S) \in \left[\frac{\sum_{v\in S}(1-\gamma_v)}{n}, \frac{\Gamma+\sum_{v\in S}(1-\gamma_v)}{n}\right]$ we obtain another result on the variance of $Z_j(S)$ as follows.

Lemma 11. *The variance of random variable $Z_j(S)$ satisfies*

$$\mathrm{Var}[Z_j(S)] \leq \frac{\mathbb{I}(S)}{n}\frac{\Gamma}{n}. \tag{2.94}$$

Using the above result with the general form of Chernoff's bound of Lemma 2 in [105], we derive the following concentration inequalities for random variables $Z_j(S)$ of SKIS samples.

Lemma 12. *Given a SKIS sketch $\mathcal{R} = \{R_1, \ldots, R_T\}$ with random variables $Z_1(S), \ldots, Z_T(S)$, we have,*

$$\Pr\Big[\frac{\sum_{j=1}^{T} Z_j(S)}{T} n - \mathbb{I}(S) \geq \epsilon \mathbb{I}(S)\Big] \leq \exp\Big(\frac{-\epsilon^2 T}{2\frac{\Gamma}{n} + \frac{2}{3}\epsilon} \frac{\mathbb{I}(S)}{n}\Big)$$
$$\Pr\Big[\frac{\sum_{j=1}^{T} Z_j(S)}{T} n - \mathbb{I}(S) \leq -\epsilon \mathbb{I}(S)\Big] \leq \exp\Big(\frac{-\epsilon^2 T}{2\frac{\Gamma}{n}} \frac{\mathbb{I}(S)}{n}\Big).$$

Compared with the bounds for RIS sketch in Corollaries 1 and 2 in [105], the above concentration bounds for SKIS sketch (Lemma 12) are stronger, i.e. tighter. Specifically, we have the factor $\frac{\Gamma}{n}$ (note that $\frac{\Gamma}{n} \ll 1$ in most practical scenarios) in the denominator of the $\exp(.)$ function while for RIS random variables, it is simply 1.

Sufficient Size of SKIS Sketch for High-quality Estimations. There are multiple strategies to determine the number of ROIS samples generated in the preprocessing step. For example, [92] generates samples until total size of all samples reaches $O(\frac{1}{\epsilon^3}(n+m)\log(n))$. Generating ROIS samples to reach such a specified threshold is vastly faster than using RIS due to the bigger size of ROIS samples. This method provides an *additive* estimation error guarantee within ϵ. Alternatively, by Lemma 12, we derive the sufficient number of ROIS samples to provide the more preferable (ϵ, δ)-estimation of $\mathbb{I}(S)$.

Lemma 13. *Given a set S, $\epsilon, \delta \geq 0$, if the SKIS sketch $\mathcal{R}$ has at least $(2\frac{\Gamma}{n} + \frac{2}{3}\epsilon)\ln(\frac{2}{\delta})\frac{n}{\mathbb{I}(S)}\epsilon^{-2}$ ROIS samples, $\hat{\mathbb{I}}_{\mathcal{R}}(S)$ is an (ϵ, δ)-estimate of $\mathbb{I}(S)$, i.e.,*

$$\Pr[(1-\epsilon)\mathbb{I}(S) \leq \hat{\mathbb{I}}_{\mathcal{R}}(S) \leq (1+\epsilon)\mathbb{I}(S)] \geq 1 - \delta. \tag{2.95}$$

In practice, $\mathbb{I}(S)$ is unknown in advance and a lower-bound of $\mathbb{I}(S)$, e.g. $|S|$, can be used to compute the necessary number of samples to provide the same guarantee. Compared to RIS with weaker concentration bounds, we save a significant factor of $O(\frac{\Gamma}{n})$.

2.3.6 SKIS-based IM Algorithms

With the help of SKIS sketch that is better in estimating the influences compared to the well-known successful RIS, we can largely improve the efficiency of IM algorithms in the broad class of RIS-based methods, i.e. RIS [11], TIM/TIM+ [106], IMM [105], BCT [80, 85], SSA/DSSA [84]. This improvement is possible since these methods heavily rely on the concentration of influence estimations provided by RIS samples.

SKIS-based framework. Let $\mathcal{R} = \{R_1, R_2, \dots\}$ be a SKIS sketch of ROIS samples. $\mathcal{R}$ gives an influence estimate

$$\hat{\mathbb{I}}_{\mathcal{R}}(S) = \frac{\mathrm{Cov}_{\mathcal{R}}(S)}{|\mathcal{R}|} \cdot \Gamma + \sum_{v \in S} (1 - \gamma_v), \tag{2.96}$$

for any set S. Thus, instead of optimizing over the exact influence, we can intuitively find the set S to maximize the estimation function $\hat{\mathbb{I}}(S)$. Then, the framework of using SKIS sketch to solve IM problem contains two main steps:

1) Generate a SKIS sketch $\mathcal{R}$ of ROIS samples,

2) Find the set S_k that maximizes the function $\hat{\mathbb{I}}_R(S)$ and returning S_k as the solution for the IM instance.

There are two essential questions related to the above SKIS-based framework : 1) Given a SKIS sketch $\mathcal{R}$ of ROIS samples, how to find S_k of k nodes that maximizes $\hat{\mathbb{I}}_{\mathcal{R}}(S_k)$ (in Step 2)? 2) How many ROIS samples in the SKIS sketch $\mathcal{R}$ (in Step 1) are sufficient to guarantee a high-quality solution for IM?

We give the answers for the above questions in the following sections. Firstly, we adapt the gold-standard greedy algorithm to obtain an $(1 - (1 - 1/k)^k)$-approximate solution over a given SKIS sketch. Secondly, we adopt recent techniques on RIS with strong solution guarantees to SKIS sketch to find an overall satisfactory solution.

Algorithm 9: Greedy Algorithm on SKIS sketch

Input: SKIS sketch $\mathcal{R}$ and k
Output: An $(1-(1-1/k)^k)$-approximate seed set $\hat{S}_k$
$\hat{S}_k = \emptyset$
for $i = 1 : k$ **do**
  $\hat{v} \leftarrow \arg\max_{v \in V \setminus \hat{S}_k} \left(\frac{\mathrm{Cov}_{\mathcal{R}}(S \cup \{v\}) - \mathrm{Cov}_{\mathcal{R}}(S)}{|\mathcal{R}|} \Gamma + (1 - \gamma_v) \right)$
  Add $\hat{v}$ to $\hat{S}_k$
end
return $\hat{S}_k$

2.3.6.1 Greedy Algorithm on SKIS Sketches

Let consider the optimization problem of finding a set S_k of at most k nodes to maximize the function $\hat{\mathbb{I}}_{\mathcal{R}}(S)$ on a SKIS sketch $\mathcal{R}$ of ROIS samples under the cardinality constraint $|S| \leq k$. The function $\hat{\mathbb{I}}_{\mathcal{R}}(S)$ is monotone and submodular since it is the weighted sum of a set coverage function $\mathrm{Cov}_{\mathcal{R}}(S)$ and a linear term $\sum_{v \in S}(1 - \gamma_v)$. Thus, we obtain the following lemma with the detailed proof in the appendix.

Lemma 14. *Given a set of ROIS samples $\mathcal{R}$, the set function $\hat{\mathbb{I}}_{\mathcal{R}}(S)$ defined in Eq. 2.96 is monotone and submodular.*

Thus, a standard greedy scheme [81], which iteratively selects a node with highest marginal gain, gives an $(1-(1-\frac{1}{k})^k)$, that converges to $(1-1/e)$ asymptotically, approximate solution $\hat{S}_k$. The marginal gain of a node v with respect to a set S on SKIS sketch $\mathcal{R}$ is defined as follows,

$$\mathsf{gain}_{\mathcal{R}}(v, S) = \Delta_{\mathcal{R}}(v, \hat{S}_k)\Gamma/|\mathcal{R}| + (1 - \gamma_v), \tag{2.97}$$

where $\Delta_{\mathcal{R}}(v, S) = \mathrm{Cov}_{\mathcal{R}}(S \cup \{v\}) - \mathrm{Cov}_{\mathcal{R}}(S)$ is called the marginal coverage gain of v w.r.t. S on SKIS sketch $\mathcal{R}$.

Given a collection of ROIS samples $\mathcal{R}$ and a budget k, the Greedy algorithm is presented in Alg. 11 with a main loop (Lines 2-4) of k iterations. Each iteration picks a node

$\hat{v}$ having largest marginal gain (Eq. 2.97) with respect to the current partial solution $\hat{S}_k$ and adds it to $\hat{S}_k$. The approximation guarantee of the Greedy algorithm (Alg. 11) is stated below.

Lemma 15. *The Greedy algorithm (Alg. 11) returns an* $(1-(1-\frac{1}{k})^k)$*-approximate solution* $\hat{S}_k$*,*

$$\hat{\mathbb{I}}_{\mathcal{R}}(\hat{S}_k) \geq (1-(1-1/k)^k)\hat{\mathbb{I}}_{\mathcal{R}}(S^*_{\mathcal{R}}), \tag{2.98}$$

where $S^*_{\mathcal{R}}$ *is the optimal cover set of size* k *on sketch* $\mathcal{R}$*.*

The lemma is derived directly from the $1-(1-\frac{1}{k})^k$ approximation factor of the ordinary greedy algorithm [81].

2.3.6.2 Sufficient Size of SKIS Sketch for IM

Since the SKIS sketch offers a similar greedy algorithm with approximation ratio $(1-(1-1/k)^k)$ to the traditional RIS, we can combine SKIS sketch with any RIS-based algorithm, e.g. RIS[11], TIM/TIM+[106], IMM[105], BCT[85], SSA/DSSA[84]. We discuss the adoptions of two most recent and scalable algorithms IMM[105] and SSA/DSSA[84].

IMM+SKIS. Tang et al. [105] provide a theoretical threshold

$$\theta_{\mathsf{RIS}} = O\Big((\log\binom{n}{k} + \log\delta^{-1})\frac{n}{\mathsf{OPT}_k}\epsilon^{-2}\Big) \tag{2.99}$$

on the number of RIS samples to guarantee an $(1-1/e-\epsilon)$-approximate solution for IM problem with probability $1-\delta$.

Replacing RIS with ROIS samples to build a SKIS sketch enables us to use the better bounds in Lemma 12. Following [105], we reduce the threshold of ROIS samples to,

$$\theta_{\mathsf{SKIS}} = O\Big(\frac{\Gamma+k}{n}\theta_{\mathsf{RIS}}\Big). \tag{2.100}$$

Combine with Lemma 15, we obtain the following theorem.

Theorem 6. *With θ_{SKIS} random* ROIS *samples (Eq.2.100) in the* SKIS *sketch $\mathcal{R}$, the* Greedy *(Alg. 11) returns an $(1 - 1/e - \epsilon)$-approximate solution with probability at least $1 - \delta$.*

SSA/D-SSA+SKIS. More recently, Nguyen et al. [84] propose SSA and D-SSA algorithms which implement the Stop-and-Stare strategy of alternating between finding candidate solutions and checking the quality of those candidates at exponential points, i.e. $2^t, t \geq 1$, to detect a satisfactory solution at the earliest time.

Combining SKIS with SSA or D-SSA brings about multiple benefits in the checking step of SSA/D-SSA. The benefits stem from the better concentration bounds which lead to better error estimations and smaller thresholds to terminate the algorithms. We give more details in the following.

Recall that the original Stop-and-Stare strategy in [84] uses two independent sets of RIS samples, called $\mathcal{R}$ and $\mathcal{R}^c$. The greedy algorithm is applied on the first set $\mathcal{R}$ to find a candidate set $\hat{S}_k$ along with an estimate $\hat{\mathbb{I}}_{\mathcal{R}}(\hat{S}_k)$ and the second set $\mathcal{R}^c$ is used to reestimate the influence of $\hat{S}_k$ by $\hat{\mathbb{I}}_{\mathcal{R}^c}(\hat{S}_k)$. Now, SSA and D-SSA have different ways to check the solution quality.

SSA. It assumes a set of fixed precision parameters $\epsilon_1, \epsilon_2, \epsilon_3$ such that $\frac{\epsilon_1+\epsilon_2+\epsilon_1\epsilon_2+\epsilon_3}{(1+\epsilon_1)(1+\epsilon_2)}(1-1/e) \leq \epsilon$. The algorithm stops when two conditions are met:

1) $\mathrm{Cov}_{\mathcal{R}}(\hat{S}_k) \geq \Lambda_1$ where $\Lambda_1 = O(\log \frac{\delta}{t_{max}}\epsilon_3^{-2})$ and t_{max} is a precomputed number depending on the size of the input graph $\mathcal{G}$.

2) $\hat{\mathbb{I}}_{\mathcal{R}}(\hat{S}_k) \leq (1+\epsilon_1)\hat{\mathbb{I}}_{\mathcal{R}^c}(\hat{S}_k)$.

$\Rightarrow$ *Improvements using* SKIS*:* Replacing RIS samples by ROIS samples to build $\mathcal{R}$ and $\mathcal{R}^c$ helps in both stopping conditions:

- Reduce Λ_1 to $\Lambda_1 = O(\frac{\Gamma}{n} \log \frac{\delta}{t_{max}} \epsilon_3^{-2})$ using the tighter form of the Chernoff's bounds in Lemma 12.

- Since ROIS samples have better influence estimation accuracy, $\hat{\mathbb{I}}_{\mathcal{R}}(\hat{S}_k)$ and $\hat{\mathbb{I}}_{\mathcal{R}^c}(\hat{S}_k)$ are closer to the true influence $\mathbb{I}(\hat{S}_k)$. Thus, the second condition is met earlier than using RIS samples.

D-SSA. Instead of assuming precision parameters, D-SSA dynamically compute the error bounds ϵ_1, ϵ_2 and ϵ_3 as follows:

- $\epsilon_1 = \frac{\hat{\mathbb{I}}_{\mathcal{R}}(\hat{S}_k)}{\hat{\mathbb{I}}_{\mathcal{R}^c}(\hat{S}_k)} - 1$.
- $\epsilon_2 = \epsilon \sqrt{\frac{n(1+\epsilon)}{2^{t-1} \hat{\mathbb{I}}_{\mathcal{R}^c}(\hat{S}_k)}}$.
- $\epsilon_3 = \epsilon \sqrt{\frac{n(1+\epsilon)(1-1/e-\epsilon)}{(1+\epsilon/3) 2^{t-1} \hat{\mathbb{I}}_{\mathcal{R}^c}(\hat{S}_k)}}$.

Here, ϵ_1 measures the discrepancy of estimations using two different sketches $\mathcal{R}$ and $\mathcal{R}^c$ while ϵ_2 and ϵ_3 are the error bounds of estimating the influences of $\hat{S}_k$ and the optimal solution S_k^* using the number of samples contained in $\mathcal{R}$ and $\mathcal{R}^c$. The algorithm stops when two conditions are met:

- $\mathrm{Cov}_{\mathcal{R}}(\hat{S}_k) \geq \Lambda_2$ where $\Lambda_2 = O(\log \frac{\delta}{t_{max}} \epsilon^{-2})$.
- $(\epsilon_1 + \epsilon_2 + \epsilon_1 \epsilon_2)(1 - 1/e - \epsilon) + (1 - 1/e)\epsilon_3 \leq \epsilon$.

$\Rightarrow$ Improvement using SKIS: Similarly to SSA, applying SKIS helps in both stopping conditions:

- Reduce Λ_2 to $\Lambda_2 = O(\frac{\Gamma}{n} \log \frac{\delta}{t_{max}} \epsilon^{-2})$.
- Reduce the value of ϵ_1, ϵ_2 and ϵ_3 due to better influence estimations of $\hat{\mathbb{I}}_{\mathcal{R}}(\hat{S}_k)$ and $\hat{\mathbb{I}}_{\mathcal{R}^c}(\hat{S}_k)$ by SKIS that leads to earlier satisfaction of the second condition.

2.3.7 Extensions to other diffusion models

The key step in extending our techniques for other diffusion models is devising an importance sketching procedure for each model. Fortunately, following the same designing principle as ROIS, we can devise importance sketching procedures for many other diffusion models. We demonstrate this ability through introducing the importance sketching algorithm for two other equally important and widely adopted diffusion models, i.e. Linear Threshold [55] and Continuous-time model [32].

The Linear Threshold model (LT) [55]. This model imposes a constraint that the total weights of incoming edges into any node $v \in V$ is at most 1, i.e. $\sum_{u \in N^{in}(v)} w(u,v) \leq 1, \forall v \in V$. Every node has a random activation threshold $\lambda_v \in [0,1]$ and gets activated if the total edge weights from active in-neighbors exceeds λ_v, i.e. $\sum_{u \in N^{in}(v), u \text{ is active}} w(u,v) \geq \lambda_v$. A RIS sampling for LT model [85] selects a random node as the source (initially activated) and iteratively picks at most one in-neighbor of the last activated node with probability being the edge weights, $w(u,v)$. It also stops when no more nodes are activated. The resulted random RIS sample consists of all the activated nodes along the way.

Similarly to ROIS, the importance sketching algorithm for the LT model has the following components:

- *Probability of having a non-singular sample*:

$$\gamma_v = \sum_{u \in N^{in}(v)} w(u,v) \tag{2.101}$$

- *Source Sampling Rate*:

$$\Pr[\mathsf{src}(R_j) = v] = \frac{\gamma_v}{\sum_{v \in V} \gamma_v} \tag{2.102}$$

- *Sample a non-singular sample from a source.*: select exactly one in-neighbor u of $\mathsf{src}(R_j) = v$ with probability $\frac{w(u,v)}{\gamma_v}$. The rest follows RIS sampling [85].

The Continuous-time diffusion model [32]. This recently emerged model exhibits in principle a similar importance sketching procedure. Here we have a deadline parameter T of the latest activation time and each edge (u, v) is associated with a length distribution, represented by a density function $\mathcal{L}_{(u,v)}(t)$, of how long it takes u to influence v. A node u is influenced if the length of the shortest path from any active node at time 0 is at most T. The RIS sampling for the Continuous-time model [105] picks a random node as the source and invokes the Dijkstra's algorithm to select nodes into $\mathsf{src}(R_j)$. When the edge (u, v) is first visited, the activation time is sampled following its length distribution $\mathcal{L}_{(u,v)}(t)$. The procedure stops when the shortest path length of the considering node exceeds the deadline T. Due to the property of Dijkstra's algorithm, at the stopping point, all the nodes with shortest path lengths less than T are visited. From the length distribution, we can compute the probability $p(u, v, T)$ of an edge (u, v) having activation time at most T as follows:

$$p(u, v, T) = \int_{t=0}^{T} \mathcal{L}_{(u,v)}(t)dt \tag{2.103}$$

The importance sketching procedure for the Continuous-time model has the following components:

- *Probability of having a non-singular sample*:

$$\gamma_v = 1 - \prod_{u \in N^{in}(v)} (1 - p(u, v, T)) \tag{2.104}$$

- *Source Sampling Rate*:

$$\Pr[\mathsf{src}(R_j) = v] = \frac{\gamma_v}{\sum_{v \in V} \gamma_v} \tag{2.105}$$

- *Sample a non-singular sample from a source.*: Use a bucket system on $p(u, v, T)$ similarly to ROIS to select the first in-neighbor u. The activation time of u follows the normalized density function $\frac{\mathcal{L}_{(u,v)}(t)}{\gamma_v}$. Subsequently, it continues by following RIS

sampling in [105].

2.3.8 Experiments

We demonstrate the advantages of our SKIS sketch through a comprehensive set of experiments on the key influence estimation and maximization problems. Due to space limit, we report the results under the IC model and partial results for the LT model. However, the implementations are released on our website to produce complete results.

2.3.8.1 Experimental Settings

Table 8.: Datasets' Statistics

Dataset	**#Nodes**	**#Edges**	**Avg. Degree**
NetPHY	$37 \cdot 10^3$	$181 \cdot 10^3$	9.8
Epinions	$75 \cdot 10^3$	$841 \cdot 10^3$	22.4
DBLP	$655 \cdot 10^3$	$2 \cdot 10^6$	6.1
Orkut	$3 \cdot 10^6$	$234 \cdot 10^6$	78.0
Twitter [65]	$41.7 \cdot 10^6$	$1.5 \cdot 10^9$	70.5
Friendster	$65.6 \cdot 10^6$	$3.6 \cdot 10^9$	109.6

Datasets. We use 6 real-world datasets from [103, 65] with size ranging from tens of thousands to as large as 65.6 million nodes and 3.6 billion edges. Table 18 gives a statistical summary of the testing datasets.

Algorithms compared. On influence estimation, we compare our SKIS sketch with:

- RIS [11]: The well-known RIS sketch.
- SKIM [25]: Combined reachability sketch. We run SKIM with default parameters in [25] ($k = l = 64$).

Following [92], we generate samples into SKIS and RIS until the total size of all the samples reaches $h \cdot n \log n$ where h is a constant. Here, h is chosen in the set $\{5, 10\}$.

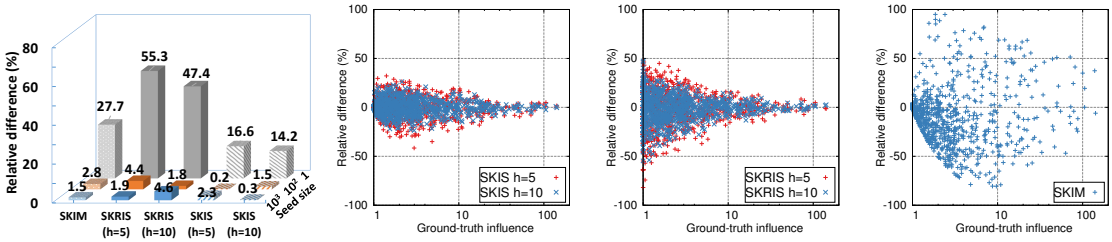
On influence maximization, we compare:

- PMC [93]: A Monte-Carlo simulation pruned method with no guarantees. It only works on the IC model.
- IMM [105]: RIS-based algorithm with quality guarantees.
- D-SSA [84]: The current fastest RIS-based algorithm with strong approximation guarantee.
- D-SSA+SKIS: A modified version of D-SSA where SKIS sketch is adopted to replace RIS.

We set $\epsilon = 0.5, \delta = 1/n$ for the last three algorithms. For PMC, we use the default parameter of 200 DAGs.

We compare the algorithms in terms of the solution quality, running time and memory usage. For quality assessment, we use the following metric.

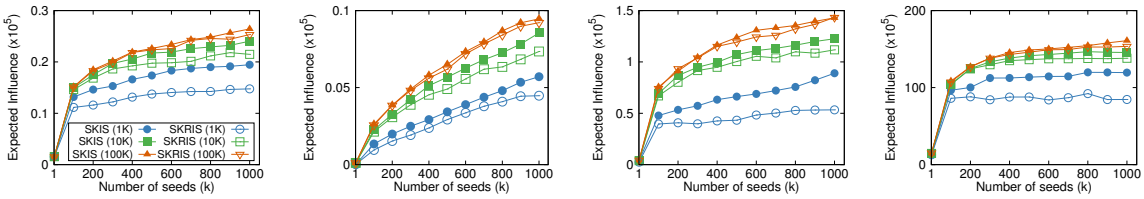
Estimation Error. To assess the estimation quality w.r.t. a seed set S, we adopt the *relative difference* which is defined as $\frac{|\hat{\mathbb{I}}(S)-\mathbb{I}(S)|}{\max\{\mathbb{I}(S),\hat{\mathbb{I}}(S)\}} \cdot 100\%$, where $\hat{\mathbb{I}}(S)$ is an estimate, and $\mathbb{I}(S)$ is the "ground-truth" influence of S.



(a) TRI model (b) SKIS-WC (c) RIS-WC (d) SKIM-WC

Fig. 8.: Relative difference on Epinions under a) TRI model and b), c), d) WC model with $|S| = 1$. SKIS are the closest to the 'ground truth' among the three sketches.

Ground-truth Influence. Unlike previous studies [105, 25, 93] using a constant number of cascade simulations, i.e. 10000, to measure the ground-truth influence with



(a) Epinions (b) NetPHY (c) DBLP (d) Twitter

Fig. 9.: Efficiency of SKIS and RIS sketches in finding the maximum seed sets. SKIS sketch is up to 80% more efficient.

unknown accuracy, we adopt the Monte-Carlo Stopping-Rule algorithm [26] that guarantees an estimation error less than ϵ with probability at least $1-\delta$ where $\epsilon = 0.005, \delta = 1/n$. Specifically, let W_j be the size of a random influence cascade and $Z_j = \frac{W_j}{n}$ with $\mathcal{E}[Z_j] = \mathbb{I}(S)/n$ and $0 \leq Z_j \leq 1$. Monte-Carlo method generates sample Z_j until $\sum_{j=1}^{T} Z_j \geq 4(e-2)\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}$ and returns $\hat{\mathbb{I}}(S) = \sum_{j=1}^{T} Z_j n/T$ as the ground-truth influence.

For Twitter and Friendster dataset, we set $\epsilon = 0.05$, and $\delta = 1/n$ to compute ground-truth due to the huge computational cost in these networks. For the other networks, we keep the default setting of ϵ and δ as specified above.

Weight Settings. We consider two widely-used models:

- *Weighted Cascade* (WC) [106, 25, 105, 84]: The weight of edge (u, v) is inversely proportional to the in-degree of node v, $d_{in}(v)$, i.e. $w(u, v) = \frac{1}{d_{in}(v)}$.
- *Trivalency* (TRI) [25, 19, 52]: The weight $w(u, v)$ is selected randomly from the set $\{0.1, 0.01, 0.001\}$.

Environment. We implemented our algorithms in C++ and obtained the implementations of others from the corresponding authors. We conducted all experiments on a CentOS machine with Intel Xeon E5-2650 v3 2.30GHz CPUs and 256GB RAM. We compute the ground-truth for our experiments in a period of 2 months on a cluster of 16 CentOS ma-

Table 9.: Average relative differences (dnf: "did not finish" within 24h). SKIS almost always returns the lowest errors.

		WC Model					**TRI Model**				
		SKIS		RIS		SKIM	SKIS		RIS		SKIM
$\|S\|$	**Nets**	$h(5)$	$h(10)$	$h(5)$	$h(10)$	$k(64)$	$h(5)$	$h(10)$	$h(5)$	$h(10)$	$k(64)$
1	PHY	6.2	3.7	14.0	7.8	7.5	1.7	1.3	11.8	8.2	4.5
	Epin.	4.7	3.0	15.7	11.8	19.6	16.6	14.2	55.3	47.4	27.7
	DBLP	3.8	4.1	13.7	11.6	5.0	0.9	0.7	9.4	6.4	3.5
	Orkut	10.3	9.2	13.5	8.8	77.6	9.3	9.9	14.5	10.8	dnf
	Twit.	10.9	10.5	21.4	16.0	29.1	81.4	81.9	80.8	81.5	dnf
	Frien.	15.9	10.2	22.2	13.3	dnf	29.8	21.3	28.5	23.6	dnf
10^2	PHY	0.9	0.6	1.0	0.7	2.1	0.3	0.2	1.1	0.9	1.8
	Epin.	1.0	0.7	1.0	1.0	7.6	0.2	1.5	4.4	1.8	2.8
	DBLP	0.9	0.6	1.9	1.4	5.0	0.8	0.7	5.5	5.3	5.5
	Orkut	0.9	0.6	1.1	0.7	56.5	0.1	0.2	4.2	0.9	dnf
	Twit.	1.1	1.2	1.3	1.1	60.2	4.3	3.1	6.4	5.5	dnf
	Frien.	0.9	0.7	0.9	0.7	dnf	1.9	1.9	0.6	2.0	dnf
10^3	PHY	0.6	0.8	0.9	1.0	0.6	0.3	0.4	1.2	1.3	1.1
	Epin.	0.6	0.6	0.6	0.7	2.3	2.3	0.3	1.9	4.6	1.5
	DBLP	0.2	0.3	0.2	0.2	1.7	0.1	0.0	0.3	0.2	0.3
	Orkut	0.3	0.3	0.3	0.3	50.7	2.5	1.1	6.8	2.1	dnf
	Twit.	0.9	0.9	1.0	0.9	36.3	0.9	2.4	4.1	2.8	dnf
	Frien.	0.3	0.3	0.3	0.2	dnf	1.9	1.9	0.6	2.0	dnf

chines, each with 64 Intel Xeon CPUs X5650 2.67GHz and 256GB RAM.

2.3.8.2 Influence Estimation

We show that SKIS sketch consumes much less time and memory space while consistently obtaining better solution quality, i.e. very small errors, than both RIS and SKIM.

Solution Quality: Table 9 and Figure 8 present the relative estimation errors of all three sketches.

The solution quality of SKIS is consistently better than RIS and SKIM across all the networks and edge models. As shown in Table 9, the errors of SKIS are 110% and

Table 10.: Performance of IM algorithms with $k = 100$ (dnf: "did not finish" within 6h, mem: "out of memory").

		Running Time [s (or h)]				**Total Memory [M (or G)]**				**Expected Influence (%)**				**#Samples [$\times 10^3$]**		
	Nets	IMM	PMC	D-SSA	D-SSA +SKIS	IMM	PMC	D-SSA	D-SSA +SKIS	IMM	PMC	D-SSA	D-SSA +SKIS	IMM	D-SSA	D-SSA +SKIS
WC	PHY	0.1	3.1	**0.0**	**0.0**	31	86	26	**9**	6.64	6.7	5.33	5.34	103.3	8.9	**3.8**
	Epin.	0.2	10.5	**0.0**	**0.0**	39	130	34	**17**	19.4	19.8	17.9	16.6	39.8	4.48	**0.9**
	DBLP	1.1	137.4	**0.1**	**0.1**	162	**60**	136	113	10.8	11.2	9.3	8.5	93.0	5.4	**2.6**
	Orkut	24.1	1.4h	2.6	**0.9**	4G	6G	**2G**	**2G**	6.7	8.7	5.7	5.1	174.4	11.52	**2.6**
	Twit.	67.3	mem	5.5	6.3	30G	mem	17G	**16G**	25.80	mem	24.1	21.0	54.0	18.0	**0.8**
	Frien.	dnf	mem	78.3	**43.6**	dnf	mem	**35G**	36G	dnf	mem	0.35	0.35	mem	215.0	**102.4**
TRI	PHY	0.2	1.5	**0.0**	**0.0**	50	61	30	**9**	1.77	1.73	1.4	1.5	370.1	35.8	**3.8**
	Epin.	13.9	6.9	2.0	**0.6**	483	40	72	**33**	5.7	5.9	5.47	5.46	123.0	8.9	**0.5**
	DBLP	3.2	20.1	0.3	**0.2**	389	**54**	191	118	0.32	0.31	0.28	0.24	3171.0	348.2	**20.5**
	Orkut	dnf	0.3h	1.3h	**0.2h**	dnf	16G	28G	**11G**	dnf	67.3	67.9	67.8	dnf	1.4	**0.3**
	Twit.	dnf	mem	5.2h	**0.6h**	dnf	mem	100G	**28G**	dnf	mem	24.2	24.4	dnf	3.4	**0.4**
	Frien.	dnf	mem	mem	**3.1h**	dnf	mem	mem	**99G**	dnf	mem	mem	40.1	dnf	mem	**0.2**

400% smaller than those of RIS with $k = 1$ while being as good as or better than RIS for $k = 100, 1000$. On the other hand, SKIM shows the largest estimation errors in most of the cases. Particularly, SKIM's error is more than 60 times higher than SKIS and RIS on Twitter when $|S| = 100$. Similar results are observed under TRI model. Exceptionally, on Twitter and Friendster, the relative difference of RIS is slightly smaller than SKIS with $h = 5$ but larger on $h = 10$. In TRI model, estimating a random seed on large network as Twitter produces higher errors since we have insufficient number of samples.

Figures 8b, c, and d draw the error distributions of sketches for estimating the influences of random seeds. Here, we generate 1000 uniformly random nodes and consider each node to be a seed set. We observe that SKIS's errors are highly concentrated around 0% even when the influences are small while errors of RIS and SKIM spread out widely. RIS reveals extremely high errors for small influence estimation, e.g. up to 80%. The error distribution of SKIM is the most widely behaved, i.e. having high errors at every influence level. Under TRI model (Figure 8a), SKIS also consistently provides significantly smaller estimation errors than RIS and SKIM.

Performance: We report indexing time and memory of different sketches in Table 11.

Table 11.: Sketch construction time and index memory of algorithms on different edge models. SKIS and RIS uses roughly the same time and memory and less than those of SKIM.

		Index Time [second (or h for hour)]					**Index Memory** [MB (or G for GB)]				
		SKIS		RIS		SKIM	SKIS		RIS		SKIM
	Nets	$h(5)$	$h(10)$	$h(5)$	$h(10)$	$k(64)$	$h(5)$	$h(10)$	$h(5)$	$h(10)$	$k(64)$
WC	PHY	0	1	1	1	2	41	83	52	105	105
	Epin.	1	1	1	1	10	63	126	81	162	220
	DBLP	10	18	7	14	37	702	1G	848	2G	2G
	Orkut	92	157	69	148	0.6h	2G	5G	3G	5G	9G
	Twit.	0.6h	0.9h	0.4h	1.0h	5.2h	38G	76G	42G	84G	44G
	Frien.	0.8h	1.8h	0.8h	1.9h	dnf	59G	117G	61G	117G	dnf
TRI	PHY	0	1	1	2	1	46	90	97	194	99
	Epin.	1	1	1	1	29	41	82	41	84	230
	DBLP	11	34	18	36	22	1G	2G	2G	5G	2G
	Orkut	88	206	89	197	dnf	2G	4G	2G	4G	dnf
	Twit.	0.6h	1.2h	0.5h	1.3h	dnf	36G	69G	36G	69G	dnf
	Frien.	0.9h	2.3h	1.0h	2.4h	dnf	54G	108G	54G	108G	dnf

Indexing Time. SKIS and RIS use roughly the same amount of time for build the sketches while SKIM is much slower than SKIS and RIS and failed to process large networks in both edge models. On larger networks, SKIS is slightly faster than RIS. SKIM markedly spends up to 5 hours to build sketch for Twitter on WC model while SKIS, or RIS spends only 1 hour or less on this network.

Index Memory. In terms of memory, the same observations are seen as with indexing time essentially because larger sketches require more time to construct. In all the experiments, SKIS consumes the same or less amount of memory with RIS. SKIM generally uses more memory than SKIS and RIS.

In summary, SKIS consistently achieves better solution quality than both RIS and

SKIM on all the networks, edge models and seed set sizes while consuming the same or less time/memory. The errors of SKIS is highly concentrated around 0. In contrast, RIS is only good for estimating high influence while incurring significant errors for small ranges.

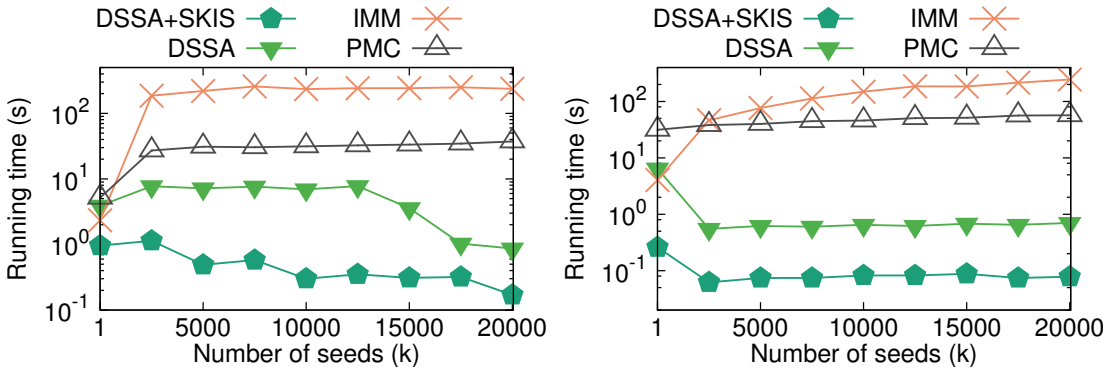


Fig. 10.: Running time of algorithms under the IC model.

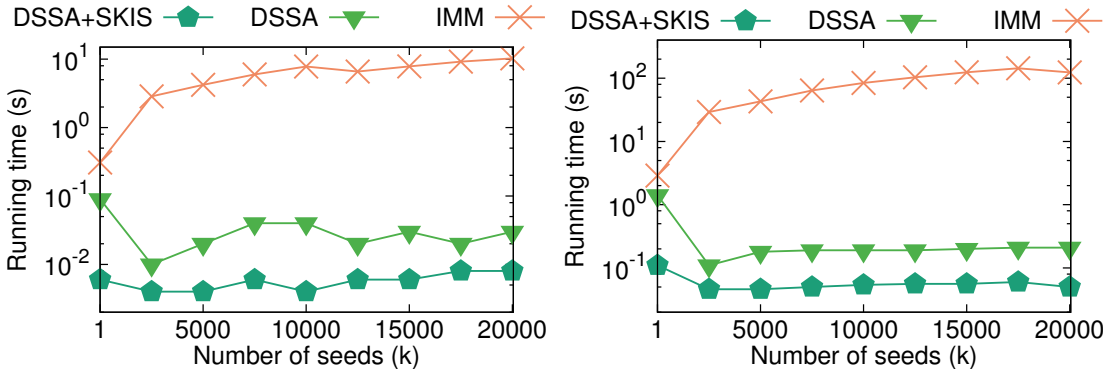


Fig. 11.: Running time of algorithms under the LT model.

2.3.8.3 Influence Maximization

This subsection illustrates the advantage of ROIS sketch in finding the seed set with maximum influence. The results show that ROIS samples drastically speed up the computation time. D-SSA+SKIS is the first to handle billion-scale networks on the challenging

TRI edge model. We limit the running time for algorithms to 6 hours and put "dnf" if they cannot finish.

Identifiability of the Maximum Seed Sets: We compare the ability of the new ROIS with the traditional RIS sampling in terms of identifying the seed set with maximum influence. We fix the number of samples generated to be in the set $\{1000, 10000, 100000\}$ and then apply the Greedy algorithm to find solutions. We recompute the influence of returned seed sets using Monte-Carlo method with precision parameters $\epsilon = 0.005, \delta = 1/n$. The results is presented in Figure 9.

From Figure 9, we observe a recurrent consistency that ROIS samples return a better solution than RIS over all the networks, k values and number of samples. Particularly, the solutions provided by ROIS achieve up to 80% better than those returned by RIS. When more samples are used, the gap gets smaller.

Efficiency of SKIS on IM problem: Table 10 presents the results of D-SSA-SKIS, D-SSA, IMM and PMC in terms of running time, memory consumption and samples generated.

Running Time. From Table 10, the combination D-SSA+SKIS outperforms the rest by significant margins on all datasets and edge models. D-SSA-SKIS is up to 10x faster than the original D-SSA. D-SSA+SKIS is the first and only algorithm that can run on the largest network on TRI model.

Figure 10 compares the running time of all IM algorithms across a wide range of budget $k = 1..20000$ under IC and TRI edge weight model. D-SSA+SKIS always maintains significant performance gaps to the other algorithms, e.g. 10x faster than D-SSA or 1000x faster than IMM and PMC.

Number of Samples and Memory Usage. On the same line with the running time, the memory usage and number of samples generated by D-SSA+SKIS are much less than those required by the other algorithms. The number of samples generated by D-SSA+SKIS

is up to more 10x smaller than D-SSA on TRI model, 100x less than IMM. Since the memory for storing the graph is counted into the total memory, the memory saved by D-SSA+SKIS is only several times smaller than those of D-SSA and IMM. PMC exceptionally requires huge memory and is unable to run on two large networks.

Experiments on the Linear Threshold (LT) model. We carry another set of experiments on the LT model with multiple budget k. Since in LT, the total weights of incoming edge to every node are bounded by 1, for each node, we first normalized the weights of incoming edges and then multiply them with a random number uniformly generated in $[0, 1]$.

The results are illustrated in Figure 11. Similar observations to the IC are seen in the LT model that D-SSA+SKIS runs faster than the others by orders of magnitude.

Overall, D-SSA+SKIS reveals significant improvements over the state-of-the-art algorithms on influence maximization. As a result, D-SSA+SKIS is the only algorithm that can handle the largest networks under different models.

2.3.9 Conclusion

We propose SKIS - a novel sketching tools to approximate influence dynamics in the networks. We provide both comprehensive theoretical and empirical analysis to demonstrate the superiority in size-quality trade-off of SKIS in comparisons to the existing sketches. The application of SKIS to existing algorithms on Influence Maximization leads to significant performance boost and easily scale to billion-scale networks. In future, we plan to extend SKIS to other settings including evolving networks and time-based influence dynamics.

2.3.10 Omitted Proofs of Lemmas and Theorems

2.3.10.1 Proof of Lemma 9

Given a stochastic graph $\mathcal{G}$, recall that $\Omega_{\mathcal{G}}$ is the space of all possible sample graphs $g \sim \mathcal{G}$ and $\Pr[g]$ is the probability that g is realized from $\mathcal{G}$. In a sample graph $g \in \Omega_{\mathcal{G}}$, $\eta_g(S, v) = 1$ if v is reachable from S in g. Consider the graph sample space $\Omega_{\mathcal{G}}$, based on a node $v \in V \backslash S$, we can divide $\Omega_{\mathcal{G}}$ into two partitions: 1) $\Omega^{\emptyset}_{\mathcal{G}}(v)$ contains those samples g in which v has *no incoming live-edges*; and 2) $\bar{\Omega}^{\emptyset}_{\mathcal{G}}(v) = \Omega_{\mathcal{G}} \backslash \Omega^{\emptyset}_{\mathcal{G}}$. We start from the definition of influence spread as follows,

$$\begin{aligned}\mathbb{I}(S) &= \sum_{v \in V} \sum_{g \in \Omega_{\mathcal{G}}} \eta_g(S, v) \Pr[g] \\ &= \sum_{v \in V} \Big(\sum_{g \in \Omega^{\emptyset}_{\mathcal{G}}(v)} \eta_g(S, v) \Pr[g] + \sum_{g \in \bar{\Omega}^{\emptyset}_{\mathcal{G}}(v)} \eta_g(S, v) \Pr[g] \Big).\end{aligned}$$

In each $g \in \Omega^{\emptyset}_{\mathcal{G}}(v)$, the node v does not have any incoming nodes, thus, $\eta_g(S, v) = 1$ only if $v \in S$. Thus, we have that $\sum_{v \in V} \sum_{g \in \Omega^{\emptyset}_{\mathcal{G}}(v)} \eta_g(S, v) \Pr[g] = \sum_{v \in S} \sum_{g \in \Omega^{\emptyset}_{\mathcal{G}}(v)} \Pr[g]$. Furthermore, the probability of a sample graph which has no incoming live-edge to v is $\sum_{g \in \Omega^{\emptyset}_{\mathcal{G}}(v)} \Pr[g] = 1 - \gamma_v$. Combine with the above equiation of $\mathbb{I}(S)$, we obtain,

$$\mathbb{I}(S) = \sum_{v \in S} (1 - \gamma_v) + \sum_{v \in V} \sum_{g \in \bar{\Omega}^{\emptyset}_{\mathcal{G}}(v)} \eta_g(S, v) \Pr[g \in \Omega_{\mathcal{G}}]. \tag{2.106}$$

Since our ROIS sketching algorithm only generates samples corresponding to sample graphs from the set $\bar{\Omega}^{\emptyset}_{\mathcal{G}}(v)$, we define $\bar{\Omega}^{\emptyset}_{\mathcal{G}}(v)$ to be a graph sample space in which the sample graph $\bar{g} \in \bar{\Omega}^{\emptyset}_{\mathcal{G}}(v)$ has a probability $\Pr[\bar{g} \in \bar{\Omega}^{\emptyset}_{\mathcal{G}}(v)] = \frac{\Pr[\bar{g} \in \Omega_{\mathcal{G}}]}{\gamma_v}$ of being realized (since $\sum_{\bar{g} \in \bar{\Omega}^{\emptyset}_{\mathcal{G}}(v)} \Pr[\bar{g} \in \Omega_{\mathcal{G}}] = \gamma_v$ is the normalizing factor to fulfill a probability distribution of a

sample space). Then, Eq. 2.106 is rewritten as follows,

$$\begin{aligned}\mathbb{I}(S) &= \sum_{v \in V} \sum_{g \in \bar{\Omega}^{\emptyset}_{\mathcal{G}}(v)} \eta_g(S, v) \frac{\Pr[g \in \Omega_{\mathcal{G}}]}{\gamma_v} \gamma_v + \sum_{v \in S} (1 - \gamma_v) \\ &= \sum_{v \in V} \sum_{\bar{g} \in \bar{\Omega}^{\emptyset}_{\mathcal{G}}(v)} \eta_{\bar{g}}(S, v) \Pr[\bar{g} \in \bar{\Omega}^{\emptyset}_{\mathcal{G}}(v)] \gamma_v + \sum_{v \in S} (1 - \gamma_v)\end{aligned}$$

Now, from the node v in a sample graph $\bar{g} \in \bar{\Omega}^{\emptyset}_{\mathcal{G}}(v)$, we have a ROIS sketch $R_j(\bar{g}, v)$ starting from v and containing all the nodes that can reach v in $\bar{g}$. Thus, $\eta_{\bar{g}}(S, v) = \mathbf{1}_{R_j(\bar{g},v) \cap S \neq \emptyset}$ where $\mathbf{1}_x$ is an indicator function returning 1 iff $x \neq 0$. Then,

$$\begin{aligned}&\sum_{\bar{g} \in \bar{\Omega}^{\emptyset}_{\mathcal{G}}(v)} \eta_{\bar{g}}(S, v) \Pr[\bar{g} \in \bar{\Omega}^{\emptyset}_{\mathcal{G}}(v)] \\ &= \sum_{\bar{g} \in \bar{\Omega}^{\emptyset}_{\mathcal{G}}(v)} \mathbf{1}_{R_j(\bar{g},v) \cap S \neq \emptyset} \Pr[\bar{g} \in \bar{\Omega}^{\emptyset}_{\mathcal{G}}(v)] = \Pr[R_j(v) \cap S \neq \emptyset]\end{aligned}$$

where $R_j(v)$ is a random ROIS sketch with $\mathsf{src}(R_j(v)) = v$. Plugging this back into the computation of $\mathbb{I}(S)$ gives,

$$\begin{aligned}\mathbb{I}(S) &= \sum_{v \in V} \Pr[R_j(v) \cap S \neq \emptyset] \gamma_v + \sum_{v \in S} (1 - \gamma_v) \\ &= \sum_{v \in V} \Pr[R_j(v) \cap S \neq \emptyset] \frac{\gamma_v}{\Gamma} \Gamma + \sum_{v \in S} (1 - \gamma_v) \\ &= \sum_{v \in V} \Pr[R_j(v) \cap S \neq \emptyset] \Pr[\mathsf{src}(R_j) = v] \Gamma + \sum_{v \in S} (1 - \gamma_v) \\ &= \Pr[R_j \cap S \neq \emptyset] \cdot \Gamma + \sum_{v \in S} (1 - \gamma_v) \qquad (2.107)\end{aligned}$$

That completes the proof.

2.3.10.2 Proof of Lemma 10

From the basic properties of variance, we have,

$$\begin{aligned}\mathrm{Var}[Z_j(S)] &= \mathrm{Var}[\frac{X_j(S)\cdot\Gamma + \sum_{v\in S}(1-\gamma_v)}{n}] \\ &= \frac{\Gamma^2}{n^2}\mathrm{Var}[X_j(S)]\end{aligned}$$

Since $X_j(S)$ is a Bernoulli random variable with its mean value $\mathcal{E}[X_j(S)] = \frac{\mathbb{I}(S)-\sum_{v\in S}(1-\gamma_v)}{\Gamma}$, the variance $\mathrm{Var}[X_j(S)]$ is computed as follows,

$$\begin{aligned}&\mathrm{Var}[X_j(S)] \\ &= \frac{\mathbb{I}(S)-\sum_{v\in S}(1-\gamma_v)}{\Gamma}(1-\frac{\mathbb{I}(S)-\sum_{v\in S}(1-\gamma_v)}{\Gamma}) \\ &= \frac{\mathbb{I}(S)}{\Gamma} - \frac{\mathbb{I}^2(S)}{\Gamma^2} - \frac{\sum_{v\in S}(1-\gamma_v)}{\Gamma^2}(\Gamma + \sum_{v\in S}(1-\gamma_v) - 2\mathbb{I}(S))\end{aligned}$$

Put this back into the variance of $Z_j(S)$ proves the lemma.

2.3.10.3 Proof of Lemma 11

Since $Z_j(S)$ takes values of either $\frac{\sum_{v\in S}(1-\gamma_v)}{n}$ or $\frac{\Gamma+\sum_{v\in S}(1-\gamma_v)}{n}$ and the mean value $\mathcal{E}[Z_j(S)] = \frac{\mathbb{I}(S)}{n}$, i.e. $\frac{\sum_{v\in S}(1-\gamma_v)}{n} \leq \frac{\mathbb{I}(S)}{n} \leq \frac{\Gamma+\sum_{v\in S}(1-\gamma_v)}{n}$. The variance of $Z_j(S)$ is computed as follows,

$$\begin{aligned}&\mathrm{Var}[Z_j(S)] \\ &\quad= \Big(\frac{\mathbb{I}(S)}{n} - \frac{\sum_{v\in S}(1-\gamma_v)}{n}\Big)\Big(\frac{\Gamma+\sum_{v\in S}(1-\gamma_v)}{n} - \frac{\mathbb{I}(S)}{n}\Big) \\ &\quad\leq \frac{\mathbb{I}(S)}{n}\Big(\frac{\Gamma+\sum_{v\in S}(1-\gamma_v)}{n} - \frac{\sum_{v\in S}(1-\gamma_v)}{n}\Big) \\ &\quad= \frac{\mathbb{I}(S)}{n}\frac{\Gamma}{n}\end{aligned} \tag{2.108}$$

2.3.10.4 Proof of Lemma 12

Lemma 2 in [105] states that:

Lemma 16. *Let $M_1, M_2, \ldots$ be a martingale, such that $|M_1| \leq a$, $|M_j - M_{j-1}| \leq a$ for any $j \in [2, T]$, and*

$$\mathrm{Var}[M_1] + \sum_{j=2}^{T} \mathrm{Var}[M_j | M_1, M_2, \ldots, M_{j-1}] \leq b, \tag{2.109}$$

where $\mathrm{Var}[.]$ *denotes the variances of a random variable. Then, for any $\lambda > 0$,*

$$\Pr[M_T - \mathcal{E}[M_T] \geq \lambda] \leq \exp\Big(- \frac{\lambda^2}{\frac{2}{3}a\lambda + 2b} \Big) \tag{2.110}$$

Note that uniform random variables are also a special type of martingale and the above lemma holds for random variable as well. Let $p = \frac{\mathbb{I}(S)}{n}$. For RIS samples, since

- $|M_1| \leq 1$,
- $|M_j - M_{j-1}| \leq 1, \forall j \in [2, T]$,
- $\mathrm{Var}[M_1] + \sum_{j=2}^{T} \mathrm{Var}[M_j | M_1, \ldots, M_{j-1}] = \sum_{j=1}^{i} \mathrm{Var}[Y_j(S)] = Tp(1-p) \leq Tp$,

applying Eq. 2.110 for $\lambda = \epsilon Tp$ gives the following Chernoff's bounds,

$$\Pr\Big[\sum_{j=1}^{T} X_j(S) - Tp \geq \epsilon Tp \Big] \leq \exp\Big(- \frac{\epsilon^2}{2 + \frac{2}{3}\epsilon} Tp \Big), \tag{2.111}$$

and,

$$\Pr\Big[\sum_{j=1}^{T} X_j(S) - Tp \leq -\epsilon Tp \Big] \leq \exp\Big(- \frac{\epsilon^2}{2} Tp \Big). \tag{2.112}$$

However, for ROIS samples in SKIS sketch, the corresponding random variables $Z_j(S)$ replace $Y_j(S)$ and have the following properties:

- $|M_1| \leq \frac{\Gamma + \sum_{v \in S}(1 - \gamma_v)}{n} \leq 1$,

- $|M_j - M_{j-1}| \leq 1, \forall j \in [2, T]$,

- The sum of variances:

$$\begin{aligned} &\mathrm{Var}[M_1] + \sum_{j=2}^{T} \mathrm{Var}[M_j | M_1, \ldots, M_{j-1}] \\ &= \sum_{j=1}^{i} \mathrm{Var}[Z_j(S)] = Tp\frac{\Gamma}{n} \end{aligned} \tag{2.113}$$

Thus, applying the general bound in Eq. 2.110 gives,

$$\Pr\Big[\sum_{j=1}^{T} X_j(S) - Tp \geq \epsilon Tp\Big] \leq \exp\Big(-\frac{\epsilon^2}{2\frac{\boldsymbol{\Gamma}}{\boldsymbol{n}} + \frac{2}{3}\epsilon} Tp\Big), \tag{2.114}$$

and,

$$\Pr\Big[\sum_{j=1}^{T} X_j(S) - Tp \leq -\epsilon Tp\Big] \leq \exp\Big(-\frac{\epsilon^2}{2\frac{\boldsymbol{\Gamma}}{\boldsymbol{n}}} Tp\Big). \tag{2.115}$$

Note the factor $\frac{\Gamma}{n}$ is added in the denominator of the terms in the $\exp(.)$ function. Since $2\frac{\Gamma}{n}$ dominates $\frac{2}{3}\epsilon$, the concentration bounds for $Z_j(S)$ for SKIS are tighter than those of $Y_j(S)$ for RIS given in Eqs. 2.111 and 2.112.

2.3.10.5 Proof of Lemma 14

Since the function $\hat{\mathbb{I}}_{\mathcal{R}}(S)$ contains two additive terms, it is sufficient to show that each of them is monotone and submodular. The second term $\sum_{v \in S}(1 - \gamma_v)$ is a linear function and thus, it is monotone and submodular. For the first additive term, we see that $\frac{\Gamma}{|\mathcal{R}| \cdot n}$ is a constant and only need to show that $\mathrm{Cov}_{\mathcal{R}}(S)$ is monotone and submodular. Given the collection of ROIS samples $\mathcal{R}$ in which $R_j \in \mathcal{R}$ is a list of nodes, the function $\mathrm{Cov}_{\mathcal{R}}(S)$ is just the count of ROIS samples that intersect with the set S. In other words, it is equivalent to a covering function in a set system where ROIS samples are elements and nodes are sets. A set covers an element if the corresponding node is contained in the corresponding

ROIS sample. It is well known that any covering function is monotone and submodular [107] and thus, the $\mathrm{Cov}_{\mathcal{R}}(S)$ has the same properties.

CHAPTER 3

INFLUENCE MAXIMIZATION

In this topic, we study three key problems and propose near-optimal approximation algorithms that significantly surpass the state-of-the-art methods in terms of efficiency by several orders of magnitudes. Our proposed algorithms are able to handle billion-scale networks within a matter of seconds.

3.1 Optimal Sampling Algorithms for Influence Maximization

Problem Definition: Given the propagation models defined previously, the Influence Maximization (IM) problem is defined as follows,

Definition 6 (Influence Maximization (IM)). *Given a graph $G = (V, E, w)$, an integer $1 \leq k \leq |V|$ and a propagation model, the Influence Maximization problem asks for a seed set $\hat{S}_k \subset V$ of k nodes that maximizes the influence spread $\mathbb{I}(\hat{S}_k)$ under the given propagation model.*

Summary of contributions:

- We generalize the RIS sampling methods in [11, 106, 105] into a general framework which characterizes the necessary conditions to guarantee the $(1 - 1/e - \epsilon)$-approximation factor. Based on the framework, we define classes of RIS thresholds and two types of minimum thresholds, namely, type-1 and type-2.

- We propose the Stop-and-Stare Algorithm (SSA) and its dynamic version, D-SSA, which both guarantee a $(1 - 1/e - \epsilon)$-approximate solution and are the first algorithms to achieve, within constant factors, the type-1 and type-2 minimum thresholds,

respectively. Our proposed methods are not limited to solve influence maximization problem but also can be generalized for an important class of hard optimization problems over samples/sketches.

- Our framework and approaches are generic and can be applied in principle to sample-based optimization problems to design high-confidence approximation algorithm using (asymptotically) *minimum number of samples*.

- We carry extensive experiments on various real networks with up to several billion edges to show the superiority in performance and comparable solution quality. To test the applicability of the proposed algorithms, we apply our methods on an IM-application, namely, Targeted Viral Marketing (TVM). The results show that our algorithms are up to 1200 times faster than the current best method on IM problem and, for TVM, the speedup is up to 500 times.

We summarize the frequently used notations in Table 24.

Table 12.: Table of notations

Notation	**Description**
n, m	#nodes, #edges of graph $G = (V, E, w)$.
$\mathbb{I}(S)$	Influence Spread of seed set $S \subseteq V$.
OPT_k	The maximum $\mathbb{I}(S)$ for any size-k seed set S.
$\hat{S}_k$	The returned size-k seed set of SSA/D-SSA.
S_k^*	An optimal size-k seed set, i.e., $\mathbb{I}(S_k^*) = \mathsf{OPT}_k$.
R_j	A random RR set.
$\mathcal{R}$	A collection of random RR sets.
$\mathrm{Cov}_{\mathcal{R}}(S)$	#RR sets $R_j \in \mathcal{R}$ covered by S, i.e., $R_j \cap S \neq \emptyset$.
$\hat{\mathbb{I}}_{\mathcal{R}}(S), \hat{\mathbb{I}}(S)$	$\frac{\mathrm{Cov}_{\mathcal{R}}(S)}{\lvert\mathcal{R}\rvert}$.
$\Upsilon(\epsilon, \delta)$	$\Upsilon(\epsilon, \delta) = (2 + \frac{2}{3}\epsilon) \ln \frac{1}{\delta} \frac{1}{\epsilon^2}$.

3.1.1 Unified RIS framework

This section presents the unified RIS framework, generalizing all the previous methods of using RIS sampling [11, 106, 105, 89] for IM. The unified framework characterizes the sufficient conditions to guarantee an $(1 - 1/e - \epsilon)$-approximation. Subsequently, we will introduce the concept of RIS threshold in terms of the number of necessary samples to guarantee the solution quality and two types of minimum RIS thresholds, i.e., type-1 and type-2.

3.1.1.1 Preliminaries

RIS sampling: The major bottle-neck in the traditional methods for IM [55, 70, 47, 87] is the inefficiency in estimating the influence spread. To address that, Borgs et al. [11] introduced a novel sampling approach for IM, called Reverse Influence Sampling (in short, RIS), which is the foundation for TIM/TIM+[106] and IMM[105], the state-of-the-art methods.

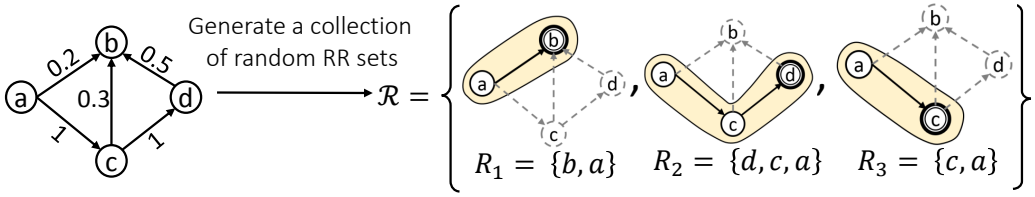


Fig. 12.: An example of generating random RR sets under the LT model. Three random RR sets R_1, R_2 and R_3 are generated. Node a has the highest influence and is also the most frequent element across the RR sets.

Given a graph $G = (V, E, w)$, RIS captures the influence landscape of G through generating a set $\mathcal{R}$ of random *Reverse Reachable (RR) sets*. The term 'RR set' is also used in TIM/TIM+ [106, 105] and referred to as 'hyperedge' in [11]. Each RR set R_j is a subset of V and constructed as follows,

Definition 7 (Reverse Reachable (RR) set)**.** *Given $G = (V, E, w)$, a random RR set R_j is generated from G by 1) selecting a random node $v \in V$ 2) generating a sample graph g from G and 3) returning R_j as the set of nodes that can reach v in g.*

Node v in the above definition is called the *source* of R_j. Observe that R_j contains the nodes that can influence its source v.

If we generate multiple random RR sets, influential nodes will likely appear frequently in the RR sets. Thus a seed set S that *covers* most of the RR sets will likely maximize the influence spread $\mathbb{I}(S)$. Here a seed set S covers an RR set R_j, if $S \cap R_j \neq \emptyset$. For convenience, we denote the coverage of set S as follows,

$$\mathrm{Cov}_{\mathcal{R}}(S) = \sum_{R_j \in \mathcal{R}} \min\{|S \cap R_j|, 1\} \tag{3.1}$$

An illustration of this intuition and how to generate RR sets is given in Fig. 12. In the figure, three random RR sets are generated following the LT model with sources b, d and c, respectively. The influence of node a is the highest among all the nodes in the original graph and also is the most frequent node across the RR sets. This observation is captured in the following lemma in [11].

Lemma 17 ([11])**.** *Given $G = (V, E, w)$, a seed set $S \subset V$, for a random RR set R_j generated from G,*

$$\mathbb{I}(S) = n \Pr[S \textit{ covers } R_j]. \tag{3.2}$$

Lemma 30 says that the influence of a node set S is proportional to the probability that S intersects with a random RR set. Define

$$\hat{\mathbb{I}}_{\mathcal{R}}(S) = \frac{\mathrm{Cov}_{\mathcal{R}}(S)}{|\mathcal{R}|},$$

an approximate of $\mathbb{I}(S)$. When the context is clear, we also ignore $\mathcal{R}$ and write $\hat{\mathbb{I}}(S)$ instead

of $\hat{\mathbb{I}}_{\mathcal{R}}(S)$. Thus, to find S that maximize $\mathbb{I}(S)$ we can find S to maximize $\hat{I}(S)$, i.e., to find subset S that covers as many R_j as possible. The most important question addressed in this paper is about the *minimum size of $\mathcal{R}$ to provide bounded-error guarantees*.

(ϵ, δ)-approximation: The bounded-error guarantee we seek for in our influence maximization algorithms, $(1-1/e-\epsilon)$ with probability at least $(1-\delta)$, is based on the concept of (ϵ, δ)-approximation.

Definition 8 ((ϵ, δ)-approximation)**.** *Let $Z_1, Z_2, ...$ be i.i.d. random variables in $[0, 1]$ with mean μ_Z and variance σ_Z^2. A Monte Carlo estimator,*

$$\hat{\mu}_Z = \frac{1}{T}\sum_{i=1}^{T} Z_i \tag{3.3}$$

is said to be an (ϵ, δ)-approximation of μ_Z if

$$\Pr[(1-\epsilon)\mu_Z \leq \hat{\mu}_Z \leq (1+\epsilon)\mu_Z] \geq 1-\delta \tag{3.4}$$

Let $R_1, R_2, R_3, \ldots, R_j, \ldots$ be the random RR sets generated in either SSA or D-SSA algorithms. Given a subset of nodes $S \subset V$, define $Z_j = \min\{|R_j \cap S|, 1\}$, the Bernouli random variable with mean $\mathcal{E}[Z_j] = \mathbb{I}(S)/n$. Further, define $Y_j = Z_j - \mathcal{E}[Z_j]$, then Y_j is a *martingale* [105], i.e., $\mathcal{E}[Y_i|Y_1, Y_2, \ldots, Y_{i-1}] = Y_{i-1}$ and $\mathcal{E}[Y_i] < +\infty$. This martingale view of Y_j is adopted from [105] to cope with the fact that Y_j might be weakly dependent due to the stopping conditions. Let $\hat{\mu}_Z = \frac{1}{T}\sum_{i=1}^{T} Z_i$, an estimate of μ_Z. Corollaries 1 and 2 in [105] gives the following two concentration inequalities.

Lemma 18 ([105])**.** *For $T > 0$ and $\epsilon > 0$, the following inequalities hold,*

$$\Pr[\hat{\mu} > (1+\epsilon)\mu] \leq \exp\left(\frac{-T\mu\epsilon^2}{2+\frac{2}{3}\epsilon}\right), \tag{3.5}$$

$$\Pr[\hat{\mu} < (1-\epsilon)\mu] \leq \exp\left(\frac{-T\mu\epsilon^2}{2}\right). \tag{3.6}$$

Equivalently, we can derive from Lem. 18 the sufficient number of samples to provide

an (ϵ, δ)-approximation.

Corollary 1. *For fixed* $\epsilon > 0$ *and* $\delta \in (0, 1)$*,*

$$\Pr[\hat{\mu} > (1+\epsilon)\mu] \leq \delta, \textit{if } T \geq \frac{2+\frac{2}{3}\epsilon}{\epsilon^2} \ln \frac{1}{\delta} \frac{1}{\mu} = \Upsilon(\epsilon, \delta)\frac{1}{\mu},$$
$$\Pr[\hat{\mu} < (1-\epsilon)\mu] \leq \delta, \textit{if } T \geq \frac{2}{\epsilon^2} \ln(\frac{1}{\delta})\frac{1}{\mu}.$$

3.1.1.2 RIS Framework and Thresholds

Based on Lem. 30, the IM problem can be solved by the following two-step algorithm.

- Generate a collection of RR sets, $\mathcal{R}$, from G.
- Use the greedy algorithm for the Max-coverage problem [56] to find a seed set $\hat{S}_k$ that covers the maximum number of RR sets and return $\hat{S}_k$ as the solution.

As mentioned, the core issue is to determine the minimum $\theta(\epsilon, \delta)$ given a predefined setting of ϵ, δ. For IM, this means *"How many RR sets are sufficient to provide a good approximate solution?"*. [106, 105] propose two such theoretical thresholds and two probing techniques to realistically estimate those thresholds. However, their thresholds are not known to be any kind of minimum and the probing method is *ad hoc* in [106] or far from the proposed threshold in [105]. Thus, they cannot provide any guarantee on the optimality of the number of samples generated.

We look into the cores of the techniques in [106, 105, 11, 89] and capture the essential conditions to achieve an $(1 - 1/e - \epsilon)$ approximation for Influence Maximization problem. By satisfying these critical conditions, we aim to achieve a better approach rather than the prescribing a explicit threshold θ as in previous work [106, 105, 11, 89].

RIS Critical conditions. Suppose that there is an optimal seed set S_k^*, which has the

maximum influence in the network[1]. Given $0 \leq \epsilon, \delta \leq 1$, our unified RIS framework enforces two conditions:

$$\Pr[\hat{\mathbb{I}}(\hat{S}_k) \leq (1+\epsilon_a)\mathbb{I}(\hat{S}_k)] \geq 1-\delta_a \tag{3.7}$$

and

$$\Pr[\hat{\mathbb{I}}(S_k^*) \geq (1-\epsilon_b)\mathsf{OPT}_k] \geq 1-\delta_b \tag{3.8}$$

where $\delta_a + \delta_b \leq \delta$ and $(1-\frac{1}{e})\frac{\epsilon_a+\epsilon_b}{1+\epsilon_a} \leq \epsilon$.

Based on the above conditions, we define the RIS threshold as the following.

Definition 9 (RIS Threshold)**.** *Given a graph G, $\epsilon_a \in (0,\infty)$ and $\epsilon_b, \delta_a, \delta_b \in (0,1)$, $N(\epsilon_a, \epsilon_b, \delta_a, \delta_b)$ is called an RIS Threshold in G w.r.t $\epsilon_a, \epsilon_b, \delta_a, \delta_b$, if $|\mathcal{R}| \geq N(\epsilon_a, \epsilon_b, \delta_a, \delta_b)$ implies Eqs. 3.7 and 3.8 hold together.*

The RIS threshold gives a *sufficient condition* to achieve a $(1-1/e-\epsilon)$-approximation as stated below. The proof of this theorem as well as those of latter lemmas/theorems are located in our appendix.

Theorem 7. *Given a graph G, $\epsilon_a \in [0,\infty)$, and $\epsilon_b, \delta_a, \delta_b \in (0,1)$, let $\epsilon = (1-\frac{1}{e})\frac{\epsilon_a+\epsilon_b}{1+\epsilon_a}$ and $\delta \geq \delta_a + \delta_b$, if the number of RR sets $|\mathcal{R}| \geq N(\epsilon_a, \epsilon_b, \delta_a, \delta_b)$, then the two-step algorithm in our RIS framework returns $\hat{S}_k$ satisfying*

$$\Pr[\mathbb{I}(\hat{S}_k) \geq (1-1/e-\epsilon)\mathsf{OPT}_k] \geq 1-\delta. \tag{3.9}$$

That is $\hat{S}_k$ is an $(1-1/e-\epsilon)$-approximate solution with high probability (w.h.p.)

Existing RIS thresholds. For any $\epsilon, \delta \in (0,1)$, Tang et al. established in [106] an

[1]If there are multiple optimal sets with influence, OPT_k, we choose the first one alphabetically to be S_k^*.

RIS threshold,

$$N\left(\frac{\epsilon}{2},\frac{\epsilon}{2},\frac{\delta}{2}\left(1-\binom{n}{k}^{-1}\right),\frac{\delta}{2}\binom{n}{k}\right)=(8+2\epsilon)n\frac{\ln 2/\delta+\ln\binom{n}{k}}{\epsilon^2\mathsf{OPT}_k}$$

In a later study [105], they reduced this number to another RIS threshold from Theorem 1 in [105],

$$N\left(\epsilon_1,\epsilon-\epsilon_1,\frac{\delta}{2}\left(1-\binom{n}{k}^{-1}\right),\frac{\delta}{2}\binom{n}{k}^{-1}\right)=2n\frac{((1-1/e)\alpha+\beta)^2}{\epsilon^2\mathsf{OPT}_k},$$

where $\alpha=(\ln\frac{2}{\delta})^{\frac{1}{2}}$, $\beta=\ (1-1/e)^{\frac{1}{2}}\left(\ln\frac{2}{\delta}+\ln\binom{n}{k}\right)^{\frac{1}{2}}$ and $\epsilon_1=\frac{\epsilon\cdot\alpha}{(1-1/e)\alpha+\beta}$.
Simplify the above equation, we have

$$\begin{aligned}&((1-1/e)\alpha+\beta)^2\leq 2((1-1/e)^2\alpha^2+\beta^2)\\ =\ &2(1-1/e)((1-1/e)\ln\frac{2}{\delta}+\ln\frac{2}{\delta}+\ln\binom{n}{k})\\ \leq\ &2(1-1/e)\left(2\ln\frac{2}{\delta}+\ln\binom{n}{k}\right).\end{aligned}$$

Thus, we obtain a simplified threshold,

$$N=4\left(1-\frac{1}{e}\right)n\frac{2\ln(2/\delta)+\ln\binom{n}{k}}{\epsilon^2\mathsf{OPT}_k} \tag{3.10}$$

$$\leq 8\left(1-\frac{1}{e}\right)\frac{\ln(2/\delta)+\ln\binom{n}{k}}{\epsilon^2}\frac{n}{k} \tag{3.11}$$

Unfortunately, computing OPT_k is intractable, thus, the proposed algorithms have to generate $\theta\frac{\mathsf{OPT}_k}{KPT^+}$ RR sets, where KPT^+ is the expected influence of a node set obtained by sampling k nodes with replacement from G and the ratio $\frac{\mathsf{OPT}_k}{KPT^+}\geq 1$ is not upper-bounded. That is they may generate many times more RR sets than needed as in [106].

3.1.1.3 Two Types of Minimum Thresholds

Based on the definition of RIS threshold, we now define two strong theoretical limits, i.e. type-1 minimum and type-2 minimum thresholds. In Section 4.4, we will prove that

our first proposed algorithm, SSA, achieves, within a constant factor, a type-1 minimum threshold and later, in Section 3.1.4, our dynamic algorithm, D-SSA, is shown to obtain, within a constant factor, the strongest type-2 minimum threshold.

If $N(\epsilon_a, \epsilon_b, \delta_a, \delta_b)$ is an RIS threshold, then any N such that $N \geq N(\epsilon_a, \epsilon_b, \delta_a, \delta_b)$ is also an RIS threshold. We choose the smallest number over all such RIS thresholds to be type-1 minimum as defined in Def. 10.

Definition 10 (type-1 minimum threshold)**.** *Given* $0 \leq \epsilon, \delta \leq 1$ *and* $\epsilon_a \in (0, \infty)$, $\epsilon_b, \delta_a, \delta_b \in (0, 1)$ *satisfying* $\delta_a + \delta_b \leq \delta$ *and* $(1 - \frac{1}{e})\frac{\epsilon_a + \epsilon_b}{1+\epsilon_a} \leq \epsilon$, $N^{(1)}_{min}(\epsilon_a, \epsilon_b, \delta_a, \delta_b)$ *is called a type-1 minimum threshold w.r.t* $\epsilon_a, \epsilon_b, \delta_a, \delta_b$ *if* $N^{(1)}_{min}(\epsilon_a, \epsilon_b, \delta_a, \delta_b)$ *is the smallest number of RR sets that satisfies both Eq. 3.7 and Eq. 3.8.*

All the previous methods [11, 106, 105] try to approximate $N^{(1)}_{min}(\epsilon_a, \epsilon_b, \delta_a, \delta_b)$ for some setting of $\epsilon_a, \epsilon_b, \delta_a, \delta_b$, however, they fail to provide any guarantee on how close their numbers are to that threshold. In contrast, we show that SSA achieves, within a constant factor, a type-1 minimum threshold in Section 4.4. Next, we give the definition of a stronger type-2 minimum threshold which is achieved by D-SSA as shown in Section 3.1.4.

Definition 11 (type-2 minimum threshold)**.** *Given* $0 \leq \epsilon, \delta \leq 1$, $N^{(2)}_{min}(\epsilon, \delta)$ *is called the type-2 minimum threshold if*

$$N^{(2)}_{min}(\epsilon, \delta) = \min_{\epsilon_a, \epsilon_b, \delta_a, \delta_b} N^{(1)}_{min}(\epsilon_a, \epsilon_b, \delta_a, \delta_b) \tag{3.12}$$

where $(1 - \frac{1}{e})\frac{\epsilon_a + \epsilon_b}{1+\epsilon_a} = \epsilon$ *and* $\delta_a + \delta_b = \delta$ *and* $\epsilon_a \in (0, \infty)$, $\epsilon_b, \delta_a, \delta_b \in (0, 1)$.

Type-2 minimum threshold is the tightest threshold that one can achieve using the RIS-framework.

3.1.2 Stop-and-Stare Algorithm (SSA)

In this section, we present Stop-and-Stare Algorithm (SSA), the first approximation algorithm that meets (asymptotically) a type-1 minimum threshold.

Algorithm 10: SSA Algorithm

Input: Graph G, $0 \leq \epsilon, \delta \leq 1$, and a budget k

Output: An $(1 - 1/e - \epsilon)$-optimal solution, $\hat{S}_k$ with at least $(1 - \delta)$-probability

Choose $\epsilon_1, \epsilon_2, \epsilon_3$ satisfying Eqs. 3.14. For example, recommended values for $\epsilon_1, \epsilon_2, \epsilon_3$ are in Eq. 3.15

$N_{max} = 8\frac{1-1/e}{2+2\epsilon/3}\Upsilon\left(\epsilon, \frac{\delta}{6}/\binom{n}{k}\right)\frac{n}{k}$; $i_{max} = \lceil \log_2 \frac{2N_{max}}{\Upsilon(\epsilon,\delta/3)} \rceil$;

$\Lambda = \Upsilon(\epsilon, \frac{\delta}{3i_{max}})$; $\Lambda_1 \leftarrow (1+\epsilon_1)(1+\epsilon_2)\Upsilon(\epsilon_3, \frac{\delta}{3i_{max}})$

$\mathcal{R} \leftarrow$ Generate Λ random RR sets

repeat

  Double the size of $\mathcal{R}$ with new random RR sets

  $<\hat{S}_k, \hat{\mathbb{I}}(\hat{S}_k)> \leftarrow$ Max-Coverage($\mathcal{R}$, k, n)

  if $\mathrm{Cov}_{\mathcal{R}}(\hat{S}_k) \geq \Lambda_1$ **then** ▷ *[f]Condition C1

    $\delta_2' = \frac{\delta_2}{3i_{max}}$; $T_{max} = 2|\mathcal{R}|\frac{1+\epsilon_2}{1-\epsilon_2}\frac{\epsilon_3^2}{\epsilon_2^2}$

    $\mathbb{I}_c(\hat{S}_k) \leftarrow$ Estimate-Inf($G, \hat{S}_k, \epsilon_2, \delta_2', T_{max}$)

    if $\hat{\mathbb{I}}(\hat{S}_k) \leq (1+\epsilon_1)\mathbb{I}_c(\hat{S}_k)$ **then** ▷ *[f]Condition C2

      return $\hat{S}_k$

    end

  end

until $|\mathcal{R}| \geq N_{max}$;

return $\hat{S}_k$

3.1.2.1 SSA Algorithm

At a high level, SSA, presented in Alg. 10, consists of multiple iterations. In each iteration, it follows the RIS framework to generate (additional) RR sets and uses the Max-Coverage (Alg. 11) to find a candidate solution $\hat{S}_k$. If $\hat{S}_k$ passes the quality check, Lines 8-12, the algorithm stops and outputs $\hat{S}_k$. Otherwise, it doubles the number of RR sets and advances to the next iteration. The name Stop-and-Stare is based on the view that the algorithm "scans" through a stream of samples and *stops* at exponential check points to *stare* at the the generated samples to see if it can find a provably good solution. We enforce a nominal cap on the number of samples $N_{max} = 8\frac{1-1/e}{2+2\epsilon/3}\Upsilon\left(\epsilon, \frac{\delta}{6}/\binom{n}{k}\right)\frac{n}{k}$. Thus, the number of iterations is at most $i_{max} = \lceil \log_2 \frac{2N_{max}}{\Upsilon(\epsilon,\delta/3)} \rceil = O(\log_2 n)$ (Line 2).

Specifically, the algorithm starts by determining parameters $\epsilon_1, \epsilon_2, \epsilon_3$ satisfying $(1 - \frac{1}{e})\frac{\epsilon_a+\epsilon_b}{1+\epsilon_a} = \epsilon$ (Line 1). For each iteration $t = 1, 2, \ldots, i_{max}$, SSA doubles the number of generated RR sets in $\mathcal{R}$. Thus, the number of samples at an iteration t is $|\mathcal{R}| = \Lambda 2^{t-1}$, where $\Lambda = \Upsilon(\epsilon, \delta/(3i_{max}))$. After that, SSA invokes Max-Coverage (Alg. 11) to find a candidate solution $\hat{S}_k$ and its influence estimation

$$\hat{\mathbb{I}}(\hat{S}_k) = \frac{\mathrm{Cov}_{\mathcal{R}}(\hat{S}_k)n}{|\mathcal{R}|}.$$

The condition $\mathrm{Cov}_{\mathcal{R}}(\hat{S}_k) \geq \Lambda_1$ (Line 8) is to guarantee that there are sufficient samples to estimate the influence accurately within a relative error ϵ_3. If the condition is met, SSA independently generates another collection of RR sets $\mathcal{R}'$ in Estimate-Inf (Alg. 12) to obtain an accurate estimation of $\hat{S}_k$ influence (with a relative error ϵ_2). This estimation is compared against $\hat{\mathbb{I}}(\hat{S}_k)$ and the SSA stops when the two estimations are close (Line 11), i.e., when

$$\hat{\mathbb{I}}(\hat{S}_k) \leq (1 + \epsilon_1)\mathbb{I}_c(\hat{S}_k).$$

Stopping conditions. Ignore the rare case that SSA reaches the cap N_{max} on the

number of samples. SSA stops only when the following two *stopping conditions* are met.

(C1) The 1st condition $\text{Cov}_{\mathcal{R}}(\hat{S}_k) \geq \Lambda_1$ (Line 8) ensures that the influence of S_k^* can be estimated with a relative error at most ϵ_3 as shown in Lems. 21 and 22.

(C2) The 2nd condition $\hat{\mathbb{I}}(\hat{S}_k) \leq (1+\epsilon_1)\mathbb{I}_c(\hat{S}_k)$ (Line 11) guarantee that the estimation $\hat{\mathbb{I}}(\hat{S}_k)$ is not far from the error-bounded estimation $\mathbb{I}_c(\hat{S}_k)$ returned by the Estimate-Inf procedure. Recall that $\mathbb{I}_c(\hat{S}_k)$ has a relative error at most ϵ_2 comparing to the true influence $\mathbb{I}(\hat{S}_k)$.

As we will prove in Sec. 4.4, the two stopping conditions are sufficient to guarantee the $(1-1/e-\epsilon)$-approx. of $\hat{S}_k$.

Algorithm 11: Max-Coverage procedure

Input: RR sets ($\mathcal{R}$), k and number of nodes (n)

Output: An $(1-1/e)$-optimal solution, $\hat{S}_k$ and its estimated influence $\mathbb{I}_c(\hat{S}_k)$

$\hat{S}_k = \emptyset$

for $i = 1 : k$ **do**

  $\hat{v} \leftarrow \arg\max_{\{v \in V\}}(\text{Cov}_{\mathcal{R}}(\hat{S}_k \cup \{v\}) - \text{Cov}_{\mathcal{R}}(\hat{S}_k))$

  Add $\hat{v}$ to $\hat{S}_k$

end

return $<\hat{S}_k, \text{Cov}_{\mathcal{R}}(\hat{S}_k) \cdot n/|\mathcal{R}|>$

Finding Max-Coverage. Standard greedy algorithm in Max-coverage is used to find $\hat{S}_k$. The algorithm repeatedly selects node u with maximum marginal gain, the number of RR sets that are covered by u but not the previously selected nodes. The well-known result in [81] states that $\text{Cov}_{\mathcal{R}}(\hat{S}_k)$ is at leat $(1-1/e)$ the maximum coverage obtained by any size-k seed set. This algorithm can be implemented in linear time in terms of the total size of the RR sets [11].

Influence Estimation. Estimate-Inf, presented in Alg. 12, gives an estimation $\mathbb{I}_c(S)$ with one-side error guarantee

$$\Pr[\mathbb{I}_c(S) \leq (1+\epsilon')\mathbb{I}(S)] \geq 1-\delta'.$$

Algorithm 12: Estimate-Inf procedure

Input: A seed set $S \subset V$, $\epsilon' > 0$, $\delta' \in (0,1)$ and maximum number of samples, T_{max}

Output: $\mathbb{I}_c(S)$ or -1 if exceeds T_{max} samples.

$\Lambda_2 = 1 + (1+\epsilon')\Upsilon(\epsilon', \delta')$

$\mathrm{Cov} = 0$

for $T = 1 : T_{max}$ **do**

  Generate $R_j \leftarrow$ RIS(G)

  $\mathrm{Cov} = \mathrm{Cov} + \min\{|R_j \cap S|, 1\}$

  if $\mathrm{Cov} \geq \Lambda_2$ **then**

    return $n\Lambda_2/T$; // n: number of nodes

  end

end

return -1 ; // Exceeding T_{max} RR sets

The algorithm generates RR sets R_j and counts the number of "successes", defined as the number of RR sets that intersect with S. When the number of successes reaches $\Lambda_2 = 1 + (1+\epsilon')\Upsilon(\epsilon', \delta')$, the algorithm returns $\mathbb{I}_c(S) = \frac{\Lambda_2 n}{T}$, where T is the number of generated RR sets.

Estimate-Inf is based on the Stopping-Rule algorithm in [26] with an important difference. The algorithm stops and return -1 if T_{max} samples has been generated. Choosing T_{max} proportional to the number of samples in $\mathcal{R}$ (Line 9, SSA) avoid time-wasting on es-

timating influence for $\hat{S}_k$ at early iterations in SSA. Those early $\hat{S}_k$ candidates often have small influence, thus, require up to $\Omega(n)$ samples to estimate. Without the cap T_{max}, SSA will suffer a quadratic (or worse) time complexity.

Similar to the proof of the stopping theorem in [26], we obtain the following lemma with the proof in the appendix.

Lemma 19. *When Estimate-Inf terminates within T_{max} samples, the returned estimation $\mathbb{I}_c(S)$ satisfies*

$$\Pr[\mathbb{I}_c(S) \leq (1+\epsilon')\mathbb{I}(S)] \geq 1-\delta'. \tag{3.13}$$

In SSA Lines 9 and 10, Estimate-Inf is invoked with the parameters $\epsilon' = \epsilon_2$, $\delta' = \delta_2/(3i_{max})$, and $T_{max} = \Theta(|\mathcal{R}|)$.

3.1.2.2 Parameter Settings for SSA

In SSA, we can select arbitrary $\epsilon_1, \epsilon_2, \epsilon_3 \in (0,1)$ as long as they satisfy the following,

$$(1-\frac{1}{e})\frac{\epsilon_1+\epsilon_2+\epsilon_1\epsilon_2+\epsilon_3}{(1+\epsilon_1)(1+\epsilon_2)} \leq \epsilon \tag{3.14}$$

In practice, the selection of ϵ_1, ϵ_2 and ϵ_3 has considerate effect on the running time. Through our experiments, we observe good performance yields when

- $\epsilon_1 > \epsilon \approx \epsilon_3$ for small networks
- $\epsilon_1 \approx \epsilon \approx \epsilon_3$ for moderate network (few million edges)
- $\epsilon_1 \ll \epsilon_2 \approx \epsilon_3$ for large networks (hundreds of millions of edges).

For simplicity, we use the following *default setting* for SSA.

$$\epsilon_2 = \epsilon_3 = (1 - 1/e)^{-1}\epsilon/2 \tag{3.15}$$

$$\epsilon_1 = \frac{1 + (1 - 1/e - \epsilon)^{-1}\epsilon/2}{1 + \epsilon_2} - 1. \tag{3.16}$$

For example, when $\epsilon = 0.1$ we can set

$$\epsilon_1 = 1/78, \epsilon_2 = \epsilon_3 = 2/25. \tag{3.17}$$

In Sect. 3.1.4, we will later propose D-SSA, a Stop-and-Stare algorithm with "dynamic" parameters. D-SSA can automatically select a near-optimal setting of $\epsilon_1, \epsilon_2, \epsilon_3$.

3.1.3 SSA Theoretical Analysis

In this section, we will prove that SSA returns a $(1 - 1/e - \epsilon)$-approximate solution w.h.p. in Subsec. 6.1.2.2. Subsequently, SSA is shown to require no more than a constant factor of a type-1 minimum threshold of RR sets w.h.p. in Subsec. 3.1.3.2.

3.1.3.1 Approximation Guarantee

We will prove that SSA returns a $(1 - 1/e - \epsilon)$-approximate solution $\hat{S}_k$ w.h.p. The major portion of the proof is to bound the probabilities of the following three bad events

1. $|R| \geq N_{max}$ and $\mathbb{I}(\hat{S}_k) < (1 - 1/e - \epsilon)$
2. The error in the estimation $\mathbb{I}_c(\hat{S}_k)$ exceeds ϵ_2 (Lem. 21)
3. The error in the estimation $\hat{\mathbb{I}}(S_k^*)$, the estimation of the OPT_k, exceeds ϵ_3 (Lem. 22).

Finally, Theorem 24, assuming none of the bad events happen, shows that $\mathbb{I}(\hat{S}_k) \geq (1 - 1/e - \epsilon)\mathsf{OPT}_k$.

The probability of the first bad event follows directly from the threshold θ in Eq. 3.10 with δ replaced by $\delta/3$.

Lemma 20. *We have*

$$\Pr[|\mathcal{R}| \geq N_{max} \text{ and } \mathbb{I}(\hat{S}_k) < (1 - 1/e - \epsilon)\mathsf{OPT}_k] \leq \delta/3.$$

Since, we do not know the iteration that SSA will stop, we will bound the probabilities of the other two bad events for all iterations. The bound on the relative error of $\mathbb{I}_c(\hat{S}_k)$:

Lemma 21. *For any iteration $t = 1, 2, \ldots, i_{max}$ in SSA,*

$$\Pr[\mathbb{I}_c(\hat{S}_k) > (1 + \epsilon_2)\mathbb{I}(\hat{S}_k)] \leq \delta/3i_{max}. \tag{3.18}$$

Proof. The inequality holds trivially if Estimate-Inf return -1. Otherwise, it follows from Lem. 19 with $\epsilon' = \epsilon_2$, $\delta' = \delta_2/(3i_{max})$. □

Since $|\mathcal{R}| = \Lambda 2^{t-1}$ is fixed, we apply the Chernoff's bound in Lem. 18 over $|\mathcal{R}|$ random variables to obtain the following error bound on the estimation of $\hat{\mathbb{I}}(S_k^*)$.

Lemma 22. *For any iteration $i = 1, 2, \ldots, i_{max}$ in SSA,*

$$\Pr[\hat{\mathbb{I}}(S_k^*) < (1 - \epsilon_3^{(i)})\mathsf{OPT}_k] \leq \delta/(3i_{max}) \tag{3.19}$$

where $\epsilon_3^{(i)} = \sqrt{\frac{2n \ln \frac{3i_{max}}{\delta}}{|\mathcal{R}|\mathsf{OPT}_k}}$, and $|\mathcal{R}| = \Lambda 2^{t-1}$ at iteration i.

Lem. 21 and 22 are sufficient to prove the approximation guarantee of SSA as stated by the following theorem.

Theorem 8. *Given $0 \leq \epsilon, \delta \leq 1$, SSA returns a seed set $\hat{S}_k$ satisfying*

$$\Pr[\mathbb{I}(\hat{S}_k) \geq (1 - 1/e - \epsilon)\mathsf{OPT}_k] \geq 1 - \delta. \tag{3.20}$$

3.1.3.2 Achieving Type-1 Minimum Threshold

We will show that for any $\epsilon_a, \epsilon_b, \delta_a, \delta_b$ satisfying the conditions of RIS threshold (Def. 9), there exists a setting of $\epsilon_1, \epsilon_2, \epsilon_3$ such that SSA stops within $O(N_{min}^{(1)}(\epsilon_a, \epsilon_b, \delta_a, \delta_b))$

samples (w.h.p.)

We need to bound the total number of RR sets generated by SSA. Recall that SSA generates two different types of RR sets: 1) RR sets in $\mathcal{R}$ to find $\hat{S}_k$ through solving Max-Coverage and 2) RR sets in Estimate-Inf for the stopping condition C2. At each iteration, the number of type 2 RR sets is at most $2\frac{1+\epsilon_2}{1-\epsilon_2}\frac{\epsilon_3^2}{\epsilon_2^2}|\mathcal{R}| = \Theta(|\mathcal{R}|)$. Thus, the core part is to prove that: *"SSA will stop w.h.p. when $|\mathcal{R}| = O(N_1(\epsilon_a, \epsilon_b, \delta_a, \delta_b))$"*.

Our assumptions. Under the assumptions that make the Chernoff's bound (Lem. 18) tight up to a constant in the exponent, we show that SSA stops within $O(N^{(1)}_{min}(\epsilon_a, \epsilon_b, \delta_a, \delta_b))$. The assumptions, referred to as the *range conditions*, are as follows.

- $\mathsf{OPT}_k \leq \frac{1}{2}|V|$. That is no k nodes can influence more than half of the nodes in the network. This assumption guarantees $\mu \leq 1/2$, needed for the tightness of Chernoff's bound in Lem. 13 in the appendix.

- $\epsilon \leq 1/4$. The constant $1/4$ can be replaced by any constant $c < 1$, assuming δ is sufficiently small. This assumption guarantees that $\epsilon_b \leq 1/2$, which is also needed for Lem. 13.

- $1/\delta = \Omega(n)$. This assumption guarantee that δ is sufficiently small (Lem. 13). This is compatible with the settings in the previous works [106, 105, 89], in which $\delta = 1/n$.

Consider positive $\epsilon_a, \epsilon_b, \delta_a, \delta_b \in (0, 1)$ satisfying

$$(1 - \frac{1}{e})\frac{\epsilon_a + \epsilon_b}{1 + \epsilon_a} = \epsilon \leq \frac{1}{4} \text{ and} \tag{3.21}$$

$$\delta_a + \delta_b = \delta < \frac{1}{\log_2 n}. \tag{3.22}$$

We will determine suitable parameters $\epsilon_1, \epsilon_2, \epsilon_3$ for SSA so that

$$(1 - \frac{1}{e})\frac{\epsilon_1 + \epsilon_2 + \epsilon_1\epsilon_2 + \epsilon_3}{(1 + \epsilon_1)(1 + \epsilon_2)} = \epsilon. \tag{3.23}$$

Denote $T_1 = N^{(1)}_{min}(\epsilon_a, \epsilon_b, \delta_a, \delta_b)$. From Def. 10 of the type-1 threshold, $|\mathcal{R}| \geq T_1$ leads to

$$\Pr[\hat{\mathbb{I}}_{\mathcal{R}}(\hat{S}_k) > (1+\epsilon_a)\mathbb{I}(\hat{S}_k)] \leq \delta_a \text{ and} \tag{3.24}$$

$$\Pr[\hat{\mathbb{I}}_{\mathcal{R}}(S^*_k) < (1-\epsilon_b)\mathsf{OPT}_k] \leq \delta_b. \tag{3.25}$$

An upper bound on the number of RR sets needed in $\mathcal{R}$ is given in the following lemma.

Lemma 23. *Let* $\epsilon_0 = \min\{\epsilon_2, \epsilon_3, \epsilon_b\}$*, and*

$$T_{\mathsf{SSA}} = \max\{T_1, \alpha\Upsilon(\epsilon_0, \frac{\delta}{3i_{max}})\frac{n}{\mathsf{OPT}_k}\},$$

for some constant $\alpha > 1$*. Under the range conditions,*

$$T_{\mathsf{SSA}} = O(T_1).$$

Now we bound the estimation error in the Estimate-Inf procedure at each iteration. At iteration $1 \leq i \leq i_{max}$,

$$T_{max} = 2|\mathcal{R}|\frac{1+\epsilon_2}{1-\epsilon_2}\frac{\epsilon_3^2}{\epsilon_2^2} = 2^i\Lambda\frac{1+\epsilon_2}{1-\epsilon_2}\frac{\epsilon_3^2}{\epsilon_2^2} \tag{3.26}$$

is a fixed number. Denote by $\mathcal{R}_c$, the set of RR sets generated in Estimate-Inf. Apply the concentration inequality in Eq. (3.5), for T_{max} RR sets in $\mathcal{R}_c$ we have

Lemma 24. *For iteration* $1 \leq i \leq i_{max}$ *in SSA, let* $\epsilon_2^{(i)} = \sqrt{\frac{(\ln 1/\delta + \ln 3i_{max})n}{T_{max}}}$*. The following holds*

$$\Pr[(|\mathcal{R}_c| \geq T_{max}) \text{ and } \hat{\mathbb{I}}_{\mathcal{R}_c}(\hat{S}_k) < (1-\epsilon_2^{(i)})\mathbb{I}(\hat{S}_k)] \leq \frac{\delta}{3i_{max}}.$$

Theorem 9. *Consider* $\epsilon_a, \epsilon_b, \delta_a, \delta_b$ *satisfying Eqs. (3.21) and (3.22). Under the range conditions, there exist SSA parameters* $\epsilon_1, \epsilon_2, \epsilon_3$*, satisfying Eq. (3.23), and a constant* $c > 1$ *such that if* $|\mathcal{R}| \geq cN^{(1)}_{min}(\epsilon_a, \epsilon_b, \delta_a, \delta_b)$*, SSA will stop w.h.p.*

Similarly, we can show the reverse direction.

Theorem 10. *Consider SSA's with $\epsilon_1, \epsilon_1, \epsilon_2, \epsilon_3$, satisfying Eq. (3.23) and $\epsilon_2 \leq \frac{\epsilon_1}{1+\epsilon_1}$. Under the range conditions, there exist $\epsilon_a, \epsilon_b, \delta_a, \delta_b$ satisfying Eqs. (3.21) and (3.22), and a constant $c > 1$ such that if $|\mathcal{R}| \geq cN_1(\epsilon_a, \epsilon_b, \delta_a, \delta_b)$, SSA will stop w.h.p.*

The proof is similar to that of Theorem 9 and is omitted.

SSA limitation. First, the performance of SSA depends on the selection of the parameters $\epsilon_1, \epsilon_2, \epsilon_3$. While the presetting in Eq. 3.15 provides decent performance for most cases, there will be certain input that results in less than ideal performance. Secondly, the samples in $\mathcal{R}'$, the sample pool to verify the quality of the candidate solution $\hat{S}_k$ are not used efficiently. They are only used once and then discarded. Alternative strategies that reuse the sample in $\mathcal{R}'$ may potentially reduce the number of the generated samples and provide better performance.

3.1.4 Dynamic Stop-and-Stare Algorithm

In this section, we present D-SSA, a stop-and-stare algorithm that automatically selects near-optimal $\epsilon_1, \epsilon_2, \epsilon_3$ settings. That is the sample size of D-SSA meets, asymptotically, the type-2 minimum threshold, the strongest guarantee for methods following the RIS framework.

The algorithm D-SSA, summarized in Alg. 13, works on a *single stream* of RR sets $R_1, R_2, ..., R_i, ...$ The algorithm consists of multiple iterations $t = 1, 2, \ldots, t_{max}$, where $t_{max} = O(\log n)$ is the maximum number of iterations.

At an iteration t, the algorithm looks into the first $\Lambda \times 2^t$ RR sets, for a fixed Λ (Line 3), and divide those samples into two halves.

- The first half $\mathcal{R}_t = \{R_1, \ldots, R_{\Lambda 2^{t-1}}\}$ will be used to find the candidate solution $\hat{S}_k$ via solving a max-coverage problem Max-Coverage$(\mathcal{R}_t, k)$.

Algorithm 13: D-SSA Algorithm

Input: Graph G, $0 \leq \epsilon, \delta \leq 1$, and k

Output: An $(1 - 1/e - \epsilon)$-optimal solution, $\hat{S}_k$

$N_{max} = 8\frac{1-1/e}{2+2\epsilon/3}\Upsilon\left(\epsilon, \frac{\delta}{6}/\binom{n}{k}\right)\frac{n}{k}$;

$t_{max} = \lceil \log_2(2N_{max}/\Upsilon(\epsilon, \frac{\delta}{3})) \rceil$; $t = 0$;

$\Lambda = \Upsilon(\epsilon, \frac{\delta}{3t_{max}})$; $\Lambda_1 = 1 + (1+\epsilon)\Upsilon(\epsilon, \frac{\delta}{3t_{max}})$;

repeat

  $t \leftarrow t + 1$;

  $\mathcal{R}_t = \{R_1, \ldots, R_{\Lambda 2^{t-1}}\}$;

  $\mathcal{R}_t^c = \{R_{\Lambda 2^{t-1}+1}, \ldots, R_{\Lambda 2^t}\}$;

  $< \hat{S}_k, \hat{\mathbb{I}}_t(\hat{S}_k) > \leftarrow$ Max-Coverage$(\mathcal{R}_t, k)$;

  if $\mathrm{Cov}_{\mathcal{R}_t^c}(\hat{S}_k) \geq \Lambda_1$ **then** ▷ Condition D1

    $\mathbb{I}_t^c(\hat{S}_k) \leftarrow \mathrm{Cov}_{\mathcal{R}_t^c}(\hat{S}_k) \cdot n/|\mathcal{R}_t^c|$

    $\epsilon_1 \leftarrow \hat{\mathbb{I}}_t(\hat{S}_k)/\mathbb{I}_t^c(\hat{S}_k) - 1$; $\epsilon_2 \leftarrow \epsilon\sqrt{\frac{n(1+\epsilon)}{2^{t-1}\mathbb{I}_t^c(\hat{S}_k)}}$; $\epsilon_3 \leftarrow \epsilon\sqrt{\frac{n(1+\epsilon)(1-1/e-\epsilon)}{(1+\epsilon/3)2^{t-1}\mathbb{I}_t^c(\hat{S}_k)}}$

    $\epsilon_t = (\epsilon_1 + \epsilon_2 + \epsilon_1\epsilon_2)(1 - 1/e - \epsilon) + (1 - \frac{1}{e})\epsilon_3$

    if $\epsilon_t \leq \epsilon$ **then** ▷ Condition D2

      return $\hat{S}_k$

    end

  end

until $|\mathcal{R}_t| \geq N_{max}$;

return $\hat{S}_k$;

- The second half $\mathcal{R}_t^c = \{R_{\Lambda 2^{t-1}+1}, \ldots, R_{\Lambda 2^t}\}$ will be used to verify the quality of the candidate solution $\hat{S}_k$.

Note that $\mathcal{R}_{t+1} = \mathcal{R}_t \cup \mathcal{R}_t^c$, thus, the samples used in verifying $\hat{S}_k$ will be reused to find the candidate solution in next iteration.

To verify whether $\hat{S}_k$ meets the approximation guarantee with high probability (whp),

D-SSA, in Line 9, will first apply the stopping rule condition in [26] to check if the number of samples in $\mathcal{R}_t^c$ are sufficient to guarantee an $(\epsilon, \frac{\delta}{3t_{max}})$-approximation of $\mathbb{I}(\hat{S}_k)$. If not, it advances to the next iteration. Otherwise, it will automatically estimate the best possible precision parameters $\epsilon_1, \epsilon_2, \epsilon_3$ in Lines 11 and 12. Once the combination of those precision parameter is sufficiently small, i.e.,

$$\epsilon_t = (\epsilon_1 + \epsilon_2 + \epsilon_1\epsilon_2)(1 - 1/e - \epsilon) + (1 - 1/e)\epsilon_3 \leq \epsilon,$$

the algorithm returns $\hat{S}_k$ as an $(1 - 1/e - \epsilon)$-approximation solution (whp).

In the unfortunate event that the algorithm does not meet the condition $\epsilon_t \leq \epsilon$ for any t, it will terminate when the number of samples in the algorithm reaches to the cap $N_{max} = 8\frac{1-1/e}{2+2\epsilon/3}\Upsilon\left(\epsilon, \frac{1}{6}\delta/\binom{n}{k}\right)\frac{n}{k}$.

3.1.4.1 Theoretical Guarantees Analysis

We will subsequently show that D-SSA achieves the $(1 - 1/e - \epsilon)$-approximation factor (whp) in Subsec. 3.1.4.1 and requires only, to within a constant factor, the strongest type-2 minimum threshold of the RR sets (whp) in Subsec. 3.1.4.1.

Approximation Guarantee: We will show that D-SSA returns a $(1-1/e-\epsilon)$ solution with probability at least $1 - \delta$. For clarity, we present most of the proofs in the appendix.

Recall that D-SSA stops when either 1)the number of samples exceeds the cap, i.e., $|\mathcal{R}_t| \geq N_{max}$ or 2) $\epsilon_t \leq \epsilon$ for some $t \geq 1$. In the first case, N_{max} were chosen to guarantee that $\hat{S}_k$ will be a $(1 - 1/e - \epsilon)$-approximation solution w.h.p.

Lemma 25. *Let $B^{(1)}$ be the bad event that*

$$B^{(1)} = (|\mathcal{R}_t| \geq N_{max}) \cap (\mathbb{I}(\hat{S}_k) < (1 - 1/e - \epsilon)\mathsf{OPT}_k).$$

We have

$$\Pr[B^{(1)}] \leq \delta/3.$$

In the second case, the algorithm stops when $\epsilon_t \leq \epsilon$ for some $1 \leq t \leq t_{max}$. The maximum number of iterations t_{max} is bounded by $O(\log n)$ as stated below.

Lemma 26. *The number of iterations in D-SSA is at most* $t_{max} = O(\log n)$.

For each iteration t, we will bound the probabilities of the bad events that lead to inaccurate estimations of $\mathbb{I}(\hat{S}_k)$ through $\mathcal{R}_t^c$, and $\mathbb{I}(S_k^*)$ through $\mathcal{R}_t$(Lines 9 and 12).

Lemma 27. *For each* $1 \leq t \leq t_{max}$*, let*

$$\hat{\epsilon}_t \text{ be the unique root of } f(x) = \frac{\delta}{3t_{max}},$$

where $f(x) = \exp\left(-\frac{N_t \frac{\mathbb{I}(\hat{S}_k)}{n} x^2}{2+2/3x}\right)$*, and*

$$\epsilon_t^* = \epsilon\sqrt{\frac{n}{(1+\epsilon/3)2^{t-1}\mathsf{OPT}_k}}.$$

Consider the following bad events

$$B_t^{(2)} = \left(\hat{\mathbb{I}}_t^{(c)}(\hat{S}_k) > (1+\hat{\epsilon}_t)\mathbb{I}(\hat{S}_k)\right),$$
$$B_t^{(3)} = \left(\hat{\mathbb{I}}_t(S_k^*) < (1-\epsilon_t^*)\mathsf{OPT}_k]\right).$$

We have

$$\Pr[B_t^{(2)}], \Pr[B_t^{(3)}] \leq \frac{\delta}{3t_{max}}.$$

Lemma 28. *Assume that none of the bad events* $B^{(1)}$*,* $B_t^{(2)}$*,* $B_t^{(3)}$ *(*$t = 1..t_{max}$*) happen and D-SSA stops with some* $\epsilon_t \leq \epsilon$*. With* $\hat{\epsilon}_t$ *defined in Lem. 27, we have,*

$$\hat{\epsilon}_t < \epsilon \text{ and consequently} \tag{3.27}$$
$$\mathbb{I}_t^{(c)}(\hat{S}_k) \leq (1+\hat{\epsilon}_t)\mathbb{I}(\hat{S}_k) \leq (1+\epsilon)\mathbb{I}(\hat{S}_k) \tag{3.28}$$

We now achieve the approximation guarantee of D-SSA.

Theorem 11. *D-SSA returns an* $(1 - 1/e - \epsilon)$*-approximate solution with probability at least* $(1 - \delta)$.

Achieving the Type-2 Minimum Threshold: Denote by $T_2 = N^{(2)}_{min}(\epsilon, \delta)$, the type-2 minimum threshold defined in Def. 11. Under the *range conditions*, we will prove that D-SSA meets the Type-2 minimum threshold, i.e., it requires $O(T_2)$ samples w.h.p. This is the strongest efficiency guarantee for algorithms following the RIS framework.

The proof is based on the observation that there must exist $\epsilon^*_a, \epsilon^*_b, \delta^*_a, \delta^*_b$ that $N^{(1)}_{min}(\epsilon^*_a, \epsilon^*_b, \delta^*_a, \delta^*_b) = T_2$. Further, within $O(T_2)$ samples, we will have $\epsilon_2, \epsilon_3 \leq \epsilon^*_b/3$ and $\epsilon_1 \approx \epsilon^*_a$. Then both conditions D1 ($\mathrm{Cov}_{\mathcal{R}^c_t}(\hat{S}_k) \geq \Lambda_1$) and D2 ($\epsilon_t \leq \epsilon$) will be met and the algorithm will stop w.h.p.

Theorem 12. *Given* ϵ, δ*, assume the* range conditions *D-SSA will stop w.h.p within* $O(N^{(2)}_{min}(\epsilon, \delta))$ *samples.*

3.1.5 Experiments

Backing by the strong theoretical results, we will experimentally show that SSA and D-SSA outperform the existing state-of-the-art IM methods by a large margin. Specifically, SSA and D-SSA are several orders of magnitudes faster than IMM and TIM+, the best existing IM methods with approximation guarantee, while having the same level of solution quality. SSA and D-SSA also require several times less memory than the other algorithms. To demonstrate the applicability of the proposed algorithms, we apply our methods on a critical application of IM, i.e., Targeted Viral Marketing (TVM) introduced in [71] and show the significant improvements in terms of performance over the existing methods.

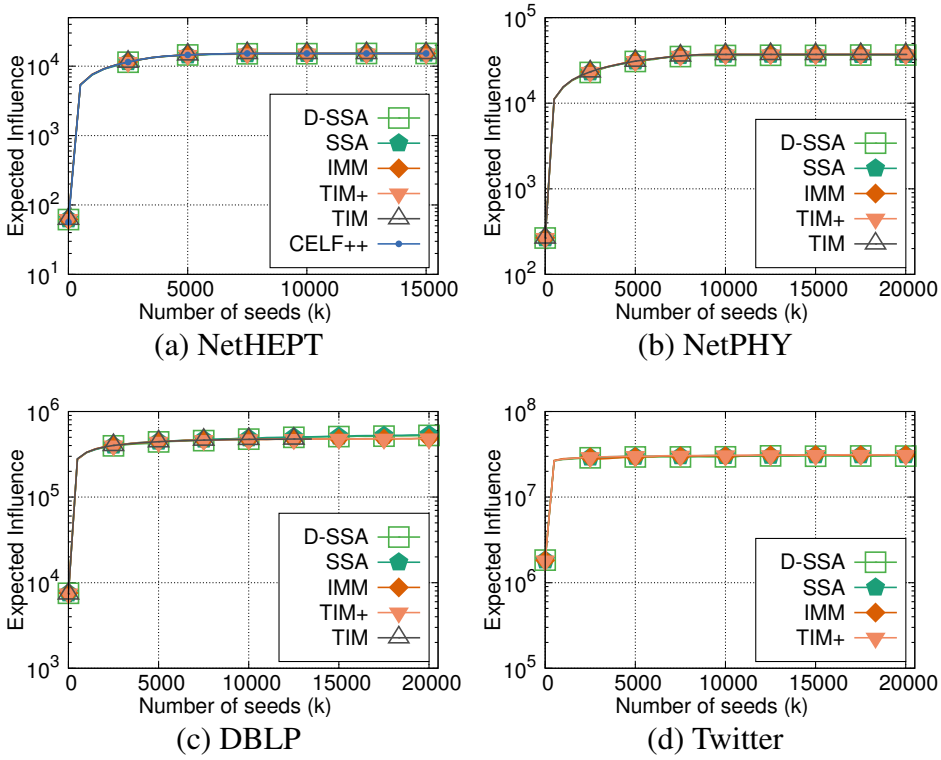


Fig. 13.: Expected Influence under LT model.

Table 13.: Datasets' Statistics

Dataset	**#Nodes**	**#Edges**	**Avg. degree**
NetHELP[2]	15K	59K	4.1
NetPHY[2]	37K	181K	13.4
Enron[2]	37K	184K	5.0
Epinions[2]	132K	841K	13.4
DBLP[2]	655K	2M	6.1
Orkut[2]	3M	234M	78
Twitter [65]	41.7M	1.5G	70.5
Friendster[2]	65.6M	**3.6G**	54.8

[2]From http://snap.stanford.edu

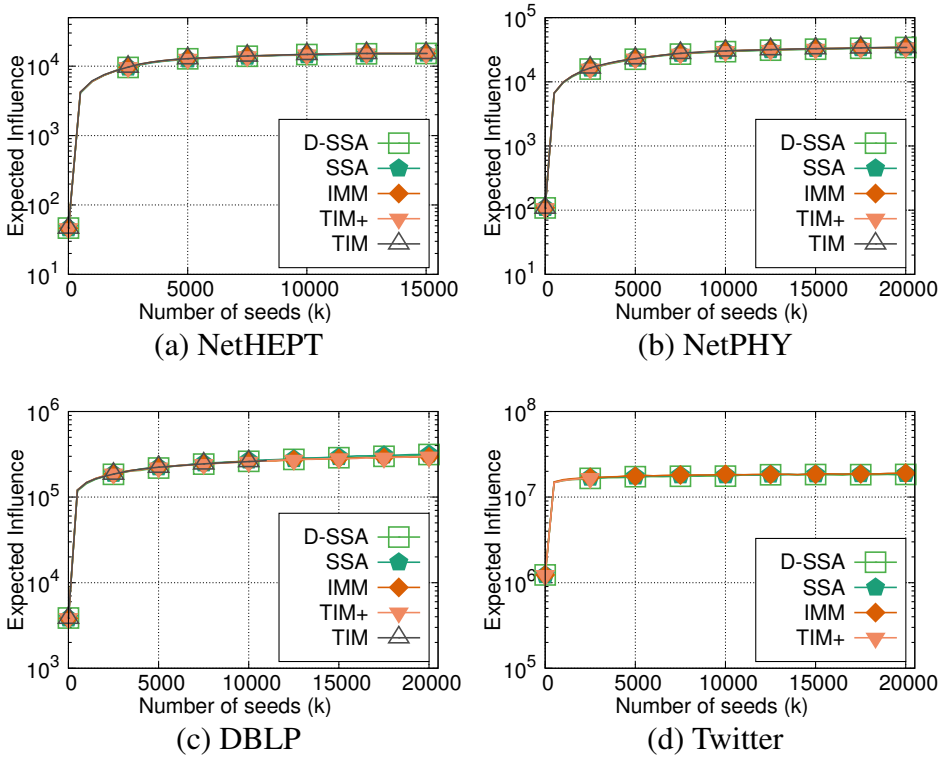


Fig. 14.: Expected Influence under IC model.

3.1.5.1 Experimental Settings

All the experiments are run on a Linux machine with 2.2Ghz Xeon 8 core processor and 100GB of RAM. We carry experiments under both LT and IC models on the following algorithms and datasets.

Algorithms compared. On IM experiments, we compare SSA and D-SSA with the group of top algorithms that provide the same $(1 - 1/e - \epsilon)$-approximation guarantee. More specifically, CELF++ [44], one of the fastest greedy algorithms, and IMM [105], TIM/TIM+ [106], the best current RIS-based algorithms, are selected. For experimenting with TVM problem, we apply our Stop-and-Stare algorithms on this context and compare with the most efficient method for the problem, termed KB-TIM, in [71].

Datasets. For experimental purposes, we choose a set of 8 datasets from various dis-

ciplines: NetHEPT, NetPHY, DBLP are citation networks, Email-Enron is communication network, Epinions, Orkut, Twitter and Friendster are online social networks. The description summary of those datasets is in Table 18. On Twitter network, we also have the actual tweet/retweet dataset and we use these data to extract the target users whose tweets/retweets are relevant to a certain set of keywords. The experiments on TVM are run on the Twitter network with the extracted targeted groups of users.

Remark. Since Orkut and Friendster are **undirected** networks, within those networks we *replace each edge by two oppositely directed edges* (arcs). This contrasts to the conference version of this paper in which the Orkut and Friendster networks are treated as directed networks.

Parameter Settings. For computing the edge weights, we follow the conventional computation as in [106, 20, 47, 87], the weight of the edge (u, v) is calculated as $w(u, v) = \frac{1}{d_{in}(v)}$ where $d_{in}(v)$ denotes the in-degree of node v.

In all the experiments, we keep $\epsilon = 0.1$ and $\delta = 1/n$ as a general setting or explicitly stated otherwise. For the other parameters defined for particular algorithms, we take the recommended values in the corresponding papers if available. We also limit the running time of each algorithm in a run to be within 24 hours.

3.1.5.2 Experiments with IM problem

To show the superior performance of the proposed algorithms on IM task, we ran the first set of experiments on four real-world networks, i.e., NetHEPT, NetPHY, DBLP, Twitter. We also test on a wide spectrum of the value of k, typically, from 1 to 20000, except on NetHEPT network since it has only 15233 nodes. The solution quality, running time, memory usage are reported sequentially in the following. We also present the actual number of RR sets generated by SSA, D-SSA and IMM when testing on four other datasets, i.e., Enron, Epinions, Orkut and Friendster.

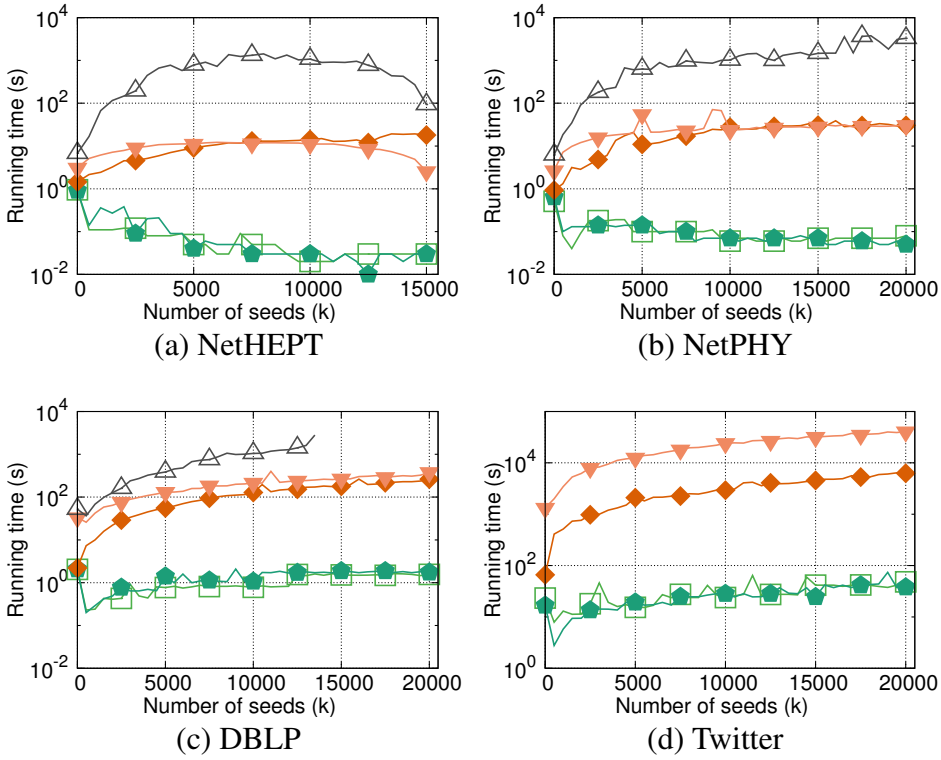


Fig. 15.: Running time under LT model

Solution Quality: We first compare the quality of the solution returned by all the algorithms on LT and IC models. The results are presented in Fig. 13 and Fig. 14, respectively. The CELF++ algorithm is only able to run on NetHEPT due to time limit. From those figures, all the methods return comparable seed set quality with no significant difference. The results directly give us a better viewpoint on the basic network property that a small fraction of nodes can influence a very large portion of the networks. Most of the previous researches only find up to 50 seed nodes and provide a limited view of this phenomenon. Here, we see that after around 2000 nodes have been selected, the influence gains of selecting more seeds become very slim.

Running time: We next examine the performance in terms of running time of the tested algorithms. The results are shown in Fig. 15 and Fig. 16. Both SSA and D-SSA significantly outperform the other competitors by a huge margin. Comparing to IMM, the

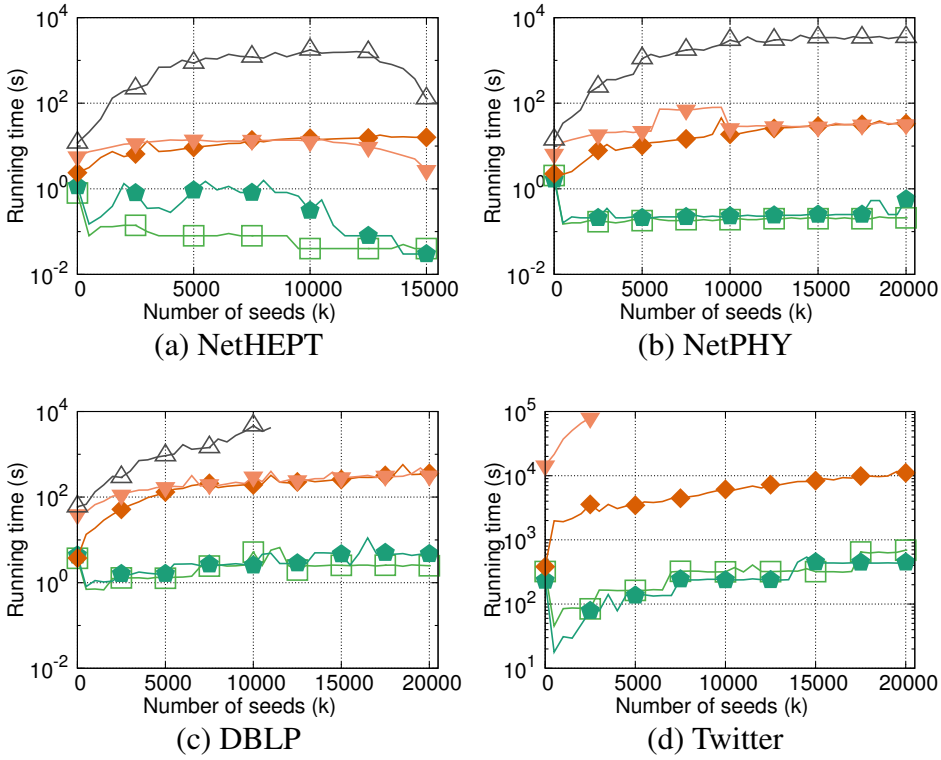


Fig. 16.: Running time under IC model

best known algorithm, SSA and D-SSA run up to several orders of magnitudes faster. TIM+ and IMM show similar running time since they operate on the same philosophy of estimating optimal influence first and then calculating the necessary samples to guarantee the approximation for all possible seed sets. However, each of the two steps displays its own weaknesses. In contrast, SSA and D-SSA follows the Stop-and-Stare mechanism to thoroughly address those weaknesses and thus exhibit remarkable improvements. In particular, the speedup factor of D-SSA to IMM can go up to 1200x in the case of NetHEPT network on the LT model. On most of other cases, the factor stabilizes at several hundred times.

Comparing between SSA and D-SSA, since D-SSA possesses the type-2 minimum threshold compared to the weaker type-1 threshold of SSA with the same precision settings ϵ, δ, D-SSA performs at least as good as SSA and outperforms in many cases in which the

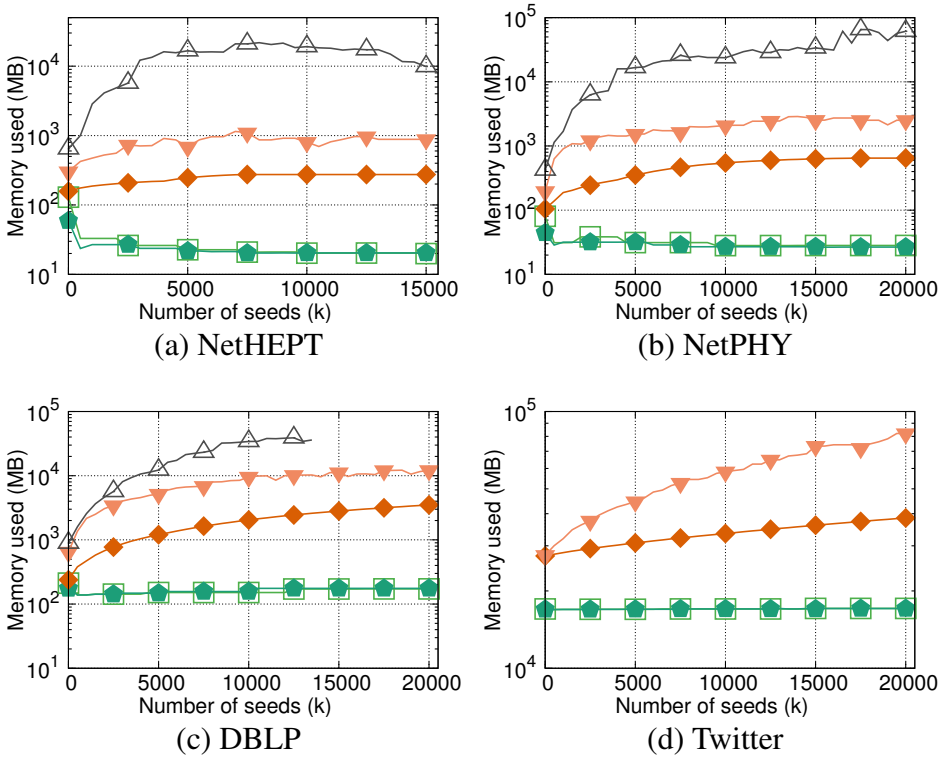


Fig. 17.: Memory usage under LT model

fixed setting of SSA falls out of the effective ranges for that network and value k. This problem is resolved in D-SSA thanks to the dynamic error computation at every iteration.

Memory Usage and Number of RR sets: This experiment is divided into two parts: 1) we report the memory usage in the previous experiments and 2) since the gain in influence peaks at the selection of 1 to 1000 nodes, we carry new experiments on four other datasets, i.e., Enron, Epinion, Orkut and Friendster, with $k \in \{1, 500, 1000\}$ to show the view across datasets of SSA, D-SSA and IMM.

Memory Usage. The results on memory usage of all the algorithms are shown in Fig. 17 and Fig. 18. We can see that there is a strong correlation between running time and memory usage. It is not a surprise that SSA and D-SSA require much less memory, up to orders of magnitude, than the other methods since the complexity is represented by the number of RR sets and these methods achieve type-1 and type-2 minimum thresholds of

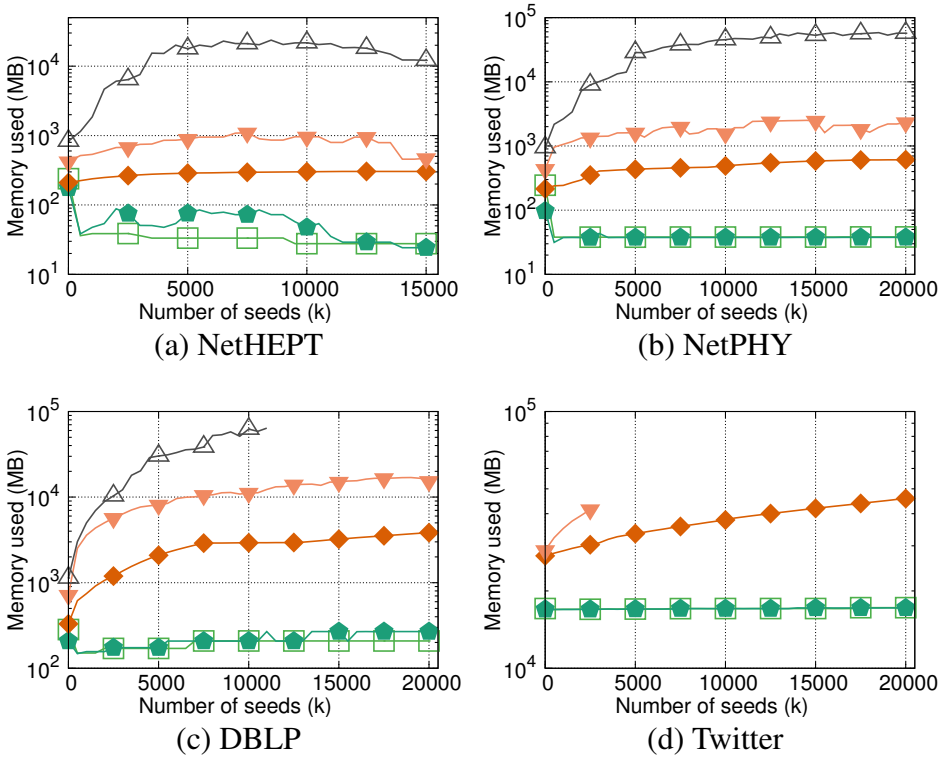


Fig. 18.: Memory usage under IC model

RR sets.

Across datasets view. We ran SSA, D-SSA and IMM on four other datasets, i.e., Enron, Epinions, Orkut and Friendster, with $k \in \{1, 500, 1000\}$ under LT model. The results are presented in Table 19. In terms of running time, the table reflects our previous results that SSA and D-SSA largely outperform IMM, up to several orders of magnitudes. The same pattern happens in terms of the number of RR sets generated. As shown, even in the most extreme cases of selecting a single node, SSA and D-SSA require several times fewer RR sets than IMM.

We note that, in the most challenging case of Friendster network with over 3.6 billion edges, IMM uses 172 GB of main memory while D-SSA and SSA require much lower memory resource of only 69 and 72 GB respectively.

Table 14.: Performance of SSA, D-SSA and IMM on various datasets under LT model.

Data	running time (in second (s) or hour (h))								
	$k = 1$			$k = 500$			$k = 1000$		
	D-SSA	SSA	IMM	D-SSA	SSA	IMM	D-SSA	SSA	IMM
Enron	**0.5 s**	0.6 s	0.7 s	**0.1 s**	**0.1 s**	3.1 s	**0.1** s	**0.1 s**	6.9 s
Epinions	**0.6 s**	0.7 s	0.8 s	**0.2 s**	**0.2 s**	4.4 s	**0.2 s**	0.3 s	12.1 s
Orkut	**86.2 s**	108.2 s	179.9 s	**11.8 s**	12.1 s	317.8 s	**23.8 s**	25.8 s	548.9 s
Friendster	**4.1 h**	4.7 h	8.1 h	**0.3 h**	0.5 h	n/a	**0.3 h**	0.5 h	n/a
	number of RR sets								
Enron	**96 K**	272 K	280 K	**24 K**	42 K	580 K	**24 K**	61 K	910 K
Epinions	**205 K**	570 K	400 K	**51 K**	97 K	1.2 M	**51 K**	131 K	1.9 M
Orkut	**512 K**	1.5 M	1.2 M	**64 K**	177 K	2.1 M	**128 K**	230 K	3.3 M
Friendster	**77 M**	161 M	175 M	**4.8 M**	17 M	n/a	**4.8 M**	15 M	n/a

3.1.5.3 Experiments with TVM problem

In this experiments, we will modify our Stop-and-Stare algorithms to work on Targeted Viral Marketing (TVM) problem and compare with the best existing method, i.e., KB-TIM in [71] to show the drastic improvements when applying our methods. In short, we will describe how we select the targeted groups from actual tweet/retweet datasets of Twitter and how to modify D-SSA and SSA for TVM problem. Then, we will report the experimental results. **TVM problem and methods:** Targeted Viral Marketing (TVM) is a central problem in economics in which, instead of maximizing the influence over all the nodes in a network as in IM, it targets a specific group whose users are relevant to a certain topic and aims at optimizing the influence to that group only. Each node in the targeted group is associated with a weight which indicates the relevance of that user to the topic. The best current method for solving TVM is proposed in [71] in which the authors introduce weighted RIS sampling (called WRIS) and integrate it into TIM+ method [106] to

derive an approximation algorithm, termed KB-TIM. WRIS only differs from the original RIS at the point of selecting the sampling root. More specifically, WRIS selects the root node proportional to the node weights instead of uniform selection as in RIS.

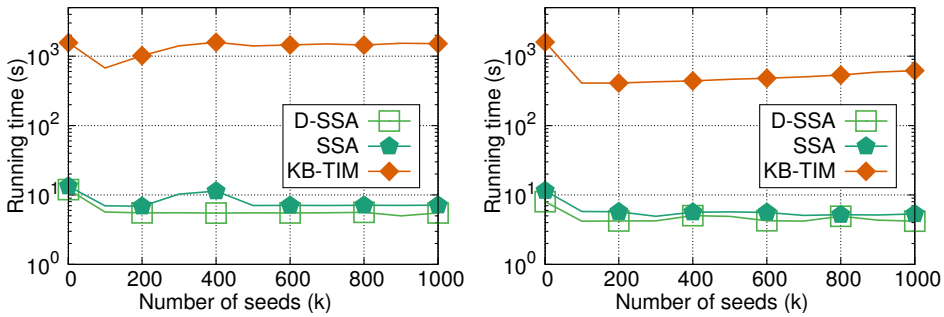
In the same way, we incorporate WRIS into D-SSA and SSA for solving TVM problem. By combining the analysis of WRIS in [71] and our previous proofs, it follows that the modified D-SSA and SSA preserve the $(1 - 1/e - \epsilon)$-approximation property as in IM problem. **Extracting the targeted groups:** We use tweet/retweet dataset to extract

Table 15.: Topics, related keywords

Topic	*Keywords*	#Users
1	**bill clinton, iran, north korea, president obama, obama**	997,034
2	**senator ted kenedy, oprah, kayne west, marvel, jackass**	507,465

the users' interests on two political topics as described in [65]. We choose two groups of most popular keywords as listed in Table 21, and mine from the tweet data who posted tweets/reweets containing at least one of those keywords in each group and how many times. We consider those users to be the targeted groups in TVM experiments with the relevance/interest of each user on the topic proportional to the frequency of having those keywords in their tweets. **Experimental results:**

We run SSA, D-SSA and KB-TIM on Twitter network under LT model with the targeted groups extracted from tweet dataset as described previously. Since all the algorithms have the same guarantee on the returned solution, we only measure the performance of these methods in terms of running time and the results are depicted in Fig. 19. In both cases, D-SSA and SSA consistently witness at least two order of magnitude improvements (up to 500 times) in running time compared to KB-TIM. D-SSA is also consistently faster than SSA due to the more optimal type-2 threshold.



(a) Topic 1 (b) Topic 2

Fig. 19.: Running time on Twitter network

3.1.6 Conclusion

In this paper, we make several significant contributions in solving the fundamental influence maximization (IM) problem. We provide the unified RIS framework which generalizes the best existing technique of using RIS sampling to find an $(1-1/e-\epsilon)$-approximate solution in billion-scale networks. We introduce the RIS threshold that all the algorithms following the framework need to satisfy and two minimum thresholds, i.e., type-1 and type-2. Interestingly, we are able to develop two novel algorithms, SSA and D-SSA, which are the first methods meeting the two minimum thresholds. Since IM plays a central roles in a wide range of practical applications, e.g., viral marketing, controlling diseases, virus/worms, detecting contamination and so on, the developments of SSA and D-SSA will immediately result in a burst in performance and allow their applications to work in billion-scale domains. Our approach here can be further coupled with advanced sampling techniques to produce even more efficient algorithms for IM [85].

Tightness of Chernoff's bounds

In the following proofs, we use an intermediate results on the optimality of Chernoff-like in Lem. 2. According to Lemma 4 in [39], we have the following results regarding the

tightness of the Chernoff-like bounds in the above lemma.

Lemma 29 ([39]). *Let $X_1, X_2, \ldots, ..., X_T$ be i.i.d random variables taking values 0 or 1, and $\Pr[X_i = 1] = \mu \leq 1/2$. Denote by $\hat{\mu} = \frac{1}{T}\sum_{i=1}^{T} X_i$ the average of the random variables. For every $\epsilon \in (0, 1/2]$, if $\epsilon^2 \mu T \geq 3$, the following hold:*

$$\Pr[\hat{\mu} \leq (1-\epsilon)\mu] \geq \exp\left(-9\epsilon^2 \mu T\right) \text{ and} \tag{3.29}$$

$$\Pr[\hat{\mu} \geq (1+\epsilon)\mu] \geq \exp\left(-9\epsilon^2 \mu T\right). \tag{3.30}$$

Note that the condition $\epsilon \in (0, 1/2]$ can be relaxed into $\epsilon \in (0, c]$ for any $c < 1$ if we assume sufficiently small δ.

Corollary 2 (Tightness of Chernoff's bound). *Let $X_1, X_2, \ldots$ be i.i.d random variables taking values 0 or 1, and $\Pr[X_i = 1] = \mu \in (0, 1/2]$. For $\epsilon \in (0, 1/2]$, $\delta < 1/e$ and $T > 0$, the following hold*

- *If $\Pr\left[\frac{1}{T}\sum_{i=1}^{T} X_i < (1-\epsilon)\mu\right] \leq \delta$, then $T = \Omega(\Upsilon(\epsilon, \delta)\frac{1}{\mu})$.*
- *If $\Pr\left[\frac{1}{T}\sum_{i=1}^{T} X_i > (1+\epsilon)\mu\right] \leq \delta$, then $T = \Omega(\Upsilon(\epsilon, \delta)\frac{1}{\mu})$.*

Proof. If $T < \frac{1}{9}\frac{1}{\epsilon^2}\ln\frac{1}{n}$, then by Lem. 29, $\Pr[\hat{\mu} \leq (1-\epsilon)\mu] \geq \exp(-9\epsilon^2\mu T) = \delta$ (contradiction). Thus, $T \geq \frac{1}{9}\frac{1}{\epsilon^2}\ln\frac{1}{n} = \Omega(\Upsilon(\epsilon, \delta))$.

Similarly, if $T < \frac{1}{9}\frac{1}{\epsilon^2}\ln\frac{1}{n}$, then $\Pr[\hat{\mu} \geq (1+\epsilon)\mu] \geq \exp\left(-9\epsilon^2\mu T\right) = \delta$ (contradiction). Thus,

$$\Pr\left[\frac{1}{T}\sum_{i=1}^{T} X_i > (1+\epsilon)\mu\right] \leq \delta$$

implies $T \geq \frac{1}{9}\frac{1}{\epsilon^2}\ln\frac{1}{n} = \Omega(\Upsilon(\epsilon, \delta))$. □

The lower bounds also hold for the case when $X_1, \ldots, X_T$ are weakly dependent (martingales) as the random variables in Lem. 2.

Omitted Proofs of Lemmas and Theorems

Proof of Theorem 1

Apply the union bound. The following two inequalities from Eqs. 7 and 8 hold together with probability at least $1 - (\delta_a + \delta_b)$.

$$\hat{\mathbb{I}}(\hat{S}_k) \leq (1 + \epsilon_a)\mathbb{I}(\hat{S}_k) \tag{3.31}$$

$$\mathbb{I}(\hat{S}_k^*) \geq (1 - \epsilon_b)\mathbb{I}(S_k^*). \tag{3.32}$$

Assume that the above two inequalities hold. We show, by contradiction, that $\mathbb{I}(\hat{S}_k) \geq (1 - 1/e - \epsilon)\mathsf{OPT}_k$, where $\epsilon = (1 - \frac{1}{e})\frac{\epsilon_a + \epsilon_b}{1 + \epsilon_a}$. Assume the opposite, i.e.,

$$\mathbb{I}(\hat{S}_k) < (1 - 1/e - \epsilon)\mathsf{OPT}_k. \tag{3.33}$$

Since the greedy algorithm used in Max-Coverage algorithm returns a $(1 - 1/e)$ approximation [81], the greedy solution $\hat{S}_k$ satisfies $\mathrm{Cov}_{\mathcal{R}}(\hat{S}_k) \geq (1 - 1/e)\mathrm{Cov}_{\mathcal{R}}(S_k^*)$. It follows that

$$\hat{\mathbb{I}}(\hat{S}_k) \geq (1 - 1/e)\hat{\mathbb{I}}(S_k^*).$$

Extend (3.31) and use the assumption (3.33).

$$\begin{aligned}\mathbb{I}(\hat{S}_k) &\geq \hat{\mathbb{I}}(\hat{S}_k) - \epsilon_a\mathbb{I}(\hat{S}_k) \geq (1 - 1/e)\hat{\mathbb{I}}(S_k^*) - \epsilon_a\mathbb{I}(\hat{S}_k) \\ &\geq (1 - 1/e)\hat{\mathbb{I}}(S_k^*) - \epsilon_a(1 - 1/e - \epsilon)\mathsf{OPT}_k \end{aligned} \tag{3.34}$$

Apply Eq. (3.32), we yield

$$\begin{aligned}\mathbb{I}(\hat{S}_k) &\geq (1 - 1/e)(1 - \epsilon_b)\mathbb{I}(S_k^*) - \epsilon_a(1 - 1/e - \epsilon)\mathsf{OPT}_k \\ &= (1 - 1/e - (1 - 1/e - \epsilon)\epsilon_a + (1 - 1/e)\epsilon_b)\mathsf{OPT}_k \\ &= (1 - 1/e - \epsilon)\mathsf{OPT}_k \textit{ (contradiction)}\end{aligned}$$

where $\epsilon = (1 - 1/e - \epsilon)\epsilon_a + (1 - 1/e)\epsilon_b)$, or equivalently, $\epsilon = (1 - \frac{1}{e})\frac{\epsilon_a + \epsilon_b}{1+\epsilon_a}$.

Thus, $\Pr[\mathbb{I}(\hat{S}_k) \geq (1 - 1/e - \epsilon)OPT_k] \geq 1 - (\delta_a + \delta_b)$.

Proof of Lemma 3

We follow the proof of the Stopping Rule Theorem in [26].

Since $\mathbb{I}_c(\hat{S}_k) = n\Lambda_2/T$, it suffices to show that

$$\Pr[T \leq \frac{n\Lambda_2}{(1+\epsilon_2)\mathbb{I}(\hat{S}_k)}] \leq \frac{\delta_2}{3\log_2 n}, \tag{3.35}$$

where $T \leq T_{max}$ is the number of RR sets generated.

Let $L = \lfloor \frac{n\Lambda_2}{(1+\epsilon_2)\mathbb{I}(\hat{S}_k)} \rfloor$. From the definition of Λ_2, we obtain that,

$$L = \lfloor \frac{n(1 + (1+\epsilon_2)(2 + \frac{2}{3}\epsilon_2)\ln(\frac{1}{\delta_2'})\frac{1}{\epsilon_2^2})}{(1+\epsilon_2)\mathbb{I}(\hat{S}_k)} \rfloor \tag{3.36}$$

$$\geq (2 + \frac{2}{3}\epsilon_2)\ln(\frac{1}{\delta_2'})\frac{n}{\mathbb{I}(\hat{S}_k)\epsilon_2^2}. \tag{3.37}$$

Since T is an integer, $T \leq \frac{n\Lambda_2}{(1+\epsilon_2)\mathbb{I}(\hat{S}_k)}$ if and only if $T \leq L$. But $T \leq L$ if and only if $\mathrm{Cov}_L = \sum_{j=1}^{L} Z_j \geq \Lambda_2$. Thus,

$$\Pr[T \leq \frac{n\Lambda_2}{(1+\epsilon_2)\mathbb{I}(\hat{S}_k)}] = \Pr[T \leq L] = \Pr[\mathrm{Cov}_L \geq \Lambda_2] \tag{3.38}$$

$$= \Pr[\mathrm{Cov}_L n/L \geq \Lambda_2 n/L] \tag{3.39}$$

$$\leq \Pr[\hat{\mu}_L \geq (1+\epsilon_2)\mu]. \tag{3.40}$$

Apply the Chernoff's bound in Lem. 2 on the last probability and note that $L \geq (2 + \frac{2}{3}\epsilon_2)\ln(\frac{1}{\delta_2'})\frac{n}{\mathbb{I}(\hat{S}_k)\epsilon_2^2}$, we achieve the following bound,

$$\Pr[\hat{\mu}_L \geq (1+\epsilon_2)\mu] \leq \delta_2' = \frac{\delta_2}{3\log_2 n}. \tag{3.41}$$

Thus, we have,

$$\Pr[\mathbb{I}_c(\hat{S}_k) \geq (1+\epsilon_2)\mathbb{I}(\hat{S}_k)] \leq \frac{\delta_2}{3\log_2 n}, \tag{3.42}$$

which completes the proof of Lem. 3.

Proof of Lemma 6

Note that there are $|\mathcal{R}|$ RR sets to estimate the influence of the optimal solution S_k^*. We use Chernoff-Hoeffding's inequality (Lem. 2) on the optimal solution, S_k^*, with random variable $Z = \min\{1, |R_j \cap S_k^*|\}$ and $\mu_Z = \mathsf{OPT}_k/n$ to obtain

$$\Pr[\hat{\mathbb{I}}(S_k^*) \leq (1-\epsilon_3^{(i)})\mathsf{OPT}_k] \leq e^{-\frac{|\mathcal{R}|\mathsf{OPT}_k(\epsilon_3^{(i)})^2}{2n}} \leq \delta/(3i_{max}), \tag{3.43}$$

which completes the proof of Lem. 6.

Proof of Theorem 2

Assume that none of the bad events in Lemmas 4, 5, and 6 happens. By union bound, this assumption holds with probability at least

$$1 - (\delta/3 + \delta/(3i_{max}) \times 3i_{max} + \delta/(3i_{max}) \times 3i_{max}) = 1 - \delta.$$

We will show that $\mathbb{I}(\hat{S}_k) \geq (1-1/e-\epsilon)\mathsf{OPT}_k$.

If SSA terminates with $|\mathcal{R}| \geq N_{max}$, since the bad event $\left[|R| \geq N_{max} \text{ and } \mathbb{I}(\hat{S}_k) < (1-1/e-\epsilon)\right]$ (Lem. 4) does not happen, we have $\mathbb{I}(\hat{S}_k) \geq (1-1/e-\epsilon)\mathsf{OPT}_k$.

Otherwise, SSA will stop due to the two stopping conditions (C1), Line 8 Alg. 1, and (C2), Line 11 Alg. 1.

Proving $\epsilon_3^{(t)} \leq \epsilon_3$. Since the bad event in Lem. 5 does not happen, we have

$$\mathbb{I}_c(\hat{S}_k) \leq (1+\epsilon_2)\mathbb{I}(\hat{S}_k).$$

Thus,

$$\hat{\mathbb{I}}(\hat{S}_k) = (1+\epsilon_1)\mathbb{I}_c(\hat{S}_k) \leq (1+\epsilon_1)(1+\epsilon_2)\mathbb{I}(\hat{S}_k). \tag{3.44}$$

From the stopping condition (C1) $\mathrm{Cov}_{\mathcal{R}}(\hat{S}_k) \geq \Lambda_1$, we have

$$\begin{aligned}\hat{\mathbb{I}}(\hat{S}_k) &= \frac{\mathrm{Cov}_{\mathcal{R}}(\hat{S}_k)n}{|\mathcal{R}|} \geq \frac{\Lambda_1 n}{|\mathcal{R}|} = \frac{(1+\epsilon_1)(1+\epsilon_2)\Upsilon(\epsilon_3, \frac{\delta}{3i_{max}})n}{|\mathcal{R}|} \\ \Rightarrow |\mathcal{R}| &\geq \frac{\Lambda_1 n}{\hat{\mathbb{I}}(\hat{S}_k)} = \frac{(1+\epsilon_1)(1+\epsilon_2)\Upsilon(\epsilon_3, \frac{\delta}{3i_{max}})n}{\hat{\mathbb{I}}(\hat{S}_k)}\end{aligned} \tag{3.45}$$

Combine with Eq. (3.44), we obtain

$$\begin{aligned}|\mathcal{R}| &\geq \frac{(1+\epsilon_1)(1+\epsilon_2)\Upsilon(\epsilon_3, \frac{\delta}{3i_{max}})n}{(1+\epsilon_1)(1+\epsilon_2)\mathbb{I}(\hat{S}_k)} \\ &= \frac{\Upsilon(\epsilon_3, \frac{\delta}{3i_{max}})n}{\mathbb{I}(\hat{S}_k)} \geq \frac{\Upsilon(\epsilon_3, \frac{\delta}{3i_{max}})n}{\mathsf{OPT}_k}.\end{aligned}$$

Substitute the above into the definition of $\epsilon_3^{(t)}$. We have

$$\epsilon_3^{(t)} = \sqrt{\frac{2n\ln\frac{3i_{max}}{\delta}}{|\mathcal{R}|\mathsf{OPT}_k}} \leq \sqrt{\frac{2n\ln\frac{3i_{max}}{\delta}}{\frac{\Upsilon(\epsilon_3, \frac{\delta}{3i_{max}})n}{\mathsf{OPT}_k}\mathsf{OPT}_k}} \leq \epsilon_3. \tag{3.46}$$

Proving the approximation ratio. Combine the above with the assumption that the bad event in the Lem. 6 does not happen, we have

$$\hat{\mathbb{I}}(S_k^*) \geq (1-\epsilon_3^{(t)})\mathsf{OPT}_k \geq (1-\epsilon_3)\mathsf{OPT}_k. \tag{3.47}$$

Let $\epsilon_a = \epsilon_1 + \epsilon_2 + \epsilon_1\epsilon_2$. We can rewrite Eq. (3.44) into

$$\hat{\mathbb{I}}(\hat{S}_k) \leq (1+\epsilon_a)\mathbb{I}(\hat{S}_k).$$

Follow the same contradiction proof in the Theorem 1 with ϵ_a and $\epsilon_b = \epsilon_3$, we have $\mathbb{I}(\hat{S}_k) \geq (1-1/e-\epsilon)\mathsf{OPT}_k$, where $\epsilon = (1-\frac{1}{e})\frac{\epsilon_a+\epsilon_b}{1+\epsilon_a} = (1-\frac{1}{e})\frac{\epsilon_1+\epsilon_2+\epsilon_1\epsilon_2+\epsilon_3}{(1+\epsilon_1)(1+\epsilon_2)}$.

Therefore, $\Pr[\mathbb{I}(\hat{S}_k) \geq (1-1/e-\epsilon)\mathsf{OPT}_k] \geq 1-\delta$.

Proof of Lemma 7

Since $|\mathcal{R}| \geq T_1$ implies

$$\Pr[\hat{\mathbb{I}}_{\mathcal{R}}(S_k^*) \geq (1-\epsilon_b)OPT_k] \geq 1-\delta_b.$$

From the assumption that $1/\delta = \Omega(\ln n)$ and the fact that $i_{max} \leq 2\log_2 n$ and $\delta_b \leq \delta$, we have

$$\begin{aligned}\Upsilon(\epsilon_0, \frac{\delta}{3i_{max}}) &= (2+2/3\epsilon_0)\frac{1}{\epsilon_0^2}\ln\frac{3i_{max}}{\delta} \\ &\leq 3\frac{\epsilon_b^2}{\epsilon_0^2}\frac{1}{\epsilon_b^2}(\ln 1/\delta + \ln 3i_{max}) = O(\Upsilon(\epsilon_b, \delta_b))\end{aligned} \tag{3.48}$$

The values of ϵ_2, ϵ_3 specified later at the end of Theorem 3 will guarantee that $\frac{\epsilon_b^2}{\epsilon_0^2}$ is also a constant that depends only on ϵ_b.

Apply Corollary 2, we have $T_1 = \Omega(\Upsilon(\epsilon_b, \delta_b)\frac{n}{\mathsf{OPT}_k})$. Thus,

$$\begin{aligned}T_{\mathsf{SSA}} &= \max\{T_1, \alpha\Upsilon(\epsilon_0, \frac{\delta}{3i_{max}})\frac{n}{\mathsf{OPT}_k}\} \\ &= O(\Upsilon(\epsilon_b, \delta_b))\frac{n}{\mathsf{OPT}_k} = O(T_1)\end{aligned} \tag{3.49}$$

This yields the proof.

Proof of Theorem 3

SSA stops when either $|\mathcal{R}| \geq N_{max}$ or all the following stopping conditions hold simultaneously.

- $\mathrm{Cov}_{\mathcal{R}}(\hat{S}_k) \geq \Lambda_1 = \Theta(\Upsilon(\epsilon_3, \frac{\delta}{3i_{max}}))$ (Condition C1)
- Estimate-Inf$(G, \hat{S}_k, \epsilon_2, \delta_2', T_{max})$ returns an estimation $\mathbb{I}_c(\hat{S}_k)$ but not -1. (Line 10,

Alg. 1).

- $\hat{\mathbb{I}}(\hat{S}_k) \leq (1+\epsilon_1)\mathbb{I}_c(\hat{S}_k)$ (Condition C2).

Assume $|\mathcal{R}| \geq T_{\mathsf{SSA}} = O(T_1)$ (Lem. 7). If $T_{\mathsf{SSA}} \geq N_{max}$, then SSA will stop within $O(T_1)$ samples. Otherwise, $|\mathcal{R}| \geq T_{\mathsf{SSA}}$ at some iteration $i \leq i_{max}$.

Assume that none of the bad events in Lemmas 5, 6, 8, and Eqs. (6.4), and Eq. (6.5) happen. By union bound, the assumption holds with a probability at least

$$1 - (\frac{\delta}{3i_{max}}3i_{max} + \frac{\delta}{3i_{max}}3i_{max} + \frac{\delta}{3i_{max}}3i_{max} + \delta_a + \delta_b) \geq 1 - 2\delta.$$

Since the bad events in Eqs. (24) and (25) do not happen,

$$\hat{\mathbb{I}}(\hat{S}_k) \leq (1+\epsilon_a)\mathbb{I}(\hat{S}_k), \tag{3.50}$$

$$\hat{\mathbb{I}}(S_k^*) \geq (1-\epsilon_b)\mathsf{OPT}_k, \text{ and} \tag{3.51}$$

Similar to the proof of Theorem 1, it follows that

$$\mathbb{I}(\hat{S}_k) \geq (1-1/e-\epsilon)\mathsf{OPT}_k \tag{3.52}$$

Condition C1: From the $(1-1/e)$ approximation guarantee of the Max-Coverage algorithm, it follows that

$$\mathrm{Cov}_{\mathcal{R}}(\hat{S}_k) \geq (1-1/e)\mathrm{Cov}_{\mathcal{R}}(S_k^*)$$

From Eq. (3.51),

$$\mathrm{Cov}_{\mathcal{R}}(S_k^*) \geq (1-\epsilon_b)\frac{\mathsf{OPT}_k}{n}|\mathcal{R}|.$$

Thus,

$$\begin{aligned}\mathrm{Cov}_{\mathcal{R}}(\hat{S}_k) &\geq (1-1/e)(1-\epsilon_b)\frac{\mathsf{OPT}_k}{n}|\mathcal{R}| \\ &\geq (1-1/e)(1-\epsilon)\frac{\mathsf{OPT}_k}{n}\alpha\Upsilon(\epsilon_0, \frac{\delta}{3i_{max}})\frac{n}{\mathsf{OPT}_k} \\ &\geq (1-1/e)(1-\epsilon)\alpha\Upsilon(\epsilon_3, \frac{\delta}{3i_{max}}) \text{ (since } \epsilon_0 \leq \epsilon_3)\end{aligned}$$

Select $\alpha > \frac{(1+\epsilon_1)(1+\epsilon_2)}{(1-1/e)(1-\epsilon)}$, we have

$$\mathrm{Cov}_{\mathcal{R}}(\hat{S}_k) > \Lambda_1 = (1+\epsilon_1)(1+\epsilon_2)\Upsilon(\epsilon_3, \frac{\delta}{3i_{max}}) \tag{3.53}$$

Termination of Estimate-Inf: We show that Estimate-Inf does not return -1. If Estimate-Inf terminates in line 7, Alg. 3, for some $T < T_{max}$, then nothing left to prove. Otherwise, we show that when $|\mathcal{R}_c| = T_{max}$, then

$$\mathrm{Cov}_{\mathcal{R}_c}(\hat{S}_k) \geq \Lambda_2 = 1 + (1+\epsilon_2)\Upsilon(\epsilon_2, \frac{\delta}{3i_{max}}) \text{(Line 2, Alg. 3)},$$

and, hence, Estimate-Inf returns an estimate but not -1.

By definition of T_{SSA},

$$|\mathcal{R}| \geq T_{\mathsf{SSA}} \geq \alpha\Upsilon(\epsilon_0, \frac{\delta}{3i_{max}})\frac{n}{\mathsf{OPT}_k} \geq \alpha\Upsilon(\epsilon_2, \frac{\delta}{3i_{max}})\frac{n}{\mathsf{OPT}_k}.$$

The last inequality is due to $\epsilon_0 \leq \epsilon_2$. Thus,

$$T_{max} = 2|\mathcal{R}|\frac{1+\epsilon_2}{1-\epsilon_2}\frac{\epsilon_3^2}{\epsilon_2^2} \geq 2\alpha\Upsilon(\epsilon_2, \frac{\delta}{3i_{max}})\frac{\epsilon_3^2}{\epsilon_2^2}\frac{n}{\mathsf{OPT}_k}. \tag{3.54}$$

Select large enough α, says $\alpha > \frac{\epsilon_2^2}{\epsilon_3^2}$, we obtain

$$\epsilon_2^{(i)} = \sqrt{\frac{(\ln 1/\delta + \ln 3i_{max})n}{T_{max}}\mathbb{I}(\hat{S}_k)} \leq \epsilon_2$$

Since the bad event in Lem. 8 does not happen we have

$$\mathbb{I}_c(\hat{S}_k) \geq (1-\epsilon_2^{(i)})\mathbb{I}(\hat{S}_k)$$

Since $\epsilon_2^{(i)} \leq \epsilon_2$, it follows that

$$\mathbb{I}_c(\hat{S}_k) \geq (1-\epsilon_2)\mathbb{I}(\hat{S}_k) \tag{3.55}$$

and $\hat{\mathbb{I}}_c(\hat{S}_k) = \frac{\mathrm{Cov}_{\mathcal{R}_c}(\hat{S}_k)n}{|\mathcal{R}_c|}$, we have

$$\mathrm{Cov}_{\mathcal{R}_c}(\hat{S}_k) \geq (1-\epsilon_2)\frac{\mathbb{I}(\hat{S}_k)}{n}|\mathcal{R}_c| = (1-\epsilon_2)\frac{\mathbb{I}(\hat{S}_k)}{n}T_{max}$$

From Eqs. (3.52) and (3.54),

$$\begin{aligned}\mathrm{Cov}_{\mathcal{R}_c}(\hat{S}_k) &\geq (1-\epsilon_2)\frac{(1-\frac{1}{e}-\epsilon)\mathsf{OPT}_k}{n}2\alpha\Upsilon(\epsilon_2, \frac{\delta}{3i_{max}})\frac{\epsilon_3^2}{\epsilon_2^2}\frac{n}{\mathsf{OPT}_k} \\ &\geq 1+(1+\epsilon_2)\Upsilon(\epsilon_2, \frac{\delta}{3i_{max}}).\end{aligned} \tag{3.56}$$

Here, we select $\alpha > 1 + \frac{\epsilon_2^2}{\epsilon_3^2}\frac{1}{2(1-\epsilon_2)(1-1/e-\epsilon)}$.

Condition C2: We show that the condition C2, $\hat{\mathbb{I}}(\hat{S}_k) \leq (1+\epsilon_1)I_c(\hat{S}_k)$, in line 11, Alg. 1 is satisfied with proper selection of $\epsilon_1, \epsilon_2, \epsilon_3$. The condition C2 is equivalent to

$$\frac{\hat{\mathbb{I}}(\hat{S}_k)}{\mathbb{I}_c(\hat{S}_k)} - 1 \leq \epsilon_1 \Leftrightarrow \frac{\hat{\mathbb{I}}(\hat{S}_k)}{\mathbb{I}(\hat{S}_k)}\frac{\mathbb{I}(\hat{S}_k)}{\mathbb{I}_c(\hat{S}_k)} - 1 \leq \epsilon_1$$

From Eqs. (3.50) and (3.55), we have

$$\frac{\hat{\mathbb{I}}(\hat{S}_k)}{\mathbb{I}(\hat{S}_k)}\frac{\mathbb{I}(\hat{S}_k)}{\mathbb{I}_c(\hat{S}_k)} - 1 \leq (1+\epsilon_a)\frac{1}{1-\epsilon_2} - 1 = \frac{\epsilon_a+\epsilon_2}{1-\epsilon_2}$$

Set $\epsilon_1 = \frac{\epsilon_a+\epsilon_b/2}{1-\epsilon_b/2}, \epsilon_2 = \epsilon_b/2, \epsilon_3 = \frac{\epsilon_b^2}{2-\epsilon_b}$, the following holds

$$\begin{cases} \epsilon_1 \in (0,\infty), \epsilon_2, \epsilon_3 \in (0,1) \\ (1-\frac{1}{e})\frac{\epsilon_a+\epsilon_b}{1+\epsilon_a} = (1-\frac{1}{e})\frac{\epsilon_1+\epsilon_2+\epsilon_1\epsilon_2+\epsilon_3}{(1+\epsilon_1)(1+\epsilon_2)} = \epsilon \\ \frac{\hat{\mathbb{I}}(\hat{S}_k)}{\mathbb{I}_c(\hat{S}_k)} - 1 \leq \frac{\epsilon_a+\epsilon_2}{1-\epsilon_2} = \epsilon_1 \end{cases} \tag{3.57}$$

Thus, SSA with the setting in (3.57) will stop with a probability at least $1-2\delta$ if $|\mathcal{R}| \geq T_{\mathsf{SSA}} = O(N_1(\epsilon_a, \epsilon_b, \delta_a, \delta_b))$.

Constants Justification: The factors that are assumed to be constants within our proofs for SSA are 1) the factor $3\frac{\epsilon_b^2}{\epsilon_0}$ in Eq. (3.48), 2) $\alpha > \frac{(1+\epsilon_1)(1+\epsilon_2)}{(1-1/e)(1-\epsilon)}$ before Eq. (3.53), 3) $\alpha > \frac{\epsilon_2^2}{\epsilon_3^2}$, after Eq. (3.54), 4) $\alpha > 1 + \frac{\epsilon_2^2}{\epsilon_3^2}\frac{1}{2(1-\epsilon_2)(1-1/e-\epsilon)}$, after Eq. (3.56). With the above setting of $\epsilon_1, \epsilon_2, \epsilon_3$, we can verify that those factors are constants that depend only on ϵ, ϵ_a, and ϵ_b.

Proof of Lemma 10

$$\begin{aligned} t_{\max} &= \log_2(\frac{2N_{\max}}{\Upsilon(\epsilon, \delta/3)}) \\ &= \log_2\left(2(2-\frac{1}{e})^2\frac{(2+\frac{2}{3}\epsilon)n \cdot \frac{\ln(6/\delta)+\ln\binom{n}{k}}{k\epsilon^2}}{(2+\frac{2}{3}\epsilon)\ln(\frac{3}{\delta})\frac{1}{\epsilon^2}}\right) \\ &= \log_2\left(2(2-\frac{1}{e})^2\frac{n(\ln(6/\delta)+\ln\binom{n}{k})}{k\ln(3/\delta)}\right) \\ &\leq \log_2\left(2(2-\frac{1}{e})^2\frac{n(\ln(6/\delta)+k\ln n)}{k\ln(3/\delta)}\right) \\ &\leq 2\log_2 n + 2 = O(\log_2 n) \end{aligned} \tag{3.58}$$

The last inequality follows from our assumption $1/\delta = \Omega(\log_2 n)$.

Proof of Lemma 11

One can verify that $f(x)$ is a strictly decreasing function for $x > 0$. Moreover, $f(0) = 1$ and $\lim_{x\to\infty} f(x) = 0$. Thus, the equation $f(x) = \frac{\delta}{3t_{max}}$ has an *unique solution* for $0 < \delta < 1$ and $t_{max} \geq 1$.

Bound the probability of $B_t^{(2)}$: Note that $\hat{\epsilon}_t$ and the samples generated in $\mathcal{R}_t^c$ are independent. Thus, we can apply the concentration inequality in Eq. (5):

$$\Pr[\mathbb{I}_t^c(\hat{S}_k) \geq (1+\hat{\epsilon}_t)\mathbb{I}(\hat{S}_k)] \leq \exp\left(-\frac{N_t\mathbb{I}(\hat{S}_k)\hat{\epsilon}_t^2}{(2+\frac{2}{3}\hat{\epsilon}_t)n}\right) = \frac{\delta}{3t_{max}}.$$

The last equation is due to the definition of $\hat{\epsilon}_t$.

Bound the probability of $B_t^{(3)}$: Since ϵ_t^* is fixed and independent from the generated samples, we have

$$\begin{aligned}
\Pr[\hat{\mathbb{I}}_t(S_k^*) \leq (1-\epsilon_t^*)\mathsf{OPT}_k] &\leq \exp\left(-\frac{|\mathcal{R}_t|\mathsf{OPT}_k\epsilon_t^{*2}}{2n}\right) \\
&= \exp\left(-\frac{\Lambda 2^{t-1}\mathsf{OPT}_k\epsilon^2 n}{2n2^{t-1}\mathsf{OPT}_k}\right) \qquad (3.59)\\
&= \exp\left(-\frac{(2+\frac{2}{3}\epsilon)\ln(\frac{3t_{max}}{\delta})\frac{1}{\epsilon^2}2^{t-1}\mathsf{OPT}_k\epsilon^2 n}{2(1+\epsilon/3)n2^{t-1}\mathsf{OPT}_k}\right) \\
&\leq \exp\left(-\ln\frac{3t_{max}}{\delta}\right) = \frac{\delta}{3t_{max}}, \qquad (3.60)
\end{aligned}$$

which completes the proof of Lemma. 11.

Proof of Lemma 12

Since the bad event $B_t^{(2)}$ doesn't happen, we have

$$\hat{\mathbb{I}}_t^{(c)}(\hat{S}_k) \leq (1+\hat{\epsilon}_t)\mathbb{I}(\hat{S}_k) \Rightarrow \mathrm{Cov}_{\mathcal{R}_t^c}(\hat{S}_k) \leq (1+\hat{\epsilon}_t)N_t\frac{\mathbb{I}(\hat{S}_k)}{n}$$

When D-SSA stops with $\epsilon_t \leq \epsilon$, it must satisfy the condition on line 9 of D-SSA

$$\mathrm{Cov}_{\mathcal{R}_t^c}(\hat{S}_k) \geq \Lambda_1.$$

Thus, we have

$$(1+\hat{\epsilon}_t)N_t\frac{\mathbb{I}(\hat{S}_k)}{n} \geq \Lambda_1 = 1+(1+\epsilon)\frac{2+2/3\epsilon}{\epsilon^2}\ln\frac{3t_{max}}{\delta} \tag{3.61}$$

From the definition of $\hat{\epsilon}_t$, it follows that

$$N_t = \frac{2+2/3\hat{\epsilon}_t}{\hat{\epsilon}_t^2}\ln\left(\frac{3t_{max}}{\delta}\right)\frac{n}{\mathbb{I}(\hat{S}_k)} \tag{3.62}$$

Substitute the above into (3.61) and simplify, we obtain:

$$(1+\hat{\epsilon}_t)\frac{2+2/3\hat{\epsilon}_t}{\hat{\epsilon}_t^2}\ln\left(\frac{3t_{max}}{\delta}\right) \tag{3.63}$$

$$\geq (1+\epsilon)\frac{2+2/3\epsilon}{\epsilon^2}\ln\frac{3t_{max}}{\delta}+1 \tag{3.64}$$

Since the function $(1+x)\frac{2+2/3x}{x^2}$ is a decreasing function for $x>0$, it follows that $\hat{\epsilon}_t < \epsilon$.

Proof of Theorem 5

Assume that none of the bad events $B^{(1)}, B_t^{(2)}, B_t^{(3)}$ ($t = 1..t_{max}$) in Lemmas 9 and 11 happens. Apply union bound, the probability that the assumption holds is at least

$$1-(\delta/3+(\delta/(3t_{max})+\delta/(3t_{max}))\times t_{max}) \geq 1-\delta \tag{3.65}$$

We shall show that the returned $\hat{S}_k$ is a $(1-1/e-\epsilon)$-approximation solution. If D-SSA stops with $|\mathcal{R}_t| \geq N_{max}$, $\hat{S}_k$ is a $(1-1/e-\epsilon)$-approximation solution, since the bad event $B^{(1)}$ does not happen.

Otherwise, D-SSA stops at some iteration t and $\epsilon_t \leq \epsilon$. We use contradiction method. Assume that

$$\mathbb{I}(\hat{S}_k) < (1-1/e-\epsilon)\mathsf{OPT}_k. \tag{3.66}$$

The proof will continue in the following order

(A) $\mathbb{I}(\hat{S}_k) \geq (1 - 1/e - \epsilon'_t)\mathsf{OPT}_k$

where $\epsilon'_t = (\epsilon_1 + \hat{\epsilon}_t + \epsilon_1\hat{\epsilon}_t)(1 - 1/e - \epsilon) + (1 - 1/e)\epsilon^*_t$.

(B) $\hat{\epsilon}_t \leq \epsilon_2$ and $\epsilon^*_t \leq \epsilon_3$.

(C) $\epsilon'_t \leq \epsilon_t \leq \epsilon \Rightarrow \mathbb{I}(\hat{S}_k) \geq (1 - \frac{1}{e} - \epsilon)\mathsf{OPT}_k$ (*contradiction*).

Proof of (A). Since the bad events $B^{(2)}_t$ and $B^{(3)}_t$ do not happen, we have

$$\hat{\mathbb{I}}^{(c)}_t(\hat{S}_k) \leq (1 + \hat{\epsilon}_t)\mathbb{I}(\hat{S}_k), \text{and} \tag{3.67}$$

$$\hat{\mathbb{I}}_t(S^*_k) \leq (1 - \epsilon^*_t)\mathsf{OPT}_k. \tag{3.68}$$

Since $\epsilon_1 \leftarrow \hat{\mathbb{I}}_t(\hat{S}_k)/\mathbb{I}^c_t(\hat{S}_k) - 1$, it follows from (3.67) that

$$\hat{\mathbb{I}}_t(\hat{S}_k) = (1 + \epsilon_1)\mathbb{I}^c_t(\hat{S}_k) \leq (1 + \epsilon_1)(1 + \hat{\epsilon}_t)\mathbb{I}(\hat{S}_k)$$

Expand the right hand side and apply (3.66), we obtain

$$\begin{aligned}\mathbb{I}(\hat{S}_k) &\geq \hat{\mathbb{I}}_t(\hat{S}_k) - (\epsilon_1 + \hat{\epsilon}_t + \epsilon_1\hat{\epsilon}_t)\mathbb{I}(\hat{S}_k) \\ &\geq \hat{\mathbb{I}}_t(\hat{S}_k) - (\epsilon_1 + \hat{\epsilon}_t + \epsilon_1\hat{\epsilon}_t)(1 - 1/e - \epsilon)\mathsf{OPT}_k\end{aligned}$$

Since the greedy algorithm in the Max-Coverage guarantees a $(1 - 1/e)$-approximation, $\hat{\mathbb{I}}_t(\hat{S}_k) \geq (1 - 1/e)\hat{\mathbb{I}}_t(S^*_k)$. Thus

$$\begin{aligned}\mathbb{I}(\hat{S}_k) &\geq (1 - 1/e)\hat{\mathbb{I}}_t(S^*_k) - (\epsilon_1 + \hat{\epsilon}_t + \epsilon_1\hat{\epsilon}_t)(1 - 1/e - \epsilon)\mathsf{OPT}_k \\ &\geq (1 - 1/e)(1 - \epsilon^*_t)\mathsf{OPT}_k - (\epsilon_1 + \hat{\epsilon}_t + \epsilon_1\hat{\epsilon}_t)(1 - 1/e - \epsilon)\mathsf{OPT}_k \\ &\geq (1 - 1/e - \epsilon'_t)\mathsf{OPT}_k,\end{aligned}$$

where $\epsilon'_t = (\epsilon_1 + \hat{\epsilon}_t + \epsilon_1\hat{\epsilon}_t)(1 - 1/e - \epsilon) + (1 - 1/e)\epsilon^*_t$.

Proof of (B). We show that $\hat{\epsilon}_t \leq \epsilon_2$. Since, $\epsilon_2 = \epsilon\sqrt{\frac{n(1+\epsilon)}{2^{t-1}\mathbb{I}_t^c(\hat{S}_k)}}$, we have

$$\frac{1}{\epsilon^2} = \frac{1}{\epsilon_2^2}\frac{n}{2^{t-1}}\frac{1+\epsilon}{\mathbb{I}_t^c(\hat{S}_k)}.$$

Expand the number of RR sets in iteration t, $N_t = 2^{t-1}\Lambda$, and apply the above equality, we have

$$N_t = 2^{t-1}(2+2/3\epsilon)\frac{1}{\epsilon^2}\ln\frac{3t_{max}}{\delta} \tag{3.69}$$

$$= 2^{t-1}(2+2/3\epsilon)\frac{1}{\epsilon_2^2}\frac{n}{2^{t-1}}\frac{1+\epsilon}{\mathbb{I}_t^c(\hat{S}_k)}\ln\frac{3t_{max}}{\delta} \tag{3.70}$$

$$= (2+2/3\epsilon)\frac{1}{\epsilon_2^2}\frac{(1+\epsilon)n}{\mathbb{I}_t^c(\hat{S}_k)}\ln\frac{3t_{max}}{\delta} \tag{3.71}$$

On the other hand, according to Eq. (3.62), we also have,

$$N_t = \frac{2+2/3\hat{\epsilon}_t}{\hat{\epsilon}_t^2}\ln\left(\frac{3t_{max}}{\delta}\right)\frac{n}{\mathbb{I}(\hat{S}_k)}. \tag{3.72}$$

Thus

$$(2+2/3\epsilon)\frac{1}{\epsilon_2^2}\frac{1+\epsilon}{\mathbb{I}_t^c(\hat{S}_k)} = \frac{2+2/3\hat{\epsilon}_t}{\hat{\epsilon}_t^2}\frac{1}{\mathbb{I}(\hat{S}_k)}$$

$$\Rightarrow \frac{\hat{\epsilon}_t^2}{\epsilon_2^2} = \frac{2+2/3\hat{\epsilon}_t}{2+2/3\epsilon}\frac{\mathbb{I}_t^c(\hat{S}_k)}{(1+\epsilon)\mathbb{I}(\hat{S}_k)} \leq 1$$

The last step is due to Lemma 12, i.e., $\mathbb{I}_t^c(\hat{S}_k) \leq (1+\epsilon)\mathbb{I}(\hat{S}_k)$ and $\hat{\epsilon}_t \leq \epsilon$. Therefore, $\hat{\epsilon}_t \leq \epsilon_2$.

We show that $\epsilon_t^* \leq \epsilon_3$. According to the definition of ϵ_t^* and ϵ_3, we have

$$\frac{(\epsilon_t^*)^2}{\epsilon_3^2} = \frac{n}{(1+\epsilon/3)2^{t-1}\mathsf{OPT}_k}\Big/\frac{n(1+\epsilon)(1-1/e-\epsilon)}{(1+\epsilon/3)2^{t-1}\mathbb{I}_t^c(\hat{S}_k)}$$

$$= \frac{\mathbb{I}_t^c(\hat{S}_k)}{\mathsf{OPT}_k(1+\epsilon)(1-1/e-\epsilon)} \leq \frac{\mathbb{I}_t(\hat{S}_k)}{\mathsf{OPT}_k(1-1/e-\epsilon)} \leq 1$$

The last two steps follow from Lem. 12, $\mathbb{I}_t^c(\hat{S}_k) \leq (1+\epsilon)\mathbb{I}(\hat{S}_k)$ and the assumption (3.66), respectively. Thus, $\epsilon_t^* \leq \epsilon_3$.

Proof of (C). Since $1+\epsilon_1 = \hat{\mathbb{I}}_t(\hat{S}_k)/\mathbb{I}_t^c(\hat{S}_k) \geq 0$ and $\epsilon_2 \geq \hat{\epsilon}_t > 0$ and $\epsilon_3 \geq \epsilon_t^* > 0$, we

have

$$\epsilon'_t = (\epsilon_1 + \hat{\epsilon}_t + \epsilon_1\hat{\epsilon}_t)(1 - 1/e - \epsilon) + (1 - 1/e)\epsilon^*_t \tag{3.73}$$

$$= (\epsilon_1 + \hat{\epsilon}_t(1 + \epsilon_1))(1 - 1/e - \epsilon) + (1 - 1/e)\epsilon^*_t \tag{3.74}$$

$$\leq (\epsilon_1 + \epsilon_2(1 + \epsilon_1))(1 - 1/e - \epsilon) + (1 - 1/e)\epsilon_3 \tag{3.75}$$

$$= \epsilon_t \leq \epsilon. \tag{3.76}$$

Proof of Theorem 6

Since $T_2 \geq 1$, there exist $\epsilon^*_a, \epsilon^*_b, \delta^*_a, \delta^*_b$ that satisfy

$$N^{(1)}_{min}(\epsilon^*_a, \epsilon^*_b, \delta^*_a, \delta^*_b) = T_2, \tag{3.77}$$

$$(1 - \frac{1}{e})\frac{\epsilon^*_a + \epsilon^*_b}{1 + \epsilon^*_a} = \epsilon \leq \frac{1}{4}, \tag{3.78}$$

$$\delta^*_a + \delta^*_b \leq \delta < \frac{1}{\log_2 n}. \tag{3.79}$$

Let $\epsilon_0 = \min\{\epsilon, \epsilon^*_b\}$, and

$$T_{\textsf{D-SSA}} = \max\{T_2, \alpha\Upsilon(\epsilon_0, \frac{\delta}{3t_{max}})\frac{n}{\textsf{OPT}_k}\}, \tag{3.80}$$

for some constant α specified later. Note that $\frac{\epsilon^*_b}{\epsilon} \leq 1/(1 - 1/e)$ (from Eq. 3.78). Similar to the proof in Lem. 7, we can show that

$$T_{\textsf{D-SSA}} = O(T_2)$$

under the *range conditions*.

From Def. 5 of the type-1 minimum threshold, if $|\mathcal{R}| \geq T_{\textsf{D-SSA}} \geq T_2$ then

$$\Pr[\hat{\mathbb{I}}(\hat{S}_k) > (1 + \epsilon^*_a)\mathbb{I}(\hat{S}_k)] \leq \delta^*_a \text{ and} \tag{3.81}$$

$$\Pr[\hat{\mathbb{I}}(S^*_k) < (1 - \epsilon^*_b)\textsf{OPT}_k] \leq \delta^*_b. \tag{3.82}$$

Similar to the proof of Theorem 1, it follows that

$$\mathbb{I}(\hat{S}_k) \geq (1 - 1/e - \epsilon)\mathsf{OPT}_k \tag{3.83}$$

Assume that D-SSA reaches to a round $t \leq t_{max}$ with $|\mathcal{R}| = \Lambda 2^{t-1} \geq T_{\mathsf{D\text{-}SSA}}$. If $T_{\mathsf{D\text{-}SSA}} > N_{max}$ then D-SSA will stop and the proof is complete. Otherwise consider the bad events that $\mathbb{I}_t^c(\hat{S}_k) = \frac{\mathrm{Cov}_{\mathcal{R}_t^c}(\hat{S}_k)n}{|\mathcal{R}_t^c|}$ is an underestimate of $\mathbb{I}(\hat{S}_k)$. Specifically, define for each $t = 1, \ldots, t_{max}$ the event

$$B_t^{(4)} = \left(\hat{\mathbb{I}}_t^c(\hat{S}_k) < (1 - \tilde{\epsilon}_t)\mathbb{I}(\hat{S}_k)\right),$$

where $\tilde{\epsilon_t} = \epsilon\sqrt{\frac{n}{(1+\epsilon/3)2^{t-1}\mathbb{I}(\hat{S}_k)}}$. Similar to the proof of Lem. 11, we can show that

$$\Pr[B_t^{(4)}] \leq \frac{\delta}{3t_{max}}.$$

Assume that neither the bad events in Eqs. (6.4) and (6.5) nor the bad events $B_t^{(2)}, B_t^{(3)}, B_t^{(4)}$ happen for any $t \in [1, t_{max}]$. Apply the union bound, this assumption holds with a probability at least

$$1 - (\delta_a^* + \delta_b^* + \frac{\delta}{3t_{max}}t_{max} + \frac{\delta}{3t_{max}}t_{max} + \frac{\delta}{3t_{max}}t_{max}) \geq 1 - 2\delta.$$

Under the above assumption, we will show that the two conditions D1 and D2 are met, and, thus, D-SSA will stop.

Condition D1: We will prove that $\mathrm{Cov}_{\mathcal{R}_t^c}(\hat{S}_k) \geq \Lambda_1$. Since $|\mathcal{R}| \geq T_{\mathsf{D\text{-}SSA}} \geq \alpha\Upsilon(\epsilon_0, \frac{\delta}{3t_{max}})\frac{n}{\mathsf{OPT}_k}$ and $\epsilon_0 \leq \epsilon$, we have

$$\begin{aligned} 2^{t-1} &\geq \frac{|\mathcal{R}|}{\Upsilon(\epsilon, \frac{\delta}{3t_{max}})} \geq \alpha\Upsilon(\epsilon_0, \frac{\delta}{3t_{max}})\frac{n}{\mathsf{OPT}_k}/\Upsilon(\epsilon, \frac{\delta}{3t_{max}}) \\ &\geq \alpha\frac{n}{\mathsf{OPT}_k}\frac{\epsilon^2}{\epsilon_0^2} \end{aligned} \tag{3.84}$$

Select $\alpha > 9/(1-1/e-\epsilon)$ and apply Eq. 3.83, we have

$$\begin{aligned}\tilde{\epsilon}_t &= \epsilon\sqrt{\frac{n}{(1+\epsilon/3)2^{t-1}\mathbb{I}(\hat{S}_k)}} \\ &\leq \epsilon\sqrt{\frac{n}{(1+\epsilon/3)\alpha\frac{n}{\mathsf{OPT}_k}\frac{\epsilon^2}{\epsilon_0^2}(1-1/e-\epsilon)\mathsf{OPT}_k}} \leq \frac{\epsilon_0}{3}\end{aligned} \tag{3.85}$$

Since $B_t^{(4)}$ does not happen, we have

$$\hat{\mathbb{I}}_t^c(\hat{S}_k) \geq (1-\tilde{\epsilon}_t)\mathbb{I}(\hat{S}_k). \tag{3.86}$$

We have

$$\begin{aligned}&\mathrm{Cov}_{\mathcal{R}_t^c}(\hat{S}_k) = \frac{\mathrm{Cov}_{\mathcal{R}_t^c}(\hat{S}_k)n}{|R_t^c|}\frac{|R_t^c|}{n} = \hat{\mathbb{I}}_t^c(\hat{S}_k)\frac{|\mathcal{R}_t^c|}{n} \\ &\geq (1-\tilde{\epsilon}_t)\frac{\mathbb{I}(\hat{S}_k)}{n}\Upsilon(\epsilon, \frac{\delta}{3t_{max}})2^{t-1} \\ &\geq (1-\epsilon/3)(1-1/e-\epsilon)\frac{\mathsf{OPT}_k}{n}\Upsilon(\epsilon, \frac{\delta}{3t_{max}})\alpha\frac{n}{\mathsf{OPT}_k} \\ &\geq 1+(1+\epsilon)\Upsilon(\epsilon, \frac{\delta}{3t_{max}}) = \Lambda_1,\end{aligned}$$

when selecting $\alpha > 1 + \frac{1}{(1-1/e-\epsilon)(1-\epsilon/3)}$.

Condition D2: $\epsilon_t \leq \epsilon$. The condition D2 is equivalent to

$$\begin{aligned}&(1-1/e)\frac{\epsilon_1+\epsilon_2+\epsilon_1\epsilon_2+\epsilon_3}{(1+\epsilon_1)(1+\epsilon_2)} \leq \epsilon = (1-1/e)\frac{\epsilon_a^*+\epsilon_b^*}{1+\epsilon_a^*} \\ \Leftrightarrow &1-\frac{1-\epsilon_3}{(1+\epsilon_1)(1+\epsilon_2)} \leq 1-\frac{1-\epsilon_b^*}{1+\epsilon_a^*} \\ \Leftrightarrow &1-\epsilon_3 \geq \frac{1-\epsilon_b^*}{1+\epsilon_a^*}(1+\epsilon_1)(1+\epsilon_2)\end{aligned} \tag{3.87}$$

From Eqs. (6.4), (3.86), and (3.85), we have

$$1+\epsilon_1 = \frac{\hat{\mathbb{I}}(\hat{S}_k)}{\mathbb{I}(\hat{S}_k)}\frac{\mathbb{I}(\hat{S}_k)}{\mathbb{I}_t^c(\hat{S}_k)} \leq (1+\epsilon_a^*)\frac{1}{1-\tilde{\epsilon}_t} \leq \frac{1+\epsilon_a^*}{1-\epsilon_0/3} \leq \frac{1+\epsilon_a^*}{1-\epsilon_b^*/3}$$

Thus, it is sufficient to show that

$$
\begin{aligned}
1-\epsilon_3 &\geq \frac{1-\epsilon_b^*}{1+\epsilon_a^*}\frac{1+\epsilon_a^*}{1-\epsilon_b^*/3}(1+\epsilon_2) \\
\Leftrightarrow &(1-\epsilon_3)(1-\epsilon_b^*/3) \geq (1-\epsilon_b^*)(1+\epsilon_2) \\
\Leftrightarrow &\frac{2}{3}\epsilon_b^* + \frac{\epsilon_b^*}{3}\epsilon_3 + \epsilon_b^*\epsilon_2 \geq \epsilon_2 + \epsilon_3
\end{aligned} \tag{3.88}
$$

Apply the inequalities $2^{t-1} \geq \alpha\frac{n}{\mathsf{OPT}_k}$, Eq. (3.84), and $\mathbb{I}(\hat{S}_k) \geq (1-1/e-\epsilon)\mathsf{OPT}_k$, Eq. (3.83). For sufficiently large $\alpha > \frac{9(1+\epsilon)}{(1-1/e-\epsilon)(1-\epsilon/3)}$ which implies from Eqs. 3.86 and 3.83 that,

$$
\mathbb{I}_t^c(\hat{S}_k) \geq (1-\tilde{\epsilon}_t)\mathbb{I}(\hat{S}_k) \geq (1-\frac{\epsilon_0}{3})\mathbb{I}(\hat{S}_k) \leq (1-\epsilon/3)\mathbb{I}(\hat{S}_k) \leq (1-\epsilon/3)(1-1/e-\epsilon)\mathsf{OPT}_k, \tag{3.89}
$$

then, we have

$$
\epsilon_2 = \epsilon\sqrt{\frac{n(1+\epsilon)}{2^{t-1}\mathbb{I}_t^c(\hat{S}_k)}} \leq \epsilon_0/3 \leq \epsilon_b^*/3 \tag{3.90}
$$

$$
\epsilon_3 = \epsilon\sqrt{\frac{n(1+\epsilon)(1-1/e-\epsilon)}{(1+\epsilon/3)2^{t-1}\mathbb{I}_t^c(\hat{S}_k)}} \leq \epsilon_0/3 \leq \epsilon_b^*/3 \tag{3.91}
$$

Therefore, $\epsilon_2 + \epsilon_3 \leq 2/3\epsilon_b^*$ with a constant $\alpha > \frac{9(1+\epsilon)}{(1-1/e-\epsilon)(1-\epsilon/3)}$ and the inequality (3.88) holds. This completes the proof.

3.2 Cost-aware Targeted Viral Marketing

Summary of contributions:

- We propose the *Cost-aware Targeted Viral Marketing* (CTVM) problem that consider *heterogeneous costs and benefits* for nodes in the network. Our problem generalizes other viral marketing problems including TVM, BIM, and the fundamental IM problems.

- We propose BCT, an efficient algorithm that returns $(1 - 1/\sqrt{e} - \epsilon)$-approximate solutions for CTVM with a high probability. The two novel aspects of BCT are an efficient benefit sampling strategy (Section III) and an efficient stopping rule (Section IV) that guarantees an asymptotic minimal number of samples. Interestingly, the time complexity is independent of the edges, making BCT the *first sub-linear time algorithm* for CTVM (and IM) in dense graphs, under the LT model.

- We perform extensive experiments on various real networks. BCT, considering both cost and benefit, provides significantly higher quality solutions than existing methods, while running multiple times faster than the state-of-the art ones. Further, we also demonstrate the ability of BCT to identify key influencers in trending topics in a Twitter dataset of 1.5 billion social relations and 106 million tweets within few minutes.

A comprehensive comparison of the state-of-the-art algorithms for IM and extensions is provided in Table 16.

Table 16.: Main results of related methods (k is the number of selected seed nodes, n is the number of nodes in the graph, m is the number of edges)

Method	IM	BIM	TIM	CTVM	Model	Time Complexity
Naive Greedy [55]	✓				LT+IC	$O(kmnR)$, R is the #Monte Carlo simulations
CELF [69]	✓				LT+IC	$O(kmnR)$, empirically faster than Naive Greedy
CELF++ [44]	✓				LT+IC	$O(kmnR)$, optimized CELF
Simpath [45]	✓				LT	$O(kmnR)$, empirically faster than Naive Greedy
LDAG [19]	✓				MIA	$O(nt_{i\theta} + kn_{O\theta}n_{i\theta}\log(n))$ (see [19] for details)
Borgs's method [11]	✓				LT+IC	$O(kl^2(m+n)\log^2(n)/\epsilon^3)$ with probability $1/n^l$
TIM/TIM+ [106]	✓				LT+IC	$O((k+l)(n+m)\log(n)/\epsilon^2)$
IMM [105]	✓				IC	$O((k+l)(n+m)\log(n)/\epsilon^2)$
BIM [87]		✓			LT+IC	$O(n_0(n(\log(n_0)+d)+kn_0(1+d)))$
KB-TIM [71]			✓		IC	-
BCT (this paper)	✓	✓	✓	✓	LT+IC	$O((k+l)n\log(n)/\epsilon^2)$ for the LT model; $O((k+l)(n+m)\log(n)/\epsilon^2)$ for the IC model

3.2.1 Models and Problem Definitions

In this section, we formally define the CTVM problem and present an overview of the Reverse Influence Sampling approaches in Borgs et al. [11] and Tang et al. [106, 105]. For readability, we focus on the *Linear Threshold* (LT) propagation model [55] and summarize our similar results for the *Independent Cascade* (IC) model in Subsection 3.2.4.5.

3.2.1.1 Model and Problem Definition

Let $G = (V, E, c, b, w)$ be a social network with a node set V and a directed edge set E, with $|V| = n$ and $|E| = m$. Each node $u \in V$ has a selecting cost $c(u) \geq 0$ and a benefit $b(u)$ if u is influenced. Each directed edge $(u, v) \in E$ is associated with an influence weight $w(u, v) \in [0, 1]$ such that $\sum_{u \in V} w(u, v) \leq 1$.

Our model assumes that all the parameters, $c(u), b(u)\, \forall u \in V$ and $w(u, v)\, \forall (u, v) \in E$ are given. In fact, these can be estimated depending on the specific context when applying our method. The cost of node u, $c(u)$, manifests how hard (how much effort) it is to initially influence the respective person, e.g., convince him to adopt the product. Thus, $c(u)$ is usually regarded proportionally to some centrality measures, e.g., the degree centrality [87].

Similarly, the node benefit $b(u)$ refers to the gain of influencing node u and hence is context-dependent, e.g., in targeted viral marketing, $b(u)$ is assigned 1 if u is in our targeted group and 0 outside [9, 18] or learned from the interest level on the relevant topic, e.g., number of tweets/retweets with specific keywords on Twitter network. Additionally, $w(u, v)$ indicates the probability of u influencing v which is widely evaluated as the interaction frequency from u to v [55, 106] or learned from action logs [43].

Similarly to influence of a set, he *benefit* of a seed set S is defined as the expected total

benefit over all influenced nodes, i.e.,

$$\mathbb{B}(S) = \sum_{g \sqsubseteq G} \Pr[g] \sum_{u \in R(g,S)} b(u). \tag{3.92}$$

We now define our problem as follows.

Definition 12 (Cost-aware Targeted Viral Marketing -CTVM)**.** *Given a graph $G = (V, E, c, b, w)$ and a budget $B > 0$, find a seed set $S \subset V$ with total cost $c(S) \leq B$ to maximize $\mathbb{B}(S)$.*

CTVM generalizes the viral marketing problems:

- Influence Maximization (IM): IM is a special case of CTVM with $c(u) = 1$ and $b(u) = 1 \; \forall u \in V$.
- Budgeted Influence Maximization (BIM)[87]: find a seed set with total cost at most B, that maximizes $\mathbb{I}(S)$. That is $b(u) = 1 \; \forall u \in V$.
- Targeted Viral Marketing (TVM): find a set of k node to maximize the number of influenced nodes in a targeted set T. This is $c(u) = 1 \; \forall u \in V$ and benefits $c(v) = 1$ if $v \in T$, and $c(w) = 0$ otherwise.

Since IM is a special case of CTVM, CTVM inherits the IM's complexity and hardness of approximation. Thus CTVM is an NP-hard problem and cannot be approximated within a factor $1 - 1/e + \epsilon$ for any $\epsilon > 0$, unless $P = NP$.

In Table 24, we summarize the frequently used notations.

3.2.1.2 Summary of the RIS Approach

The major bottle-neck in previous methods for IM [55, 70, 45, 87] is the inefficiency in estimating the influence spread. To address this, Borgs et al. [11] introduced a novel approach for IM, called Reverse Influence Sampling (RIS), which is the foundation for TIM/TIM+ algorithms, the state-of-the-art methods for IM [106].

Table 17.: Table of Notations

Notation	Description
n, m	#nodes, #links in G, respectively
$\mathbb{I}(S), \mathbb{I}(S, u)$	Influence Spread of seed set $S \subseteq V$ and influence of S on a node v. For $v \in V$, $\mathbb{I}(v) = \mathbb{I}(\{v\})$
Γ	Sum of all node benefits, $\sum_{v \in V} b(v)$
$\mathbb{B}(S)$	Benefit of seed set $S \subseteq V$
$\hat{\mathbb{B}}(S)$	$\hat{\mathbb{B}}(S) = \frac{deg_{\mathcal{H}}(S)}{m_{\mathcal{H}}}\Gamma$ - an estimator of $\mathbb{B}(S)$
OPT_k	The maximum $\mathbb{B}(S)$ for any size-k seed set S
S_k^*	An optimal size-k seed node, $\mathbb{B}(S_k^*) = OPT_k$
$m_{\mathcal{H}}$	#hyperedges in hypergraph $\mathcal{H}$
$deg_{\mathcal{H}}(S), S \subseteq V$	#hyperedges incident at some node in S. Also, $deg_{\mathcal{H}}(v)$ for $v \in V$
α	$\alpha = \sqrt{\ln(1/\delta) + \ln 2}$
β	$\beta = \sqrt{(1 - 1/e) \cdot (\ln \binom{n}{k} + \ln(1/\delta) + \ln 2)}$
ϵ_2	$\epsilon_2 = \frac{\epsilon\beta}{((1-1/e)\alpha+\beta)}$
Λ_L	$\Lambda_L = (1+\epsilon_2)\frac{(2+2\epsilon_2/3)\Gamma(\ln(6/\delta)+\ln\binom{n}{k})}{\epsilon_2^2}$

Given a graph $G = (V, E, c, b, w)$, RIS captures the influence landscape of G through generating a hypergraph $\mathcal{H} = (V, \{\mathcal{E}_1, \mathcal{E}_2, \ldots\})$. Each hyperedge $\mathcal{E}_j \in \mathcal{H}$ is a subset of nodes in V and constructed as follows.

Definition 13 (Random Hyperedge)**.** *Given $G = (V, E, w)$, a random hyperedge $\mathcal{E}_j$ is generated from G by 1) selecting a random node $v \in V$ 2) generating a sample graph $g \sqsubseteq G$ and 3) returning $\mathcal{E}_j$ as the set of nodes that can reach v in g.*

Node v in the above definition is called the *source* of $\mathcal{E}_j$ and denoted by $\mathsf{src}(\mathcal{E}_j)$. Observe that $\mathcal{E}_j$ contains the nodes that can influence its source v. If we generate multiple random hyperedges, influential nodes will likely appear more often in the hyperedges. Thus a seed set S that *covers* most of the hyperedges will likely maximize the influence spread $\mathbb{I}(S)$. Here a a seed set S covers a hyperedge $\mathcal{E}_j$, if $S \cap \mathcal{E}_j \neq \emptyset$. This is captured in the

following lemma in [11].

We denote by $m_{\mathcal{H}}$ the number of hyperedges in $\mathcal{H}$.

Lemma 30. *[11] Given $G = (V, E, w)$ and a random hyperedge $\mathcal{E}_j$ generated from G. For each seed set $S \subset V$,*

$$\mathbb{I}(S) = n \Pr[S \textit{ covers } \mathcal{E}_j]. \tag{3.93}$$

RIS framework. Based on the above lemma, the IM problem can be solved using the following framework.

- Generate multiple random hyperedges from G
- Use the greedy algorithm for the Max-coverage problem [56] to find a seed set S that covers the maximum number of hyperedges and return S as the solution.

Thresholds for Sufficient Number of Samples. The core issue in applying the above framework is that: *How many hyperedges are sufficient to provide a good approximation solution?* For any $\epsilon, \delta \in (0, 1)$, Tang et. al. established in [106] a theoretical threshold

$$\theta = (8 + 2\epsilon) n \frac{\ln 2/\delta + \ln \binom{n}{k}}{\epsilon^2 \mathsf{OPT}_k}, \tag{3.94}$$

and proved that when the number of hyperedges in $\mathcal{H}$ reaches θ, the above framework returns an $(1 - 1/e - \epsilon)$-approximate solution with probability $1 - \delta$. Here OPT_k denotes the maximum influence spread $\mathbb{I}(S)$.

Unfortunately, computing OPT_k is intractable, thus, TIM/TIM+ in [106] have to approximate OPT_k by a heuristic KPT^+ and thus, generate $\theta \frac{\mathsf{OPT}_k}{KPT^+}$ hyperedges, where the ratio $\frac{\mathsf{OPT}_k}{KPT^+} \geq 1$ is not upper-bounded. That is TIM/TIM+ may generate many times more hyperedges than needed. In contrast, our BCT algorithm in Section 3.2.3 guarantees that the number of hyperedges is at most a constant time of the theoretical threshold (with high probability). Thus, its running time is smaller and more predictable.

The same group of authors further reduce the threshold θ (Theorem 1 in [105]) to,

$$\theta = \frac{2n \cdot ((1-1/e) \cdot \alpha + \beta)^2}{\mathsf{OPT}_k \cdot \epsilon^2}, \tag{3.95}$$

where

$$\alpha = \sqrt{\ln(1/\delta) + \ln 2}, \text{ and} \tag{3.96}$$

$$\beta = \sqrt{(1-1/e) \cdot (\ln \tbinom{n}{k} + \ln(1/\delta) + \ln 2)}. \tag{3.97}$$

Define

$$\epsilon_2 = \frac{\epsilon\beta}{(1-1/e)\alpha + \beta}, \tag{3.98}$$

then, the threshold θ can be rewritten as follows,

$$\theta = \frac{2n\beta^2}{\mathsf{OPT}_k \epsilon_2^2} = \frac{(2-2/e)n(\ln \tbinom{n}{k} + \ln(1/\delta) + \ln 2)}{\mathsf{OPT}_k \epsilon_2^2} \tag{3.99}$$

which is shown in [105] to be 5 times smaller than that of Eq. 3.102. IMM also improves the estimation of KPT^+ to be bounded by some constant times OPT_k with high probability. However, the bound is loose and the estimation process is complicated. On the other hand, the proposed BCT algorithm in this paper adopts the better threshold in [105] with our approach in [88] which: 1) avoids a possibly complicated and expensive estimation phase, 2) achieves a better bound on the actual number of samples and 3) solves the more general CTVM problem (covers IM problem).

Remark. The most intuitive way to extend the RIS framework to cope with benefit of the nodes is to modify the RIS framework to find a seed set S that covers the maximum *weighted* number of hyperedges, where the weight of a hyperedge $\mathcal{E}_j$ is the benefit of the source $\mathsf{src}(\mathcal{E}_j)$. However following the same analysis in Tang et al. [106, 105], we need

$$\theta_B = \theta \mathbf{b_{max}}, \tag{3.100}$$

where $b_{max} = \max\{b(u)|u \in V\}$.

Unfortunately, θ_B can be as large as n times θ in the worst-case. To see this, we can (wlog) normalize the node benefit $b(u)$ so that $\sum_{u \in V} b(u) = n$. Then note that b_{max} could be as large as $\sum_{u \in V} b(u) = n$.

3.2.2 Benefit-aware Reverse Influence Sampling

In this section, we show that the well-studied convectional sampling strategy, called Reverse Influence Sampling (RIS), does not work well in CTVM problem when we have benefit. In fact, straightforwardly applying RIS to CTVM may lead to an extremely inefficient algorithm with the number of samplings being up to n times the number needed for IM task. Therefore, we propose an adapted version of RIS, named Benefit-aware Reverse Influence Sampling (BSA), for estimating expected benefit.

3.2.2.1 Summary of the RIS method

Borgs et al. [11] introduced a novel approach, called Reverse Influence Sampling (RIS), to estimate the influence in IC model. RIC generates a hypergraph $\mathcal{H}$ consisting of random hyperedges where each hyperedge $\mathcal{E}_j$ is constructed as follows. First, a node u is chosen uniformly at random and then they travel in the *reversed graph* to infer which nodes can influence u. Repeating that process multiple times will provide us with information on the influence landscape of the network. In $\mathcal{H}$, the degree of a set $S \in V$ is defined as $deg_{\mathcal{H}}(S) = |\{\mathcal{E}|\mathcal{E} \cap S \neq \emptyset\}|$. Intuitively, if node u has higher influence to other people than node v, with high probability, node u will appear more often in a random set of hyperedges than node v. The other direction is also very inherent. The following Lemma (see [11] for details) capture this intuition

Lemma 31. *For each subset of nodes $S \in V$,*

$$\mathbb{I}(S) = n \Pr[\mathcal{E} \cap S \neq \emptyset] \tag{3.101}$$

where $\mathcal{E}$ is a random hyperedge.

Thus, the problem of estimating the influence of a subset S ($\mathbb{I}(S)$) is transformed into the problem of estimating $\Pr[\mathcal{E} \cap S \neq \emptyset]$. Based on this result, the framework of using RIS for IM problem consists of two steps:

- Generate a sufficiently large random hyperedges using RIS to capture the influence landscape in the network.
- Find a seed set that covers the maximum number of hyperedges by using greedy approach on hyperedges.

In the first step, with the number of hyperedges

$$\theta = (8 + 2\epsilon)\frac{n(\ln n + \ln \binom{n}{k} + \ln 2)}{OPT_k \epsilon^2}, \tag{3.102}$$

TIM/TIM+ [106] return a $(1 - 1/e - \epsilon)$-approximate solution with probability $1/n$.

We, now, determine the number of samples required to approximate accurately a fixed subset $S \in V$ with high probability. If we define i.i.d. random variables $X_1, ..., X_T$ having mean value $\mathcal{E}[X_i] = \mu = \Pr[\mathcal{E} \cap S \neq \emptyset] = \frac{\mathbb{I}(S)}{n}$, then an estimation of $\mathbb{I}(S)$ is given by

$$\hat{\mathbb{I}}(S) = n\hat{\Pr}[\mathcal{E} \cap S \neq \emptyset] = \frac{\sum_{i=1}^{T} X_i}{T} n = \frac{deg_{\mathcal{H}}(S)}{T} n. \tag{3.103}$$

where hypergraph $\mathcal{H}$ contains T hyperedges. However, how large T have to be in order to approximate $\Pr[\mathcal{E} \cap S \neq \emptyset]$ accurately. The following Lemmas provide us the number of samples needed to bound approximation error.

Lemma 32. *Let $X_1, ..., X_T$ be i.i.d. random variables with $\mathcal{E}[X_i] = \mu$. For any fixed*

$T > 0$,

$$\Pr[\hat{\mu} \geq (1+\epsilon)\mu] \leq e^{\frac{-T\mu\epsilon^2}{2c}}$$

and

$$\Pr[\hat{\mu} \leq (1-\epsilon)\mu] \leq e^{\frac{-T\mu\epsilon^2}{2c}}.$$

where $\hat{\mu} = \frac{\sum_{i=1}^{T} X_i}{T}$.

Lemma 33. *Given* $0 \leq \epsilon \leq 1$ *and* $0 \leq \delta \leq 1$*, if we have*

$$T = 2c\ln(\frac{2}{\delta})\frac{1}{\epsilon^2\mu} \tag{3.104}$$

i.i.d. random variables $X_1, ..., X_T$ *with* $\mu_{X_i} = \mu$ *then*

$$\Pr[|\hat{\mu} - \mu| \geq \epsilon\mu] \geq 1 - \delta \tag{3.105}$$

The probability in 3.105 is called (ϵ, δ)-approximation. Lemma 41 is a tight version of Chernoff-Hoeffding theorem and Lemma 33 is the Generalized Zero-One Estimator Theorem (See [26] for both). Applying Lemma 41 to the approximation of $\Pr[\mathcal{E} \cap S \neq \emptyset]$, we find that

$$T = 2c\ln(23/\delta)\frac{1}{\epsilon^2\mu_{X_i}} = 2c\ln(\frac{2}{\delta})\frac{n}{\epsilon^2\mathbb{I}(S)}. \tag{3.106}$$

is the sufficient number of samples to guarantee

$$\Pr[|\hat{\mathbb{I}}(S) - \mathbb{I}(S)| \leq \epsilon\mathbb{I}(S)] \geq 1 - \delta \tag{3.107}$$

Thus, the most straightforward way of using RIS on CTVM problem, where node u has a benefit $b(u)$, is to define i.i.d. random variables $X'_1, ..., X'_n$ with mean $E[X'_i] = \frac{\mathbb{B}(S)}{n} = \frac{\sum_{v \in V} \mathbb{I}(S,v)b(v)}{n}$ and a sample corresponds to $1_{(S \cap \mathcal{E}_v \neq \emptyset)} b(v)$ where v is random. However, this intuitively adapted algorithms requires the number of samples up to n times larger than that for IM problem as shown by the following Lemma with the proof in the Appendix.

Lemma 34. *The naive algorithm using RIS sampling for CTVM problem requires*

$$T' = 2c \ln(2/\delta) \frac{n b_{max}}{\epsilon^2 \mathbb{B}(S)} \tag{3.108}$$

samples to estimate expected benefit of a set S.

Comparing T in 3.106 and T' in 3.108: 1) In case of Influence Maximization problem where all nodes have the same benefit of 1, $T = T'$; 2) In the worst case where $b_{max} = n$ and $\mathbb{B}(S) = O(\mathbb{I}(S))$, then the adapted TIM+ to CTVM requires n times that number for IM problem.

3.2.2.2 Efficient Sampling Strategy for CTVM

Due to the inefficiency of RIS in CTVM, we propose an efficient adapted version of RIS, called Benefit-aware Reverse Influence Sampling (BSA), for estimating expected benefit of a seed set. The BSA procedure to generate a random hyperedge $\mathcal{E}_j \subseteq V$ under LT model is summarized in Algorithm 17. A procedure for IC model can be similarly constructed. The great deal of difference of BSA is where we choose the starting node proportional to node benefit as opposed to choosing uniformly at random in RIS. That is the probability of choosing node u is $P(u) = \frac{b(u)}{\sum_{v \in V} b(v)} = \frac{b(u)}{\Gamma}$. After choosing a starting node u, we attempt to select an *in-neighbor* v of u, i.e. (v, u) is an edge of $\mathcal{G}$, according to the edge weights. Then we "move" to v and repeat, i.e. to continue the process with v replaced by u. The procedure stops when we encounter a previously visited vertex or no edge is selected. The hyperedge is then returned as the set of nodes visited along the process.

The key insight into why random hyperedges generated via BSA can capture the benefit landscape is stated in the following Lemma.

Algorithm 14: Benefit-aware Reverse Influence Sampling under LT model (BSA-LT)

Input: Weighted graph $\mathcal{G} = (V, E, w)$

Output: A random hyperedge $\mathcal{E}_j \subseteq V$.

1: $\mathcal{E}_j \leftarrow \emptyset$

2: Pick a node u with probability $\frac{b(u)}{\Gamma}$.

3: **Repeat**

4: Add u to $\mathcal{E}_j$

5: Attempt to select an edge (v, u) using live-edge model

6: **if** edge (v, u) is selected **then** Set $u \leftarrow v$.

7: **Until** $(u \in \mathcal{E}_j)$ OR (no edge is selected)

8: Return $\mathcal{E}_j$

Lemma 35. *Given a fixed seed set $S \subseteq V$, for a random hyperedge $\mathcal{E}$,*

$$\Pr[\mathcal{E} \cap S \neq \emptyset] = \frac{\mathbb{B}(S)}{\Gamma}$$

Proof.

$$\begin{aligned}
\mathbb{B}(S) &= \sum_{u \in V} \Pr[\text{ u is reachable from a node in } S] b(u) \\
&= \sum_{u \in V} \Pr[\exists v \in S \text{ such that } v \in \mathcal{E}(u)] b(u) \\
&= \Gamma \sum_{u \in V} \Pr[\exists v \in S \text{ such that } v \in \mathcal{E}(u)] \frac{b(u)}{\Gamma} \\
&= \Gamma \Pr[\exists v \in S \text{ such that } v \in \mathcal{E}] \\
&= \Gamma \Pr[S \cap \mathcal{E} \neq \emptyset] \qquad (3.109)
\end{aligned}$$

The transition from the third to forth equality follows from the distribution of choosing node u as a starting node of BSA. We select u with probability $P(u) = \frac{b(u)}{\Gamma}$, hence, the

forth equality contains the expected probability taken over the node benefit distribution.

Now, we analyze the number of samples needed to (ϵ, δ)-approximate a random set S. First, define variable $X = 1_{(S \cap \mathcal{E} \neq \emptyset)}$ where $\mathcal{E}$ is a random hyperedge with the mean value $\mu = \Pr[S \cap \mathcal{E}] = \frac{\mathbb{B}(S)}{\Gamma}$ (followed by Eq. 3.129). Thus, applying Lemma 33 gives

$$T = 2c\frac{\Gamma \ln 2/\delta}{\epsilon^2 \mathbb{B}(S)}. \tag{3.110}$$

This number of samples is asymptotically optimal based on the results in [26]. By comparing Eq. 3.110 with Eq. 3.108, we see that algorithm using BSA requires up to n times fewer samples than the straightforward strategy.

3.2.3 BCT Approximation Algorithm

In this section, we present BCT - a scalable approximation algorithm for CTVM. BCT combines two novel techniques: BSA (Alg. 15), a sampling strategy to estimate the benefit and a powerful stopping condition to smartly detect when the sufficient number of hyperedges is reached.

Algorithm 15: BSA - Benefit Sampling Alg. for LT model

Input: Weighted graph $\mathcal{G} = (V, E, w)$.
Output: A random hyperedge $\mathcal{E}_j \subseteq V$.
1: $\mathcal{E}_j \leftarrow \emptyset$;
2: Pick a node u with probability $\frac{b(u)}{\Gamma}$;
3: **repeat**
4: Add u to $\mathcal{E}_j$;
5: Attempt to select an edge (v, u) using live-edge model;
6: **if** edge (v, u) is selected **then** Set $u \leftarrow v$;
7: **until** $(u \in \mathcal{E}_j)$ OR (no edge is selected);
8: **return** $\mathcal{E}_j$;

3.2.3.1 Efficient Benefit Sampling Algorithm - BSA

Due to the inefficiency of RIS when applying to CTVM problem, we propose a generalized version of RIS, called Benefit Sampling Algorithm - BSA, for estimating benefit

$\mathbb{B}(S)$. The BSA for generating a random hyperedge $\mathcal{E}_j \subseteq V$ under LT model is summarized in Algorithm 15. A similar BSA procedure for IC model can be derived by changing the generating of live-edges in the Lines 5 and 6 of Algorithm 15 to the equivalent live-edge model for IC [55]. The great deal of difference of BSA from RIS is that it *chooses the source node proportional to benefit of each node* as opposed to choosing uniformly at random in RIS. That is the probability of choosing node u is $P(u) = b(u)/\Gamma$ with $\Gamma = \sum_{v \in V} b(v)$. After choosing a starting node u, it attempts to select an *in-neighbor* v of u according to the LT model and make (v, u) a *live* edge. Then it "moves" to v and repeat the process. The procedure stops when we encounter a previously visited vertex or no edge is selected. The hyperedge is the set of nodes visited along the process.

Note that the selection of a source node with the probability proportional to the benefit can be done in $O(1)$ after an $O(n)$ preprocessing using the Alias method [109]. Similarly, the selection of the live edge according to the influence weight can also be done in $O(1)$. In contrast, in the IC model [11], it takes a time $\theta(d(v))$ at a node v to generate all live edges pointing to v. *This key difference makes the generating hyperedges in the LT model more efficient than that in the IC.*

The key insight into why random hyperedges generated via BSA can capture the benefit landscape is stated in the following Lemma.

Lemma 36. *Given a fixed set $S \subseteq V$, for a random hyperedge $\mathcal{E}$,*

$$\Pr_{g \sqsubseteq G, u \in V}[\mathcal{E}_j \cap S \neq \emptyset] = \frac{\mathbb{B}(S)}{\Gamma}. \tag{3.111}$$

The above Lemma on computing benefit is similar to Lemma 30 on influence except having the *normalizing constant* Γ *in the place of* n in Lemma 30. Thus, the RIS framework can be applied and a similar result to Theorem 1 in [105] on the threshold of hyperedges can be derived as follows.

Corollary 3. *Let*

$$\theta_B(\epsilon, \delta) = \frac{(2 - 2/e)\Gamma(\ln \binom{n}{k} + \ln(1/\delta) + \ln 2)}{\mathsf{OPT}_k \epsilon_2^2}. \tag{3.112}$$

For any fixed $T \geq \theta_B$, the RIS framework with T random hyperedges, generated by BSA, will return an $(1 - 1/e - \epsilon)$-approximate solution for the CTVM problem.

3.2.3.2 Solving Budgeted Max-Coverage Problem

Finding a candidate seed set $\hat{S}_k$ that appears most frequently in the hyperedges is a special version of the Budgeted Max-Coverage problem [57]. Each hyperedge represents an element in the *Budgeted Max-Coverage* problem and each node $v \in V$ is associated with a subset of hyperedges that contains v. The cost to select a subset is given by the cost to select the corresponding node into the seed set.

We use the greedy algorithm, denoted by Budgeted-Max-Coverage, in [57] to find a maximum covering set within the budget B is applied. This procedure considers two candidates and chooses the one with higher coverage. The first one is taken from greedy strategy which sequentially selects nodes with highest efficiency, i.e. ratio between marginal coverage gain and its cost of selecting,

$$\forall i = 1..k, v_i = \arg \max_{v \in V \setminus S_{i-1}} \Delta(S_{i-1}, v), S_i = S_{i-1} \cup \{v_i\}$$

where $\Delta(S_{i-1}, v) = \frac{\mathrm{Cov}(S_{i-1} \cup \{v\}) - \mathrm{Cov}(S_{i-1})}{c(v)}$ and $\mathrm{Cov}(S_{i-1})$ is the number of hyperedges incident to at least a node in S_{i-1}. The second solution is just a node with highest coverage within the budget. [57] proved that this procedure returns a $(1 - 1/\sqrt{e})$-approximate cover if the nodes' cost are non-uniform, or, $(1 - 1/e)$-approximate cover, otherwise.

Note that we can improve the approximation ratio to $(1 - 1/e)$ for the case of non-uniform costs, however, the time complexity ($\Omega(n^4)$) becomes impractical.

Algorithm 16: BCT Algorithm

Input: Graph $G = (V, E, b, c, w)$, budget $B > 0$, and two precision parameters $\epsilon, \delta \in (0, 1)$.
Output: $\hat{S}_k$ - An $(1 - 1/e - \epsilon)$-approximate seed set.

$\Lambda_L = (1 + \epsilon_2)\frac{2+2\epsilon_2/3}{2-2/e}\theta_B(\epsilon, \delta/3)\mathsf{OPT}_k$
(Or $\Lambda_L = \Lambda_L^{cost}$ in Eq. 3.121 for non-uniform costs;)
$N_t = \Lambda_L$; $\mathcal{H} \leftarrow (V, \mathcal{E} = \emptyset)$; $t \leftarrow 0$;
repeat
  for $j = 1$ **to** $N_t - |\mathcal{E}|$ **do**
    Generate $\mathcal{E}_j \leftarrow$BSA($\mathcal{G}$); Add $\mathcal{E}_j$ to $\mathcal{E}$;
  end for
  $t \leftarrow t + 1$; $N_t = 2N_{t-1}$;
  $\hat{S}_k =$ Budgeted-Max-Coverage($\mathcal{H}$, B);
until $deg_{\mathcal{H}}(\hat{S}_k) \geq \Lambda_L$;
return $\hat{S}_k$;

3.2.3.3 BCT - The Main Algorithm

BCT algorithm for the CTVM problem is presented in Algorithm 16. The algorithm uses BSA (Algorithm 15) to generate hyperedges and Budgeted-Max-Coverage [88] to find a candidate seed set $\hat{S}_k$ following the RIS framework.

BCT keeps generating hyperedges by BSA sampling (Algorithm 15) until the degree of the seed set selected by Budgeted-Max-Coverage exceeds a threshold Λ_L (the stopping condition). Specifically, at iteration $1 \leq t \leq O(\log n)$, it consider the hypergraph $\mathcal{H}$ that consists of the first $2^{t-1}\Lambda_L$ hyperedges. That is the number of samples (aka hyperedges) are double after each iteration. In each iteration, Budgeted-Max-Coverage algorithm is called to select a seed set $\hat{S}_k$ within the budget B and stops the algorithm if the degree of $\hat{S}_k$ exceeds Λ_L, $deg_{\mathcal{H}}(\hat{S}_k) \geq \Lambda_L$. Otherwise, it advances to the next iteration.

3.2.4 Approximation and Complexity Analysis

We prove that BCT will stop within $O(\theta_B)$ samples (aka hyperedges) and return an $(1 - 1/e - \epsilon)$-approximate solution.

Note that BCT can be used with any threshold for the sufficient number of samples (not only the one in [105]). That is if a better threshold $\theta' < \theta$ exists, we can use θ' in BCT

to guarantee BCT will stop within $O(\theta')$ samples whp.

3.2.4.1 Approximation Guarantee for uniform cost CTVM

Assume the case of uniform node cost. The proof consists of two steps: 1) the "stopping time" (aka the number of hyperedges) $m_{\mathcal{H}}$ concentrates on an interval $[T^*, \mathsf{c}T^*]$ for some fixed $\mathsf{c} > 4$ (Lemma 37 and 39); and 2) for that interval the candidate seed set $\hat{S}_k$ is a $(1 - 1/e - \epsilon)$-approximate solution whp (Lemma 38).

Given a seed set $S \subset V$, denote by $\hat{\mathbb{B}}_T(S)$ and $deg_T(S)$ the estimate of $\mathbb{B}(S)$ and the degree of S of the hypergraph with the first T random hyperedges, respectively.

Lemma 37. *Let $T^* = \frac{2+2\epsilon_2/3}{2-2/e}\theta_B(\epsilon, 2\delta_2) = \frac{\Lambda_L \Gamma}{(1+\epsilon_2)\mathsf{OPT}_k}$ hyperedges, where ϵ_2 is defined in Eq. 3.98 and $\delta_2 = \delta/6$. We have,*

$$\Pr[m_{\mathcal{H}} \leq T^*] \leq \delta_2. \tag{3.113}$$

Let $t_0 = \left\lceil \log_2 \frac{T^*}{\Lambda_L} \right\rceil + 1$ be the smallest iteration such that $2^{t_0-1}\Lambda_L \geq T^*$. The above lemma is equivalent to,

$$\Pr[t < t_0] \leq \delta_2. \tag{3.114}$$

For iterations $t \geq t_0$, we now show that the candidate solution $\hat{S}_k$ will be an $(1-\frac{1}{e}-\epsilon)$-approximate solution whp.

Lemma 38. *For any iteration $t \geq t_0$, the candidate solution $\hat{S}_k$ satisfies that*

$$\Pr[\mathbb{B}(\hat{S}_k) \leq (1 - 1/e - \epsilon)\mathsf{OPT}_k] \leq (2\delta_2)^{2^{t-t_0}}. \tag{3.115}$$

Proof. This is a direct consequence of Corollary 3. We can verify that the number of samples in iteration t is

$$|\mathcal{E}| = 2^{t-1}\Lambda_L \geq 2^{t-t_0}\theta_B(\epsilon, 2\delta_2) \geq \theta_B(\epsilon, (2\delta_2)^{2^{t-t_0}}). \tag{3.116}$$

This yields the proof. □

The upper-bound on the number of hyperedges generated by BCT is stated in the following lemma.

Lemma 39. *For* $\epsilon \in (0, 1-1/e)$ *and* $\mathsf{c} = 4\left\lceil \frac{1+\epsilon_2}{1-1/e-\frac{\epsilon+\epsilon_2}{2}} \right\rceil$,

$$\Pr[m_{\mathcal{H}} \geq \mathsf{c}T^*] \leq \delta_2. \tag{3.117}$$

Finally, we prove the overall approximation guarantee of BCT in the following Theorem 24.

Theorem 13. *Given* $0 < \epsilon < 1 - 1/e, 0 < \delta < 1$,

$$\Pr[m_{\mathcal{H}} = O(\theta_B(\epsilon, \delta)) \textit{and } \mathbb{B}(\hat{S}_k) \geq (1 - \frac{1}{e} - \epsilon)\mathsf{OPT}_k] \geq 1 - \delta.$$

Proof. Assume that none of the following "bad" events in Lemmas 37, 39 and 38 happens.

(b1) $\Pr[m_{\mathcal{H}} \leq T^*] \leq \delta_2$

(b2) $\Pr[m_{\mathcal{H}} \geq \mathsf{c}T^*] \leq \delta_2$

(b3) $\forall t \geq t_0, \Pr[\mathbb{B}(\hat{S}_k) \leq (1 - 1/e - \epsilon)\mathsf{OPT}_k] \leq (2\delta_2)^{2^{t-t_0}}$

That is the following inequalities

(i1) $m_{\mathcal{H}} \geq T^*$,

(i2) $m_{\mathcal{H}} \leq \mathsf{c}T^*$, and

(i3) $\forall t \geq t_0, \mathbb{B}(\hat{S}_k) \geq (1 - 1/e - \epsilon)\mathsf{OPT}_k$

hold together with probability at least

$$\begin{aligned} &1 - [\delta_2 + \delta_2 + (2\delta_2 + (2\delta_2)^2 + (2\delta_2)^4 + \dots)] \\ \geq &1 - (2\delta_2 + \frac{2\delta_2}{1 - 2\delta_2}) \geq 1 - \delta \end{aligned}$$

The last one is due to $\delta_2 = \delta/6 \leq 1/6$.

From the above inequalities, we will have $T^* \leq m_{\mathcal{H}} \leq cT^*$. And the algorithm will stop in one of (at most) $\log_2 \mathsf{c} + 1$ iterations, starting from t_0. Further, no matterwhat the iteration that the algorithm will stop at, the candidate seed set $\hat{S}_k$ satisfies $\mathbb{B}(\hat{S}_k) \geq (1 - 1/e - \epsilon)\mathsf{OPT}_k$. Since $T^* = O(\theta_B(\epsilon, \delta)$,

$$\Pr[m_{\mathcal{H}} = O(\theta_B(\epsilon, \delta)) \text{and } \mathbb{B}(\hat{S}_k) \geq (1 - \frac{1}{e} - \epsilon)\mathsf{OPT}_k] \geq 1 - \delta.$$

That completes the proofs. □

3.2.4.2 Time Complexity

The overall time complexity of BCT comprises of two components: 1) for generating hyperedges and 2) for running Greedy algorithm for Max-Coverage. The result is stated in the following theorem and the proof is presented in our conference paper [88].

Theorem 14. *BCT has an expected running time for uniform cost CTVM problem under LT model of* $O(\frac{\log(\binom{n}{k}/\delta)}{\epsilon_2^2} n)$.

Remark. From Theorem 14, under the LT model, the time complexity does not depend on the number of edges in the original graph, hence, uniform-cost BCT has a sublinear time complexity in dense graphs.

3.2.4.3 Sample Complexity and Comparison to IMM

Since the number of samples (hyperedges) decides the complexity of BCT, IMM [105] and any algorithm using sampling techniques, we compare number of hyperedges generated by BCT with the current state-of-the-art IMM. We can prove a tighter version of Lemma 39 as stated in the lemma below.

Lemma 40. *Let* $\delta_2 \in (0, 1), 0 < \epsilon < (1 - 1/e)$, *BCT returns* $\hat{S}_k$,

$$deg_{\mathcal{H}}(\hat{S}_k) \leq 2\frac{(1 + \epsilon_2) \cdot (2 + 2\epsilon_2/3) \cdot \log(6\binom{n}{k}/\delta_2)}{\epsilon_2^2}, \tag{3.118}$$

due to doubling hyperedges every round and,

$$\Pr[m_{\mathcal{H}} \geq \frac{2}{(1-\epsilon_2)(1-1/e-\epsilon)} T^*] \leq \delta_2 = \delta/6. \tag{3.119}$$

In comparison with IMM [105], BCT theoretically generates at least $3/2$ times fewer samples than IMM. IMM approaches the problem by trying to achieve an estimate KPT^+ of OPT_k such that $KPT^+ \leq OPT_k$ and then deriving the sufficient number of samples by replacing OPT_k in T^* of Lemma 37 by KPT^+ and the constant $(2 + 2\epsilon_2/3)$ by $(2 - 2/e)$ to get T_2^*. Thus, Lemma 9 in [105] states the number of samples generated by IMM, $|\mathcal{R}|$, as follows,

$$\Pr\left[|\mathcal{R}| \leq 3\frac{(1+\epsilon')^2}{1-1/e} \max\{T_2^*, T_2'\}\right] \geq 1 - \delta, \tag{3.120}$$

where $\epsilon' = \sqrt{2}\epsilon$ and

$$T_2' = \frac{(2 + \frac{2}{3}\epsilon')(\log \binom{n}{k} + \log(1/\delta) + \log\log_2(n)) \cdot n}{\epsilon'^2 OPT_k}.$$

Comparing Eqs. 3.120 and 3.119, we see that $\max\{T_2^*, T_2'\} \geq T^*$ and $3\frac{1+\sqrt{2}\epsilon}{1-1/e} > 2\frac{1}{(1-\epsilon_2)(1-1/e-\epsilon)}$ (assume that ϵ is small). Thus, the number of samples generated by BCT is always less than that of IMM and the ratio between the two is approximately $3/2$ (when $\epsilon \leq \sqrt{2} - 1 > 0.4$ which is usually the case). In fact, our experiments show that BCT is up to 10x faster than IMM proving the practical efficiency.

3.2.4.4 Approximation Algorithm for Arbitrary Cost CTVM

We analyze the CTVM algorithm under the heterogeneous selecting costs. First observe that in this case, the candidate seed sets may have different sizes since the total cost of each set must be less that the given budget B. However, we can obtain an upper-bound $k_{max} = \max\{k : \exists S \subset V, |S| = k, c(S) \leq B\}$ by iteratively selecting the smallest cost nodes until reaching the budget B. We then guarantee that all subsets of size up to k_{max}

are well approximated. The number of such seed sets is subsequently bounded above by $\sum_{k \leq k_{max}} \binom{n}{k} \leq n^{k_{max}}$. Thus, the computation of α and β at the step of calculating ϵ_1 are updated to ϵ_1',

$$\epsilon_2' = \frac{\epsilon\sqrt{(1-1/e)k_{max}\log(n \cdot 2/\delta)}}{(1-1/e)\sqrt{\log(2/\delta)} + \sqrt{(1-1/e)k_{max}\log(n \cdot 2/\delta)}}$$

Thus, Λ_L is also updated to Λ_L^2 as follows,

$$\Lambda_L^{cost} = \frac{(1+\epsilon_2') \cdot (2 + 2\epsilon_2'/3) \cdot \log(6\binom{n}{k}/\delta)}{\epsilon_2'^2}. \tag{3.121}$$

In addition, the Weighted-Max-Coverage algorithm used in CTVM only guarantees $(1 - 1/\sqrt{e})$ approximate solutions, as shown in [56]. Putting these modifications together, we have the following Theorem 15. The proofs are similar to that of Theorem 24 and 14 and is omitted for clarity.

Theorem 15. *Given a budget B, $0 \leq \epsilon \leq 1$ and $0 \leq \delta \leq 1$, BCT for arbitrary cost CTVM problem returns a solution $\hat{S}$ that,*

$$\Pr[\mathbb{B}(\hat{S}) \geq (1 - 1/\sqrt{e} - \epsilon)OPT] \geq 1 - \delta, \tag{3.122}$$

and runs in time $O(\frac{\log(\binom{n}{k}/\delta)}{\epsilon_2'^2} n)$.

3.2.4.5 Extension to IC model

When applying BCT for IC model, the only change is in the BSA procedure to generate hyperedges following the IC model, as originally presented in [11]. Thus, our results for LT model translate directly over for IC model. Specifically, the following theorem states the solution guarantee and time complexity of BCT to the uniform cost version.

Theorem 16. *Given a budget B, $0 \leq \epsilon \leq 1$ and $0 \leq \delta \leq 1$, BCT for uniform cost CTVM problem returns $\hat{S}$ where*

$$\Pr[\mathbb{B}(\hat{S}) \geq (1 - 1/e - \epsilon)OPT] \geq 1 - \delta, \tag{3.123}$$

and runs in time $O(\frac{\log(\binom{n}{k}/\delta)}{\epsilon_2^2}(m+n))$.

Similar to Theorem 17, we obtain the performance guarantee for the arbitrary cost version under IC model in the following theorem.

Theorem 17. *Given a budget* B, $0 \leq \epsilon \leq 1$ *and* $0 \leq \delta \leq 1$, *for arbitrary cost* CTVM *problem,* BCT *returns a solution* $\hat{S}$,

$$\Pr[\mathbb{B}(\hat{S}) \geq (1 - 1/\sqrt{e} - \epsilon)OPT] \geq 1 - \delta, \tag{3.124}$$

and runs in time $O(\frac{\log(\binom{n}{k}/\delta)}{\epsilon_2'^2}(m+n))$.

3.2.5 Experiments

In this section, we experimentally evaluate and compare the performance of BCT to other influence maximization methods on three aspects: *the solution quality, the scalability, and the applicability* of BCT on various network datasets including our case study on a billion-scale dataset with both links and content.

3.2.5.1 Experimental Settings

All the experiments are carried on a Linux machine with a 2.2Ghz Xeon 8 core processor and 64GB of RAM.

Algorithms compared: We choose three groups of methods to test on:

(1) Designed for IM task, including the top four state-of-the-art algorithms, i.e., IMM [105], TIM/TIM+ [106], CELF++ [44] and SIMPATH [45].

(2) Designed for BIM task, namely, BIM algorithm [87].

(3) Our method BCT for the general CTVM problem.

In the first experiment, we will compare between these groups of methods on CTVM problem and the second experiment reports results on IM task. Our last set of experiments are on Twitter - a billion-scale network where we first test the scalability of BCT against IMM and TIM+ (the current most scalable methods for solving IM problem) on IM task. Next, we acquire a Twitter's tweet dataset and extract two groups of users who tweet/retweet the same topic and run our BCT algorithm to find the users who attract the most interested people in the same topics.

Table 18.: Datasets' Statistical Summary

Dataset	#Nodes	#Edges	Type	Avg. degree
NetHEPT [19]	15K	59K	undirected	4.1
NetPHY [19]	37K	181K	undirected	13.4
Enron [60]	37K	184K	undirected	5.0
Epinions[19]	132K	841K	directed	13.4
DBLP [19]	655K	2M	undirected	6.1
Twitter [65]	41.7M	1.5G	directed	70.5

Datasets: For a comprehensive experimental purpose, we select a diverse set of 6 datasets with sizes from thousands to millions in various disciplines: NetHEPT, NetPHY, DBLP are citation networks, Email-Enron is communication network, Twitter and Epinions are online social networks. The description summary of those datasets is provided in Table 18. **Parameter Settings:** *Computing the edge weights*. Following the conventional computation as in [106, 20, 45, 87], the weight of the edge (u, v) is calculated as follows,

$$w(u, v) = 1/d_{in}(v) \tag{3.125}$$

where $d_{in}(v)$ denotes the in-degree of node v.

Computing the node costs. Intuitively, the more famous one is, the more difficult it is to

convince that person. Hence, we assign the cost of a node proportional to the out-degree:

$$c(u) = nd^{out}(u)/\sum_{v\in V} d^{out}(v) \tag{3.126}$$

where $d^{out}(v)$ is the out-degree of node v.

Computing the node benefits. In the first experiment, we choose a random $p = 20\%$ of all the nodes to be the target set and assign benefit 1 to all of them while in case studies, the benefit is learned from a separate dataset.

In all the experiments, we keep $\epsilon = 0.1$ and $\delta = 1/n$ as a general setting or directly mentioned otherwise. For the other parameters, we take the recommended values in the corresponding papers if available.

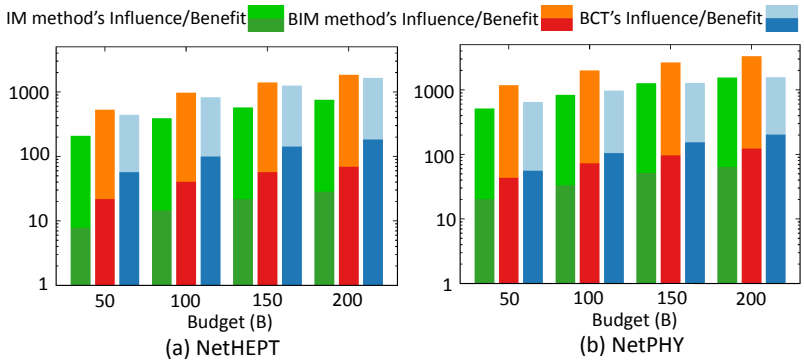


Fig. 20.: Comparisons on CTVM problem. The whole column indicates influence of the selected seeds while the darker colored portion reflects the benefit gained from that set.

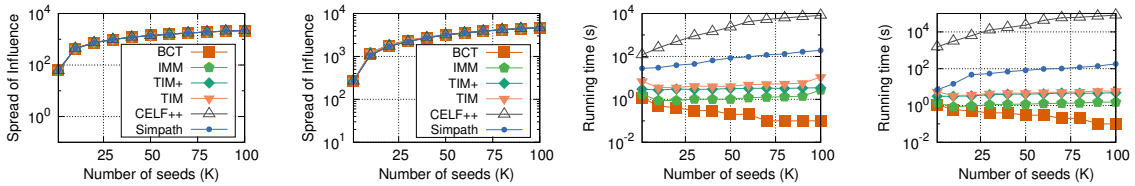
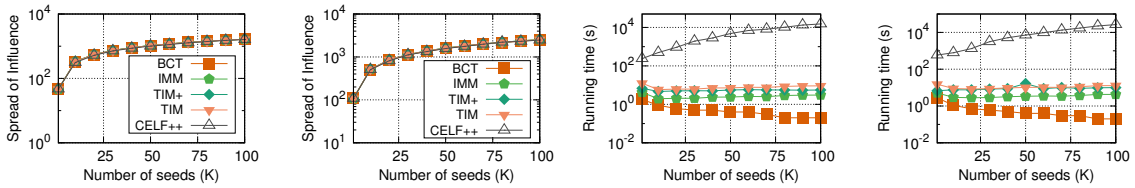


Fig. 21.: Comparison on IM problem under the LT model.



(a) NetHEPT **(b)** NetPHY **(c)** NetHEPT **(d)** NetPHY

Fig. 22.: Comparison on IM problem under the IC model.

3.2.5.2 Experimental results

We carry three experiments on both CTVM and IM tasks to compare the performance of BCT with other state-of-the-art methods. In the first experiments, we compare three groups of algorithms, namely, IM based methods, BIM and BCT on CTVM problem. We choose four algorithms in the category of IM methods: CELF++, SIMPATH, TIM/TIM+ and IMM, which are well known algorithms for IM. The results are presented in Fig. 20. We conduct the second and third experiments on the classical IM task with different datasets and various k values. The results are shown in Table 19 and Fig. 21 for LT model and Fig. 22 for IC model.

Table 19.: Comparison between different methods on IM problem and various datasets (with $\epsilon = 0.1, k = 50, \delta = \frac{1}{n}$).

Method	**Spread of Influence**			**Running Time (s)**		
	Epin.	*Enron*	*DBLP*	*Epin.*	*Enron*	*DBLP*
BCT	16280	16726	108400	**0.19**	**0.14**	**0.58**
IMM	16290	16716	108430	2	1.5	3.5
TIM++	16293	16732	108343	6	3	12
TIM+	16306	16749	107807	8	4	17
Simpath	16291	16729	103331	23	18	136

3.2.5.3 Comparison of solution quality on CTVM

Fig. 20 shows the results of the three groups of methods (CELF++ with 10000 sampling times represents the first group) for solving CTVM problem on NetHEPT and NetPHY networks. We see that BCT outperforms the other methods by a large margin on CTVM problem. With the same amount of budget, CTVM returns a solution which is up to order of magnitudes better than that of BIM and IM based methods in terms of benefit. Because IM algorithms only desire to maximize the influence and thus usually aim at the most influential nodes, unfortunately, those nodes are very expensive or have high cost. As a consequence, when nodes have heterogeneous cost, IM methods suffer severely in terms of both influence and benefit. On the other hand, BIM optimizes cost and influence while ignoring benefit of influencing nodes that causes BIM to select cheaper nodes with high influence. Hence, the seed sets returned by BIM have the highest influence among all but relatively low benefit. Even though BCT returns seed set with lower value of influence than BIM, the majority of the influenced nodes are our target and thus achieves the most benefit.

3.2.5.4 Comparison of solution quality on IM

In the previous experiment, one can argue that CTVM performs better because it focuses on optimizing the benefit and the others do not. This experiment compares BCT to the other algorithms with IM problem where the node costs are all 1 and so as the node benefits on various datasets. Fig. 21 and Fig. 22 display the spread of influence and running time on NetHEPT and NetPHY under the LT and IC models respectively. Table 19 shows the cross-dataset view of the results when we fix a setting and run on multiple data.

Fig. 21, Fig. 22 and Table 19 reveal that all the tested algorithms including BCT and the top methods on IM problem achieve the same level of performance in terms of spread of influence. Specifically, they all expose the phenomenon that the first few seed nodes (≤ 25)

can influence a large portion of the networks and after that point, the gains of adding more seeds are marginal. The phenomenon is explained by the submodularity property of the influence function.

3.2.5.5 Comparison of running time

The experimental results in Fig. 21, Fig. 22 and Table 19 also confirm our theoretical establishment in Section 3.2.4 that BCT for uniform cost CTVM requires much less number of hyperedges needed by IMM, TIM and TIM+. As such, the running time of BCT in all the experiments are significantly lower than the other methods. In average, BCT is up to 10 and 50 times faster IMM and TIM/TIM+, respectively. Since both Simpath and CELF++ require intensive graph simulation, these methods have very poor performance compared to BCT, TIM/TIM+ and IMM which apply advanced techniques to approximate the influence landscape. That is illustrated by the distinct separation of two groups.

3.2.5.6 Robustness Testing

In this experiment, we test the robustness of our algorithm against noise possibly incurred in computing the edge weights in the diffusion model. We take NetHEPT with the previously calculated edge weights as our 'ground-truth' network and then add various noises, e.g., in different levels and noise models to it. In particular, we consider Gaussian and Uniform noise models where the added noise follows either a Gaussian or Uniform distribution respectively. To account for noise levels, we select 4 different values $0.2, 0.4, 0.6, 0.8$, which correspond to 20% to 80% noise since the edge weight is between 0 and 1, and assign them to be the variance of the distribution (larger variance signifies noisier data) while the mean values are set at 0. Thus, each pair of noise level and model, we have a specific distribution of noise and use that for generating noise. For each such pair, we generate 30 noisy networks and run BCT to find 50 seed nodes and then take the

average over 30 runs. We then use the original HetHEPT without noise to recompute the influence and quantify the effect of noise.

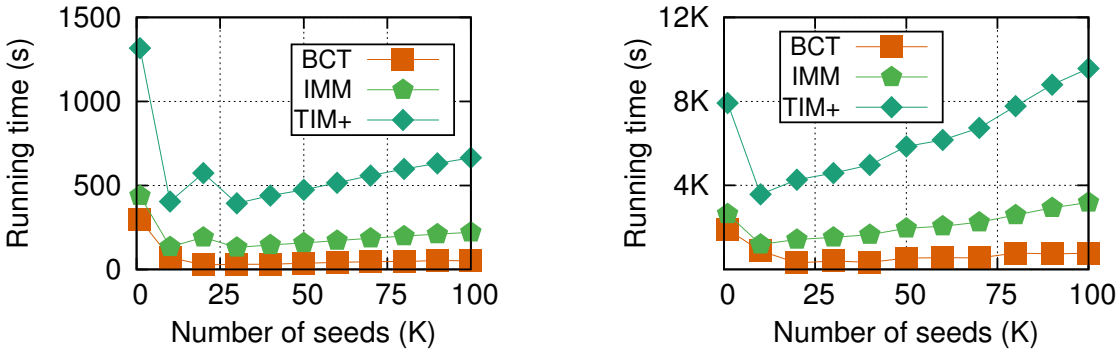
Table 20.: Robustness results (% to true value).

		Uniform Noise				**Gaussian Noise**			
ϵ	**True**	*0.2*	*0.4*	*0.6*	*0.8*	*0.2*	*0.4*	*0.6*	*0.8*
0.1	1588	99.7	99.2	98.6	97.9	98.9	98.0	96.7	95.7
0.2	1504	99.9	99.7	99.2	98.8	100.1	99.7	98.3	95.8
0.4	1312	99.9	99.5	98.4	98.8	100.9	99.4	98.7	98.2

The experimental results under the IC model are depicted in Table 20 where the 'true value' refers to the results run on the original network. Interestingly, BCT performs very well under the noises introduced to the network. For example, with 80% noise, the quality of BCT only degrades by less than 2% with Uniform noise and 5% with Gaussian case in average. In some rare cases on network with 20% Gaussian noise, we see the qualities of over 100% compared to true values. This happens when ϵ is large implying the provided solution guarantee $1-1/e-\epsilon$ is small. Thus, the seed set found on small noisy network may get better than that on the original. Moreover, different from the Uniform case, Gaussian noise is highly concentrated at 0.

3.2.6 Twitter: A billion-scale social network

In this subsection, we design two case studies on Twitter network: one is to compare the scalability of BCT with IMM and TIM++ - the fastest existing methods and the another is using BCT to find a set of users who have highest benefit with respect to a particular topic in Twitter.



(a) LT model. **(b)** IC model.

Fig. 23.: BCT, IMM and TIM++ on Twitter

3.2.6.1 Compare BCT against IMM and TIM+

Figure 23 shows the results of running BCT and TIM+ on Twitter network dataset using both LT and IC models with k ranging from 1 to 100. Twitter has 1.5 billion edges and all the other methods, except BCT, IMM and TIM+, fail to process it within a day in our experiments. The results, here, are consistent with the other results in the previous experiments. Regardless of the values of k, in LT model, BCT is always several times faster than IMM or TIM+ and in IC model, this ratio is in several orders of magnitude since influence in IC model is much larger and, thus, harder for IMM or TIM+ to have a close estimate of the optimality which decides the complexity of these algorithms.

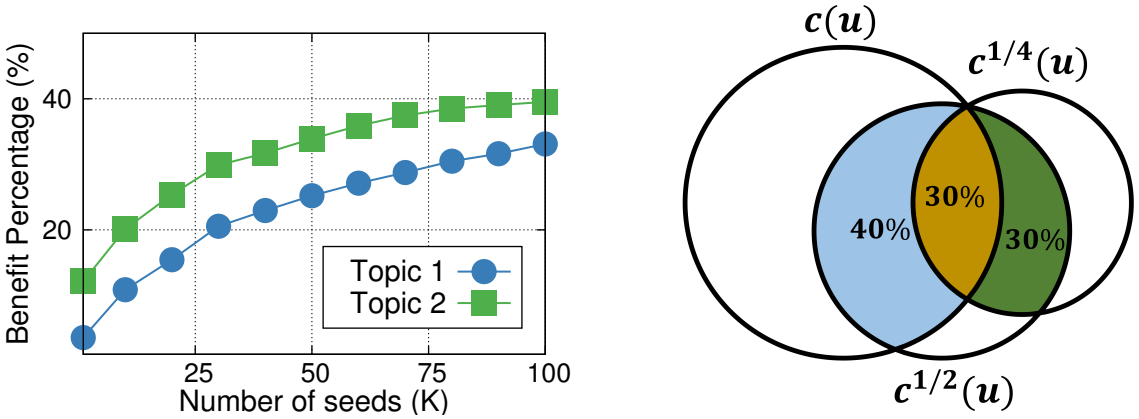
We also measure the memory consumed by these two algorithms and observe that, in average, BCT requires around 20GB but IMM and TIM+ always need more than 30GB. This is a reasonable since in addition the memory for the original graph, BCT needs much less number of hyperedges than that generated by IMM or TIM+.

3.2.6.2 A Case Study on Twitter network.

We study the twitter network using BCT by extracting some trending topics from the retrieved tweet dataset and find who are most influential in those topics based on Twitter

Table 21.: Topics, related keywords

Topic	Keywords	#Users
1	**bill clinton, iran, north korea, president obama, obama**	997K
2	**senator ted kenedy, oprah, kayne west, marvel, jackass**	507K



(a) Benefit on Twitter **(b)** Different Cost Schemes

Fig. 24.: Case Study on real-world Twitter

network. First we choose two most popular topics with related keywords (Table 21) as reported in [65]. Based on the list of keywords, we use a Twitter's tweet dataset to extract a list of users who mentioned the keywords in their posts and the number of those tweets/retweets. The number of tweets/retweets reveals the interest of the user on the topic, thus, we consider this as the benefit of that node. Lastly, we run BCT on Twitter with the extracted parameters.

Fig. 24a shows the benefit percentage, which is computed as the percentage of benefit gained by the selected seed set over the total benefit. We see that apparently the very first chosen nodes have high benefit and it continues increasing later but with much lower rate. Looking into the first 5 Twitters chosen by the algorithm, they are users with only few thousands of followers (unlike Katy Perry or President Obama who got more than

50 millions followers) but highly active posters in the corresponding topic. For example, on the first political topic, the first selected users is a Canadian poster, who is originally from Iran and has about 4000 followers and but generates more than 210K posts on the movements of governments in the US and Iran.

On Twitter we test different schemes of assigning node costs, i.e., proportional to a concave function. We employ square and fourth roots, denoted by $c^{1/2}(u)$ and $c^{1/4}(u)$ respectively, w.r.t. out-degree and run BCT on each case. The results are presented in Figure 24b. We see that BCT is relatively robust with different concave cost functions, e.g., 70% of nodes returned in case of $c^{1/2}(u)$ overlaps with that of $c(u)$-linear cost, and 60% overlap for the pair $c^{1/4}(u)$ to $c^{1/2}(u)$. Another interesting result is that the number of selected seeds gets smaller when the cost function gets farther from linear, i.e., $c(u) \rightarrow c^{1/2}(u)$ and $c^{1/2}(u) \rightarrow c^{1/4}(u)$.

3.2.7 Martingale View on Benefit Estimation

To recognize the connection between the expected benefit and martingales, we give a general definition as follows,

Definition 14 (Martingale)**.** *A sequence of random variables $Y_1, Y_2, \ldots$ is a martingale if and only if $\mathcal{E}[Y_i] \leq +\infty$ and $\mathcal{E}[Y_i|Y_1, Y_2, \ldots, Y_{i-1}] = Y_{i-1}$.*

For a random hyperedge $\mathcal{E}_j$, we define random variable,

$$X_j = \begin{cases} 1 & \text{if } S \cap \mathcal{E}_j \neq \emptyset \\ 0 & \text{otherwise.} \end{cases} \tag{3.127}$$

Thus, we have a sequence of random variables $X_1, X_2, \ldots$ corresponding to the series of random hyperedges. Then, we define the second sequence of random variables based on

$X_1, X_2, \ldots$ as follows,

$$Z_j = \sum_{i=1}^{j} (X_i - \mathbb{B}(S)/\Gamma) \tag{3.128}$$

The sequence $Z_1, Z_2, \ldots$ has the following properties [105]:

(1) $\mathcal{E}[Z_j] = 0, \forall j \geq 1$ (since $\mathcal{E}[Z_i] = \mathbb{B}(S)/\Gamma, \forall i \geq 1$)

(2) $\mathcal{E}[Z_j | Z_1, Z_2, \ldots, Z_{j-1}] = Z_{j-1}$

Thus, the two conditions (1) and (2) hold and make the sequence $Z_1, Z_2, \ldots$ a martingale. Hence the following inequalities for martingales follow from [105].

Lemma 41. *For any fixed $T > 0$ and $\epsilon > 0$, we have*

$$\Pr[\hat{\mu} \geq (1+\epsilon)\mu] \leq e^{\frac{-T\mu\epsilon^2}{2+\frac{2}{3}\epsilon}},$$

and

$$\Pr[\hat{\mu} \leq (1-\epsilon)\mu] \leq e^{\frac{-T\mu\epsilon^2}{2}}.$$

where $\hat{\mu} = \frac{\sum_{i=1}^{T} X_i}{T}$ is an estimate of $\mu = \mathbb{B}(S)/\Gamma$.

3.2.8 Proofs of lemmas and theorems

3.2.8.1 Proof of Lemma 51

We start with the definition of $\mathbb{B}(S)$ in Eq. 3.92 and prove the equivalent formula $\mathbb{B}(S) = \Gamma \Pr_{g \sqsubseteq G, u \in V}[\mathcal{E}_j \cap S \neq \emptyset]$.

$$\begin{aligned}
\mathbb{B}(S) &= \sum_{u \in V} \Pr_{g \sqsubseteq G}[u \in R(g, S)] b(u) \\
&= \sum_{u \in V} \Pr_{g \sqsubseteq G}[\exists v \in S \text{ such that } v \in \mathcal{E}_j(u)] b(u) \\
&= \Gamma \sum_{u \in V} \Pr_{g \sqsubseteq G}[\exists v \in S \text{ such that } v \in \mathcal{E}_j(u)] \frac{b(u)}{\Gamma} \\
&= \Gamma \Pr_{g \sqsubseteq G, u \in V}[\exists v \in S \text{ such that } v \in \mathcal{E}_j] \\
&= \Gamma \Pr_{g \sqsubseteq G, u \in V}[S \cap \mathcal{E}_j \neq \emptyset]. \qquad (3.129)
\end{aligned}$$

The transition from the third to forth equality follows from the distribution of choosing node u as a starting node of BSA. Since we select u with probability $P(u) = b(u)/\Gamma$, the forth equality contains the expected probability taken over the benefit distribution. That completes our proof.

3.2.8.2 Proof of Lemma 37

Consider the first $T^* = \frac{(2+2\epsilon_2/3)\cdot\Gamma\cdot\ln\left(\binom{n}{k}/\delta_2\right)}{\mathsf{OPT}_k\cdot\epsilon_2^2}$ hyperedges, for any set S_k of k nodes, by Chernoff's bound (Lemma 41),

$$\begin{aligned}
&\Pr[\hat{\mathbb{B}}_{T^*}(S_k) \geq \mathbb{B}(S_k) + \epsilon_2\mathsf{OPT}_k] \\
=&\Pr[\hat{\mathbb{B}}_{T^*}(S_k) \geq (1+\epsilon_2\frac{\mathsf{OPT}_k}{\mathbb{B}(S_k)})\mathbb{B}(S_k)] \\
\leq&\exp\left(-\frac{T^*\mathbb{B}(S_k)(\epsilon_2\frac{\mathsf{OPT}_k}{\mathbb{B}(S_k)})^2}{(2+\frac{2}{3}\epsilon_2\frac{\mathsf{OPT}_k}{\mathbb{B}(S_k)})\Gamma}\right) \\
=&\exp\left(-\frac{(2+2\epsilon_2/3)\Gamma\ln(\binom{n}{k}/\delta_2)\mathbb{B}(S_k)\epsilon_2^2\mathsf{OPT}_k^2}{\mathsf{OPT}_k^2\epsilon_2^2\mathbb{B}^2(S_k)(2+\frac{2}{3}\frac{\mathsf{OPT}_k}{\mathbb{B}(S_k)})\Gamma}\right) \\
=&\exp\left(-\frac{(2+2\epsilon_2/3)\ln(\binom{n}{k}/\delta_2)}{2\frac{\mathbb{B}(S_k)}{\mathsf{OPT}_k}+2\epsilon_2/3}\right) \leq \delta_2/\binom{n}{k}.
\end{aligned} \tag{3.130}$$

Moreover,

$$\begin{aligned}
&\Pr[\hat{\mathbb{B}}_{T^*}(S_k) \geq \mathbb{B}(S_k) + \epsilon_2\mathsf{OPT}_k] \\
\geq&\Pr[\hat{\mathbb{B}}_{T^*}(S_k) \geq \mathsf{OPT}_k + \epsilon_2\mathsf{OPT}_k] \\
=&\Pr[deg_{T^*}(S_k)\Gamma/T^* \geq (1+\epsilon_2)\mathsf{OPT}_k] \\
=&\Pr[deg_{T^*}(S_k) \geq (1+\epsilon_2)\mathsf{OPT}_k T^*/\Gamma] \\
=&\Pr[deg_{T^*}(S_k) \geq \Lambda_L]
\end{aligned} \tag{3.131}$$

Combine Eqs. 3.130 and 3.131, we obtain,

$$\Pr[deg_{T^*}(S_k) \geq \Lambda_L] \leq \delta_2/\binom{n}{k} \tag{3.132}$$

Apply union bound over all seed sets S_k of size k, we have,

$$\Pr[\exists S_k, deg_{T^*}(S_k) \geq \Lambda_L] \leq \delta_2 \tag{3.133}$$

which means that with T^* hyperedges, the probability of having a seed set S_k of k nodes

with $deg_{T^*}(S_k) \geq \Lambda_L$ is less than δ_2. INote that BCT stops only when the returned seed set $\hat{S}_k$ has $deg_{\mathcal{H}} \geq \Lambda_L$ that implies having a seed set with the coverage at least Λ_L. Thus, the number of hyperedges generated by BCT is at least T^* with probability of $1 - \delta_2$.

3.2.8.3 Proof of Lemma 39

Since $\mathsf{c} > 1$, with $\mathsf{c}T^*$ hyperedges, we have,

$$\begin{aligned}
&\Pr[\hat{\mathbb{B}}_{\mathsf{c}T^*}(S_k) \leq \mathbb{B}(S_k) - \frac{\epsilon_2}{2}\mathsf{OPT}_k] \\
=&\Pr[\hat{\mathbb{B}}_{\mathsf{c}T^*}(S_k) \leq (1 - \frac{\epsilon_2}{2}\frac{\mathsf{OPT}_k}{\mathbb{B}(S_k)})\mathbb{B}(S_k)] \\
\leq&\exp\left(-\frac{\mathsf{c}T^*\mathbb{B}(S_k)(\frac{\epsilon_2}{2}\frac{\mathsf{OPT}_k}{\mathbb{B}(S_k)})^2}{2\Gamma}\right) \\
=&\exp\left(-\frac{\mathsf{c}(2+2\epsilon_2/3)\Gamma \cdot \ln\left(\binom{n}{k}/\delta_2\right)\mathbb{B}(S_k)\epsilon_2^2\mathsf{OPT}_k^2}{2\Gamma\mathbb{B}^2(S_k)\mathsf{OPT}_k\epsilon_2^2 4}\right) \\
=&\exp\left(-\frac{\mathsf{c}(2+2\epsilon_2/3)\ln(\binom{n}{k}/\delta_2)\mathsf{OPT}_k}{2\mathbb{B}(S_k)4}\right) \leq \left(\delta_2/\binom{n}{k}\right)^{\mathsf{c}/4}
\end{aligned}$$

Thus, since there are at most $\binom{n}{k}$ such sets of size k,

$$\Pr[\exists S_k, \hat{\mathbb{B}}_{\mathsf{c}T^*}(S_k) \leq \mathbb{B}(S_k) - \frac{\epsilon_2}{2}\mathsf{OPT}_k] \leq \delta_2^{\mathsf{c}/4} \tag{3.134}$$

Moreover, since $\mathsf{c} \geq 8$, thus, $\mathsf{c}T^* > 8T^*$, with $\mathsf{c}T^*$ hyperedges, applying Corollary 3 with parameter settings $\epsilon = \epsilon/2$ and $\delta = (2\delta_2)^{\mathsf{c}/4}$,

$$\Pr[\hat{\mathbb{B}}_{\mathsf{c}T^*}(\hat{S}_k) \leq (1 - 1/e - \epsilon/2)\mathsf{OPT}_k] \leq (2\delta_2)^{\mathsf{c}/4} \tag{3.135}$$

From Eqs. 3.134 and 3.135, consider two events:

(e1) $\forall S_k, \hat{\mathbb{B}}_{\mathsf{c}T^*}(S_k) \geq \mathbb{B}(S_k) - \frac{\epsilon_2}{2}\mathsf{OPT}_k$

(e2) $\hat{\mathbb{B}}_{\mathsf{c}T^*}(\hat{S}_k) \geq (1 - 1/e - \epsilon/2)\mathsf{OPT}_k$

which together happen with probability of at least $1-\delta_2^{\mathsf{c}/4}-(2\delta_2)^{\mathsf{c}/4} \geq 1-\frac{1}{6}\delta_2-\frac{4}{6}\delta_2 = 1-\delta_2$

since $\mathsf{c} > 1$ and $\delta_2 = \delta/6 \leq 1/6$. In that case, we further derive,

$$\begin{aligned}
&\hat{\mathbb{B}}_{\mathsf{c}T^*}(\hat{S}_k) \geq \mathbb{B}(\hat{S}_k) - \frac{\epsilon_2}{2}\mathsf{OPT}_k \\
\Leftrightarrow\ &\mathbb{B}_{\mathsf{c}T^*}(\hat{S}_k) \geq (1 - 1/e - \frac{\epsilon}{2})\mathsf{OPT}_k - \frac{\epsilon_2}{2}\mathsf{OPT}_k \\
\Leftrightarrow\ &deg_{\mathsf{c}T^*}(\hat{S}_k)\frac{\Gamma}{\mathsf{c}T^*} \geq (1 - 1/e - \frac{\epsilon}{2} - \frac{\epsilon_2}{2})\mathsf{OPT}_k \\
\Leftrightarrow\ &deg_{\mathsf{c}T^*}(\hat{S}_k) \geq \frac{(1 - 1/e - \frac{\epsilon}{2} - \frac{\epsilon_2}{2})\mathsf{OPT}_k \mathsf{c}T^*}{\Gamma} \\
\Leftrightarrow\ &deg_{\mathsf{c}T^*}(\hat{S}_k) \geq \frac{(1 - 1/e - \frac{\epsilon}{2} - \frac{\epsilon_2}{2})}{(1 + \epsilon_2)}\mathsf{c}\Lambda_L
\end{aligned} \tag{3.136}$$

Now, since we set $\mathsf{c} = 4\left\lceil \frac{1+\epsilon_2}{1-1/e-\epsilon-\epsilon_2} \right\rceil > 4$, then,

$$deg_{\mathsf{c}T^*}(\hat{S}_k) > 4\Lambda_L \tag{3.137}$$

with probability $1 - \delta_2$. Note that c exists due to the condition on ϵ that $\epsilon < (1 - 1/e)$ and $\epsilon_2 < \epsilon$.

In other words, with a probability of at least $1 - \delta_2$, with $\mathsf{c}T^*$ hyperedges where $\mathsf{c} = 4\left\lceil \frac{1+\epsilon_2}{(1-1/e-\frac{\epsilon}{2}-\frac{\epsilon_2}{2})} \right\rceil$, the stopping condition in BCT will be satisfied. Thus, BCT generates at most $\mathsf{c}T^*$ hyperedges with probability at least $1 - \delta_2$, or

$$\Pr[m_{\mathcal{H}} \leq \mathsf{c}T^*] \geq 1 - \delta_2. \tag{3.138}$$

3.3 Social Influence Spectrum with Guarantees

3.3.1 Problem Definition

Definition 15 (Influence Spectrum (IS)). *Given two integers k_{lower} and k_{upper}, for all $k = k_{lower}, \ldots, k_{upper}$, find S^k of size k that maximizes $\mathbb{I}(S^k)$.*

When the context is clear, we also use IS to indicate the influence values $(\mathbb{I}(k_{lower}), \ldots, \mathbb{I}(k_{upper}))$.

Complexity and Hardness. For each value of $k \in \{k_{lower}, \ldots, k_{upper}\}$, we have an

IM problem that finds the k-size seed set S^k of maximum influence. Thus, IS is at least as hard as IM. Since IM is an NP-hard problem, it follows that IS is also an NP-hard problem. More than that, we cannot approximate IM with a factor $(1 - 1/e + \epsilon)$ unless $NP \subseteq DTIME(n^{\log \log n})$ [55], we also obtain the same result for IS problem.

Greedy algorithm for IM. The Greedy approach in [55], referred to as the Greedy, starts with an empty seed set $S = \emptyset$, and iteratively adds to S a node u that leads to the largest increase in the objective, i.e.,

$$u = \arg\max_{v \notin S} (\mathbb{I}(S \cup \{v\}) - \mathbb{I}(S))$$

To estimate $\mathbb{I}(S)$, we first generate a sample graph G of $\mathcal{G}$ using the live-edge model: select for each node $v \in \mathcal{G}$ at most one of its incoming edges at random, such that the edge (u, v) is selected with probability $w(u, v)$, and no edge is selected with probability $1 - \sum_u w(u, v)$. We then measure the number of nodes reachable from S in G, say $R_G(S)$. After generating "enough" sample graphs G (typically $n_s = 10,000$ samples [55]), we can take the average of $R_G(S)$ as an estimation of $\mathbb{I}(S)$.

To a select a node u, we may have to perform up to n estimations of $\mathbb{I}(.)$ that require generating n_s samples each. Thus, Greedy with its $O(k \times n_s \times mn)$ time complexity is computationally prohibitive for networks with millions of nodes. Later the heuristics CELF and CELF++ [46] are proposed to scale up the computation. Nevertheless, the greedy approach do not scale well for large networks.

3.3.1.1 IS through Greedy approach.

Let $\{u_1, u_2, \ldots, u_n\}$ be the order that nodes are added to S. Knowing this order will give an approximate solution $\{S^{k_{lower}}, ..., S^{k_{upper}}\}$ for IS in which $S^k = \{u_1, u_2, u_3, ..., u_k\}$ the k-prefix of the order. However, this approach has two main drawbacks

1. No error bounds are given on the influence of the seed sets $\mathbb{I}(S^{k_{lower}}), ..., \mathbb{I}(S^{k_{upper}})$

(with respect to the number of samples n_S).

2. The approach is not scalable for large networks due to its high time complexity.

We will address these drawbacks in the following sections.

Table 22.: Table of Notations

Notation	Description
k_{lower}, k_{upper}	lower and upper numbers of selected seed nodes
OPT^k	The maximum $\mathbb{I}(S^k)$ for any size-k node set S^k
S^{k*}	An optimal size-k seed set, i.e., $\mathbb{I}(S^{k*}) = OPT^k$
c	Sampling constant $c = 2(e-2) \approx \sqrt{2}$
$M_k, \mathcal{M}$	$M_k = \binom{n}{k} + 2, \mathcal{M} = \sum_{k=k_{lower}}^{k_{upper}} M_k$
Υ	$\Upsilon = 8c(1-\frac{1}{2e})^2(\log\frac{1}{\delta} + \log\mathcal{M})\frac{1}{\epsilon^2}$ (Note that $\log\mathcal{M} < k_{upper}\log\frac{n(k_{upper}-k_{lower}+1)}{k_{upper}}$)
Λ	$\Lambda = (1+\frac{e\epsilon}{2e-1})\Upsilon$

3.3.2 Simultaneous High-confident Estimation of Influence Spectrum

In this section, we investigate the problem of obtaining high-confident and *bounded-error* estimation of influence of *multiple* seed sets simultaneously. This is critical for knowing whether or not we have sufficient samples to provide the guarantees on the solution of IS. Given a node order $S = \{v_1, \ldots, v_n\}$, e.g., like the one obtained through the greedy approach, we wish to compute the influence of all k-prefixes with $k_{lower} \leq k \leq k_{upper}$, i.e., to calculate IS $\mathbb{I}(S^k), \forall k = k_{lower}, \ldots, k_{upper}$ where $S^k = \{v_1, \ldots, v_k\}$. Computing exact IS is intractable as computing the influence of a single seed set is already #P-hard [55].

Even approximating these influence values with an ϵ-error is difficult with existing methods. Previous works [55, 19] have to resort to estimation by simulating the influence cascades from the selected seeds many times and take the average of those simulated influence. This approach has the complexity of $O((m+n)R)$ where R is the number of simula-

tions. However, there are no rigorous methods to determine R and R is chosen using ad hoc method of experiment-based tuning. Recent methods [11, 106, 105] use Reverse Influence Sampling (RIS) to give high-confidence estimations for the influence of a single seed set. However, these methods are still not scalable for large ranges of $k \in [k_{lower}, k_{upper}]$.

To this end, we propose an efficient algorithm to give high-confidence estimation for multiple seed sets at scale. In fact, the time needed to estimate for multiple seed sets is the same with the time to evaluate the influence for a single seed set.

Our algorithm is built on top of the reverse influence sampling technique [11]. The RIS procedure to generate a random RR set $R_j \subseteq V$ in LT model is summarized in Algorithm 17. After choosing a starting node u randomly, we attempt to select an *in-neighbor* v of u, i.e. (v, u) is an edge of $\mathcal{G}$, according to the edge weights. Then we "move" to v and repeat, i.e. to continue the process with v replaced by u. The procedure stops when we encounter a previously visited vertex or no edge is selected. The RR set is then returned as the set of nodes visited along the process.

Algorithm 17: RIS-LT: Reverse Influence Sampling in LT model

Input: Weighted graph $\mathcal{G} = (V, E, w)$
Output: A random RR set $R_j \subseteq V$
$R_j \leftarrow \emptyset$
Pick a node v uniformly at random.
repeat
  Add v to R_j
  Attempt to select an edge (u, v) using live-edge model
  if *edge* (u, v) *is selected* **then**
    Set $v \leftarrow u$
  end
until *($v \in R_j$) OR (no edge is selected)*;
Return R_j

The key insight into why random RR sets generated via RIS can capture the influence landscape is stated in the following lemma.

Lemma 42. *Given a fixed seed set $S \subseteq V$, for a random RR set R_j,*

$$\Pr[R_j \cap S \neq \emptyset] = \frac{\mathbb{I}(S)}{n}$$

The proof is similar to that for IC model in [11] and is omitted.

Thus we can apply Monte-Carlo method to estimate the influence of a given seed set S, i.e., to generate enough RR sets (aka samples) and compute the frequency that the RR sets intersect with S. Even better, we only need to generate the RR sets once, and can reuse the RR sets to approximate the influence of as many seed sets as we want. This is a huge advantage comparing to the traditional Greedy [55], in which we have to perform an excessive number of BFS to estimate nodes' influence. All we need to figure out is the number of sample times (i.e. number of RR sets) needed to estimate nodes' influence at a desired level of accuracy.

3.3.2.1 Number of Samples (RR sets)

This section focuses on the number of samples (RR sets) needed to achieve a pre-determined performance guarantee. As the number of samples directly decides the running time, it is critical to minimize the number of samples (preserving the same performance guarantees). For example, Borgs et al.'s method requires at least $48\frac{(m+n)\log n}{\epsilon^3 OPT^k}$ RR sets to find a $(1-1/e-\epsilon)$-approximate of IM with probability at least $1-1/n^l$, while Tang et al.'s [106] needs only $(8+\epsilon)\frac{k(m+n)}{\epsilon^2 OPT^k}$ RR sets to provide the same guarantees. Here $OPT^k = \max_{|S|=k, S\subseteq V}\{\mathbb{I}(S)\}$, the maximum influence of any size-k seed set. Hence, the Tang et al.'s is asymptotically $\frac{1}{\epsilon}\log n$ times faster than the Borgs et al.'s.

Let Z be a random variable distributed in $[0,1]$ with mean $\mathbb{E}[Z] = \mu$ and variance σ_Z^2. Let $Z_1, Z_2, \ldots, Z_T$ be independently and identically distributed (i.i.d.) realizations of Z. A *Monte Carlo estimator* of μ_Z is,

$$\tilde{\mu} = \frac{1}{T}\sum_{i=1}^{T} Z_i. \tag{3.139}$$

$\tilde{\mu}$ is said to be an (ϵ, δ)-approximation of μ, for $0 < \epsilon, \delta \leq 1$, if

$$\Pr[|\tilde{\mu} - \mu| \leq \epsilon\mu] \geq 1 - \delta. \tag{3.140}$$

Let $\rho(\epsilon) = \max\{\sigma^2, \epsilon\mu\}$. The Generalized Zero-One Estimator Theorem in [26] proves that if

$$T = 2c \ln \frac{2}{\delta} \frac{\rho(\epsilon)}{\epsilon^2 \mu^2} \tag{3.141}$$

then $\tilde{\mu} = \frac{1}{T}\sum_{i=1}^{T} Z_i$ is an (ϵ, δ)-approximation of μ. Moreover, the number of sampling time is (asymptotically) optimal (by a constant factor) [26]. Additionally from [26], the second way of achieving an (ϵ, δ)-approximation of μ is based on the condition that,

$$\sum_{i=1}^{T} Z_i \geq 1 + (1+\epsilon)2c \ln(\frac{2}{\delta})\frac{1}{\epsilon^2}, \tag{3.142}$$

and the necessary number of samples to achieve this condition is also asymptotically optimal.

In this paper, we are interested in the random variable Z with realizations

$$Z_j = \min\{|S \cap R_j|, 1\}, \tag{3.143}$$

where S is a fixed seed set and R_j is a random RR set generated by Algorithm 17. From Lemma 51, Z is a random variable with mean $\mu_Z = \mathbb{I}(S)/n$ and variance $\sigma_Z^2 = (1-\mu_Z)\mu_Z$.

A major obstacle in using Eq. (3.141) to derive the optimal number of samples is that we do not know σ_Z^2 and μ_Z, the quantity we are trying to estimate. Let $S^{k*} = \arg\max_{|S|=k, S\subseteq V}\{\mathbb{I}(S)\}$, and $OPT^k = \mathbb{I}(S^{k*})$. If we can come up with a close bound

on OPT^k, we will know the necessary number of RR sets to capture the influence landscape. After that, IM and IS can be reduced to the classic Max-Coverage problem [107] as shown in [11, 106].

Thus, the key to the efficiency of the two previous studies in [11, 106] are the methods to probe and estimate the value of OPT^k. With the better probing and estimating techniques, TIM and TIM+ in [11] reduce the time-complexity in [106] by a factor $O(1/\epsilon \log n)$, making the first scalable method for IM in billion-size networks. However, the number of samples in [106] is still far from optimal, especially for large seed sets. As a consequence, the two algorithms scale poorly with large number of seeds.

3.3.2.2 Efficient Influence Spectrum Estimation

Algorithm 18: Efficient Influence Spectrum Validation Algorithm (EIVA)

Input: Weighted graph $\mathcal{G}$, a seed set $S = \{v_1, v_2, \ldots, v_n\}$, $k_{lower}, \leq k_{upper}$ that $1 \leq k_{lower} \leq k_{upper} \leq n$ and $\epsilon, \delta \in (0, 1)$
Output: (ϵ, δ)-approximation of $\mathbb{I}(\hat{S}^k), \forall k = k_{lower}, \ldots, k_{upper}$
$\Lambda_L \leftarrow 1 + 2c(1+\epsilon)(\ln \frac{2}{\delta} + \ln(k_{upper} - k_{lower} + 1))\frac{1}{\epsilon^2}$
$T \leftarrow 0, \mathrm{Cov}_k \leftarrow 0, \forall k = 1, \ldots, n$
repeat
  Generate random RR set $R_j \leftarrow RIS - LT(\mathcal{G})$
  $t = \arg\min_i \{v_i \in R_j\}$
  $\mathrm{Cov}_t \leftarrow \mathrm{Cov}_t + 1$
  $T \leftarrow T + 1$
until $\sum_{i=1}^{k_{lower}} \mathrm{Cov}_i \geq \Lambda_L$;
$\hat{\mathbb{I}}_{k_{lower}} = \mathrm{Cov}_{k_{lower}} \cdot n/T$
for $k = (k_{lower} + 1) : k_{upper}$ **do**
  $\hat{\mathbb{I}}_k \leftarrow \hat{\mathbb{I}}_{k-1} + \mathrm{Cov}_k \cdot n/T$
end
Return $\hat{I} = \{\hat{\mathbb{I}}_k | k = k_{lower}, \ldots, k_{upper}\}$

In this section, we provide a fast and memory-efficient algorithm, called EIVA, to estimate the influence spectrum in arbitrarily good accuracy with high probability. Given an ordered set $S = \{v_1, \ldots, v_n\}$ and two integers $1 \leq k_{lower} \leq k_{upper} \leq n$, we want to compute all $\mathbb{I}(S^k), \forall k = k_{lower}, \ldots, k_{upper}$ where $S^k = \{v_1, v_2, \ldots, v_k\}$. Here we assume

$S = \{v_1, \ldots, v_n\}$ is fixed and given as a set of seed nodes which can be the output of an IS algorithm. Denote $\hat{\mathbb{I}}(S^k)$ the estimated value of $\mathbb{I}(S^k)$ returned by EIVA algorithm. EIVA guarantees that $\hat{\mathbb{I}}(S^k), \forall k = k_{lower}, \ldots, k_{upper}$ are within $(1-\epsilon)$ the actual values with probability at least $1-\delta$ where $\epsilon, \delta \in (0,1)$ are arbitrarily small values chosen by the users.

Specifically, EIVA, shown in Algorithm 18, repeatedly generates a RR set R_j in each step. It then looks for the smallest index t that $v_t \in R_j$. Observe that all seed sets $S^k, k \geq t$ will cover RR set R_j. Instead of increasing the value of all $\mathrm{Cov}_k, k \geq t$, EIVA only increases Cov_k by one. Finally, the values of Cov_k will be aggregated at the end, lines 10 and 11. This smart update strategy reduces the worst-case time-complexity per RR set from $O(n)$ to $O(1)$. Hence, we will be able to compute all the influence of the seed sets much faster.

Lemma 43. *EIVA (Algorithm 18) computes (ϵ, δ)-approximate for the influences of all seed sets in time $O(\epsilon^{-2}(\ln \frac{2}{\delta} + \ln(k_{upper} - k_{lower} + 1))(m+n))$ and **only an** $\theta(\mathbf{n})$ **additional space** (excluding the space to store the graph).*

Proof. The complexity analysis is similar to that of LISA algorithm which will be presented in Subsection 3.3.3.2. The space complexity is followed directly from the fact that EIVA does not store RR sets but only need a single array to store the values of $\mathrm{Cov}_k, k = 1, \ldots, n$, thus its space-complexity is $\theta(n)$. Here we prove the $(1-\epsilon)$-approximation factor. First, due to the condition in line 8, we have,

$$\sum_{i=1}^{k_{upper}} \mathrm{Cov}_i \geq \sum_{i=1}^{k_{upper}-1} \mathrm{Cov}_i \geq \cdots \geq \sum_{i=1}^{k_{lower}} \mathrm{Cov}_i \geq \Lambda_L \tag{3.144}$$

Thus, for any $k = k_{lower}, \ldots, k_{upper}$, based also on the condition of achieving an (ϵ, δ)-

approximation of $\mathbb{I}(S^k)$ for a single set S^k in Eq. 3.142, we obtain,

$$\Pr[|\hat{\mathbb{I}}(S^k) - \mathbb{I}(S^k)| \leq \epsilon \mathbb{I}(S^k)] \leq 1 - \frac{\delta}{k_{upper} - k_{lower} + 1} \tag{3.145}$$

Taking the union bound of the above inequality over all values of k from k_{lower} to k_{upper} (there are $k_{upper} - k_{lower} + 1$ such values) gives,

$$\Pr[|\hat{\mathbb{I}}(S^k) - \mathbb{I}(S^k)| \leq \epsilon \mathbb{I}(S^k), \forall k = k_{lower}, \ldots, k_{upper}] \leq 1 - \delta \tag{3.146}$$

which proves the (ϵ, δ)-approximate for the influences of all seed sets S_k where $k = k_{lower}, \ldots, k_{upper}$ and completes the proof. □

Comparing with the complexity of naive algorithm for computing IS by cascade simulation, we see that EIVA saves a factor of $R\frac{k_{upper}-k_{lower}+1}{\ln(k_{upper}-k_{lower}+1)}$. More importantly, EIVA guarantees an (ϵ, δ)-approximation for the returned IS estimation.

3.3.3 LISA Approximation Algorithm for Identify Multiple Seed Sets

In this section, we propose LISA approximation algorithm that returns a $(1-1/e-\epsilon)$-approximate IS with probability at least $(1 - \delta)$ for any constant $\epsilon, \delta \in (0, 1)$.

Our algorithm, named LISA, is presented in Algorithm 19. It consists of a main loop of iterations. In each iteration, LISA 1) doubles the number of RR sets (except for the first iteration where it generates $N_t = \Lambda$ RR sets) using RIS (Algorithm 17) and 2) solves an instance of Max-Coverage using a greedy approach (Algorithm 20). The algorithm terminates when the stopping condition in line 9 is satisfied.

Borgs et al. [11] generates RR sets until a pre-defined number of edges explored by the algorithm and only provide a low successful probability $2/3$. While the authors suggest that their algorithm can be repeated multiple times to boost up the success probability, this approach leads to a very inefficient implementation. Tang et al. [106] estimates OPT^k via

Algorithm 19: LISA Influence Spectrum Algorithm

Input: Precision $\epsilon \in (0,1)$, $\delta \in (0,1)$, weighted graph $\mathcal{G}$ and min/max seed set sizes k_{lower}, k_{upper}

Output: $\{\hat{S}^k, k = k_{lower}, \ldots, k_{upper}\}$ and their influences $\{\hat{\mathbb{I}}(\hat{S}^k), k = k_{lower}, \ldots, k_{upper}\}$.

$\Lambda = (1 + \frac{e\epsilon}{2e-1})\Upsilon$

$N_t = \Lambda$

$\mathcal{R} \leftarrow \emptyset$

repeat

  Generate N_t RR sets by RIS-LT and add to $\mathcal{R}$

  $N_t = m_{\mathcal{R}}$

  $\hat{S} = \text{Max-Coverage}(\mathcal{R}, k_{upper})$

  $\hat{S}^k = \{\hat{S}_t | t = 1, \ldots, k\}, \forall k = k_{lower}, \ldots, k_{upper}$

until $\text{Cov}_{\mathcal{R}}(\hat{S}^{k_{lower}}) \geq \Lambda$;

Compute $\hat{I}(\hat{S}^k) = \frac{\text{Cov}_{\mathcal{R}}(\hat{S}^k) \cdot n}{m_{\mathcal{R}}}, \forall k = k_{lower}, \ldots, k_{upper}$

Return $\{\hat{S}^k, k = k_{lower}, \ldots, k_{upper}\}$ and $\{\hat{\mathbb{I}}(\hat{S}^k), k = k_{lower}, \ldots, k_{upper}\}$

Algorithm 20: Max-Coverage

Input: Collection $\mathcal{R}$ and maximum number of seeds k_{upper}.

Output: Seed set $\hat{S}$

$S = \emptyset$

for $i = 1 : k_{upper}$ **do**

  $\hat{v} \leftarrow \arg\max_{v \in V}(\text{Cov}_{\mathcal{R}}(\hat{S} \cup \{v\}) - \text{Cov}_{\mathcal{R}}(\hat{S}))$

  Add $\hat{v}$ to $\hat{S}$

end

Return $\hat{S}$

the average cost of RIS, called EPT^k. However, their approach still requires generating as many as k times more RR sets than necessary. Differently, we propose a novel stopping rule: *we stop generating RR sets once the degrees of all seed sets in the collection $\mathcal{R}$ reach their corresponding constants Λ.* Here, the degree of a node in the collection $\mathcal{R}$ is the number of RR sets that contain the node. Later we show our stopping rule guarantees a 'rich' enough collection $\mathcal{R}$ to estimate nodes' influence and small enough RR sets to make an efficient algorithm.

Our algorithm is easy to implement and requires no parameters rather than ϵ and δ. In practice, it scales very well with billion-size networks and large seed sets. It proves to be the fastest algorithm known for IM while maintaining superior solution quality at the same time.

3.3.3.1 Approximation Guarantees

In this subsection, we will prove the approximation factor of LISA to be $(1 - \frac{1}{e} - \epsilon)$. In our context of IS, the $(1 - \frac{1}{e} - \epsilon)$-approximation guarantee means that for all sizes $k = k_{lower}, \ldots, k_{upper}$, $\mathbb{I}(\hat{S}^k) \geq (1 - \frac{1}{e} - \epsilon)\mathbb{I}(S^{k*})$ where $\hat{S}^k = \{\hat{S}_t | t = 1, \ldots, k\}$ and S^{k*} is an optimal seed set of size k with the optimal influence of OPT^k. We say that an IS algorithm returns an $(1 - \frac{1}{e} - \epsilon)$-approximate solution $\hat{S}$ with probability at least $(1 - \delta)$ if,

$$\Pr[\mathbb{I}(\hat{S}^k) \geq (1 - \frac{1}{e} - \epsilon)\mathbb{I}(S^{k*}), \forall k = k_{lower}, \ldots, k_{upper}] \geq 1 - \delta \tag{3.147}$$

In other words, it is equivalent to that,

$$\Pr[\mathbb{I}(\hat{S}^k) < (1 - \frac{1}{e} - \epsilon)\mathbb{I}(S^{k*}), \exists k = k_{lower}, \ldots, k_{upper}] < \delta \tag{3.148}$$

The following will prove that LISA returns a $(1 - 1/e - \epsilon)$-approximate solution $\hat{S}$ with probability at least $(1 - \delta)$ where ϵ and δ are parameters in Alg. 19 (Theorem 47).

Roadmap. To prove the Eq. 3.148, we will intermediately prove a stronger inequality,

$$\sum_{k=k_{lower}}^{k_{upper}} \Pr[\mathbb{I}(\hat{S}^k) < (1 - \frac{1}{e} - \epsilon)\mathbb{I}(S^{k*})] < \delta \tag{3.149}$$

which implies Eq. 3.148 due to the inequality between probability of a union of events and sum of probabilities of individual events. More specifically, we will show that LISA's stopping condition (Line 9 of Alg. 19) guarantees the kth term in the sum to be bounded by $\delta\frac{M_k}{\mathcal{M}}$ where $M_k = \binom{n}{k} + 2, \mathcal{M} = \sum_{k=k_{lower}}^{k_{upper}} M_k$ and thus, the Eq. 3.149 follows.

In order to prove the approximation guarantee of each seed set $\hat{S}^k$, we show that LISA generates at least $T^k = \frac{n\Upsilon}{OPT^k}$ RR sets with a high probability in Lemma 45, i.e., our stopping condition. Thereafter, we prove that T^k RR sets are sufficient to guarantee that LISA returns an $(1-1/e-\epsilon)$-approximate solution of the seed size k with a high probability (Lemma 46). Combining these results gives us the approximation guarantee of LISA for a seed set $\hat{S}^k$ in Theorem 47. Thus, the IS guarantee of LISA follows in Theorem 18.

We first present the following results that will be used in our proofs. For a node set S and a RR set R_j, recall the random sample Z_j defined in Eq. 3.143,

$$Z_j = \min\{|S \cap R_j|, 1\}, \tag{3.150}$$

Thus, the series of RR sets in $\mathcal{H}$ corresponds to a sequence of samples of Z_j, denoted by $\{X_1, X_1, \dots\}$. Intuitively, since the RR sets are generated independently, the resulted sample sequence of Z_j should also be independent and identically distributed in $[0, 1]$. However, similar to the Stopping Rule Algorithm in [26] that LISA creates a dependency on the samples by stopping the algorithm when some condition is satisfied. LISA jumps to the next round when $\mathrm{Cov}_{\mathcal{H}}(\hat{S}^{k_{lower}}) \geq \Lambda$ or, $\sum_{i=1}^{|\mathcal{H}|} Z_i \geq \Lambda$ where Z_j corresponds to $\hat{S}^{k_{lower}}$, is not met and hence, whether we generate more samples depending on the current set of RR sets. Interestingly, similar to the case of Stopping Rule Algorithm in [26], the sequence

$\{Z_1, Z_2, \dots\}$ forms a *martingle* and the following results follow from [26]:

Let $Z_1, Z_2, ...$ samples according to Z random variable in the interval $[0, 1]$ with mean μ_Z and variance σ_Z^2 form a martingale and $\hat{\mu}_Z = \frac{1}{T}\sum_{i=1}^{T} Z_i$ be an estimate of μ_Z, for any fixed $T > 0, 0 \geq \epsilon \geq 1$,

$$\Pr[\hat{\mu}_Z \geq (1+\epsilon)\mu_Z] \leq e^{\frac{-T\mu_Z\epsilon^2}{2c}} \tag{3.151}$$

and,

$$\Pr[\hat{\mu}_Z \leq (1-\epsilon)\mu_Z] \leq e^{\frac{-T\mu_Z\epsilon^2}{2c}}. \tag{3.152}$$

To prove the bound on the number of RR sets generated by LISA (Lemma 45), we first need to show the following result which relates the number of RR sets to the approximation quality of any set of a specific size.

Lemma 44. *Given a size-i set S^k, if the collection $\mathcal{R}$ has at least $T^k = \frac{n\Upsilon}{OPT^k}$ RR sets, then*

$$\Pr[\mathbb{I}(S^k) \leq \hat{\mathbb{I}}(S^k) - \frac{\epsilon e}{2e-1} OPT^k] \leq \frac{\delta}{\mathcal{M}} \tag{3.153}$$

where $\mathcal{M} = \sum_{k=k_{lower}}^{k_{upper}} M_k$ and $M_k = \binom{n}{k} + 2$.

Proof. Denote $\hat{\mu}_k = \frac{\hat{\mathbb{I}}(S^k)}{n} = \frac{\mathrm{Cov}_{\mathcal{R}}(S^k)}{m_{\mathcal{R}}}$ which is an estimation of $\mu_k = \frac{\mathbb{I}(S^k)}{n}$ and $\mu_k^* = \frac{OPT^k}{n}$. The inequality Eq. 3.153 is equivalent to,

$$\begin{aligned} &\Pr[\mu_k \leq \hat{\mu}_k - \frac{\epsilon e}{2e-1}\mu_k^*] \leq \frac{\delta}{\mathcal{M}} \\ \Leftrightarrow &\Pr[\hat{\mu}_k \geq \mu_k + \frac{\epsilon e}{2e-1}\frac{\mu_k^*}{\mu_k}\mu_k] \leq \frac{\delta}{\mathcal{M}} \end{aligned} \tag{3.154}$$

Applying Eq. 4.25 to the left-hand side of the above inequality gives,

$$\Pr[\hat{\mu}_k \geq \mu_k + \frac{\epsilon e}{2e-1}\frac{\mu_k^*}{\mu_k}\mu_k] \leq e^{-\frac{m_{\mathcal{R}}\mu_k e^2\epsilon^2\mu_k^{*2}}{2c(2e-1)^2\mu_k^2}} \leq e^{-\frac{T^k\mu_k\epsilon^2\mu_k^{*2}}{8c(1-1/2e)^2\mu_k^2}} \tag{3.155}$$

The last inequality is due to the condition of having at least T^k samples generated by LISA. Thus, by replacing T^k with the corresponding definition and noticing that $\frac{\mu_k^*}{\mu_k} \geq 1$,

$$\Pr[\hat{\mu}_k \geq \mu_k + \frac{\epsilon e}{2e-1}\frac{\mu_k^*}{\mu_k}\mu_k] \leq e^{-\frac{T^k\mu_k\epsilon^2\mu_k^{*2}}{8c(1-1/2e)^2\mu_k^2}} \leq \frac{\delta}{\mathcal{M}} \tag{3.156}$$

which completes the proof of Lemma 44. □

Based on Lemma 51, we prove the following bounds on the number of RR sets generated by our LISA algorithm regarding a specific class of seed sets with size k.

Lemma 45 (Stopping condition)**.** *For each $k = k_{lower}, \ldots, k_{upper}$, if there exists a set S^k with $|S^k| = k$ such that $\mathrm{Cov}_{\mathcal{R}}(S^k) \geq \Lambda$, then,*

$$\Pr[m_{\mathcal{R}} \leq T^k] < \frac{\delta}{\mathcal{M}} \tag{3.157}$$

Proof. Let define $X_{S^k} = \min\{|S^k \cap \mathcal{E}_j|, 1\}$ to be a random variable corresponding to set S^k, then,

$$\begin{aligned}\Pr[m_{\mathcal{R}} \leq T^k] &= \Pr\Big[\sum_{j=1}^{m_{\mathcal{R}}} X_{S^k} \leq \sum_{j=1}^{T^k} X_{S^k}\Big] \\ &= \Pr\Big[\mathrm{Cov}_{m_{\mathcal{R}}}(S^k) \leq \mathrm{Cov}_{T^k}(S^k)\Big]\end{aligned}$$

Due to our condition that $\mathrm{Cov}_{m_{\mathcal{R}}}(S^k) \geq \Lambda = (1 + \frac{\epsilon\epsilon}{2e-1})\Upsilon$,

$$\begin{aligned}
\Pr[m_{\mathcal{R}} \leq T^k] &\leq \Pr\left[(1 + \frac{\epsilon e}{2e-1})\Upsilon \leq \mathrm{Cov}_{T^k}(S^k)\right] \\
&\leq \Pr\left[(1 + \frac{\epsilon e}{2e-1})\Upsilon \frac{n}{T^k} \leq \mathrm{Cov}_{T^k}(S^k)\frac{n}{T^k}\right] \\
&\leq \Pr\left[(1 + \frac{\epsilon e}{2e-1})OPT^k \leq \hat{\mathbb{I}}_{T^k}(S^k)\right] \\
&\leq \Pr\left[\mathbb{I}(S^k) + \frac{\epsilon e}{2e-1}OPT^k \leq \hat{\mathbb{I}}_{T^k}(S^k)\right] \\
&\leq \frac{\delta}{\mathcal{M}}
\end{aligned} \tag{3.158}$$

The third line is due to the definition of T^k that $T^k = \frac{n\Upsilon}{OPT^k}$ and the estimation $\hat{\mathbb{I}}(S^k) = \mathrm{Cov}_{T^k}(S^k)\frac{n}{T^k}$ when there are T^k RR sets. The last inequality is followed from Eq. 3.153 of Lemma 44 when having T^k RR sets in the collection $\mathcal{R}$. □

Based on Lemma 45, if we can find a set S^k such that $\mathrm{Cov}_{\mathcal{R}}(S^k) \geq \Lambda$, then with a very high probability, the number of generated RR sets is at least T^k. Next, we show the second component of our proof that T^k RR sets are sufficient to guarantee that the size-k seed set $\hat{S}^k$ returned by LISA is good.

Lemma 46. *For any $k \in \{k_{lower}, \ldots, k_{upper}\}$, if the number of samples (RR sets) $m_{\mathcal{R}} \geq T^k$, LISA returns the corresponding seed set $\hat{S}^k$ with*

$$\Pr[\mathbb{I}(\hat{S}^k) \leq (1 - 1/e - \epsilon)OPT^k] \leq \frac{\delta(M_k - 1)}{\mathcal{M}}. \tag{3.159}$$

Proof. Similarly to the proof of Lemma 45, we obtain the following inequality,

$$\Pr[\hat{\mathbb{I}}(S^{k*}) \leq OPT^k - \frac{\epsilon e}{2e-1}OPT^k] \leq \frac{\delta}{\mathcal{M}} \tag{3.160}$$

Combine Eq. 3.153, Eq. 3.160 and apply union bound over all possible sets of size k on Eq. 3.153 and the corresponding optimal solution S^{k*} in Eq. 3.160, we have,

$$
\begin{aligned}
&\Pr\left[\left(\mathbb{I}(S^k) \leq \hat{\mathbb{I}}(S^k) - \frac{\epsilon e}{2e-1} OPT^k, \forall S^k\right) \text{ and } \left(\hat{\mathbb{I}}(S^{k*}) \leq OPT^k - \frac{\epsilon e}{2e-1} OPT^k\right)\right] \\
&\leq \sum_{S^k} \Pr\left[\mathbb{I}(S^k) \leq \hat{\mathbb{I}}(S^k) - \frac{\epsilon e}{2e-1} OPT^k\right] + \Pr\left[\hat{\mathbb{I}}(S^{k*}) \leq OPT^k - \frac{\epsilon e}{2e-1} OPT^k\right] \\
&\leq \frac{\delta}{\mathcal{M}} \binom{n}{i} + \frac{\delta}{\mathcal{M}} = \frac{\delta(M_k - 1)}{\mathcal{M}}
\end{aligned} \tag{3.161}
$$

Since $\hat{S}^k$ is one of those possible sets S^k, we obtain the following corollary of the above inequality by keeping just the term related to S^k in the first half,

$$
\Pr[\mathbb{I}(\hat{S}^k) \geq \hat{\mathbb{I}}(\hat{S}^k) - \frac{\epsilon e}{2e-1} OPT^k \text{ and } \hat{\mathbb{I}}(S^{k*}) \geq OPT^k - \frac{\epsilon e}{2e-1} OPT^k] \leq \frac{\delta(M_k - 1)}{\mathcal{M}} \tag{3.162}
$$

Since Max-Coverage (Algo. 20) returns $\hat{S}_k$ with,

$$
\mathrm{Cov}_{\mathcal{R}}(\hat{S}^k) \geq (1 - 1/e)\mathrm{Cov}_{\mathcal{R}}(S^k_{max}) \;\geq\; (1 - 1/e)\,\mathrm{Cov}_{\mathcal{R}}(S^{k*}), \tag{3.163}
$$

where S^k_{max} is the optimal size-k solution of Max-Coverage [37].

Assume that $\mathbb{I}(\hat{S}^k) \geq \hat{\mathbb{I}}(\hat{S}^k) - \frac{\epsilon e}{2e-1} OPT^k$, $\hat{\mathbb{I}}(S^{k*}) \geq OPT^k - \frac{\epsilon e}{2e-1} OPT^k$, based on Eq. 3.163, we derive the following,

$$
\begin{aligned}
\mathbb{I}(\hat{S}^k) &\geq \hat{\mathbb{I}}(\hat{S}^k) - \frac{\epsilon e}{2e-1} OPT^k \\
&\geq \frac{\mathrm{Cov}_{\mathcal{R}}(\hat{S}^k)}{m_{\mathcal{R}}} n - \frac{\epsilon e}{2e-1} OPT^k \\
&\geq (1 - 1/e) \frac{\mathrm{Cov}_{\mathcal{R}}(S^{k*})}{m_{\mathcal{R}}} n - \frac{\epsilon e}{2e-1} OPT^k \\
&\geq (1 - 1/e) \hat{\mathbb{I}}(S^{k*}) - \frac{\epsilon e}{2e-1} OPT^k \\
&\geq (1 - 1/e)(1 - \frac{\epsilon e}{2e-1}) OPT^k - \frac{\epsilon e}{2e-1} OPT^k \\
&\geq (1 - 1/e - \epsilon) OPT^k
\end{aligned} \tag{3.164}
$$

Note that to derive Eq. 6.5, we only need two conditions $\mathbb{I}(\hat{S}^k) \geq \hat{\mathbb{I}}(\hat{S}^k) - \frac{\epsilon e}{2e-1}OPT^k$, $\hat{\mathbb{I}}(S^{k*}) \geq OPT^k - \frac{\epsilon e}{2e-1}OPT^k$ which are the subject of the inequality in Eq. 3.162. Thus, from inequalities in Eq. 3.162 and Eq. 6.5, we obtain Lemma 46 that,

$$\Pr[\mathbb{I}(\hat{S}^k) \leq (1 - 1/e - \epsilon)OPT^k] \leq \frac{\delta(M_k - 1)}{\mathcal{M}}, \tag{3.165}$$

and complete our proof. □

Lemmas 45 and 46 together prove that if there exists a set S^k where $|S^k| = k$ such that $\mathrm{Cov}_{\mathcal{R}}(S^k) \geq \Lambda$, then the greedy algorithm for selecting seed set on the collection $\mathcal{R}$ $\mathcal{R}$ will return a $(1 - 1/e - \epsilon)$-approximate solution $\hat{S}^k$. As a result, the following lemma states the approximation guarantee of $\hat{S}^k$.

Lemma 47. *For any $k \in \{k_{lower}, \ldots, k_{upper}\}$, LISA selects a set of k nodes $\hat{S}^k$,*

$$\Pr[\mathbb{I}(\hat{S}^k) \leq (1 - 1/e - \epsilon)OPT^k] \leq \delta\frac{M_k}{\mathcal{M}} \tag{3.166}$$

Proof. Note that LISA algorithm keeps generating RR sets until the degree of each set $\hat{S}^k$ returned by Max-Coverage exceeds Λ. From Lemma 45, we obtain,

$$\Pr[m_{\mathcal{R}} \leq T^k] < \frac{\delta}{\mathcal{M}}, \forall k = k_{lower}, \ldots, k_{upper} \tag{3.167}$$

Assume that $m_{\mathcal{R}} \geq T^k$, from Lemma 46, we also have,

$$\Pr[\mathbb{I}(\hat{S}^k) \leq (1 - 1/e - \epsilon)OPT^k] \leq \frac{\delta(M_k - 1)}{\mathcal{M}}. \tag{3.168}$$

Combine Eqs. 3.167 and 3.168, it is followed that,

$$\begin{aligned}\Pr[\mathbb{I}(\hat{S}^k) \geq (1 - 1/e - \epsilon)OPT^k] &\geq 1 - \Pr[m_{\mathcal{R}} \leq T^k] - \Pr[\mathbb{I}(\hat{S}^k) \leq (1 - 1/e - \epsilon)OPT^k] \\ &= 1 - \frac{\delta}{\mathcal{M}} - \frac{\delta(M_k - 1)}{\mathcal{M}} = 1 - \frac{\delta M_k}{\mathcal{M}}\end{aligned} \tag{3.169}$$

Thus, we complete the proof of Lemma 47. □

Lemma 47 states the approximation guarantee of each set $\hat{S}^k$. The following theorem uses this result with a simple union bound over $(k_{upper} - k_{lower} + 1)$ seed sets $\hat{S}^k, \forall k \in \{k_{lower}, \ldots, k_{upper}\}$ to show the overall approximation factor of LISA algorithm.

Theorem 18. *LISA algorithm returns a sequence of seed sets $\hat{S}^{k_{lower}}, \ldots, \hat{S}^{k_{upper}}$ satisfying*

$$\Pr[\mathbb{I}(\hat{S}^k) \geq (1 - 1/e - \epsilon)OPT^k, \textbf{\textit{for all}}\ k \in \{k_{lower}, \ldots, k_{upper}\}] \geq 1 - \delta \qquad (3.170)$$

Equivalently, LISA has an IS approximation factor of $(1 - 1/e - \epsilon)$.

3.3.3.2 Complexity Analysis

Time complexity. The Max-Coverage procedure of LISA can be implemented in a linear-time in terms of the total size of all the RR sets. As we shall show later in the space complexity section, the expected total size of the RR sets is $O(\Lambda \cdot n)$. Thus Max-Coverage has an expected time complexity $O(\Lambda \cdot n)$.

We shall bound the time-complexity of generating RR sets via the number of edges examined. Keeping track of the maximum degree in the collection $\mathcal{R}$ is relatively easy and can be done with little additional cost.

Lemma 48. *The expected number of edges examined by LISA is at most $\Lambda \cdot m$.*

Proof. The proof consists of two parts 1) bound the expected number of RR sets $m_{\mathcal{R}}$ and 2) estimate the mean number of edges visited per reverse influence sampling.

Number of RR sets: Let $v^* = \arg\max_{v \in V} \mathbb{I}(v)$, the most influential node. Note that v^* is not necessary the same with $\hat{v}_1$, selected by LISA. Define $Y_j = \min\{|\{v^*\} \cap R_j|, 1\}|$, a random variable with mean $\mu_Y = \mathbb{I}(v^*)/n$.

Denote by $T(\Lambda)$ and $T^*(\Lambda)$ the random variables that correspond to the numbers of sampled RR sets until $\mathrm{Cov}_{\mathcal{R}}(\bar{S}^{k_{lower}}) = \Lambda$ and $\mathrm{Cov}_{\mathcal{R}}(v^*) = \Lambda$, respectively. Clearly,

$T(\Lambda) = m_{\mathcal{R}} \leq T^*(\Lambda)$, hence,

$$\mathbb{E}[T_{max}(\Lambda)] \leq \mathbb{E}[T^*(\Lambda)].$$

Using Wald's equation [108], and that $\mathbb{E}[T^*(\Lambda)] < \infty$ we have

$$\mathbb{E}[T^*(\Lambda)]\mu_Y = \Lambda$$

Therefore,

$$\mathbb{E}[m_{\mathcal{R}}] = \mathbb{E}[T(\Lambda)] \leq \mathbb{E}[T^*(\Lambda)] = \frac{\Lambda}{\mu_Y}.$$

Average number of edges visited per reverse influence sampling: The reverse influence sampling procedure picks a source vertex u uniformly at random. Then for each vertex v, it will examine all in-neighbors of v with a probability $\mathbb{I}(v, u)$, the probability that v can reach to u over all sample graphs of $\mathcal{G}$ (aka the probability that v influences u). Thus the mean number of edges examined by the procedure is

$$\begin{aligned}
&\frac{1}{n}\sum_{u\in V}(\sum_{v\in V}\mathbb{I}(v,u)d^-(v)) = \frac{1}{n}\sum_{v\in V}d^-(v)\sum_{u\in V}I(v,u) \\
&= \frac{1}{n}\sum_{v\in V}d^-(v)\mathbb{I}(v) \leq \frac{1}{n}\sum_{v\in V}d^-(v)\mathbb{I}(v^*) = \frac{m}{n}\mathbb{I}(v^*)
\end{aligned} \tag{3.171}$$

Therefore, the expected number of edges examined by LISA is at most

$$\frac{m}{n}\mathbb{I}(v^*)\frac{\Lambda}{\mu_Y} = m\mu_Y\frac{\Lambda}{\mu_Y} = \Lambda m \tag{3.172}$$

This yields the proof. □

Theorem 19. *LISA has expected running time* $O((\log\frac{1}{\delta}+\log\mathcal{M})\frac{1}{\epsilon^2}(m+n))$ *where* $\log\mathcal{M} < k_{upper}\log\frac{n(k_{upper}-k_{lower}+1)}{k_{upper}}$.

Proof. Since Max-Coverage has a time complexity $O(\Lambda n)$ and generating RR sets has an expected runtime $O(\Lambda m)$, it follows that the expected time complexity of LISA is $O(\Lambda(m+$

$n)) = O((\log \frac{1}{\delta} + \log \mathcal{M})\frac{1}{\epsilon^2}(m+n))$.

To evaluate $\log(\mathcal{M})$, we fist have the following,

$$\log(\mathcal{M}) = \log\Big(\sum_{k=k_{lower}}^{k_{upper}} M_k\Big) = \log\Big(\sum_{k=k_{lower}}^{k_{upper}} \binom{n}{k} + 2\Big) \tag{3.173}$$

Since $k_{upper} \leq n/2$,

$$\begin{aligned}
\log\Big[\sum_{k=k_{lower}}^{k_{upper}} \binom{n}{k} + 2\Big] &\leq \log\Big[\sum_{k=k_{lower}}^{k_{upper}} \binom{n}{k_{upper}}\Big] \\
&\leq \log\Big[(k_{upper} - k_{lower} + 1)\binom{n}{k_{upper}}\Big] \\
&\leq \log\Big[(k_{upper} - k_{lower} + 1)\Big(\frac{n}{k_{upper}}\Big)^{k_{upper}}\Big] \\
&\leq k_{upper} \log \frac{n(k_{upper} - k_{lower} + 1)}{k_{upper}}
\end{aligned}$$

□

Space complexity. Besides an $O(m+n)$ space to hold $\mathcal{G}$, we show that on average only an additional $O(n)$ space is sufficient to hold the RR sets. Thus, LISA has an expected linear space complexity $O(m+n)$.

Lemma 49. *The expected additional space to store all the RR sets is* $O((\log \frac{1}{\delta} + \log \mathcal{M})\frac{1}{\epsilon^2}n)$.

Proof. From the proof of Lemma 48, the expected number of RR sets is at most Λ/μ_Y with $\mu_Y = \max_{v \in V} \mathbb{I}(v)/n$. The mean size of a RR set can be computed as

$$1/n \sum_{u \in V} \sum_{v \in V} \mathbb{I}(v, u) = 1/n \sum_{v \in V} \mathbb{I}(v) \leq n\mu_Y$$

Therefore, the expected value of the total sizes of all RR sets is at most

$$\frac{\Lambda}{\mu_Y} \times n\mu_Y = \Lambda n = (\log \frac{1}{\delta} + \log \mathcal{M})\frac{1}{\epsilon^2}n.$$

This completes the proof. □

3.3.3.3 IM

The IM problem can be solved by first running LISA and returning $\hat{S}_k = \{\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_k\}$. The approximation and running time follow directly.

Theorem 20. *There exists a randomized algorithm that returns a* $(1-1/e-\epsilon)$*-approximate of the IM problem in an expected time* $O(\Lambda(m+n))$.

3.3.3.4 Extension to IC model

Our LISA algorithm is easily extended to work on the Independent Cascade (IC) model without effecting the approximation guarantee and complexity order. Differ from the Linear Threshold (LT) model, the edge weight assumption is $0 \leq w(u,v) \leq 1$ (not $\sum_{u \in V} w(u,v) \leq 1, \forall v$ in LT model). IC also operates in rounds, however, the activation criteria is modified to: instead of having a randomly chosen threshold λ_v and v is activated if the total weights form active neighbors exceed λ_v, a node in IC model becomes active through its incoming-edges from the newly activated neighbors. A newly-activated node u will have a single chance of activating its out-going neighbor v and succeed with probability equal the edge weight $w(u,v)$.

As presented in [55], the IC model is also equivalent to a live-edge model and thus, similarly to LT model, corresponds to an RIS sampling procedure [11], termed RIS-IC. By replacing the RIS-LT sampling in LISA with the IC version and following the analysis as for LT model, we obtain the same approximation guarantees (independent with sampling techniques) and complexity results.

Theorem 21. *LISA algorithm for the IC model has an expected running time* $O((\log \frac{1}{\delta} + \log \mathcal{M})\frac{1}{\epsilon^2}(m+n))$ *and returns a sequence of seed sets* $\hat{S}^{k_{lower}}, \ldots, \hat{S}^{k_{upper}}$ *that is an* $(1-$

$1/e - \epsilon$) *IS approximate solution.*

3.3.4 Experiments

In this section, we experimentally evaluate the performance of LISA against the existing state-of-the-art methods, including IMM [105], TIM/TIM+ [106], CELF++[46] and Simpath [47] on five real-world networks with a wide range of sizes from various disciplines. Since there is no easy way of extending the existing algorithms for IS problem, we have to run these algorithms multiple times for all $k \in \{k_{lower}, \ldots, k_{upper}\}$. The experimental results show that our algorithm can solve the IS problem in several orders of magnitudes faster than the runner-up IMM.

3.3.4.1 Experimental Settings

Table 23.: Datasets' Statistics

Datasets	*NetHEPT*	*NetPHY*	*Epinions*	*DBLP*	*Twitter*
Nodes	15K	37K	76K	655K	41.7M
Edges	59K	181K	509K	2M	1.5G
Type	undirected	undirected	directed	undirected	directed
Avg. degree	4.1	4.87	13.4	6.1	70.5

Datasets. We perform our experiments in five datasets: NetHEPT, NetPHY, Epinions, DBLP, and Twitter. The basic statistics of these networks are summarized in Table 23. *NetHEPT*, *NetPHY* and *DBLP* are collaboration networks taken from the "High Energy Physics - Theory", "Physics" sections of arXiv.org and "Computer Science Bibliography". These undirected networks were frequently used in previous works [43, 47, 19]. In the networks, nodes and edges represent authors and co-authorship, respectively. The Epinions dataset is the who-trust-whom online social network of a consumer review site Epinions.com. Specially, the largest network is a large portion of *Twitter*, crawled in July 2009

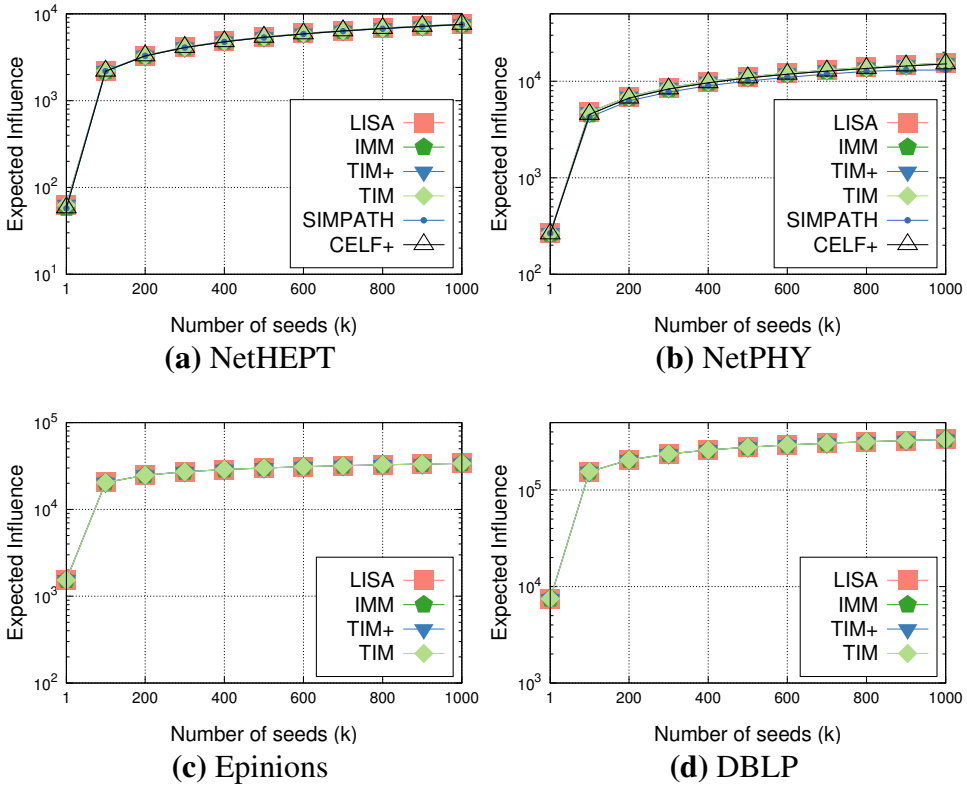
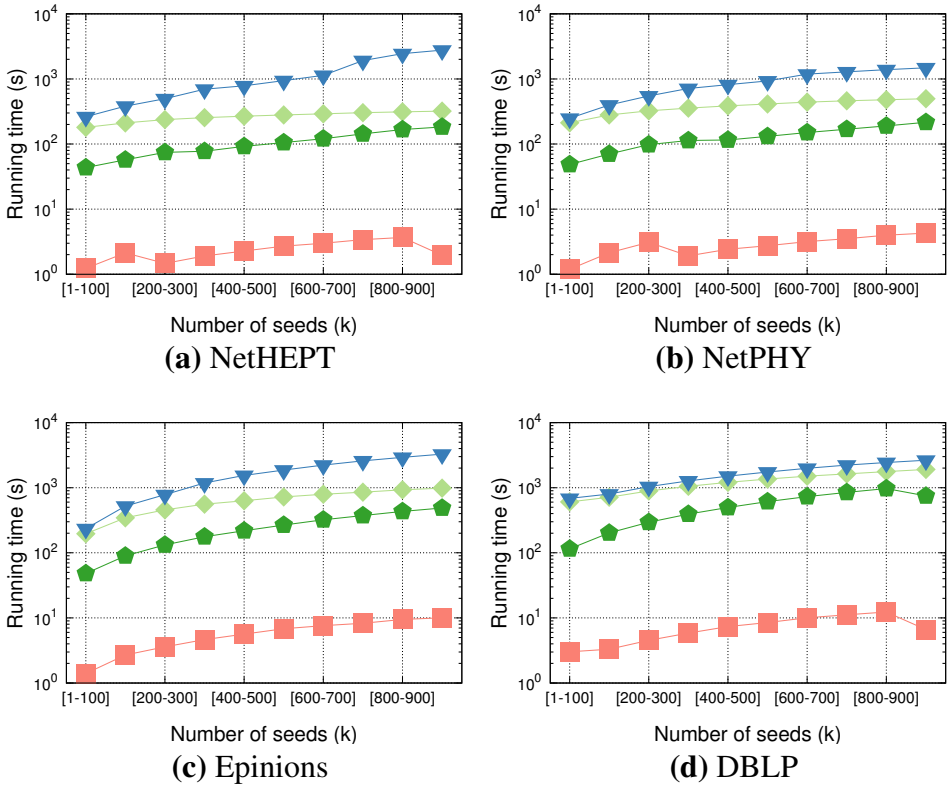


Fig. 25.: Spread of Influence under the LT model (the higher the better)

with 41.7 million nodes and 1.5 billion edges [65].

Metrics. For each algorithm, we measure 1) the *spread of influence*, i.e., the expected number of influenced nodes eventually, 2) the running time, and 3) the peak memory consumption. Note that we only need to run LISA once to get the metrics for all different $k = k_{lower}, \ldots, k_{upper}$, in contrast, we have to run the other algorithms for each value of k individually. We terminate algorithms that take more than 24 hours.

Parameters. We set $\epsilon = 0.1$ and $\delta = 1/n$ for LISA, IMM and TIM/TIM+, unless otherwise mentioned. For CELF++, we use the pruning threshold μ of 10^{-3}. For Simpath, we also set the pruning threshold μ to 10^{-3} and look-ahead value l to 4 as suggested in [47]. Finally, we validate the spread of influence of the outputted seed sets using EIVA (Section 3.3.2) with very high accuracy level: $\epsilon = 0.01$ and $\delta = 1/n$. In our experiments,



(a) NetHEPT **(b)** NetPHY

(c) Epinions **(d)** DBLP

Fig. 26.: Running time of the algorithms under the LT model (see Fig. 25 for legends)

we target the sets with sizes from 1 to 1000 ($k_{lower} = 1, k_{upper} = 1000$).

Weight settings. We adopt the methods in [55] to calculate the influence weights on edges. More precisely, we assign the weight on an edge (u, v) as $b_{uv} = \frac{A(u,v)}{D(v)}$ where $A(u, v)$ is the number of actions both u and v perform, and $D(v)$ is the in-degree of node v, i.e., $N(v) = \sum_{u \in N^in(v)} A(u, v)$.

Enviroment. Our implementation is written in C++ and compiled with GCC 4.7. All our experiments are carried out using a Linux machine with a 2.2GHz 8 core Intel Xeon CPU and 100GB memory of RAM.

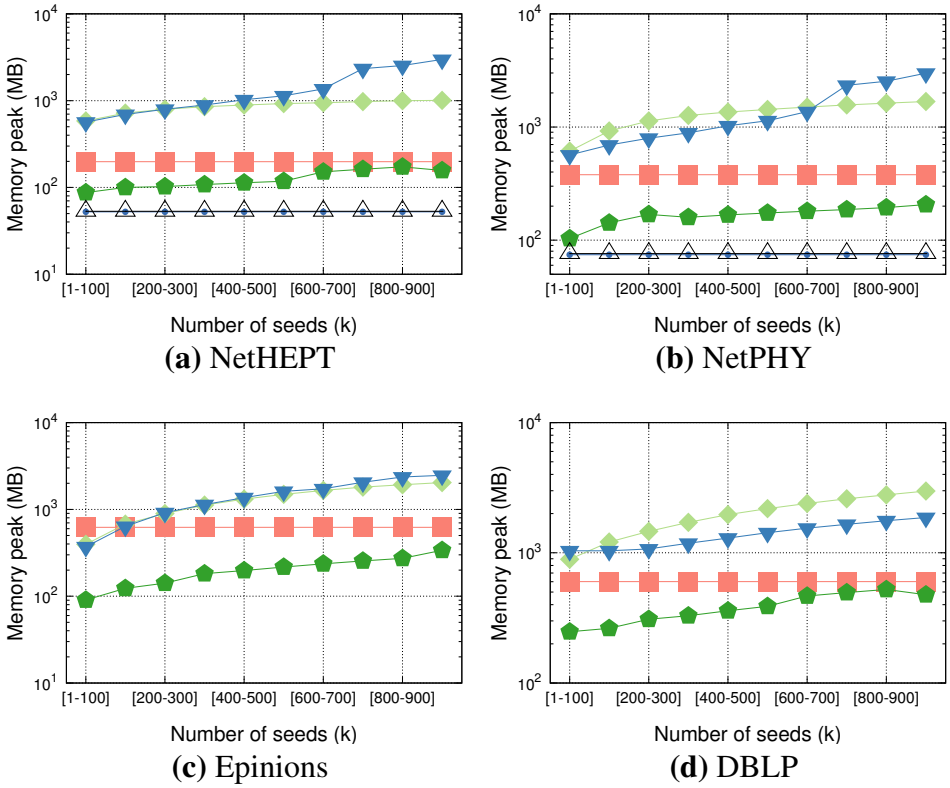


Fig. 27.: Memory usage of the algorithms under the LT model (see Fig. 25 for legends)

3.3.4.2 Results

We carry two set of experiments: 1) on moderate-size datasets, i.e., NetHEPT, NetPHY, Epinions and DBLP in which we run LISA and the competing algorithms under LT model since the results on IC model are similar and report the expected influence, running time and memory usage; 2) on the billion-scale Twitter network in which we run LISA, IMM, TIM and TIM+ (only these algorithms can handle Twitter dataset) under both the LT and IC models and report the running time and memory usage.

Solution Quality. The quality of the algorithms, measured as the expected number of influenced nodes eventually and termed *spread of influence* are shown in Figure 25. We see that all the tested algorithms admit comparable performance in all cases (on all four datasets and with any value of seed set size k). The experimental results also confirm the

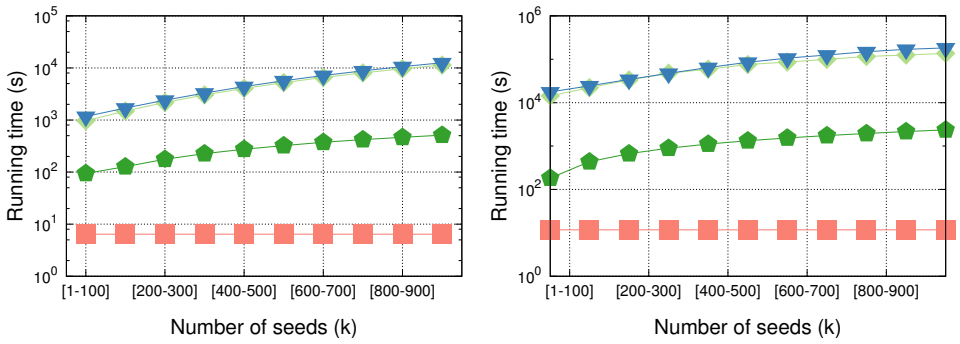
viral marketing behaviors of the influence due to the submodularity property. That is the first few selected nodes carry a huge influence gain and the later ones only bring in smaller marginal influence. Here, we emphasize that only LISA guarantees all the returned seed sets with sizes up to 1000 to have good quality and thus we only run LISA once with $k_{lower} = 1$ and $k_{upper} = 1000$. The other algorithms can guarantee at each particular size and need to run 1000 times, i.e., one for each value of k.

Running Time. In these experiments, we test the performance of all the algorithms on four moderate-size networks. Since all the methods except LISA have to rerun for each value of $k \in \{k_{lower}, \ldots, k_{upper}\}$, we need to accumulate the times for all runs to get a total running time. Thus, we choose a set of 10 small intervals $(k_{lower}, k_{upper}) \in \{(1, 100), (100, 200), \ldots, (900, 1000)\}$ so that we do not bias and have a fair comparison.

The results are presented in Fig. 34. We see that although the intervals are fairly small, LISA vastly outperform the rest of the algorithms in terms of running time. In particular, LISA is always about 100 times faster than the second best IMM methods and up to three orders of magnitudes faster than TIM+ and TIM.

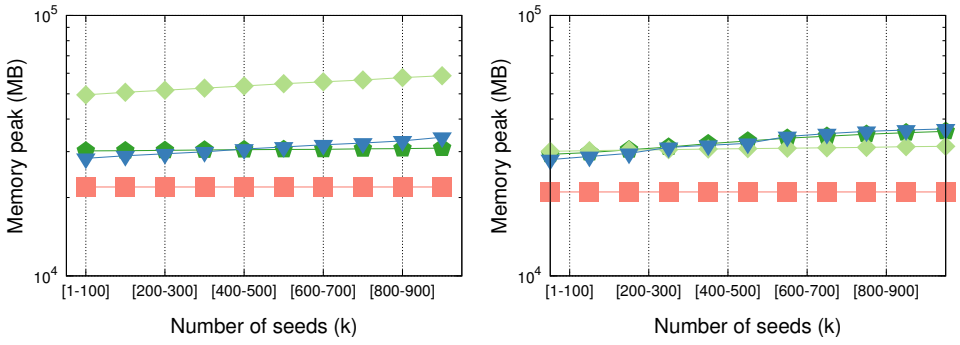
Memory Consumption. We show the memory usages of all the algorithms in Fig. 27. We see that LISA consumes much less memory than TIM+ and TIM but more than IMM, CELF++ and Simpath. However, note that these are moderate-size networks, for larger data as Twitter in the next experiment, LISA requires significantly less memory than IMM, TIM+ and TIM. CELF++ and Simpath implement the naive greedy and only need to store the graph and, indeed, they consume less memory than the others.

Experiments on the billion-scale Twitter network. Since Twitter is the largest tested dataset with millions of nodes and billions of edges, we test LISA and other algorithms under both the LT and IC models on this network. Since the solution quality is identical, we only illustrate the running time and memory usage of the algorithms under the LT and IC models. Since IMM, TIM+ and TIM take very long to run Twitter and for smaller k,



(a) LT model **(b)** IC model

Fig. 28.: Running time of the best algorithms on the billion-scale Twitter network



(a) LT model **(b)** IC model

Fig. 29.: Memory usage of the best algorithms on the billion-scale Twitter network

they requires less time than larger k, we only run them once with $k = k_{lower}$ and consider that to be running time per set size within k_{lower} and k_{upper}. With LISA, we still run for the whole interval but divide the running time by 100 to get the runtime per set size. The results are presented in Figs 28 and 29. These figure again confirm the superiority of LISA in terms of running time: it is up to several orders of magnitudes faster than the others and requires half of the memory for the others.

3.3.5 Conclusion

We propose the computation of *Influence Spectrum* (IS) to give better insights for decision making and resource planning in viral marketing campaigns. To compute IS, we design LISA, an efficient approximation algorithm for IS. LISA returns an $(1 - 1/e - \epsilon)$-approximate influence spectrum with high probability. In practice, LISA also vastly surpasses the state-of-the-art IM methods, being in several orders of magnitudes faster than the rest. While the analysis of LISA is based on LT and IC model, all the results also hold the generalized models that combine both LT and IC in [54]. In the future, we will attempt to push the limit further to develop *sublinear time approximation algorithms* for IS and IM problems.

CHAPTER 4

TRACING THE SOURCES OF MISINFORMATION CASCADES

Given an aftermath of a cascade in the network, i.e. a set V_I of "infected" nodes after an epidemic outbreak or a propagation of rumors/worms/viruses, how can we infer the sources of the cascade? Answering this challenging question is critical for computer forensic, vulnerability analysis, and risk management. Despite recent interest towards this problem, most of existing works focus only on single source detection or simple network topologies, e.g. trees or grids.

In this paper, we propose a new approach to identify infection sources by searching for a seed set S that minimizes the *symmetric difference* between the cascade from S and V_I, the given set of infected nodes. Our major result is an approximation algorithm, called SISI, to identify infection sources *without the prior knowledge on the number of source nodes*. SISI, to our best knowledge, is the first algorithm with *provable guarantee* for the problem in general graphs. It returns a $\frac{2}{(1-\epsilon)^2}\Delta$-approximate solution with high probability, where Δ denotes the maximum number of nodes in V_I that may infect a single node in the network. Our experiments on real-world networks show the superiority of our approach and SISI in detecting true source(s), boosting the F1-measure from few percents, for the state-of-the-art NETSLEUTH, to approximately 50%.

Summary of contributions:

- We propose a new approach to identify multiple infection sources through minimizing the symmetric difference between the cascade of the suspected source nodes S with the infected nodes V_I without knowing *the number of sources a priori*. Our experiments show that methods following this approach including our algorithm SISI

and the greedy algorithm outperform the other approaches in terms of detecting true sources.

- To our best knowledge, we propose the first approximation algorithm, termed SISI, for detecting multiple infection sources in general graphs and our algorithm does not require the knowledge on the number of infection sources. Given an approximation error $\epsilon > 0$, we provide rigorous analysis on sample complexity, deriving the necessary number of samples to guarantee a multiplicative error $(1 \pm \epsilon)$ on the objective estimation.

- Extensive experiments on real-world networks shows the superiority of SISI over other approaches in detecting the exact sources under both SI and IC models. The relax version of SISI is also faster than NETSLEUTH while still retaining high-quality solutions.

4.1 Related works

Infection Source Identification (ISI) under different names has recently emerged and attracted quite a number of researchers in multiple disciplines with diverse techniques. There are two main streams of works and methods that can be listed: 1) exact algorithms on tree graphs [102, 101, 78, 67, 31], 2) *ad hoc* heuristics approaches without any guarantee for general graphs [97, 78, 75].

In the first stream, Shah and Zaman in [102] established the notion of rumor-centrality which is an Maximum Likelihood estimator on regular trees under the SI model. They proposed an optimal algorithm to identify the single source of an epidemic. In [101], the same authors improved the previous results by deriving the exact expression for the probability of correct detection. Later Luo et al. [78] based on approximations of the infection sequences count to develop an algorithm that can find at most two sources in a

geometric tree. Since solely targeting trees, all these methods are unable of solving ISI problem on general graphs.

Lappas et al. [67] formulated ISI problem under the name of k-effector and introduced the minimization of the symmetric difference between the observed infection and the resulting cascade if starting from a candidate source set. While the formulation is novel, their solution is, unfortunately, limited to tree graphs and require the knowledge of the number of infection sources. The extension for general graphs by approximating a graph by a tree does not work well either as we show later in the experiment section.

Prakash et al. [97] resort to heuristic approach to find multiple sources in general graphs and propose NETSLEUTH which relies on the two-part code Minimum Description Length. They show that NETSLEUTH is able to detect both the sources and how many of them. However, besides no guarantee on solution quality, we show in our experiments that NETSLEUTH performs poorly on a simple grid graph with large overlapping region of cascades from two source nodes. Luo et al. [78] also derived an estimator to find multiple sources given that the maximum number of sources is provided. Yet similar to [67], their estimator depends on the approximation of a general graph to tree and also requires the maximum number of sources.

There are also other works on related areas: [53] studies the rumor-centrality estimator on trees under an additional constraint that the status (infected or not) of a node is revealed with probability $p \leq 1$. In case of $p = 1$, the estimator is able to reproduce the previous results and with large enough $p < 1$, it achieves performance within ϵ the optimal. Under a different model, Chen et al. [22] study the problem of detecting multiple information sources in networks under the Susceptible-Infected-Recovered (SIR) model. They propose an estimator for regular trees that can detect sources within a constant distance to the real ones with high probability and investigate a heuristic algorithm for general cases. In another study [75], Lokhov et al. take the dynamic message-passing approach under SIR

model and introduce an inference algorithm which is shown to admit good improvement.

Influence maximization problem [55] that find k nodes to maximize the expected influence is one of the most extensively studied problem. The latest references on the problem can be found in [105] and the references therein.

4.2 Models and Problem definition

We represent the network in which the infection spreads as a directed graph $G = (V, E)$ where V is the set of n nodes, e.g., computers in a computer network, and E is the set of m directed edges, e.g., connections between the computers. In addition, we are given a subset $V_I \subseteq V$ of observed infected nodes and the remaining nodes are assumed to be not infected and denoted by $\bar{V_I} = V \backslash V_I$.

Table 24.: Table of Notations

Notation	Description
n, m	#nodes, #edges of graph $G = (V, E)$.
$V_I, \bar{V_I}$	Set of infected and uninfected nodes.
β, k	Infection probability and $k = \|V_I\|$.
$V(S, \mathcal{M})$	An infection cascade from S under model $\mathcal{M}$.
$D(S, \mathcal{M}, V_I)$	Symmetric different on a graph realization.
$\mathcal{E}[D(S, \mathcal{M}, V_I)]$	The expectation of $D(S, \mathcal{M}, V_I)$ over all realizations.
$\hat{S}$	The returned source set of SISI.
OPT, S^*	The optimal value of $\mathcal{E}[D(S, \tau)]$ and an optimal solution set which achieves the optimal value.
R_j, src(R_j)	A random RR set and its source node src(R_j).
Δ	Maximum size of an RR set ($\Delta \leq V_I$).
c, M	$c = 2(e-2) \approx \sqrt{2}$, $M = 2^k + 1$.
Λ	$\Lambda = (1+\epsilon)2c(\ln \frac{2}{\delta} + k \ln 2 + 1)\frac{1}{\epsilon^2}$.

4.2.1 Infection Model

We focus on the popular *Susceptible-Infected (SI)* model.

Susceptible-Infected (SI) model. In this infection model, each node in the network is in one of two states: 1) Susceptible (S) (not yet infected) and 2) Infected (I) (infected and capable of spreading the disease/rumor). Once infected, the node starts spreading to its neighbors through their connections. While the initial model were proposed for a complete graph topology [6], the model can be extended for arbitrary graph $G = (V, E)$. We assume that the infection spreads in discrete time steps. At time $t = 0$, a subset of nodes, called the infection sources, are infected and the rest is uninfected. Once a node u gets infected at time t, it will continuously try to infect its uninfected neighbor v and succeed with probability $0 < \beta \leq 1$ from step $t + 1$ onwards until successful. The single parameter β indicates how contagious the infection is and thus the higher, the faster it contaminates the network.

Other cascade model. In principle, our formulation and proposed method will work for most progressive diffusion models in which once a node becomes infected, it stays infected. These include the two popular models *Independent Cascade IC* and *Linear Threshold (LT)* models [55]. Other non-progressive models can be first converted to a progressive ones as outlined in [21].

For simplicity, we present our method for the SI model and discuss the extension to the IC and LT models through changing the sampling method in Subsection 4.3.1.

Learning model parameters. Learning propagation model parameters is an important topic and has received a great amount of interest [58, 104, 43, 74, 63]. Our approaches can rely on these learning methods to extract influence cascade model parameters from real datasets, e.g., action logs, connection networks.

4.2.2 Problem Formulation

Intuitively, given an infection model, denoted by $\mathcal{M}$, the goal of infection source identification is to identify a set of source nodes S (unknown size) so that the resulting cascade originated from nodes in S, within a duration $\tau > 0$, matches V_I as closely as possible.

To formalize the problem, we define a cascade $V(S, \mathcal{M})$ as the set of infected nodes if we select nodes in S as the sources (initially infected) under infection model $\mathcal{M}$. Thus, the objective function which characterizes the aforementioned criteria, termed *symmetric difference*, is defined as follows,

$$D(S, \mathcal{M}, V_I) = |V_I \backslash V(S, \mathcal{M})| + |\bar{V}_I \cap V(S, \mathcal{M})| \tag{4.1}$$

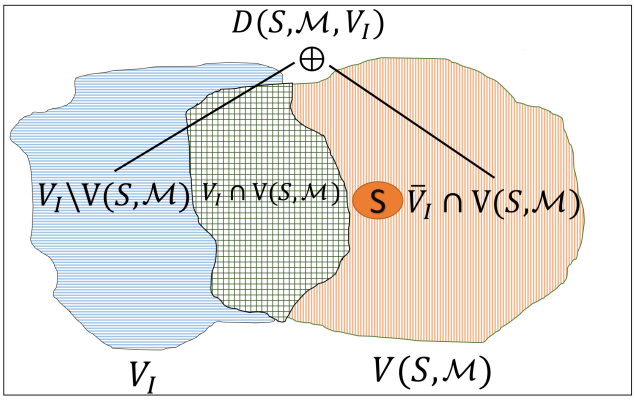


Fig. 30.: Illustration of symmetric difference.

In Eq. 4.1, the first term $|V_I \backslash V(S, \mathcal{M})|$ counts the number of nodes in V_I that are not infected by the propagation spreading from S within a duration τ and the second term indicates the number of nodes that are "mistakenly" infected during the same time interval (illustrated in Fig. 30). Together, the sum measures the similarity between the observed cascade V_I and the cascade causes by the suspected nodes S.

Due to the stochastic nature of the cascade, there are exponentially many possible cascades for a given set of source nodes S. Here cascade is used to refer to the set of infected nodes within τ steps. To account for this, we aggregate the symmetric difference over the probabilistic space of the possible cascades spreading from S. Denote by $\Pr[V(S,\mathcal{M})]$, the probability of receiving a particular cascade $V(S,\mathcal{M})$ within $t = \tau$ time steps. We compute the expected symmetric difference as follows,

$$\begin{aligned}\mathbb{E}[D(S,\mathcal{M},V_I)] &= \sum_{\text{possible } V(S,\mathcal{M})} D(S,\mathcal{M},V_I)\Pr[V(S,\mathcal{M})] \\ &= \sum_{\text{possible } V(S,\mathcal{M})} (|V_I \backslash V(S,\mathcal{M})| + |\bar{V_I} \cap V(S,\mathcal{M})|)\Pr[V(S,\mathcal{M})] \\ &= \sum_{u \in V_I} \Pr[u \text{ not infected by } S] + \sum_{v \notin V_I} \Pr[v \text{ infected by } S] \end{aligned} \tag{4.2}$$

In the last equation, the 'infected' and 'not infected' probabilities are w.r.t. a random cascade from S within τ steps.

We now state the problem of identifying the infection sources as follows.

Definition 16 (Infection Sources Identification)**.** *Given a graph $G = (V,E)$, infection model $\mathcal{M}$ (e.g., β for SI model), observation set V_I of infected nodes, and the duration of the cascade τ (could be infinity), the Infection Sources Identification (ISI) problem asks for a set $\hat{S}$ of nodes such that,*

$$\hat{S} = \arg\min_{S \subseteq V_I} \mathbb{E}[D(S,\mathcal{M},V_I)] \tag{4.3}$$

While this formulation is similar to [67], we do not require knowledge on the number of infection sources.

4.2.3 Hardness and Inapproximability

This subsection shows the NP-hardness and inapproximability results of the ISI problem. From Def. 16, there are two major difficulties in finding the sources: 1) first, by a similar argument to that of the influence maximization problem in [55], the objective function is #P-hard to compute exactly; 2) second, the objective is non-submodular, i.e., there are no easy greedy approaches to obtain approximation algorithms. In fact, we show a stronger inapproximability result in the below theorem.

Theorem 22. *ISI cannot be approximated within a factor* $O(2^{\log^{1-\epsilon} n})$ *for any* $\epsilon > 0$*, where* $n = |V|$*, unless NP* $\subseteq$ *DTIME(*$n^{polylog(n)}$*).*

Proof. To prove Theo. 22, we construct a gap-preserving polynomial-time reduction which reduces any instance of the Red-Blue Set Cover problem [13] to an instance of ISI. The Red-Blue Set Cover problem is defined as follows: an instance of Red-Blue Set Cover problem consists of two disjoint sets: $R = \{r_1, ..., r_p\}$ of red elements, $B = \{b_1, ..., b_q\}$ of blue elements, and a family $T \subseteq 2^{R \cup B}$ of $n (n \geq p, n \geq q)$ subsets of $R \cup B$. The problem asks a subfamily $C^* \subseteq T$ of subsets that covers all the blue elements but minimum number of reds,

$$C^* = \arg\min_{C \subseteq T} \{|R \cap (\cup_{i=1}^{|C|} T_i)|\} \tag{4.4}$$

Our polynomial reduction ensures that if the ISI instance has an $O(2^{\log^{1-\epsilon} n})$-approximate solution S, then there must be a corresponding $O(2^{\log^{1-\epsilon} n})$-approximate solution of the Red-Blue Set Cover polynomially induced from S. The reduction is grounded on the observation that any solution of the Red-Blue Set Cover costs at most p - the number of red elements. Then, based on the result in [13] that the Red-Blue Set Cover cannot be approximated within a factor $O(2^{\log^{1-\epsilon} N})$ where $N = n^4$ for any $\epsilon > 0$ unless NP $\subseteq$ DTIME($N^{polylog(N)}$), we obtain the Theorem 22.

We will give a polynomial reduction from an instance of the Red-Blue Set Cover to an ISI instance with $\beta = 1$ and $\tau = 1$ such that,

(**1**) The optimal solution of the ISI instance polynomially infers the optimal solution for the instance of Red-Blue Set Cover.

(**2**) If we obtain an $O(2^{\log^{1-\epsilon} n})$-approximate solution for ISI, we will also have an $O(2^{\log^{1-\epsilon} n})$-approximate solution for the Red-Blue Set Cover instance.

These two conditions are sufficient to conclude that we cannot approximate the optimal solution of ISI within a factor $O(2^{\log^{1-\epsilon} n})$ unless we can do that for Red-Blue Set Cover. Thus, the Theorem 22 follows. We will present the reduction and then prove the satisfaction of each condition.

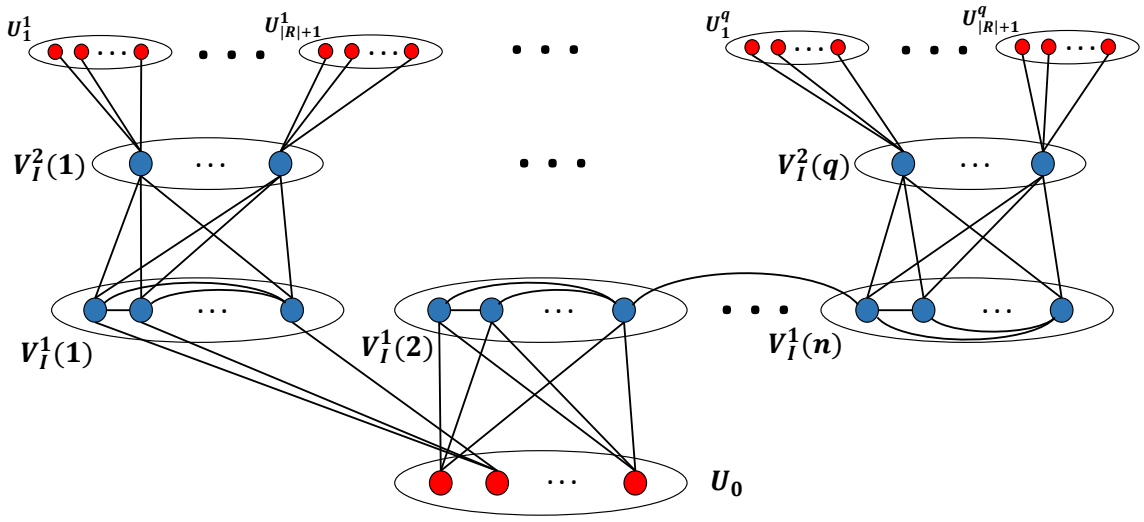


Fig. 31.: Reduction from Red-Blue Set Cover to ISI in which infected nodes are blue-colored and uninfected nodes are in red.

Given an instance of Red-Blue Set Cover with two sets R, B and a family T, we suppose all the subsets in T contains at least a blue element, otherwise we can trivially

discard all those subsets since we never select that type of subsets. In the reverse way, we also suppose every pair of subset in T has at least one red element different from each other. Otherwise we always select/reject both at the same time without changing the cost, in other words, we can group together to create one subset. We construct a corresponding ISI instance consisting of the node set V, the infected set $V_I \subseteq V$ and the set of edges E as follows (depicted in Fig. 43):

- Set of infected nodes V_I: For each subset $T_i \in T$, there is a set $V_I^1(i)$ of infected nodes whose number is the number of blues in T_i. For each blue node B_j in B, we form a set $V_I^2(j)$ of $|R| + 1$ infected nodes.
- Set of uninfected nodes $V \backslash V_I$: For each infected node l in $V_I^2(j)$, a set U_l^j of $|R| + 1$ uninfected nodes is constructed. We also have a set U_0 of p uninfected nodes corresponding to the red set R in Red-Blue Set Cover instance.
- Set of edges E: For any pair $(u, v) \in V_I^1(i)$, we connect them by an edge, so that the subgraph of nodes in $V_I^1(i)$ is a clique. For each $u \in T_i \cap T_j$, we connect the two corresponding nodes in $V_I^1(i)$ and $V_I^1(j)$ by an edge. For each $u \in V_I^1(i)$, we connect u to all $|R| + 1$ nodes in $V_I^2(u)$ and, subsequently, each node l in $V_I^2(u)$ is connected to all $|R| + 1$ nodes in U_l^u. For any pair $u, v \in V_I^1(j)$ for each $j \in \{1, ..., n\}$, we connect u with all the nodes in $V_I^2(v)$ and v with all the nodes in $V_I^2(u)$. If the subset T_i contains red element R_j, then for each $u \in V_I^1(i)$, there is an edge connecting u to the corresponding node of R_j in U_0.

Now, we will prove the two conditions consecutively. Our proof relies on two observations: the first one is that if the feasible solution for ISI contains at least an infected node from $V_I^2(j)$ for some $j \in \{1, ..., q\}$, then the number of uninfected nodes covered is at least $|R| + 1$ which causes the cost to be at least $|R| + 1$. The same phenomenon happens if an infected node v in $V_I^1(j)$ for some $j \in \{1, ..., n\}$ is not covered since there would be $|R| + 1$

infected nodes in $V_I^2(v)$ not covered. On the other hand, if all the infected nodes in $V_I^1(j)$ for all $j \in \{1, ..., n\}$ are covered, then all infected nodes in the whole network are indeed covered and at most $|R|$ uninfected nodes (in U_0) are also covered. The second observation with the previous case is that in the original Red-Blue Set Cover instance, we select those subset T_i such that the corresponding $V_I^1(i)$ contains a infection source chosen in ISI, then the cost in the two problem are equal (cover the same number of red elements/uninfected nodes).

*Prove condition **(1)**.* Based on our observation, the optimal solution S^* of the ISI instance has to cover all the nodes in $V_I^1(j)$ for all j and has the least number of uninfected nodes covered. From this solution, we construct the solution for the original Red-Blue Set Cover instance by selecting the subfamily C^* of subsets T_i such that the corresponding $V_I^1(i)$ contains a infection source in the optimal solution of ISI. First, this subfamily covers all the blue elements since each blue element corresponds to some infected nodes in $V_I^1(j)$ for some j. Secondly, if this subfamily has the lowest cost (covers the least number of red elements). Otherwise, suppose that a different subfamily $\hat{C}$ has lower cost, then we can equivalently find another solution for the reduced ISI instance and obtain the same cost (lower than that of S^*). That contradicts with the optimality of S^*.

*Prove condition **(2)**.* Based on condition **(1)** that the optimal solution of ISI instance infers the optimal solution of Ref-Blue Set Cover with the same cost. Suppose we have an $O(2^{\log^{1-\epsilon} n})$-approximate solution $\hat{S}$ for Red-Blue Set Cover instance, there are two possible cases:

- If $\hat{S}$ contains a node in $V_I^2(j)$ for some j or $\hat{S}$ does not cover a node in $V_I^1(j)$ for some j, then based on the first observation, the cost of $\hat{S}$ has to be at least $|R| + 1$. Because this is an $O(2^{\log^{1-\epsilon} n})$-approximate solution, we just select the whole family T in Red-Blue Set Cover instance which has cost of only $|R|$ and obtain an $O(2^{\log^{1-\epsilon} n})$-

approximate solution.

- Otherwise, based on the second observation, we can easily construct a solution for Red-Blue Set Cover with equal cost and thus obtain an $O(2^{\log^{1-\epsilon} n})$-approximate solution.

Lastly, note that the number of blue and red elements must be at least $|T|$, otherwise we can drop or merge some sets together without effecting any solution. Thus, by following our construction of the ISI instance, we determine the number of uninfected nodes,

$$\begin{aligned} |V| &= |U_0| + \sum_{i=1}^{n} |V_I^1(i)| + \sum_{i=1}^{q} |V_I^2(i)| \cdot \sum_{j=1}^{|R+1|} |U_j^i| \\ &\leq |T| + |T| + |T|^2(|T|+1) \leq |T|^4 \quad (—T| \geq 4) \end{aligned} \tag{4.5}$$

Since $|T| = n$ and the Red-Blue Set Cover cannot be approximated within a factor of $O(2^{\log^{1-\epsilon} N})$ where $N = n^4$ for any $\epsilon > 0$ unless NP $\subseteq$ DTIME($N^{polylog(N)}$), we obtain our results in Theo. 22. □

4.3 Sampling-based SISI algorithm

In this section, we present SISI, our sampling-based method with guarantee on achieving $\frac{2}{(1-\epsilon)^2}\Delta$-approximation factor for arbitrary small $\epsilon > 0$. Here Δ equals the maximum nodes in V_I that can infect a single node in the graph and is the same with the maximum sample size in Subsec. 4.3.1.

Outline. SISI contains two key components: 1) an efficient Truncated Reverse Infection Sampling (TRIS) to compute the objective with high accuracy and confidentiality (presented in Subsection 4.3.1) and 2) an innovative transformation of the studied problem into a submodular-cost covering problem to provide high quality solutions with performance guarantees (presented in Subsection 4.3.2). We show the combination of the two components to obtains the SISI algorithm in Subsection 4.3.3.

4.3.1 Truncated Reverse Infection Sampling

We propose the *Truncated Reverse Infection Sampling* (TRIS) strategy to generate random Reverse Reachable (RR) sets, following the *reverse influence sampling* method (RIS) pioneered in [11]. A RR set, R_j, is generated as follows.

Definition 17 (Reverse Reachable set (RR set))**.** *Given $G = (V, E)$, probability β and propagation time τ, a RR set is generated from G by 1) selecting a (uniformly) random source node $v \in V$, 2) generating a reverse random cascade from v in G within τ steps and 3) returning R_j as the set of nodes in the cascade.*

The main intuition is that each RR set R_j contains the nodes that can infect its source $v = \mathsf{src}(R_j)$ within a given time τ. Thus RR sets were used in previous works [11, 105, 89] (without the step/time limit t) to estimate influence of nodes. We shall show later in next subsection that RR sets can also be fine-tuned to estimate the chance of being infection sources.

Note that the above description of generating RR sets is model-independent, i.e., you can use it with many different cascade models for reverse cascade simulation in the step 2. For example, the reverse simulation for IC and LT, the two most popular cascades models, are presented in [11] and [80], respectively. Here we focus on the reverse sampling for SI model and highlight the necessary changes to make the method work for our problem.

4.3.1.1 Generating RR Sets under SI model.

The main difference between SI model vs. LT and IC models are SI model allows multiple attempts for an infected node to its neighbors in contrast to a single attempt in IC and LT. Given a network $G = (V, E)$ and infection probability $0 < \beta \leq 1$, RR sets in the SI model are generated as follows.

1) Select a random node u. Only u is infected at time 0 and all other nodes are not

Algorithm 21: Fast-TRIS

Input: Graph G, probability β, max time τ and V_I

Output: A random RR set R_j

Pick a random node $u \in V$

RR set $R_j = \{u\}$

Infection time $T\{v\} = \infty, \forall v \in V \backslash \{u\}, T\{u\} = 0$

Min priority queue $PQ = \{u\}$

while PQ *not empty* **do**

$u = PQ.pop()$

foreach $v \in (\textit{in-neighbors}(u) \backslash R_j) \cup PQ$ **do**

$r \leftarrow$ a random number in [0,1]

$t \leftarrow \lceil \log_{1-\beta}(1-r) \rceil$ {Assume $0 < \beta < 1$}

$T(v) = \min\{T(v), T(u) + t\}$

if $T(v) < \tau$ **then**

if $v \notin R_j$ **then**

if $v \in V_I$ **then**

$R_j = R_j \cup \{v\}$

end

$PQ.push(v)$

else

$PQ.update(v)$

end

end

end

end

Return R_j

infected.

2) For each time step $i \in [1, \tau]$, consider all edges $(u, v) \in E$ in which v is infected and u is not infected (note the direction). Toss a β-head biased coin to determine whether u succeeds in infecting v. If the coin gives head (with a probability β), we mark u as infected.

3) After τ steps, return R_j as the set of infected nodes, *removing all nodes that are not in* V_I.

Note the last step, the nodes that are not in V_I will be removed from the RR set (hence the name truncated). This truncation is due to the observation that the suspected nodes must be among the infected nodes in V_I. Our RR sets are in general smaller than the RR sets in [11] and might be empty. This saves us a considerable amount of memory in storing the RR sets.

A naive implementation of the above reserve sampling has a high complexity and does not scale when τ grows, thus we present a fast implementation using geometric distribution in Algorithm 21.

The complete pseudocode for the fast TRIS algorithm is described in Alg. 21. The key observation to speed up the TRIS procedure is that each trial in the sequence of infection attempts is a Bernoulli experiment with success probability of β. Thus this sequence of attempts until successful actually follows a geometric distribution. Instead of tossing the Bernoulli coin many times until getting a head, we can toss once and use the geometric distribution to determine the number of Bernoulli trials until successful (Lines 8,9).

Another issue is the order of attempts since a node can be infected from any of her in-neighbors but only the earliest one counts. Therefore, we will keep the list of all newly infected nodes in a min priority queue (PQ) w.r.t infection time. In each iteration, the top node is considered (Lines 6). The algorithm behaves mostly like the legacy Dijkstra's

algorithm [41] except we have time for a node w to infect a node v on each edge (w, v) instead of the length. Also, the algorithm is constrained within the region consisting of nodes at most 'distance' τ from the selected u.

The time complexities of the naive and fast implementation of TRIS are stated in the following lemma.

Lemma 50. *Expected time complexity of the naive TRIS is,*

$$C(R_j) = \frac{\Delta m \tau}{n} \tag{4.6}$$

and that of the fast implementation is,

$$C'(R_j) = \frac{\Delta m}{n} + \Delta \log(\Delta) \log(1 + \frac{\Delta m}{n^2}) \tag{4.7}$$

Proof. Similar to the analysis of Expected Performance of Dijkstra's Shortest Path Algorithm in [41] and denote the expected complexity of the fast algorithm by $C'(R_j)$, we have,

$$C'(R_j) = C(edges) + \Delta \log(\Delta) \log(1 + C(edges)/n) \tag{4.8}$$

where $C(edges)$ is the expected number of edges examined. Note that this is different from $C(R_j)$ since in this case, each edge can be checked once while, for the latter, it is multiple until successful. Δ is defined previously as the maximum size of a RR set. We also have,

$$C(edges) \leq \frac{1}{n} \sum_{u \in V} \sum_{v \in V} \Pr[u, v] d^{in}(v) \leq \frac{\Delta m}{n} \tag{4.9}$$

in which the details are similar to that of Eq. 4.6. Thus, combining with Eq. 4.8, we obtain,

$$C'(R_j) = \frac{\Delta m}{n} + \Delta \log(\Delta) \log(1 + \frac{\Delta m}{n^2}) \tag{4.10}$$

In Eq. 4.7, the first term is usually the leading factor and, then, the complexity depends mostly on $\frac{\Delta m}{n}$. We now analyze the expected time complexity $C(R_j)$ of generating R_j by

the naive way.

$$C(R_j) \leq \frac{\tau}{n} \sum_{u \in V} \sum_{v \in V} \Pr[u, v] d^{in}(v) = \frac{\tau}{n} \sum_{v \in V} d^{in}(v) \sum_{u \in V} \Pr[u, v]$$

where $\Pr[u, v]$ is the probability of v infected by u within τ steps, $d^{in}(v)$ is the in-degree of v. Here we take the average over all possible sources u of R_j (each has probability $1/n$) and the maximum number of edge checks for node v is $\tau d^{in}(v)$. Let denote the maximum size of a random RR set as Δ, we get $\sum_{u \in V} \Pr[u, v] \leq \Delta$ and thus,

$$C(R_j) \leq \frac{\tau}{n} \sum_{v \in V} d^{in}(v) \Delta = \frac{\Delta \tau}{n} \sum_{v \in V} d^{in}(v) = \frac{\Delta m \tau}{n} \tag{4.11}$$

From Eq. 4.6, the complexity depends linearly on τ and is very high with large values of τ. □

Thus, the running time $C'(R_j)$ of our fast implementation is roughly τ *times smaller than that* $C(R_j)$ of the naive implementation, especially, for large values of τ.

4.3.1.2 Chance of Being Infection Sources

We show how to utilize the generated RR sets to estimate the chance that nodes being infection sources. First, we classify each generated RR R_j into one of the two groups, based on the source of R_j, denoted by $\mathsf{src}(R_j)$.

- $\mathcal{R}_{Blue} = \{R_j | \mathsf{src}(R_j) \in V_I\}$: The set of blue RR sets that sources are.
- $\mathcal{R}_{Red} = \{R_j | \mathsf{src}(R_j) \notin V_I\}$: The set of red RR sets that sources are *not* in V_I.

Since the infection sources infect the nodes in V_I but not the nodes outside of V_I (within a time τ), thus, the infection sources should appear *frequently in blue* RR sets (of which sources are in V_I) and appear *infrequently in red* RR sets (of which sources are not in V_I.) Thus, a node v that appear in many blue RR and few red RR sets will be more likely to be among the infection sources.

The above observation can be generalized for a given a subset of nodes $S \subset V_I$, e.g., a subset of suspected nodes. A subset S that *covers* (i.e. to intersect with) *many blue* RR sets and *few red* RR sets will be more likely to be the infection sources.

Define the following two subgroups of RR sets,

$$\mathcal{R}^{-}_{Blue}(S) = \{R_j | R_j \in \mathcal{R}_{Blue} \text{ and } R_j \cap S = \emptyset\}, \text{ and} \tag{4.12}$$

$$\mathcal{R}^{+}_{Red}(S) = \{R_j | R_j \in \mathcal{R}_{Red} \text{ and } R_j \cap S \neq \emptyset\}. \tag{4.13}$$

They are the blue RR sets that a suspected subset S "fails" to cover (i.e. to intersect with) and the red RR sets that S ("mistakenly") covers. The less frequent a random RR set R_j falls into one of those two subgroups, the more likely S will be the infection sources.

Formally, we can prove that the probability of a random RR set falls into one of those two subgroups equals exactly our objective function, denoted by $\mathbb{E}[D(\hat{S}, \tau, V_I)]$. We state the result in the following lemma.

Lemma 51. *Given a fixed set $S \in V_I$, for a random RR set R_j, denote X_j a random variable such that,*

$$X_j = \begin{cases} 1 & \textit{if } R_j \in \mathcal{R}^{-}_{Blue}(S) \textit{ or } R_j \in \mathcal{R}^{+}_{Red}(S) \\ 0 & \textit{otherwise.} \end{cases} \tag{4.14}$$

then,

$$\mathbb{E}[X_j] = \frac{\mathbb{E}[D(S, \tau, V_I)]}{n} \tag{4.15}$$

Proof. Since for a random RR set R_j, $R_j \in \mathcal{R}^{-}_{Blue}(S)$ and $R_j \in \mathcal{R}^{+}_{Red}(S)$ are two mutually exclusive events,

$$\mathbb{E}[X_j] = \Pr_{R_j}[R_j \in \mathcal{R}^{-}_{Blue}(S)] + \Pr_{R_j}[R_j \in \mathcal{R}^{+}_{Red}(S)] \tag{4.16}$$

We will prove an equivalent formula of Eq. 4.15 that,

$$\mathcal{E}[D(S,\tau)] = n(\Pr_{R_j}[R_j \in \mathcal{R}^{-}_{Blue}(S)] + \Pr_{R_j}[R_j \in \mathcal{R}^{+}_{Red}(S)])$$

Let define $\mathcal{G}$ as a realization of the graph G, $\mathcal{G} \sim G$, where each edge (u, v) is assigned a length value indicating the number of trials u has to make until v gets infected from u. In one realization $\mathcal{G}$, the cascade from S at time τ, $V(S,\tau)$, is uniquely defined (the reachable nodes from S within τ-length path) and so as $D(S,\tau)$. According to the definition of $\mathcal{E}[D(S,\tau)]$ in Eq. 4.2, we have,

$$\mathcal{E}[D(S,\tau)] = \sum_{u \in V_I} \Pr_{\mathcal{G} \sim G}[u \text{ not infected}] + \sum_{v \notin V_I} \Pr_{\mathcal{G} \sim G}[v \text{ infected}]$$

Let denote $R_j(u)$ be a random RR set rooted at u, the first term in the right-hand side is equivalent to,

$$\sum_{u \in V_I} \Pr_{\mathcal{G} \sim G}[u \text{ not infected}] = \sum_{u \in V_I} \Pr_{R_j(u) \vdash \mathcal{G}}[S \cap R_j(u) = \emptyset]$$

where $R_j(u) \vdash \mathcal{G}$ denotes the consistency of $R_j(u)$ to $\mathcal{G}$ since $\mathcal{G}$ is a realization of G and thus $R_j(u)$ is well-defined. Since,

$$\Pr_{R_j(u) \vdash \mathcal{G}}[S \cap R_j(u) = \emptyset] = \Pr_{R_j}[S \cap R_j = \emptyset \mid \mathsf{src}(R_j) = u]$$

we obtain,

$$\sum_{u \in V_I} \Pr_{\mathcal{G} \sim G}[u \text{ not infected}] = \sum_{u \in V_I} \Pr_{R_j}[S \cap R_j = \emptyset \mid \mathsf{src}(R_j) = u]$$

$$= \sum_{u \in V_I} \frac{\Pr_{R_j}[S \cap R_j = \emptyset \ \& \ \mathsf{src}(R_j) = u]}{\Pr_{R_j}[\mathsf{src}(R_j) = u]}$$

$$= \sum_{u \in V_I} \Pr_{R_j}[S \cap R_j = \emptyset \ \& \ \mathsf{src}(R_j) = u] \cdot n$$

(since the source of each RR set is randomly chosen)

$$= n \sum_{u \in V_I} \Pr_{R_j}[S \cap R_j = \emptyset \ \& \ \mathsf{src}(R_j) = u]$$

$$= n \Pr_{R_j}[S \cap R_j = \emptyset \ \& \ \mathsf{src}(R_j) \in V_I] \tag{4.17}$$

$$= n \Pr_{R_j}[R_j \in \mathcal{R}^{-}_{Blue}(S)] \tag{4.18}$$

The Eq. 4.17 follows from the fact that, for all $u \in V_I$, $(S \cap R_j = \emptyset \ \& \ \mathsf{src}(R_j) = u)$ are mutually exclusive. Thus,

$$\sum_{u \in V_I} \Pr_{\mathcal{G} \sim G}[u \text{ not infected}] = n \Pr_{R_j}[R_j \in \mathcal{R}^{-}_{Blue}(S)] \tag{4.19}$$

Similarly, we can also achieve,

$$\sum_{v \in \bar{V}_I} \Pr_{\mathcal{G} \sim G}[v \text{ infected}] = n \Pr_{R_j}[R_j \in \mathcal{R}^{+}_{Red}(S)] \tag{4.20}$$

From Eq. 6.4, Eq. 6.5 and Eq. 4.16, we obtain

$$\mathcal{E}[D(S, \tau)] = n(\Pr_{R_j}[R_j \in \mathcal{R}^{-}_{Blue}(S)] + \Pr_{R_j}[R_j \in \mathcal{R}^{+}_{Red}(S)])$$

which completes the proof of Lem. 51. □

Lem. 51 suggests a two-stages approach to identify the infection sources: 1) generating many RR sets and 2) look for a subset $S \subset V_I$ that minimize the size of $|\mathcal{R}^{-}_{Blue}(S) \cup$

$R_j \in \mathcal{R}^+_{Red}(S)|$. In next two subsections, we address two key issues of this approach 1) *Optimization method* to identify S with guarantees and 2) *Sample complexity*, i.e., how many RR sets is sufficient to generate a good solution. Too few RR sets lead to biased and poor solutions, while too many RR set lead to high running time.

4.3.2 Submodular-cost Covering

We will transform the ISI problem to a submodular-cost covering problem over the generated RR sets. This allows us to apply the Δ-approximation algorithm in [61], where Δ is the maximum size of any RR set.

By Lemma 51, the problem of minimizing $\mathcal{E}[D(S, \mathcal{M}, V_I)]$ can be cast as a minimization problem of $\Pr[R_j \in \mathcal{R}^-_{Blue}(S) \cup \mathcal{R}^+_{Red}(S)]$. This, in turn, can be approximated with the following problem over the generated RR sets.

$$\min_{S \subseteq V_I} |\mathcal{R}^-_{Blue}(S) \cup \mathcal{R}^+_{Red}(S)|, \tag{4.21}$$

and, since $R_j \in \mathcal{R}^-_{Blue}(S)$ and $R_j \in \mathcal{R}^+_{Red}(S)$ are disjoint, the above minimization problem is equivalent to,

$$\min_{S \subseteq V_I} |\mathcal{R}^-_{Blue}(S)| + |\mathcal{R}^+_{Red}(S)| \tag{4.22}$$

We shall convert the above problem to the *submodular-cost covering* in [61], stated as follows.

Definition 18 (Submodular-cost covering)**.** *[61] An instance is a triple $(c, \mathcal{C}, U)$ where*

- *The cost function $c(x) : \mathbf{R}^n_{\geq 0} \to \mathbf{R}_{\geq 0}$ is submodular, continuous, and non-decreasing.*
- *The constraint set $\mathcal{C} \subseteq 2^{\mathbf{R}_{\geq 0}}$ is a collection of covering constraints, where each constraint $S \in \mathcal{C}$ is a subset of $\mathbf{R}^n_{\geq 0}$.*

- *For each $j \in [n]$, the domain U_j for variable x_j is any subset of $\mathbf{R}_{\geq 0}$.*

The problem is to find $x \in \mathbf{R}^n_{\geq 0}$, minimizing $c(x)$ subject to $x_j \in U_j, \forall j \in [n]$ and $x \in S, \forall S \in \mathcal{C}$.

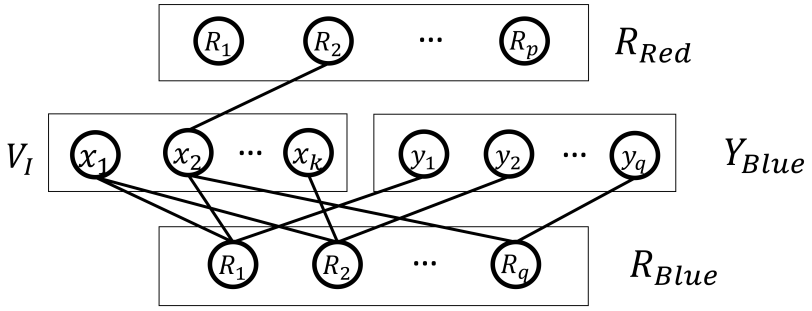


Fig. 32.: Conversion to Submodular-cost covering.

Conversion to submodular-cost covering problem. We convert the form in Eq. 4.22 into a submodular-cost covering problem as demonstrated in Fig. 32. Let $q = |\mathcal{R}_{Blue}|$ and $p = |\mathcal{R}_{Red}|$. We associate a variable $x_u \in [0, 1]$ for each $u \in V_I$ to indicate whether the corresponding node is selected as an infected source. We also assign a variable y_j to each RR set $R_j \in \mathcal{R}_{Blue}$. We require all blue RR R_j sets to be covered through the constraint $\max\{\max_{u \in R_j} x_u, y_j\} \geq 1$. Thus for each blue R_j either $x_v = 1$ for some $v \in R_j$ or the corresponding $y_j = 1$.

The objective is to minimize the cost function $\min_{x,y} c(x, y) = \sum_{R_j \in \mathcal{R}_{Red}} \max_{u \in R_j}(x_u) + \sum_{j=1}^{q} y_j$. The first part of the cost function $\max_{u \in R_j}(x_u)$ is a submodular function since the $\max$ function is submodular (see footnote 1, page 2 in [61]). The second part $\sum_{j=1}^{q} y_j$ is a linear function, and thus is also a submodular function. Therefore, *the objective is a submodular function*.

Thus, the problem in Eq. 4.22 can be converted to the following submodular-cost

covering problem,

$$\min_{x,y} c(x,y) = \sum_{R_j \in \mathcal{R}_{Red}} \max_{u \in R_j}(x_u) + \sum_{j=1}^{q} y_j \tag{4.23}$$

$$\text{subject to (for each } R_j \in \mathcal{R}_{Blue}) \ \max\{\max_{u \in R_j} x_u, y_j\} \geq 1$$

Since for any assignment of variable set x, we have a corresponding source selection: node u is selected as infection source if $x_u = 1$. The first term $\sum_{R_j \in \mathcal{R}_{Red}} \max_{u \in R_j}(x_u)$ in Eq. 4.23 is equivalent to $|\mathcal{R}^+_{Red}(S)|$ in Eq. 4.22 and similarly $\sum_{j=1}^{q} y_j$ together with the constraints is equivalent to $|\mathcal{R}^-_{Blue}(S)|$. In Eq. 4.23, each covering constraint is associated with a blue RR set R_j and says that if R_j is not covered by any variable x_u ($x_u = 1$), then $y_j = 1$ which will increase the cost function by 1. Thus, Eq. 4.23 minimizes the number of red RR sets covered and blue RR sets uncovered. **Δ-Approximation Algorithm.** Our reformulation of ISI to submodular-cost covering problem is similar to that of the facility location problem in Section 7 of [61]. According to Lemma 5 in [61], the following greedy algorithm (Alg. 22) runs in linear time with respect to the total size of all the RR sets and returns an Δ-approximate solution.

Theorem 23. *Alg. 22 returns an Δ-approximate solution for the submodular-cost covering formulation of the ISI problem, where Δ is the maximum size of an RR set (thus, $\Delta \leq V_I$), and runs in linear time.*

The Alg. 22 starts with formulating the submodular-cost covering problem from V_I and $\mathcal{R}$ by creating the necessary variables, cost function and constraints as specified previously. A variable x_u is initialized to 0 and gets updated in the iterations that node u is in the RR set considering in those iterations. The algorithm passes through all the RR sets $R_j \in \mathcal{R}_{Blue}$ and makes each of them satisfied in a single iteration in which it calculates the minimum increase θ of the cost function (Line 4-5) that satisfies the constraint. This minimum increase is computed by sequentially trying to raise each variable $x_u : u \in R_j$

Algorithm 22: Submodular-cost-Covering

Input: Infected set V_I, collection of RR sets $\mathcal{R}$

Output: An Δ-approximate set $\hat{S}$

Formulate the submodular cost covering version from $\mathcal{R}$

$x_u = 0, \forall u \in V_I$ and $y_j = 0, \forall j : R_j \in \mathcal{R}_{Blue}$

foreach $R_j \in \mathcal{R}_{Blue}$ **do**

$\theta = \min_{u \in R_j} \sum_{R_t \in \mathcal{R}^+_{Red}(u)} (1 - \max_{v \in R_t} x_v)$

$\theta = \min\{\theta, 1 - y_j\}$

foreach $u \in R_j$ **do**

if $\mathcal{R}^+_{Red}(u) = \emptyset$ **then**

$x_u = 1$

else

$x_u = \frac{1}{|\mathcal{R}^+_{Red}(u)|}\big(\theta + \sum_{R_t \in \mathcal{R}^+_{Red}(u)} \max_{v \in R_t} x_v\big)$

end

end

$y_j = y_j + \theta$

end

Add u into $\hat{S}$ if $x_u = 1$

Return $\hat{S}$

or y_j to 1 (covering) and calculating the corresponding cost. Afterwards, it updates each variable of R_j by an amount that makes the cost function increased by θ (Line 6-11). At the end, it selects the nodes in V_I that have value 1 in their variables (Line 12).

Algorithm 23: SISI Algorithm

Input: Graph $G = (V, E)$, infection probability β, a set of infected nodes V_I, an infection model $\mathcal{M}$ and $\epsilon, \delta \in (0, 1)$.

Output: Initial infected set $\hat{S}$.

$\Lambda = (1 + \epsilon)2c\Big[\ln \frac{2}{\delta} + k \ln 2 + 1\Big]\frac{1}{\epsilon^2}$

$T = \Lambda, \mathcal{R} \leftarrow \emptyset$

repeat

Generate T additional RR sets by Fast-TRIS (or the reverse sampling in [11, 89] for IC, LT models)

$\hat{S} =$ Submodular-cost-Covering$(V_I, \mathcal{R})$

$T = |\mathcal{R}|$

$\Delta = \max_{R_j} |R_j|$

if $\epsilon > 1/(1 + \Delta)$ **then**

$\epsilon = 1/(1 + \Delta)$

$\Lambda = (1 + \epsilon)2c\Big[\ln \frac{2}{\delta} + k \ln 2 + 1\Big]\frac{1}{\epsilon^2}$

end

until $|R^-_{Blue}(\hat{S})| + |R^+_{Red}(\hat{S})| \geq \Lambda$;

Post-optimization($\hat{S}$)

Return $\hat{S}$

4.3.3 SISI Approximation Algorithm

We will describe the approximation algorithm, named SISI, which combines the three key advanced components: TRIS sampling (Subsec. 4.3.1), the Δ-approximate submodular-

cost covering algorithm (Subsec. 4.3.2) and a stopping condition in [89], to solve the ISI problem and returns an $\Delta\frac{2}{(1-\epsilon)^2}$-approximate solution with at least $(1-\delta)$-probability (proved in Sec. 4.4). The description of SISI is given in Alg. 23.

SISI begins with initializing Λ which will decide the stopping condition (Line 11). The whole algorithm iterates through multiple steps: in the first step, it generates Λ RR sets and add them to $\mathcal{R}$ since, to satisfy the stopping condition (Line 11), we need at least Λ RR sets; in subsequent iterations, the algorithm doubles the number of RR sets in $\mathcal{R}$ by generating $|\mathcal{R}|$ more. In each iteration, it utilizes the submodular-cost covering algorithm to find the candidate set $\hat{S}$ (Line 5) and check whether we have sufficient statistical evidence to achieve a good solution by checking the *stopping condition* (Line 11). The stopping condition plays a decisive roles in both theoretical solution quality and the complexity of the algorithm. The condition in SISI is derived from the results of optimal sampling for Monte-Carlo estimation studied in [26]. In the next section, we will prove that with this stopping condition, SISI returns an $\Delta\frac{2}{(1-\epsilon)^2}$-approximate solution with probability of at least $(1-\delta)$, where ϵ, δ are given as inputs. The check in Lines 8-10 is to guarantee ϵ small enough and described in Sec. 4.4. At the end of the algorithm, SISI performs a post-optimization of $\hat{S}$ which incrementally removes nodes in $\hat{S}$ if that improves the objective function.

4.4 Algorithm Analysis

We will analyze the approximation guarantee and time complexity of SISI algorithm. In short, we prove that SISI returns an $\Delta\frac{2}{(1-\epsilon)^2}$-approximate solution. In the sequel, we will present the time complexity of SISI.

4.4.1 Approximation Guarantee

To prove the approximation guarantee of SISI, we show two intermediate results: 1) with $\frac{n\Lambda}{\mathcal{E}[D(\hat{S})]}$ RR sets where $\hat{S}$ is the solution returned by SISI, $\mathcal{E}[D(\hat{S})] = \mathcal{E}[D(\hat{S}, \mathcal{M}, V_I)]$ for short since the $\mathcal{M}, V_I$ are fixed, all the sets $S \subset V_I$ are well approximated from $\mathcal{R}$ with high probability (Lem. 52) and 2) the actual number of RR sets generated in SISI is greater than $\frac{n\Lambda}{\mathcal{E}[D(\hat{S})]}$ with high probability (Lem. 53). Then, combine these results and the property of submodular-cost covering, we obtain the approximation factor in Theo. 24.

Denote $D_{\mathcal{R}}(S) = \frac{n}{|\mathcal{R}|}(|\mathcal{R}^{-}_{Blue}(S)|+|\mathcal{R}^{+}_{Red}(S)|)$, which is an approximation of $\mathcal{E}[D(S)]$, achieved from the collection of RR sets $\mathcal{R}$. The following lemma states the approximation quality of a set $S \subseteq V_I$. We assume that $\mathcal{E}[D(\hat{S})] \neq 0, \forall \hat{S} \subset V_I$ since the case of equaling 0 only happens if V_I is a disconnected clique with edge weights being all 1 and then, every set $S \in V_I$ are exactly identical. In that case, the sources can be any set of nodes and are intractable to identify.

Lemma 52. *If we have $T^* = \frac{n\Lambda}{\mathcal{E}[D(\hat{S})]}$ RR sets where $\hat{S}$ is the solution returned by SISI, then for a set $S \subseteq V_I$,*

$$\Pr[|D_{\mathcal{R}}(S) - \mathcal{E}[D(S)]| \geq \epsilon\sqrt{\mathcal{E}[D(S)] \cdot \mathcal{E}[D(\hat{S})]}] \leq \frac{\delta}{M}$$

where $M = 2^k + 1$ and $k = |V_I|$.

Proof. First, for a subset $S \subseteq V_I$ and a random RR set R_j, recall the binary random variable X_j in Eq. 4.14 that,

$$X_j = \begin{cases} 1 & \text{if } R_j \in \mathcal{R}^{-}_{Blue}(S) \cup \mathcal{R}^{+}_{Red}(S) \\ 0 & \text{otherwise.} \end{cases} \tag{4.24}$$

Thus, the series of RR sets in $\mathcal{R}$ corresponds to a sequence of samples of X_j, denoted by $\{X_j^1, X_j^2, \dots\}$. Intuitively, since the RR are generated independently, the resulted sample

sequence of X_j should also be independent and identically distributed in $[0, 1]$. However, similar to the Stopping Rule Algorithm in [26] that SISI creates a dependency on the samples by stopping the algorithm when some condition is satisfied. SISI jumps to the next round when $|R^-_{Blue}(\hat{S})| + |R^+_{Red}(\hat{S})| \geq \Lambda$ or $\sum_{i=1}^{|\mathcal{R}|} X_j^i \geq \Lambda$ is not met and hence, whether we generate more samples depending on the current set of RR sets. Interestingly, similar to the case of Stopping Rule Algorithm in [26], the sequence $\{X_j^1, X_j^2, \dots\}$ forms a *martingle* and the following results follow from [26]:

Let $X_j^1, X_j^2, ...$ samples according to X_j random variable in the interval $[0, 1]$ with mean μ_{X_j} and variance $\sigma^2_{X_j}$ form a martingale and $\hat{\mu}_{X_j} = \frac{1}{T}\sum_{i=1}^{T} X_j^i$ be an estimate of μ_{X_j}, for any fixed $T > 0, 0 \geq \epsilon \geq 1$,

$$\Pr[\hat{\mu}_{X_j} \geq (1+\epsilon)\mu_{X_j}] \leq e^{\frac{-T\mu_{X_j}\epsilon^2}{2c}} \tag{4.25}$$

and,

$$\Pr[\hat{\mu}_{X_j} \leq (1-\epsilon)\mu_{X_j}] \leq e^{\frac{-T\mu_{X_j}\epsilon^2}{2c}}. \tag{4.26}$$

Recall that the value of $D_{\mathcal{R}}(S)$ is equivalent to,

$$D_{\mathcal{R}}(S) = \frac{n}{|\mathcal{R}|}\sum_{i=1}^{|\mathcal{R}|} X_j^i \tag{4.27}$$

Denote $\hat{\mu}_S = \frac{1}{|\mathcal{R}|}\sum_{i=1}^{|\mathcal{R}|} X_j^i$ which is an estimate of $\mu_S = \frac{1}{n}\mathcal{E}[D(S)]$, then $T^* = \frac{\Lambda}{\mu_{\hat{S}}}$ and the inequality in Lem. 52 can be rewritten,

$$\Pr[|\hat{\mu}_S - \mu_S| \geq \epsilon\sqrt{\mu_{\hat{S}}\mu_S}] \leq \frac{\delta}{M} \tag{4.28}$$

Now, apply the inequality in Eq. 4.26 on the left side of the above Eq. 4.28, we have,

$$\Pr[\hat{\mu}_S \leq (1-\epsilon\sqrt{\frac{\mu_{\hat{S}}}{\mu_S}})\mu_S] \leq e^{\frac{-T^*\mu_S\epsilon^2\mu_{\hat{S}}}{2c\mu_S}} = e^{-(\ln(2/\delta)+k\ln 2+1)}$$

Since $k \ln 2 + 1 > \ln(2^k + 1)$, we obtain,

$$\Pr[\hat{\mu}_S \leq \mu_S - \epsilon\sqrt{\mu_{\hat{S}}\mu_S}] \leq \frac{\delta}{2(2^k+1)} = \frac{\delta}{2M} \tag{4.29}$$

Similarly, by applying the inequality in Eq. 4.25, we obtain the following,

$$\Pr[\hat{\mu}_S \geq \mu_S + \epsilon\sqrt{\mu_{\hat{S}}\mu_S}] \leq \frac{\delta}{2(2^k+1)} = \frac{\delta}{2M} \tag{4.30}$$

Combining Eq. 4.29 and Eq. 4.30 proves Lem. 52. □

Lem. 52 states that if we have at least $T^* = \frac{n\Lambda}{\mathcal{E}[D(\hat{S})]}$ RR sets then a set $S \subset V_I$ is approximated within an additive error of $\epsilon\sqrt{\mu_{\hat{S}}\mu_S}$ with probability $(1-\frac{\delta}{M})$. As a consequence, the next lemma shows that SISI generates at least T^* RR set, thus the approximation of $S \subseteq V_I$ in SISI is also good.

Lemma 53 (Stopping condition)**.** *The number of RR sets generated by SISI when it stops satisfies,*

$$\Pr[|\mathcal{R}| \leq T^*] \leq \frac{\delta}{M} \tag{4.31}$$

Proof. We also define the random variable X_j, samples $\{X_j^1, X_j^2, \dots\}$ for the set $\hat{S}$ returned by SISI similar to the proof of Lem. 52. Starting from the left-hand side of Eq. 4.31, we manipulate as follows,

$$\Pr[|\mathcal{R}| \leq T^*] = \Pr[\sum_{i=1}^{|\mathcal{R}|} X_j^i \leq \sum_{i=1}^{T^*} X_j^i] \tag{4.32}$$

Since $|R^-_{Blue}(\hat{S})| + |R^+_{Red}(\hat{S})| = \sum_{i=1}^{|\mathcal{R}|} X_j^i$ and SISI stops when $|R^-_{Blue}(\hat{S})| + |R^+_{Red}(\hat{S})| \geq$

Λ, Eq. 4.32 is equivalent to,

$$\Pr[|\mathcal{R}| \leq T^*] \leq \Pr[\Lambda \leq \sum_{i=1}^{T^*} X_j^i] = \Pr[\frac{n}{T^*}\Lambda \leq \frac{n}{T^*}\sum_{i=1}^{T^*} X_j^i]$$
$$= \Pr[\Delta\frac{n}{T^*}\Upsilon(1+\epsilon) \leq \frac{n}{T^*}\sum_{i=1}^{T^*} X_j^i] \tag{4.33}$$

Recall that $T^* = \frac{n\Upsilon}{\mathcal{E}[D(\hat{S})]}$ or $\mathcal{E}[D(\hat{S})] = \frac{n\Upsilon}{T^*}$ and, thus,

$$\Pr[|\mathcal{R}| \leq T^*] \leq \Pr[\mathcal{E}[D(\hat{S})](1+\epsilon) \leq \frac{n}{T^*}\sum_{i=1}^{T^*} X_j^i]$$
$$= \Pr[\mathcal{E}[D(\hat{S})](1+\epsilon) \leq D_{T^*}(\hat{S})] \tag{4.34}$$

From Lem. 52, if we have T^* RR sets, we obtain,

$$\Pr[D_{\mathcal{R}}(S) \geq \mathcal{E}[D(S)] + \epsilon\sqrt{\mathcal{E}[D(S)] \cdot \mathcal{E}[D(\hat{S})]}] \leq \frac{\delta}{M}$$

for set S. Replacing S by $\hat{S}$ gives,

$$\Pr[D_{T^*}(\hat{S}) \geq \mathcal{E}[D(\hat{S})] + \epsilon\mathcal{E}[D(\hat{S})]] \leq \frac{\delta}{M}$$

The left side is exactly the Eq. 4.34 and thus,

$$\Pr[|\mathcal{R}| \leq T^*] \leq \frac{\delta}{M} \tag{4.35}$$

That completes the proof of Lem. 53. □

Based on Lem. 52 and Lem. 53, we are sufficient to prove the $\Delta\frac{2}{(1-\epsilon)^2}$-approximation factor of SISI.

Theorem 24. *Let $OPT = \mathcal{E}[D(S^*)]$ be the optimal value of $\mathcal{E}[D(S)]$ at S^*. SISI returns an $\Delta\frac{2}{(1-\epsilon)^2}$-approximate solution $\hat{S}$ with probability of at least $(1-\delta)$ or,*

$$\Pr[\mathcal{E}[D(\hat{S})] \leq \Delta\frac{2}{(1-\epsilon)^2}OPT] \geq 1-\delta \tag{4.36}$$

Proof. From Lem. 52, we obtain,

$$\Pr[|D_{\mathcal{R}}(S) - \mathcal{E}[D(S)]| \geq \epsilon\sqrt{\mathcal{E}[D(S)] \cdot \mathcal{E}[D(\hat{S})]}] \leq \frac{\delta}{M}$$

for a particular subset $S \subseteq V_I$ if there are at least T^* RR sets. Furthermore, Lem. 53 states that SISI generates at least T^* RR sets with probability at least $\frac{\delta}{M}$. Taking union bound over all subsets $S \subseteq V_I$ (note that there are 2^k such subsets) to the above probability and the probability of SISI generating at least T^* RR sets in Lem. 53, we achieve,

$$\Pr[|D_{\mathcal{R}}(S) - \mathcal{E}[D(S)]| \geq \epsilon\sqrt{\mathcal{E}[D(S)] \cdot \mathcal{E}[D(\hat{S})]}] \leq \delta$$

for every set S. Thus, both

$$D_{\mathcal{R}}(\hat{S}) \geq \mathcal{E}[D(\hat{S})] - \epsilon\mathcal{E}[D(\hat{S})] \tag{4.37}$$

and

$$D_{\mathcal{R}}(S^*) \leq OPT + \epsilon\sqrt{OPT \cdot \mathcal{E}[D(\hat{S})]} \tag{4.38}$$

happen with probability at least $(1 - \delta)$. Plugging $D_{\mathcal{R}}(\hat{S}) \leq \Delta D_{\mathcal{R}}(S^*)$ achieved by submodular-cost covering to Eq. 4.38,

$$D_{\mathcal{R}}(\hat{S}) \leq \Delta(OPT + \epsilon\sqrt{OPT \cdot \mathcal{E}[D(\hat{S})]}) \tag{4.39}$$

then combining with Eq. 4.37 gives,

$$\mathcal{E}[D(\hat{S})] - \epsilon\mathcal{E}[D(\hat{S})] \leq \Delta(OPT + \epsilon\sqrt{OPT \cdot \mathcal{E}[D(\hat{S})]})$$

or

$$\frac{\mathcal{E}[D(\hat{S})]}{OPT} \leq \frac{\Delta}{1 - \epsilon - \epsilon\Delta\sqrt{\frac{OPT}{\mathcal{E}[D(\hat{S})]}}} \tag{4.40}$$

This inequality is valid only when $1 - \epsilon - \epsilon\Delta\sqrt{\frac{OPT}{\mathcal{E}[D(\hat{S})]}} > 0$ which means $\epsilon < 1/(1 + \Delta\sqrt{\frac{OPT}{\mathcal{E}[D(\hat{S})]}})$. Since ϵ is a free parameter, we can choose $\epsilon \leq 1/(1 + \Delta)$ and satisfy the condition. By considering $\sqrt{\frac{\mathcal{E}[D(\hat{S})]}{OPT}}$ as a variable and solve the quadratic inequality with $\epsilon \leq 1/(1 + \Delta)$, we obtain,

$$\frac{\mathcal{E}[D(\hat{S})]}{OPT} \leq \Delta\frac{2}{(1-\epsilon)^2} \tag{4.41}$$

which states the $\Delta\frac{2}{(1-\epsilon)^2}$ approximation factor of SISI and happens with probability at least $(1 - \delta)$. □

4.4.2 Time Complexity

This subsection analyzes the time complexity of SISI. We analyze major procedures of the algorithm: 1) submodular-cost covering algorithm and 2) generating RR sets.

4.4.2.1 Submodular-cost covering algorithm

Recall that the total sizes of the generated RR sets is Λ on the average. Since the algorithm for solving the procedure to solve submodular-cost covering problem keeps doubling the number of RR sets after each round,the total complexity of this procedure is bounded loosely by $O(\Lambda^2)$.

4.4.2.2 Generating RR sets

To determine the time complexity of generating RR sets in SISI, we need analyze the time spent for generating a single RR set (Lemma 50) and the expected number of RR sets. Then multiplying two numbers to get the expected total complexity. The following lemma states the complexity results with the proof in our extended version [80].

Lemma 54. *Let E_s be the set of edges connecting nodes in V_I to nodes in $\bar{V}_I$, the complexity*

of generating RR sets in SISI is $O(m\Lambda\Delta/|E_s|)$

Proof. Generating a RR Set. As analyzed in Sec. 4.3, the expected complexity of generating a single RR set is as follows,

$$C'(R_j) = \frac{\Delta m}{n} + \Delta \log(\Delta) \log(1 + \frac{\Delta m}{n^2}) \approx \frac{\Delta m}{n} \tag{4.42}$$

Number of RR set generated. We will find an upper-bound for the number of RR sets generated by SISI. Using Wald's equation [108], and that $\mathcal{E}[|\mathcal{R}|] < \infty$ we have

$$\mathcal{E}[|\mathcal{R}|]\mu_{\hat{S}} = \Lambda \tag{4.43}$$

Thus,

$$\mathcal{E}[|\mathcal{R}|] = \frac{\Lambda}{\mu_{\hat{S}}} = \frac{\Lambda n}{\mathcal{E}[D(\hat{S}, \tau, V_I)]} \tag{4.44}$$

Let E_s be the set of edges connecting nodes in V_I to nodes in $\bar{V}_I$, then we have

$$\mathcal{E}[D(\hat{S}, \tau, V_I)] \geq \sum_{(u,v)\in E_s} \big[(1 - \Pr[\hat{S}, u]) + \Pr[\hat{S}, v]\big] \tag{4.45}$$

where $(1 - \Pr[\hat{S}, u])$ is the probability that $u \in V_I$ is not infected and $\Pr[\hat{S}, v]$ is the probability that $v \in V_{\bar{I}}$ is infected. Since v is uninfected and connected with u, if u is infected by $\hat{S}$, then the probability that v gets the infection from u is $\Pr[\hat{S}, v] = \beta \Pr[\hat{S}, u]$. Taking into the probability that u is infected at least 1 step before τ, we obtain $\Pr[\hat{S}, v] \geq \beta \Pr[\hat{S}, u]/(1 - \beta)$ due to the binomial distribution of successes up to τ and $\tau - 1$. Thus,

$$\begin{aligned}\mathcal{E}[D(\hat{S}, \tau, V_I)] &\geq \sum_{(u,v)\in E_s} (1 - \Pr[\hat{S}, u] + \frac{\beta}{1-\beta} \Pr[\hat{S}, u]) \\ &= |E_s| - (1 - \frac{\beta}{1-\beta}) \sum_{(u,v)\in E_s} \Pr[\hat{S}, u] \geq \frac{\beta}{1-\beta}|E_s|\end{aligned} \tag{4.46}$$

Combining this result with Eq. 4.44, we obtain,

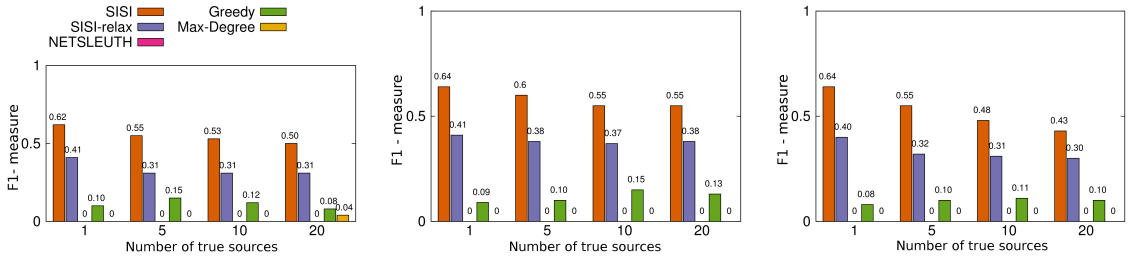
$$\mathcal{E}[|\mathcal{R}|] \leq \frac{(1-\beta)\Lambda n}{\beta |E_s|} \tag{4.47}$$

From Eq. 4.42 and Eq. 4.47, we obtains the complexity of generating RR sets. □

Therefore, the overall complexity of SISI is followed by the subsequent theorem.

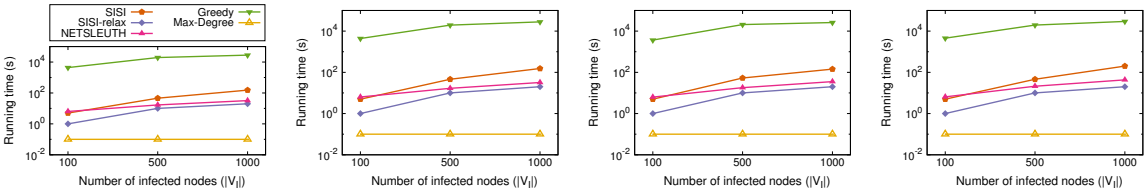
Theorem 25. *Let E_s be the set of edges connecting nodes in V_I to $\bar{V}_I$, SISI has $O(m\Delta\Lambda/|E_s| + \Lambda^2)$ time complexity.*

From Theo. 25, we see that the complexity depends on the number of connections from infected set to the outside world $|E_s|$. That is if there are many infected nodes connected to uninfected nodes, it is easier for SISI to find the sources and vice versus, if only few such connections, SISI requires more time.



(a) $|V_I| = 100$ **(b)** $|V_I| = 500$ **(c)** $|V_I| = 1000$

Fig. 33.: F1-measure scores of different algorithms. Higher is better.



(a) #sources = 1 **(b)** #sources = 5 **(c)** #sources = 10 **(d)** #sources = 20

Fig. 34.: Runtime of the tested algorithms

4.5 Experiments

In this section, we study the empirical performance of SISI and compare it with the current state-of-the-art methods under the popular SI and IC infection models. We show that SISI outperform the others in terms of detection quality, revealing major of the infection sources. In contrast, the other methods rarely find any true source of the infection.

$\lvert V_I \rvert$		100				500				1000			
#sources		1	5	10	20	1	5	10	20	1	5	10	20
	Ground-truth	**205**	**173**	**156**	**134**	**1006**	**945**	**938**	**767**	**2026**	**1835**	**1945**	**1520**
Symmetric	SISI	**211**	**181**	**168**	**142**	**1013**	**962**	**971**	**792**	**2049**	**1873**	**1959**	**1541**
Difference	SISI-relax	246	215	218	202	1141	993	1084	854	2179	1903	2012	1696
(smaller is	NETSLEUTH	294	273	280	247	1258	1147	1193	971	2297	2095	2248	1751
better)	Greedy	261	226	231	219	1152	1015	1067	914	2218	2214	2124	1707
	Max-Degree	281	325	418	387	1195	1091	1206	1105	2221	2167	2182	1876
	Ground-truth	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Jaccard	SISI	**0.92**	**0.98**	**0.96**	**0.97**	**0.99**	**0.95**	**0.82**	**0.94**	**0.96**	**0.97**	**0.97**	**0.95**
Distance	SISI-relax	0.76	0.72	0.65	0.71	0.81	0.79	0.72	0.89	0.86	0.68	0.72	0.73
(larger is	NETSLEUTH	0.21	0.24	0.31	0.37	0.16	0.29	0.26	0.41	0.20	0.17	0.18	0.21
better)	Greedy	0.32	0.19	0.39	0.35	0.26	0.28	0.34	0.37	0.22	0.26	0.21	0.19
	Max-Degree	0.32	0.35	0.24	0.29	0.24	0.27	0.26	0.18	0.14	0.16	0.17	0.12

Table 25.: Comparison on Symmetric Difference and Jaccard-based Distance of different methods.

4.5.1 Experimental Settings

4.5.1.1 Algorithms compared

Under the SI model, we compare three groups of methods:

- SISI, a relaxed version of SISI, termed SISI-relax, in which we relax the approximation guarantee of SISI by replacing $(k \ln 2)$ in Υ by a smaller constant $\ln(2 \times k)$ and the natural naive Greedy algorithm which iteratively selects one node at a time that commits the largest marginal decrease of symmetric difference. The purpose of designing SISI-relax is to test the empirical performance changes if we have fewer

RR sets.

- NETSLEUTH [97] which is the existing best algorithm in general graphs however it fails to provide any guarantee on solution quality.

- Max-Degree based method which ranks node degrees and iteratively selects nodes with highest degree until increasing the symmetric difference as the solution.

Under the IC model, we compare SISI with k-effector [67] and the naive Max-Degree algorithm on IC model.

For SISI and SISI-relax, we set the parameters $\epsilon = 0.1, \delta = 0.01$. For k-effector, k is set to the number of true sources.

4.5.1.2 Quality measures

To evaluate the solution quality, we adopt three measures:

- Symmetric difference ($\mathcal{E}[D(S, \tau, V_I)]$) which is separately calculated with high accuracy ($\epsilon = 0.01, \delta = 0.001$) through generating random RR sets as in Subsection 4.3.1.

- Jaccard distance based Q_{JD} [97]:

$$Q_{JD}(S) = \frac{\mathcal{E}[JD_S(V_I)]}{\mathcal{E}[JD_{S^*}(V_I)]} \tag{4.48}$$

where $\mathcal{E}[JD_S(V_I)]$ is the average Jaccard distance of S w.r.t. V_I and computed by generating many (10000 in our experiments) infection simulations from S and averaging over the Jaccard similarities between the infected sets and V_I. S^* contains the true sources.

- F1-measure:

$$PR(S) = \frac{|S \cap \{\text{true sources}\}|}{2|S|} + \frac{|S \cap \{\text{true sources}\}|}{2|\{\text{true sources}\}|}$$

This accurately captures our ultimate goal of ISI problem: finding both the true sources and the correct number of sources. We also define true source detection rate (%) as $100\frac{|S \cap \{\text{true sources}\}|}{|\{\text{true sources}\}|}$.

Both $Q_{JD}(S)$ and $PR(S)$ are ranging in $[0, 1]$ and larger is better. $\mathcal{E}[D(S, \tau, V_I)]$ is non-negative and smaller is better.

4.5.1.3 Datasets

For experimental purposes, we select a moderate-size real network - NetHEPT with 15233 nodes and 62796 edges that is actually the largest dataset ever tested on ISI problem. We comprehensively carry experiments on NetHEPT with various numbers of sources $\{1, 5, 10, 20\}$, chosen uniformly random, and the propagation time τ is chosen so that the infection sizes reach (or exceed) predefined values in the set $\{100, 500, 1000\}$. For each pair of the two values, we generated 10 random test cases with $\beta = 0.05$ and then ran each method on these random tests and took the average of each quality measure over 10 such results.

4.5.1.4 Testing Environments

We implement SISI, SISI-relax, Greedy and Max-Degree methods in C++, NETSLEUTH is in Matlab code and obtained from the authors of [97]. We experiment on a Linux machine with an 8 core 2.2 GHz CPU and 100GB RAM.

4.5.2 Experiments on real network and SI model

Comparing solution quality. The solution quality measured are the true infection sources discovery rate, symmetric difference (our objective) and Jaccard-based distance [97]

True source discovery. Fig. 33 reports the F1-measure scores of the tested algorithms. Note that this score has not been used in previous works [97, 78] since previous methods can only find nodes that are within few hops from the sources, but not the sources themselves. As shown in the figure, SISI and SISI-relax have the best performance. More than 50% of the true sources was discovered by SISI and 35% by SISI-relax that exquisitely surpass NETSLEUTH, Max-Degree with 0% and Greedy with roughly 10%.

#src	SISI	SISI-relax	NETS.	Greedy	Max-Degree
1	**91.4**	84.2	0	14.5	0
5	**79.7**	53.9	0	15.2	0
10	**74.1**	52.3	0	11.8	0
20	**77.3**	56.5	0	9.6	0

Table 26.: True sources detected (%) with $|V_I| = 1000$.

We also present the true source detected rates of different methods in Tab. 26 since this is an important aspect (positive rate) of ISI problem. The table shows accurate detection of SISI and SISI-relax. More than 70% and 50% of true sources are identified by SISI and SISI-relax respectively while NETSLEUTH and Max-Degree cannot detect any source.

Symmetric difference. Tab. 25 shows the $\mathcal{E}[D(S, \tau, V_I)]$ values where S is the returned solution of each algorithm with various number of true sources and sizes of infection cascades. In all the cases, SISI largely outperforms the other methods and obtains very close

values to the true sources. The superiority of SISI against the SISI-relax and Greedy confirms the good solution guarantee of SISI. NETSLEUTH and Max-Degree optimize different criteria, i.e., description length (MDL) and node degree, and thus show poor performance in terms of symmetric difference. SISI-relax is consistently the second best method and preserves very well the performance of SISI.

Jaccard distance. We use $Q_{JD}(S)$ as in [97] to evaluate the algorithms and plot the results in Tab. 25. In this case, the closer value of $Q_{JD}(S)$ to 1 indicates better solution. In terms of $Q_{JD}(S)$, we observe the similar phenomena as measured by symmetric difference that SISI achieve drastically better solution than the others and the results of SISI-relax approach those of SISI very well with much fewer RR sets.

Comparing running time. Fig. 34 illustrates the running time of the algorithms in the previous experiments. We see that SISI is slower than NETSLEUTH and SISI-relax but the differences are minor while it provides by far better accuracy than other algorithms. SISI-relax obtains possibly the best balance among all: faster than NETSLEUTH and providing good solution quality as shown previously.

4.5.3 Experiments on the IC model

Set up. We compare SISI with the dynamic programming algorithm, temporarily called k-effector, in [67] when the infection process follows the IC model. Similar to other experiments, we simulate the infection process under the IC model with 4 different numbers of sources, i.e., 1, 5, 10, 20 and run SISI and k-effector on the resulting cascades. For each setting, we carry 10 simulations and report the average results. Note that the solution for k-effector in [67] requires the number of sources as an additional input parameter and for simplicity, we provide the true number of sources used in the simulation processes. SISI, however, do not require this information. We report the results in Table 27.

Results. It is clear from Table 27 that SISI massively outperforms k-effector in terms

#src	Symmetric Difference			F1-measure		
	SISI	k-effector	Max-Deg.	SISI	k-effector	Max-Deg.
1	**6.6**	18.4	42.3	**0.57**	0	0.02
5	**55.1**	103.4	176.9	**0.53**	0.02	0
10	**25.2**	72.6	154.1	**0.49**	0.03	0
20	**203.7**	295.2	384.7	**0.52**	0.05	0.03

Table 27.: Comparison under the IC model.

of both symmetric difference and true source recovering ability. In summary, for any value of the number of true sources k, SISI always returns solution with symmetric difference equal half of the one returned by k-effector. In terms of true source discovery ability, while k-effector almost detects none of the true sources, SISI consistently achieves the F1-measure of at least 50%.

4.6 Discussion and Conclusion

We present SISI the first approximation algorithm for multiple source detection in general graphs which also works very well in practice. The algorithm can be extended to several other diffusion models and settings with little modification on the sampling procedure as outlined below.

Incomplete Observation [36, 53]. In many cases, we can only observe the states (infected/not infected) for a subset $O \subsetneq V$ of nodes in the network. In those cases, we need to modify the Fast TRIS sampling Algorithm in Line 1 and pick a node u uniformly in O (instead of V) and allow the sources to be from V_I or unknown state nodes.

However, the SISI cannot be directly adapted to non-progessive models in which a node can switch from an infected state into uninfected state. Thus approximation algorithm

for source detection in non-progressive models leaves an open question and is among our future work.

CHAPTER 5

COMMUNITY DETECTION

Community detection which partitions a network into possibly overlapping groups of nodes is a large topic in network science and found numerous applications in biology, e.g., finding function groupings, e-commerce, e.g., product recommendations, and many more. Despite a large research effort from diverse fields, e.g., computer science, computational physics, information theory, the efficiency of finding community structure is still below satisfaction. We propose to boost up this low performance by combining additional sources of data with the traditional network topology. In particular, we propose to detect communities across different networks sharing some portion of common users. The second idea is to combine the available node attribute information with the topology and cluster them together. We combine these ideas with the emerging Non-negative Matrix Factorization technique to propose efficient algorithms. We will sequentially present these works in the subsequent sections.

5.1 Non-negative Matrix Factorization (NMF)

Nonnegative matrix factorization (NMF) was first introduced by Paatero and Tapper and popularized by Lee and Seung [68]. The main idea is to approximate a nonnegative matrix V by the product of two nonnegative matrix factors W and H. Due to natural nonnegative property of the factorization, those works started a massive flow of researches covering a wide range of area, i.e., text mining, spectral data analysis, speech denoising, bioinformatics and many more [59]. Recently in social science, Lin et al. presented MetaFac [73] which uses relational hypergraph representation and tensor factorization. Wang et al. sub-

sequently used NMF algorithm and proposed three NMF solutions [110] for undirected, directed networks and compound networks of different entities (users and movies). Unfortunately, the method cannot be adapted for multiplex OSNs which have single entity type and multiple entities in different networks may refer to a same person.

5.2 Community Detection in Multiplex Social Networks

Summary of contributions:

- We propose and compare two classes of approaches. The first class, named unifying approach, finds a consistent CS in the networks by aggregating multiple accounts of the same users. The second class finds *mostly* consistent CSs in the network using coupling techniques. We also develop specialized NMF-based method for each class.
- We extend the LFR benchmark [66] to create a new benchmark for community detection in multiplex OSNs. The new extension is capable of generating layers with varying node's degree distribution and the fraction between links inside and outside communities.
- We carry intensive experiments on synthesized data. The results suggest that our approaches outperform the naive approach of finding CS in each network separately.

5.2.1 Problem formulation

We model multiplex OSNs as a collection G of graphs. G consists of p layers or p single networks. Layer i is abbreviated by $G_i = (V_i, E_i)$ where V_i and E_i are the set of nodes and the set of edges, respectively in that layer. Note that a node can appear in one or multiple layers. We define set $V = \bigcup_{i=1}^{p} V_i$ and $n = |V|$ - the capacity of set V. Now, we can represent each layer in matrix form: $\mathbf{A}_i$ is an $n \times n$ adjacency matrix of G_i. A three layer OSN is illustrated in Fig. 35.

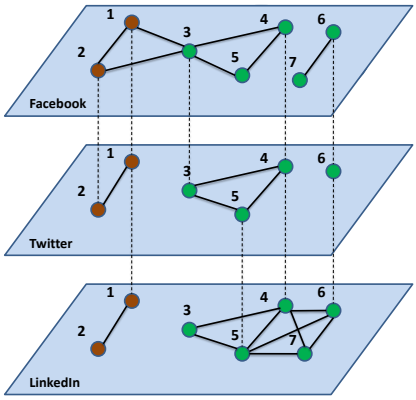


Fig. 35.: A toy example of 7 users participating in three OSNs, namely Facebook, Twitter and LinkedIn. If we analyze each layer separately, node 3 can be grouped with node 1 and 2 or with 4 and 5 in Facebook network. However, with the information from Twitter, we can surely assign node 3 to the same community with nodes 1, 2. From LinkedIn, we obtain one more structural information that nodes 3, 4, 5, 6, 7 should be in the same group.

Assume there exist k communities in layer i. We model the interaction $(A_i)_{uv}$ between nodes u and v in layer i by a mixture model of combined effect due to all the k communities. That is, we approximate $(A_i)_{uv}$ using $(A_i)_{uv} = \sum_{m,l} p_{ml} p_{m\to u} p_{l\to v}$ where p_{ml} is the interaction density between communities m and l, $p_{m\to u}$ and $p_{l\to v}$ are the probabilities that an interaction with communities m and l involves node u and v, respectively. Written in matrix form, we have $\mathbf{A}_i = \mathbf{X}_i \mathbf{S}_i \mathbf{X}_i^T$ where $\mathbf{X}_i$ is a non-negative matrix with $(X_i)_{um} = p_{m\to u}$, $\mathbf{S}_i$ is also a non-negative matrix with $(S_i)_{ml} = p_{ml}$. Our goal is to find the CS which can be represented as a $n \times k$ matrix $\mathbf{X}_i$ for each layer i where each row reflects the community membership for an user. $(X_i)_{um}$ reveals the strength of participation of user u to community m. This representation can be used for either overlapping or disjoint CS. The latter, disjoint CS, is the focus of this paper.

Central assumption. If nodes u, v are in the same community in a layer, they are more likely to belong a community in the other layers. Based on how strictly we enforce this assumption, we derive two classes of approaches:

Unifying approach: We force the instances of an user in different layers to be in the same community by aggregating all the layers into a single network where multiple instances appear as a node in aggregated network.

Coupling approach: We relax the enforcement by using coupling schema. Instead of forcing instances of a node, we suggest them to be in the same community by creating a coupling edges between matching pairs of instances.

5.2.2 Unifying approach

In this section, we present the unifying approach which finds a single CS for all layers. We consider two directions: 1) Convert multiple layers to one layer network and apply existing algorithms and 2) Adapt NMF algorithm on the original networks.

5.2.2.1 Network aggregation

To apply existing algorithms for multiplex OSNs, we need to: 1) Aggregate all layers into a single network G_c, 2) Apply existing CS algorithms, e.g., Louvain [10], Infomap [100], to find CS in G_c, 3) Project the found CS back onto each layer to find their CSs.

Given a multiple layer network G as defined in problem formulation section, the aggregated network [59] is denoted as $G_c = (V, E_c)$ where $E_c = (E_1 \cup E_2 \cup ... \cup E_p)$ and $E_1, E_2, ..., E_p$ are edge sets in the layers.

Now, we can obviously use algorithms for single networks on aggregated networks. However, the aggregation discloses itself several shortcomings, i.e., the edge types in the layers may be different from each other or some layers are weighted but the others are unweighted. Those characteristics make it difficult to aggregate the layers. Therefore, we propose the NMF-based algorithm on the original multiple layer networks.

5.2.2.2 NMF-based algorithm on the original networks

We present NMF-based algorithms for both directed and undirected networks.

Directed networks. We attempt to find a community membership matrix $\mathbf{X}$ that agrees with the structures of the given networks. Specifically, we want to minimize the sum of difference between $\mathbf{XSX}^T$ and the matrices A_i, $i = 1..p$. Here S shows the connectivity between communities. Then, the community detection problem can be cast to a nonnegative matrix factorization problem. Therefore, we obtain the following objective function

$$\min_{\mathbf{X}\geq 0,\mathbf{S}\geq 0} \sum_{i=1}^{p} d(\mathbf{A}_i \| \mathbf{XSX}^T), \tag{5.1}$$

where $d(\mathbf{A}\|\mathbf{B})$ is the measure for difference between two matrices. In the literature, we have seen two most popular and well-studied measures, the former is called the square of the Euclidean distance [68]

$$\|\mathbf{A} - \mathbf{B}\|_F^2 = \sum_{i,j} (A_{ij} - B_{ij})^2. \tag{5.2}$$

Similarly, the second measure named Kullback-Leibler divergence (KL-divergence) [68] of $\mathbf{A}$ from $\mathbf{B}$ is defined as

$$D(\mathbf{A}\|\mathbf{B}) = \sum_{i,j} (A_{ij} \log \frac{A_{ij}}{B_{ij}} - A_{ij} + B_{ij}). \tag{5.3}$$

Using KL-divergence. The cost function using KL-divergence is as follows

$$\min_{\mathbf{X}\geq 0,\mathbf{S}\geq 0} L = \sum_{i=1}^{p} D(\mathbf{A}_i \| \mathbf{XSX}^T). \tag{5.4}$$

We derive the update rules following the framework in [68]

$$X_{jk} = X_{jk}\left(\frac{\sum_{i=1}^{p}\sum_{l}(A_i)_{lj}(\mathbf{XS})_{lk}/(\mathbf{XSX}^T)_{lj}}{p\left(\sum_t((\mathbf{XS})_{tk}+(\mathbf{SX}^T)_{kt})\right)} + \frac{\sum_{i=1}^{p}\sum_{l}(A_i)_{jl}(\mathbf{SX}^T)_{kl}/(\mathbf{XSX}^T)_{jl}}{p\left(\sum_t((\mathbf{XS})_{tk}+(\mathbf{SX}^T)_{kt})\right)}\right), \tag{5.5}$$

$$S_{jk} = S_{jk}\left(\frac{\sum_{i=1}^{p}\sum_{s,t}(A_i)_{st}X_{sj}X_{tk}/(\mathbf{XSX}^T)_{st}}{p\left(\sum_{st}X_{sj}X_{tk}\right)}\right). \tag{5.6}$$

Algorithm 24: NMF-based algorithm for directed networks using KL-divergence

Input: Adjacency matrices $\{\mathbf{A}_i | i = 1..p\}$, max iterations T

Output: Membership matrix $\mathbf{X}$

Assign X_{ij}, S_{ij} (uniformly) random values in [0,1]

repeat

Update X_{jk} following Eq. 5.5

Update S_{jk} following Eq. 5.6

until *Convergence or after T iterations*;

For each row i, argmax$_j\{X_{ij}\}$ is the community that node i is assigned to

Return the list of communities corresponding to the nodes

Alg. 25 depicts NMF-based algorithms for directed networks using KL-divergence in unifying approach. The main segment is the updating procedure where X_{jk} and S_{jk} gets updated in each iteration until convergence or after T updates. Row i of matrix $\mathbf{X}$ shows the participation of user i in all the communities. Therefore, we assign user i to the community corresponding to the largest value in $\mathbf{X}_i$, if there several such communities, choose the first one.

To find the number of communities k, we adopt one of the most popular approaches used in [115]. We choose k at which the modularity function Q achieves the maximum

(see [82] for more details).

Theorem 26. *The value of the objective function in Eq. 4 is non-increasing and converged to an local minimum under the updates rules in Eq. 5.5 and Eq. 5.6.*

Proof. We will show that $\mathbf{X}$ and $\mathbf{S}$ converge and the convergence point is a local minimum.

Convergence: To prove the convergence, we need to find the auxiliary functions for $\mathbf{X}$ and $\mathbf{S}$ that lead to the update rules. We define the following auxiliary functions $Q(\mathbf{X}, \tilde{\mathbf{X}})$ and $Q(\mathbf{S}, \tilde{\mathbf{S}})$:

$$Q(\mathbf{X}, \tilde{\mathbf{X}}) = \sum_{i=1}^{p} \Big(\sum_{jk} (A_i)_{jk} \left(\log(A_i)_{jk} - 1\right) + \frac{1}{2} \sum_{jk} \Big((\mathbf{Y}\mathbf{S}\tilde{\mathbf{X}}^T)_{jk} + (\tilde{\mathbf{X}}\mathbf{S}\mathbf{Y}^T)_{jk} \Big) \Big)$$
$$- \sum_{i=1}^{p} \Big(\sum_{jk} (A_i)_{jk} \sum_{uv} \eta_{jkuv} \left(\log(X_{jv} S_{vu} X_{ku}) - \log(\eta_{jkuv})\right) \Big),$$

$$Q(\mathbf{S}, \tilde{\mathbf{S}}) = \sum_{i=1}^{p} \Big(\sum_{jk} (A_i)_{jk} \left(\log(A_i)_{jk} - 1\right) + \sum_{jk} (\mathbf{X}\mathbf{S}\mathbf{X}^T)_{jk} \Big)$$
$$- \sum_{i=1}^{p} \Big(\sum_{jk} (A_i)_{jk} \sum_{uv} \beta_{jkuv} \left(\log(X_{jv} S_{vu} X_{ku}) - \log(\beta_{jkuv})\right) \Big).$$

where

$$\beta_{jkuv} = \frac{X_{jv} \tilde{S}_{vu} X_{ku}}{\sum_{s,t} X_{jt} \tilde{S}_{ts} X_{ks}}, \eta_{jkuv} = \frac{X_{jv} S_{vu} \tilde{X}_{ku}}{\sum_{s,t} X_{jt} S_{ts} \tilde{X}_{ks}}, Y_{ij} = \frac{X_{ij}}{\tilde{X}_{ij}}.$$

Then, we only need to verify that $Q(\mathbf{X}, \tilde{\mathbf{X}}) \geq F(\mathbf{X})$ and $Q(\mathbf{S}, \tilde{\mathbf{S}}) \geq F(\mathbf{S})$. The second summation of these inequalities are equivalent (with substitution of β_{jkuv} to η_{jkuv}) to

$$-\log(\sum_{u,v} \beta_{jkuv} \frac{X_{jv} S_{vu} X_{ku}}{\beta_{jkuv}}) \leq -\sum_{u,v} \beta_{jkuv} \log(\frac{X_{jv} S_{vu} X_{ku}}{\beta_{jkuv}}),$$

which holds due to Jensen's inequality [68] and the convexity of logarithmic function. So, we can verify $Q(\mathbf{S}, \tilde{\mathbf{S}}) \geq F(\mathbf{S})$.

We also have $\frac{1}{2}(\mathbf{Y}\mathbf{S}\tilde{\mathbf{X}}^T)_{jk} + \frac{1}{2}(\tilde{\mathbf{X}}\mathbf{S}\mathbf{Y}^T)_{jk} \geq (\mathbf{X}\mathbf{S}\mathbf{X}^T)_{jk}$ and that makes the inequality $Q(\mathbf{X}, \tilde{\mathbf{X}}) \geq F(\mathbf{X})$ satisfied. Then, taking the derivatives of $Q(\mathbf{X}, \tilde{\mathbf{X}})$ and $Q(\mathbf{S}, \tilde{\mathbf{S}})$, we get the update rules.

Local minimum: We need to point out that the update rules satisfy the KKT slackness conditions [68].

Introducing the Lagrangian multipliers α_{jk} and β_{jk} to the loss function L , we have

$$J = \sum_{i=1}^{p} D(\mathbf{A}_i \| \mathbf{X}\mathbf{S}\mathbf{X}^T) = \sum_{i=1}^{p} \sum_{j,k} ((A_i)_{jk} \log \frac{(A_i)_{jk}}{(\mathbf{X}\mathbf{S}\mathbf{X}^T)_{jk}} - (A_i)_{jk} + (\mathbf{X}\mathbf{S}\mathbf{X}^T)_{jk}) + \sum_{j,k} \alpha_{jk}\mathbf{X}_{jk} + \sum_{j,k} \beta_{jk}\mathbf{S}_{jk}.$$

Take the derivatives of J in terms of X_{jk} and S_{jk}

$$\frac{\delta J}{\delta X_{jk}} = \sum_{i=1}^{p} \Big(- \sum_{l} \frac{(A_i)_{lj}(\mathbf{X}\mathbf{S})_{lk}}{(\mathbf{X}\mathbf{S}\mathbf{X}^T)_{lj}} - \sum_{l} \frac{(A_i)_{jl}(\mathbf{X}\mathbf{S})_{kl}}{(\mathbf{X}\mathbf{S}\mathbf{X}^T)_{jl}} + \sum_{t} ((\mathbf{X}\mathbf{S})_{tk} + (\mathbf{S}\mathbf{X}^T)_{kt}) \Big) - \alpha_{jk},$$

$$\frac{\delta J}{\delta S_{jk}} = \sum_{i=1}^{p} \Big(- \sum_{s,t} \frac{(A_i)_{st} X_{sj} X_{tk}}{(\mathbf{X}\mathbf{S}\mathbf{X}^T)_{st}} + \sum_{st} X_{sj} X_{tk} \Big) - \beta_{jk}.$$

Following the KKT slackness conditions, we get

$$\frac{\delta J}{\delta X_{jk}} = \sum_{i=1}^{p} \left(- \sum_{l} \frac{(A_i)_{lj}(\mathbf{X}\mathbf{S})_{lk}}{(\mathbf{X}\mathbf{S}\mathbf{X}^T)_{lj}} - \sum_{l} \frac{(A_i)_{jl}(\mathbf{X}\mathbf{S})_{kl}}{(\mathbf{X}\mathbf{S}\mathbf{X}^T)_{jl}} + \sum_{t} ((\mathbf{X}\mathbf{S})_{tk} + (\mathbf{S}\mathbf{X}^T)_{kt}) - \alpha_{jk} \right) X_{jk} = 0,$$

$$\frac{\delta J}{\delta S_{jk}} = \sum_{i=1}^{p} \left(- \sum_{s,t} \frac{(A_i)_{st} X_{sj} X_{tk}}{(\mathbf{X}\mathbf{S}\mathbf{X}^T)_{st}} + \sum_{st} X_{sj} X_{tk} - \beta_{jk} \right) S_{jk} = 0.$$

Then, we can see that the update rules satisfy the above conditions or $\mathbf{X}$ and $\mathbf{S}$ will converge to a local minimum. Since matrices $\mathbf{A}_i$, $\mathbf{S}$, and $\mathbf{X}$ are all nonnegative during the updating process, the final $\mathbf{X}$ and $\mathbf{S}$ will also be nonnegative.

□

Using Euclidean distance We can also use the Euclidean distance and obtain the cor-

responding cost function

$$\min_{X\geq 0, S\geq 0} (\sum_{i=1}^{p} \|\mathbf{A}_i - \mathbf{X}\mathbf{S}\mathbf{X}^T\|_F^2). \tag{5.7}$$

Theorem 27. *The value of the objective function in Eq. 5.7 is non-increasing and converged to an local minimum under the following updates rules*

$$X_{jk} = X_{jk} \left(\frac{\sum_{i=1}^{p} [\boldsymbol{A}_i^T \boldsymbol{X}\boldsymbol{S} + \boldsymbol{A}_i \boldsymbol{X}\boldsymbol{S}^T]_{jk}}{p[\boldsymbol{X}\boldsymbol{S}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{S}^T + \boldsymbol{X}\boldsymbol{S}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{S}]_{jk}} \right)^{1/4}, \tag{5.8}$$

$$S_{jk} = S_{jk} \left(\frac{\sum_{i=1}^{p} [\boldsymbol{X}^T \boldsymbol{A}_i \boldsymbol{X}]_{jk}}{p[\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{S}\boldsymbol{X}^T\boldsymbol{S}]_{jk}} \right). \tag{5.9}$$

We omit the proof of the Theorem 2 due to space limit.

Undirected networks. The problem in undirected networks is actually a special case of that problem in directed network. The adjacency matrix for each layer is symmetric, we, therefore, factorize $\mathbf{A}_i = \mathbf{X}\mathbf{X}^T$ and then formulate the resulting problem as:

Using KL-divergence version

$$\min_{\mathbf{X}\geq 0} \sum_{i=1}^{p} D(\mathbf{A}_i \| \mathbf{X}\mathbf{X}^T) \tag{5.10}$$

with the simplified update rule only for matrix $\mathbf{X}$

$$X_{jk} = X_{jk} \left(\frac{\sum_{i=1}^{p} \sum_l (A_i)_{lj} X_{lk} / (\mathbf{X}\mathbf{X}^T)_{lj}}{p\left(\sum_t X_{tk}\right)} \right). \tag{5.11}$$

Using Euclidean distance version

$$\min_{X\geq 0, S\geq 0} \sum_{i=1}^{p} \|\mathbf{A}_i - \mathbf{X}\mathbf{X}^T\|_F^2 \tag{5.12}$$

with the corresponding update rule

$$X_{jk} = X_{jk} \left(\frac{\sum_{i=1}^{p} [\mathbf{A}_i^T \mathbf{X}]_{jk}}{p[\mathbf{X}\mathbf{X}^T\mathbf{X}]_{jk}} \right)^{1/4}. \tag{5.13}$$

5.2.3 Coupling approach

5.2.3.1 Coupling techniques

To suggest instances of a node in multiple networks being in the same community, we create coupling edges between them and construct coupled networks. We investigate four basic coupling schema [59], namely diagonal, categorical, star and full couplings. In the article [90], the authors apply two variants of star and aggregated couplings in the context of least cost influence problem. They named lossless and lossy coupling schema for two variants, the former is constructed by creating gateway vertices as an intermediate layer similar to star coupling, whereas aggregation is used for the latter. However, they made some modifications to adapt in diffusion process, i.e. defining weights and thresholds.

Diagonal coupling [59]: Given two layers G_i and G_{i+1}, if two nodes $u \in G_i$ and $v \in G_{i+1}$ belong to an entity, there exists a coupling edges (u, v).

Categorical coupling [59]: For any pair of layers G_i and G_j, if two nodes $u \in G_i$ and $v \in G_j$ belong to an entity, there exists a coupling edges (u, v).

Star coupling [59]: We add another intermediate layer $G_{p+1} = (V, E')$ in which E' is empty and we connect each node in G_{p+1} to all nodes belonging to the same entity in all other layers.

Full coupling [59]: For two adjacent layers $G_i = (V, E_i)$ and $G_{i+1} = (V, E_{i+1})$, if there is an edge $(u, v) \in (E_i \cup E_{i+1})$, we have coupling edges (u_i, v_{i+1}) where $u_i \in G_i, v_{i+1} \in G_{i+1}$ and (u_{i+1}, v_i) where $u_{i+1} \in G_{i+1}, v_i \in G_i$.

With the knowledge of coupling, besides matrices $\mathbf{A}_i$ with $i = 1..p$, we introduce matrices $\mathbf{A}_{ij}$ representing coupling connections between layers i and j.

5.2.3.2 Directed networks

To find CS in multiplex OSNs using coupling approach, we do: 1) build coupled network by a coupling scheme, then 2) apply a CS detection algorithm on coupled network and 3) extract CS.

After constructing the coupled networks, we can simply apply existing algorithms for single networks that have an apparent advantage of requiring less effort. Let us take NMF as an example with the cost function

$$\min_{X\geq 0, S\geq 0}(\mathbf{A}\|\mathbf{X}\mathbf{S}\mathbf{X}^T), \tag{5.14}$$

where $\mathbf{A}$ is the giant $(n\times p)\times(n\times p)$ adjacency matrix for the coupled network. However, we observe that matrix $\mathbf{A}$ is very sparse because it only contains A_i and A_{ij} as its' building blocks. Therefore, we can take advantage of that structure and have the following NMF problem under KL-divergence

$$\min_{\mathbf{X}_i\geq 0\ \forall i, \mathbf{S}\geq 0}\sum_i D(\mathbf{A}_i\|\mathbf{X}_i\mathbf{S}\mathbf{X}_i^T)+\sum_{i,j}D'(\mathbf{A}_{ij}\|\mathbf{X}_i\mathbf{S}\mathbf{X}_j^T), \tag{5.15}$$

where $D'(\mathbf{A}\|\mathbf{B}) = \sum_{A_{st}\neq 0}\left(A_{st}\log\frac{A_{st}}{B_{st}}-A_{st}+B_{st}\right)$. The first summation corresponds to each layer separately, whereas the second one takes into account the couplings.

Theorem 28. *The value of the objective function in Eq. 5.15 is non-increasing and con-*

verged to an local minimum under the following updates rules

$$
\begin{aligned}
(X_i)_{uv} = (X_i)_{uv} \Bigg(& \sum_k \Big(\frac{(A_i)_{uk}}{(\boldsymbol{X}_i \boldsymbol{S} \boldsymbol{X}_i^T)_{uk}} (\boldsymbol{S} \boldsymbol{X}_i^T)_{vk} \\
& + \frac{(A_i)_{ku}}{(\boldsymbol{X}_i \boldsymbol{S} \boldsymbol{X}_i^T)_{ku}} (\boldsymbol{X}_i \boldsymbol{S})_{kv} \Big) + \sum_j \Big(\frac{(A_{ij})_{uu}}{(\boldsymbol{X}_i \boldsymbol{S} \boldsymbol{X}_j^T)_{uu}} (\boldsymbol{S} \boldsymbol{X}_j^T)_{vu} \\
& + \frac{(A_{ji})_{uu}}{(\boldsymbol{X}_j \boldsymbol{S} \boldsymbol{X}_i^T)_{uu}} (\boldsymbol{X}_j \boldsymbol{S})_{uv} \Big) \Bigg) \Big/ \Bigg(\sum_k \Big((\boldsymbol{S} \boldsymbol{X}_i)_{vk} + (\boldsymbol{X}_i \boldsymbol{S})_{kv} \Big) \\
& + \sum_j \Big((\boldsymbol{S} \boldsymbol{X}_j^T)_{vu} + (\boldsymbol{X}_j \boldsymbol{S})_{uv} \Big) \Bigg),
\end{aligned} \tag{5.16}
$$

$$
\begin{aligned}
S_{uv} = S_{uv} \Bigg(& \frac{\sum_i \sum_{p,q} (A_i)_{pq} (X_i)_{pu} (X_i)_{qv} / (\boldsymbol{X}_i \boldsymbol{S} \boldsymbol{X}_i^T)_{pq}}{\sum_i \sum_{pq} (X_i)_{pu} (X_i)_{qv} + \sum_{i,j} \sum_k (X_i)_{ku} (X_i)_{kv}} \\
& + \frac{\sum_{i,j} \sum_k (A_{ij})_{kk} (X_i)_{ku} (X_i)_{kv} / (\boldsymbol{X}_i \boldsymbol{S} \boldsymbol{X}_j^T)_{kk}}{\sum_i \sum_{pq} (X_i)_{pu} (X_i)_{qv} + \sum_{i,j} \sum_k (X_i)_{ku} (X_i)_{kv}} \Bigg).
\end{aligned} \tag{5.17}
$$

The proof the theorem are highly similar to those of unifying approach and is skipped to save space.

5.2.3.3 Undirected networks

The problem for undirected networks can be formulated as

$$
\min_{\mathbf{X}_i \geq 0 \, \forall i, \mathbf{S} \geq 0} \sum_i D(\mathbf{A}_i \| \mathbf{X}_i \mathbf{X}_i^T) + \sum_{i,j} D'(\mathbf{A}_{ij} \| \mathbf{X}_i \mathbf{X}_j^T), \tag{5.18}
$$

which is also a special case of problem for directed networks when S is an identity matrix. We, therefore, treat them in the same way and obtain the below update rule

$$
\begin{aligned}
(X_i)_{uv} = & (X_i)_{uv} \Bigg(\frac{\sum_k (A_i)_{uk} (X_i)_{kv} / (\mathbf{X}_i \mathbf{X}_i^T)_{ku}}{\sum_k (X_i)_{kv} + \sum_j (X_j)_{uv}} \\
& + \frac{\sum_j (A_{ji})_{uu} (X_j)_{uv} / (\mathbf{X}_j \mathbf{X}_i^T)_{uu}}{\sum_k (X_i)_{kv} + \sum_j (X_j)_{uv}} \Bigg).
\end{aligned} \tag{5.19}
$$

5.2.4 Experiments

In this section, we compare different algorithms and coupling schema. Specifically, we will represent how to modify the LFR bechmark [66] to create multiplex OSNs. We, then, run our NMF-based algorithms and two of the best algorithms for single network, namely Infomap [100] and Louvain [10] on coupled networks when varying the fraction of out-community-degree over total degree of each node and simultaneously changing the average node degree in each layer. We also test the algorithms on each layer without coupling to evaluate the superior of coupling techniques in multiplex OSNs.

Normalized Mutual Information (NMI) score [28] is used as the measurement for accuracy. In coupling approach, each layer has a CS and we compute NMI score for multiple networks at once by combining all the nodes in all the layers to a single CS and compute NMI score on that CS.

5.2.4.1 Extend LFR bechmark

LFR benchmark [66] was proposed by Lancichinetti et. al. in 2008 that takes into account the power law property of node degree and community size with tunable exponents. The procedure of original benchmark goes through three fundamental stages:

1 Assigning degree for each node that obeys the power-law distribution with provided exponent.

2 Assigning nodes to communities in the sense that the number of nodes in communities also follows power-law distribution with another given exponent. At the same time, the method determines the in-community-degree and out-community-degree of each node to satisfy the required fraction μ.

3 Drawing random edges with the specified degrees.

Unfortunately, the LFR benchmark is unable of generating multiplex OSNs. LFR generates single networks with totally different CS each time it runs even with identical parameters. Therefore, we make some changes to support multilayer feature while still preserving the important power law characteristics.

The point where we can alter the LFR benchmark is after step 2 when we have already assigned nodes to communities. To change the average node degree in each layer, we multiply the nodes' in-community-degree and out-community-degree with the ratio of the desired average degree to calculated average degree from the procedure. Thus, we can generate many layers with the same CS and different nodes' average degrees in each layer.

5.2.4.2 Dataset and Settings

We create four types of networks which are specified by the directed and weighted properties. For each network type, we subsequently generate 5 three-layer networks with 1000 nodes, when node average degrees in layers are (5, 5, 5); (15, 15 ,15); (20, 20, 20); (25, 25, 25); (15, 20, 25) respectively.

In reality, we may not know exactly whether two accounts in different OSNs belong to same user. Therefore, when coupling two layers, we can only connect $p\%$ out of all the vertices. For the testing dataset, we generate coupled networks when $p = 100\%$ and $p = 20\%$.

All the experiments are carried on undirected unweighted networks, the results for three other network types are similar and put in supplementary materials. We use a Linux system running on an Intel CPU Core Dual 3 GHz, 4 GB RAM machine as the testing environment.

5.2.4.3 Experimental results

We use the following captions to simplify the figures.

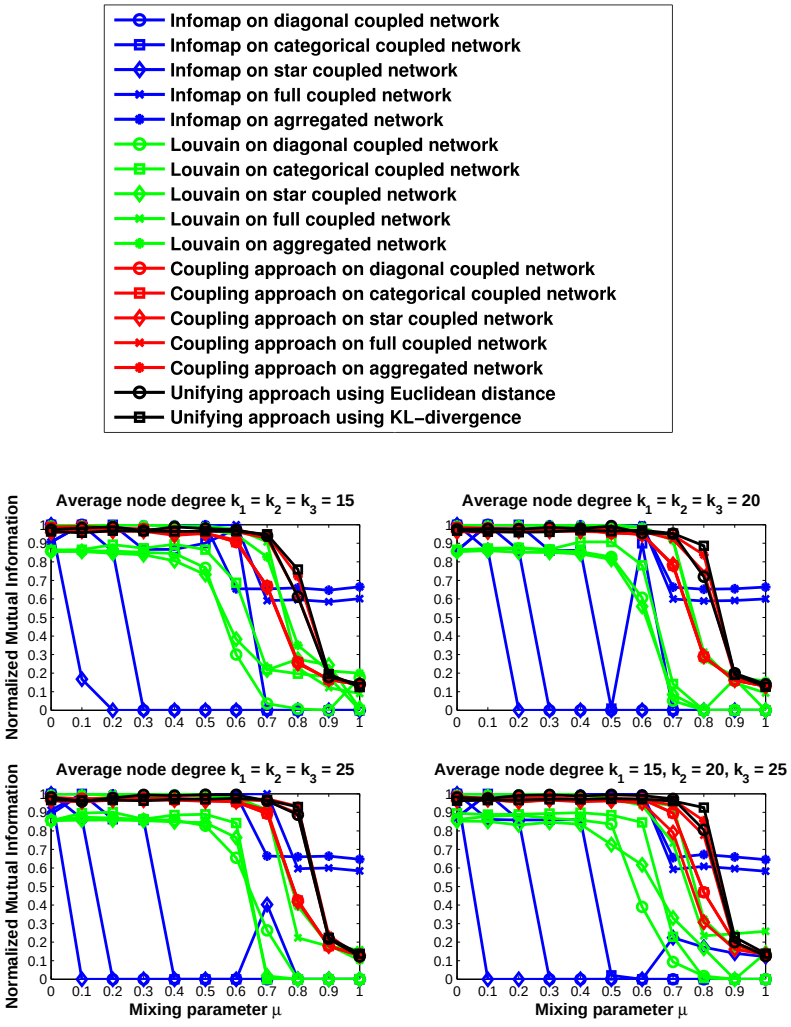


Fig. 36.: NMI scores on network with $p = 100\%$.

Comparison of the algorithms. Figs. 36 and 37 present the MNI scores for all the algorithms with varying nodes' average degrees and mixing parameters μ in undirected unweighted networks. Consistently through all three experiments, NMF-based algorithms always give the highest NMIs and remain stable in all network's settings. Infomap relies heavily on the type of coupling, i.e. performing as well as NMF-based algorithms on aggregated networks and full-coupled networks but extremely poorly on other coupled networks. Meanwhile, Louvain's results lie in the middle of two other methods on all the datasets.

Comparison of coupling schema. Observations on coupling schema used, we see

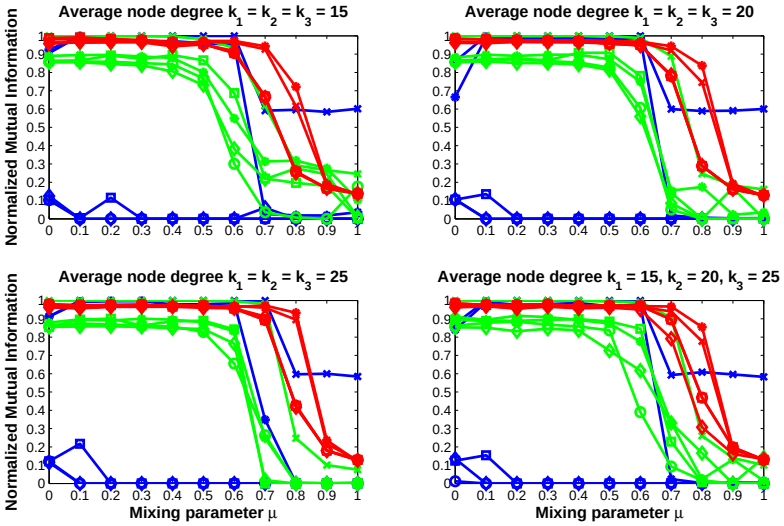


Fig. 37.: NMI scores on network with $p = 20\%$.

that aggregated and full-coupled networks support best for all the algorithms with highest NMI measures. Diagonal-coupled, categorical-coupled and star-coupled networks are only suitable for NMF-based algorithms when having identical behavior as running on aggregated or full-coupled networks. However, the results on these networks for Infomap approach 0 quickly even with very small value of mixing parameters.

Comparison of coupling and non-coupling. By *non-coupling* we refer to the approach that find CS in each layer separately. We run the three algorithms on networks without coupling and with full-coupling, the results are reported in Fig. 38. We can easily see that, for the same algorithm, running on coupled network gives better result, especially in case of NMF-based algorithm. From a general view, NMF-based algorithm show the best accuracy in term of NMI score before μ reaches 0.7. Whereas Fig. 39 shows the results when we keep $\mu = 0.3$ and change p. We see that NMF-based algorithm on coupled network is by far better than the others, for Infomap and Louvain, running on coupled networks get better than non-coupled cases when $p \geq 0.3$.

In summary, NMF-based algorithms show the best results and can be used in all the

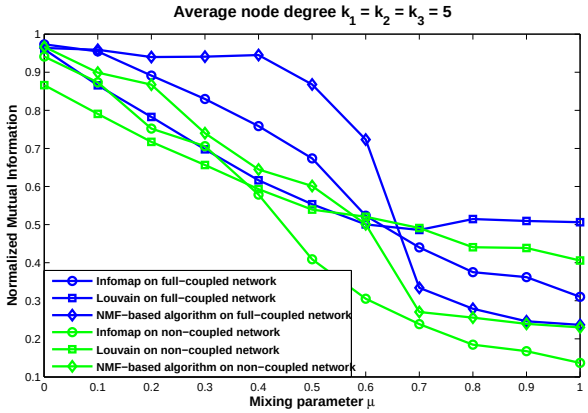


Fig. 38.: Quality of detection with different mixing parameters ($p = 20\%$)

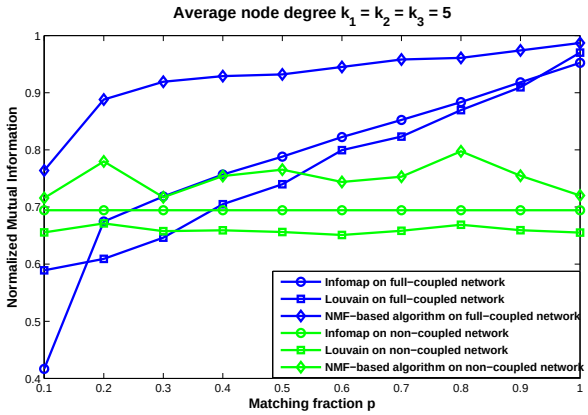


Fig. 39.: Quality of detection with different matching fractions ($\mu = 0.3$)

classes of multiplex OSNs. Louvain achieves good stability and medium accuracy when compared to NMF. Although Infomap works very well on full-coupled and aggregated networks, it is not an acceptable candidate on diagonal, categorical and star-coupled networks.

5.2.5 Conclusions

In this work, we investigate the community detection problem in multiplex OSNs. We propose and compare two classes of approaches, namely unifying and coupling, where we develop a specialization based on NMF algorithm for each approach. The intensive experiments show that NMF-based algorithms perform consistently and give better results

compared to Infomap and Louvain in our benchmark. Although Infomap and Louvain only work well on aggregated and full-coupled networks, they run much faster than NMF-based algorithms.

5.3 Community Detection in Multi-attributed Networks

Summary of contributions:

- We propose a new generative model that describes the formation of topological edges and attribute values in relation with CS. We, consequently, define finding overlapping communities as an optimization problem using NMF framework and develop 3NCD algorithm.
- We prove the convergence and provide efficient update procedure in order to speed up the computational process by a factor of n compared to the straightforward implementation of the update rules.
- We carry intensive experiments on three online social network collections with known ground-truth communities. The results show the superior performance of 3NCD in terms of both accuracy and running time compared to the best current methods.

5.3.1 Models and problem formulation

In this section, we will describe the proposed generative model and the corresponding problem formulation. Here, the network is given as a graph with node attributes, $G = (V, E, T, \mathbf{P})$ where $V = \{v_1, ..., v_n\}$ is the set of nodes, $E = \{(u, v)|u, v \in V\}$ is the set of edges, T is the set of attributes $\{a_1, ..., a_p\}$ and $\mathbf{P}$ is a matrix in which P_{va} indicates whether node v has attributes a (1 means yes and 0 otherwise). More specifically, the graph is encoded as an adjacency matrix $\mathbf{A}$ where the topological interaction between u and v is

Table 28.: Abbreviation Table

Abbreviations	Explanations
A	$A_{ua} = 1$ if there is an edge from u to v; 0 otherwise
P	$P_{ua} = 1$ if node u has attribute a; 0 otherwise
X	X_{uc} is the probability of node u belonging to community c
S	S_{cc} is the probability of an edge having an end in c
H	H_{ca} is the likelihood of a node in c having attribute a
V	Set of nodes in a network
E	Set of edges in a network
T	Set of attributes in a network
u, v	Nodes in the graph
k, c	Number of communities and the index of a community

indicated by the element A_{uv}. Within this paper, we use the abbreviations as described in Table 28.

Generative model: How community structure affects the formation of links and node attributes.

In the model, we assume that there are k communities. We define p_c as the edge density of community c in the sense that a random edge will be in community c with probability p_c. In addition, $p_{c\to u}$ denotes the probability that an edge in community c involves node u and $p_{c\leftarrow a}$ represents the probability of a node in community c having attribute a.

Link formation. The complete generative model is demonstrated in Fig. 40. We use a mixture model of combined effect due to all the k communities to describe the link formation. That is, we approximate A_{uv} using the equation $A_{uv} = \sum_{c=1}^{k} p_c p_{c\to u} p_{c\to v}$. Written in matrix form, we obtain $\mathbf{A} = \mathbf{XSX}^T$ where $X_{uc} = p_{c\to u}$ and $S_{cc} = p_c$ with the constraint

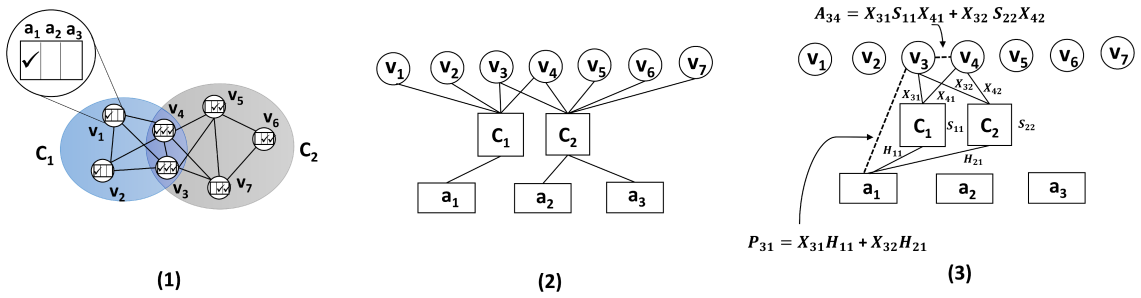


Fig. 40.: Combination model of topology and node attributes with CS: **(1)** the original graph with 7 nodes $V = \{v_1, ..., v_7\}$, two communities $\{\mathbf{C}_1, \mathbf{C}_2\}$ and three node attributes $T = \{a_1, a_2, a_3\}$, **(2)** the tripartite graph describing relationships between nodes, communities and attributes, **(3)** generative formula for A_{34} (the expected number of edge between nodes 3 and 4) and P_{31} (the likelihood of node 3 having attribute a_1).

that both **X** and **S** are non-negative.

Attribute formation. A mixture model, $P_{ua} = \sum_{c=1}^{k} p_{c \to u} p_{c \leftarrow a}$, is used to model the attribute formation. Alternatively, the model is represented in matrix decomposition perspective as $\mathbf{P} = \mathbf{XH}$ where **X** is the same matrix used in modeling topology and $H_{ca} = p_{c \leftarrow a}$. Therefore, our model contains three matrices, **X, S** and **H** as hidden parameters and they are all non-negative.

Problem definition. Community detection is formulated as the following NMF problem.

$$\min_{\mathbf{X},\mathbf{S},\mathbf{H} \geq 0} d(\mathbf{A} \| \mathbf{XSX}^T) + d(\mathbf{P} \| \mathbf{XH}), \tag{5.20}$$

where $d(\mathbf{A} \| \mathbf{B})$ measures the difference between **A** and **B**.

In this work, we use the popular KL-divergence [68] with a slight modification for measurement purpose. More precisely, we ignore the self-loops in the networks and the

measure is defined as,

$$D'(\mathbf{A}\|\mathbf{B}) = \sum_{i \neq j} \left(A_{ij}.\log \frac{A_{ij}}{B_{ij}} + A_{ij} - B_{ij} \right)$$

5.3.2 Methods

We propose 3NCD, our joint NMF method to co-factorize **A** and **P** in III. A. Since it is most important for an NMF algorithm to guarantee both the convergence and the efficient implementation, we show, in III. B, that 3NCD converges to a KKT stationary point and its adaptive update strategy reduces the time-complexity from $O(kn^3)$ to $O(kn^2)$ in an iteration, where k is the number of communities, compared to the straightforward implementation.

5.3.2.1 3NCD Algorithm

Alg. 25 describes 3NCD iterative algorithm for community detection which combines topological and node attribute information. At the beginning, all the elements of the factor matrices, **X, S** and **H**, are initialized with uniformly random values in $[0, 1]$. The main loop keeps updating matrix **X**, diagonal of matrix **S** and matrix **H** until convergence. Finally, we assign node u to community c if $X_{uc} \geq \frac{\sum_{v \in V} X_{vc}}{n}$ and return all the community memberships $\{C_u | u \in V\}$.

During the updating process, we do some pre-computation (Lines 3, 6) to speed up the procedure as analyzed in Subsection III. C. Til this point, the most important question left open is how we derive the update rules for **X, S** and **H** as performed in Lines 4-8 of Alg. 25.

Update rules. Based on the observations from previous applications of NMF method [68, 73, 110], we can compose a general framework for constructing the update rules and proving their convergence property. The framework has the central point of finding the

Algorithm 25: 3NCD algorithm for overlapping community detection combining topology and node attributes.

Input: A network $G = \{V, E, T, \mathbf{P}\}$ and a parameter ϵ

Output: Community memberships $\{C_u | u \in V\}$

Assign X_{uv}, S_{cc}, H_{ci} (uniformly) random values in [0,1]

repeat

  Pre-compute $\mathbf{XS}$, $\frac{A_{ij}}{(\mathbf{XSX})_{ij}}$, $\frac{P_{ic}}{(\mathbf{XH})_{ic}}$ for all i, j, c

  for $i = 1 : n$ **do**

    Update row i of $\mathbf{X}$ using Eq. 5.22

    Update $\mathbf{XS}$, $\frac{A_{ij}}{(\mathbf{XSX})_{ij}}$, $\frac{P_{ic}}{(\mathbf{XH})_{ic}}$ accordingly

  end

  Updates $\mathbf{S}$ and $\mathbf{H}$ using Eq. 5.23

until $\|X_{updated} - X_{previous}\| \leq \epsilon$;

The communities, that node u is assigned to, are the set $C_u = \{c | X_{uc} \geq \frac{\sum_{v \in V} X_{vc}}{n}\}$

Return Community memberships $\{C_u | u \in V\}$

auxiliary function $Q(\mathbf{X}, \tilde{\mathbf{X}})$ of $F(\mathbf{X})$ satisfying the conditions,

$$Q(\mathbf{X}, \tilde{\mathbf{X}}) \geq F(\mathbf{X}),\ Q(\mathbf{X}, \mathbf{X}) = F(\mathbf{X}). \tag{5.21}$$

Then, taking the derivative of $Q(\mathbf{X}, \tilde{\mathbf{X}})$ with respect to X_{uv}, we establish the updating rules for each X_{uv} by representing X_{uv} as an equation of $\tilde{X}_{uv}$. After that, the whole matrix $\mathbf{X}$ can be updated simultaneously. However, constructing function $Q(\mathbf{X}, \tilde{\mathbf{X}})$ is a difficult task. Our purposes of constructing function $Q(\mathbf{X}, \tilde{\mathbf{X}})$ are two-fold: We are not only finding any function $Q(\mathbf{X}, \tilde{\mathbf{X}})$ satisfying the conditions in Eq. 5.21, but also selecting the one that derives concise update rules.

In our problem, three factor matrices, i.e. **X, S** and **H**, need to be updated. From Eq. 5.20, the objective function depends linearly on either **S** or **H** and, thus, the update rules for these two matrices can be constructed similarly to that in [68, 110]. However, for **X**, it cannot be derived by simply imitating the framework and requires some delicate observation. What happens when we directly apply the framework is encountering a quadratic equation without guarantee to obtain a non-negative minimum. In Eq. 5.20, we observe that the objective is quadratic with respect to **X** but linear in terms of X_u (row uth of **X**). Hence, the matrix **X** can be updated in row manner.

Based on the above observation, we derive the following update rules,

$$X_{uv} = X_{uv}\left(\frac{\sum_{k\neq u}\left[\frac{A_{uk}(\mathbf{SX}^T)_{vk}}{(\mathbf{XSX}^T)_{uk}} + \frac{A_{ku}(\mathbf{XS})_{kv}}{(\mathbf{XSX})_{ku}}\right]}{\sum_{k\neq u}\left[(\mathbf{SX}^T)_{vk} + (\mathbf{XS})_{kv}\right] + \sum_{k=1}^{a} H_{vk}} + \frac{\sum_{k=1}^{a}\frac{P_{uk}H_{vk}}{(\mathbf{XH})_{uk}}}{\sum_{k\neq u}\left[(\mathbf{SX}^T)_{vk} + (\mathbf{XS})_{kv}\right] + \sum_{k=1}^{a} H_{vk}}\right), \tag{5.22}$$

$$S_{cc} = S_{cc}\frac{\sum_{u\neq v}\frac{A_{uv}X_{uc}X_{vc}}{(\mathbf{XSX}^T)_{uv}}}{\sum_{u\neq v} X_{uc}X_{vc}}, H_{ci} = H_{ci}\frac{\sum_{k}\frac{P_{ki}X_{kc}}{(\mathbf{XH})_{ki}}}{\sum_{k} X_{kc}}. \tag{5.23}$$

To find the number of communities k, we adopt the modularity maximization approach (see [82] for more details). We choose k at which the modularity function Q achieves the

maximum value.

5.3.2.2 Proof of convergence to a KKT stationary point

The convergence analysis plays a decisive role for the success of any iterative algorithm since we want it to converge at a minimum solution but anywhere else. Auxiliary function lies in the heart of the whole analysis, however, there is no standard way to find such functions. Finding a correct and concise auxiliary function requires subtle observations of the objective and a degree of shape reasoning. In the following, we will state the convergence property and prove it by constructing auxiliary functions.

Theorem 29. *The value of the objective function in Eq. 5.20 is non-increasing and converged to a stationary point under the updating rules in Eq. 5.22 and Eq. 5.23.*

Proof of Convergence. To prove the convergence property of the update rules, we need to find the auxiliary functions for each row of **X**, matrix **S** and **H** that lead to the stated rules. We define the auxiliary functions for each matrix as follows:
Auxiliary function for X_u:

$$Q(X_u, \tilde{X}_u) = \sum_{v \neq u} \Big(A_{uv}(\log(A_{uv}) - 1) + (\mathbf{XSX}^T)_{uv} \Big) - \sum_{v \neq u} A_{uv} \sum_c \eta_{vc} \Big(\log(X_{uc}(\mathbf{SX}^T)_{cv}) - \log(\eta_{vc}) \Big)$$
$$+ \sum_i \Big(P_{ui}(\log(P_{ui} - 1)) - (\mathbf{XH})_{ui} \Big) - \sum_i P_{ui} \sum_c \beta_{ic} \Big(\log(X_{uc}H_{ci}) - \log(\beta_{ic}) \Big) \tag{5.24}$$

where $\eta_{vc} = \frac{\tilde{X}_{uc}(\mathbf{SX}^T)_{cv}}{\sum_j \tilde{X}_{uj}(\mathbf{SX}^T)_{jv}}$, $\beta_{ic} = \frac{\tilde{X}_{uc}H_{ci}}{\sum_j \tilde{X}_{uj}H_{ji}}$.
Auxiliary function for **S**:

$$Q(\mathbf{S}, \tilde{\mathbf{S}}) = \sum_{u \neq v} \Big(A_{uv} \left(\log A_{uv} - 1 \right) + (\mathbf{XSX}^T)_{uv} \Big)$$
$$- \sum_{u,v} (A_{uv} \sum_{p,q} \lambda_{uvpq} \left(\log(X_{up}S_{pq}X_{vq}) - \log(\lambda_{uvpq}) \right),$$

where $\lambda_{uvpq} = \frac{\tilde{X}_{up}S_{pq}\tilde{X}_{vq}}{\sum_{s,t}\tilde{X}_{us}S_{st}\tilde{X}_{vt}}$.

Auxiliary function for $\mathbf{H}$:

$$Q(\mathbf{H}, \tilde{\mathbf{H}}) = \sum_{u,i}\Big(P_{ui}(\log(P_{ui}) - 1) + (\mathbf{XH})_{ui}\Big) - \sum_{u,i}P_{ui}\sum_{c=1}\psi_{uic}\Big(\log(X_{uc}H_{ci}) - \log(\psi_{uic})\Big),$$

where $\psi_{uic} = \frac{X_{uc}\tilde{H}_{ci}}{\sum_j X_{uj}\tilde{H}_{ji}}$.

It is trivial that $Q(X_u, X_u) = F(X_u)$, $Q(\mathbf{S}, \mathbf{S}) = F(\mathbf{S})$ and $Q(\mathbf{H}, \mathbf{H}) = F(\mathbf{H})$. The rest will be to show that the three inequalities, $Q(X_u, \tilde{X}_u) \geq F(X_u)$, $Q(\mathbf{S}, \tilde{\mathbf{S}}) \geq F(\mathbf{S})$ and $Q(\mathbf{H}, \tilde{\mathbf{H}}) \geq F(\mathbf{H})$, are satisfied. Notice that each of the inequalities has a positive part in both side, the only differences that are to be verified take the following form:

$$-\sum_i a_i \times \log(\frac{x_i}{a_i}) \geq -\log(\sum_i a_i \times \frac{x_i}{a_i}),$$

which is always true due to Jensen's inequality [68] and the convexity of logarithmic function.

Proof of KKT stationary convergence point. We have just shown that the update rules lead the objective function to converge by successfully constructing the auxiliary functions. Here, we prove the second part of Theorem 29 of converging to stationary point by verifying the satisfaction of the update rules with KKT conditions.

Introducing the Lagrangian multipliers α_{uc}, $\beta_c j$ and γci,

$$J = D'(\mathbf{A}\|\mathbf{XSX}^T) + D(\mathbf{P}\|\mathbf{XH}) + \sum_{u,c}\alpha_{uc}X_{uc} + \sum_c \beta_c S_{cc} + \sum_{ci}\gamma_{ci}H_{ci}$$

Take the derivatives of J with regards to X_{uc}, S_{cc} and H_{ci},

$$\frac{\delta J}{\delta X_{uc}} = \sum_{v \neq u} \Big[(\mathbf{S}\mathbf{X}^T)_{cv} + (\mathbf{X}\mathbf{S}_{vc}) \Big] - \sum_{v \neq u} \Big[\frac{A_{uv}(\mathbf{S}\mathbf{X}^T)_{cv}}{(\mathbf{X}\mathbf{S}\mathbf{X}^T)_{uv}} + \frac{A_{vu}(\mathbf{X}\mathbf{S})_{vc}}{(\mathbf{X}\mathbf{S}\mathbf{X}^T)_{vu}} \Big] + \sum_i \Big[H_{vi} - \frac{P_{ui}H_{ci}}{(\mathbf{X}\mathbf{H})_{ui}} \Big] - \alpha_{uc} \tag{5.25}$$

$$\frac{\delta J}{\delta S_{cc}} = \sum_{u \neq v} \Big[X_{uc}X_{vc} - \frac{A_{uv}X_{uc}X_{vc}}{(\mathbf{X}\mathbf{S}\mathbf{X}^T)_{uv}} \Big] - \beta_c \tag{5.26}$$

$$\frac{\delta J}{\delta H_{ci}} = \sum_v \Big[X_{vc} - \frac{P_{vi}X_{vc}}{(\mathbf{X}\mathbf{H})_{vi}} \Big] - \gamma_{ci} \tag{5.27}$$

Therefore, by multiplying $\frac{\delta J}{\delta X_{uc}}$, $\frac{\delta J}{\delta S_{cc}}$ and $\frac{\delta J}{\delta H_{ci}}$ with X_{uc}, S_{cc} and H_{ci}, respectively, and letting them equal to 0,

$$\left(\sum_{v \neq u} \Big[(\mathbf{S}\mathbf{X}^T)_{cv} + (\mathbf{X}\mathbf{S})_{vc} \Big] - \sum_{v \neq u} \Big[\frac{A_{uv}(\mathbf{S}\mathbf{X}^T)_{cv}}{(\mathbf{X}\mathbf{S}\mathbf{X}^T)_{uv}} + \frac{A_{vu}(\mathbf{X}\mathbf{S})_{vc}}{(\mathbf{X}\mathbf{S}\mathbf{X}^T)_{vu}} \Big] + \sum_i \Big[H_{vi} - \frac{P_{ui}H_{ci}}{(\mathbf{X}\mathbf{H})_{ui}} \Big] - \alpha_{uc} \right) X_{uc} = 0, \tag{5.28}$$

$$\left(\sum_{u \neq v} \Big[X_{uc}X_{vc} - \frac{A_{uv}X_{uc}X_{vc}}{(\mathbf{X}\mathbf{S}\mathbf{X}^T)_{uv}} \Big] - \beta_c \right) S_{cc} = 0, \tag{5.29}$$

$$\left(\sum_v \Big[X_{vc} - \frac{P_{vi}X_{vc}}{(\mathbf{X}\mathbf{H})_{vi}} \Big] - \gamma_{ci} \right) H_{ci} = 0, \tag{5.30}$$

we obtain the KKT condition, $\alpha_{uc}X_{uc} = 0, \beta_c S_{cc} = 0$ and $\gamma_{ci}H_{ci} = 0$, which ensures that when the objective function converges, the convergence point is a stationary point. The same results can also be inferred from Lemma 2 in [99].

This completes our proof of convergence property. We also note that since matrices **A, X, S** and **H** are all non-negative during the updating process, the final **X, S** and **H** are also non-negative.

5.3.2.3 Complexity analysis

In the algorithm, if we use the naive update rules (Lines 5-7) directly, the time complexity will be $O(kn^3)$. However, because of the pre-computation step (Lines 3 and 6), the running time is reduced to $O(kn^2)$. The algorithm iteratively updates three matrices **X, S** and **H**. The pre-computation step takes $O(kn^2)$, then, updating each row of **X** and three pre-computed matrices requires $O(kn)$. Thus, the complexity of updating the whole **X** is $O(kn^2)$. Similarly, updating **S** and **H** takes $O(kn^2)$. In other words, an iteration requires $O(kn^2)$.

5.3.3 Experiments

In this section, we evaluate the performance of the proposed method on three popular social network datasets, i.e., Facebook, Twitter and Google+. We compare the performance with the state-of-the-art methods for CS detection: CESNA [114] developed for overlapping CS detection using both topology and node attributes, Infomap (overlapping version) [34] and BigClam [113] which are two of the best methods for detecting overlapping communities using network topology. To be fair, we provide the number of communities for all methods since CESNA and BigClam require this number as an input.

5.3.3.1 Datasets.

For experimental purpose, we choose three collections of ego-network datasets stemming from most popular social networking sites, i.e., Facebook, Twitter and Google+ (Taken from [114]). Those datasets contain network topology, node attributes and, more importantly, ground-truth community structure. Ground-truth communities are defined by social circles (or 'lists' in Twitter), which are manually labeled by the owner of the ego-network. In Facebook and Google+, node attributes come from user profiles, such as gen-

der, job titles, institutions, and so on. In Twitter, node attributes are defined by hashtags used by the user in her tweets. The summary of the datasets is presented in Table 29.

Table 29.: Datasets summary.

Collections	**#Tests**	**#Nodes**	**#Edges**	**#Attributes**	**#Coms**
Facebook	10	4,089	170,174	175	193
Google+	132	250,469	30,230,905	690	437
Twitter	973	125,120	2,248,406	33,569	4056

From the above table, we can see that each Twitter's ego network has 125 nodes in average, which is the smallest number among three datasets, while Facebook's ego network contains around 409 nodes. Google+ is the collection with highest average number of nodes in a dataset which is approximately 2,500. We will make use of these statistics later to compare the scalability of the competing methods.

5.3.3.2 Measurement metrics.

We adopt the evaluation method from [114] which quantifies the performance based on ground-truth communities (C^*) and detected communities (C). More specifically, F1 score and Jaccard similarity are chosen to be the measurements between two communities. Then, for each detected, we will find a best matched ground-truth community based on F1 score or Jaccard similarity. After matching all the detected communities, we sum over all best scores and repeat the calculation for ground-truth communities. The final value is the average of the two above summations. The reason for taking average is due to the degeneration of performance if using only a summation, e.g., returning all possible subsets of nodes would lead to perfect matching from ground-truth to detected communities.

Formally, the evaluation function is,

$$\frac{1}{|C^*|}\sum_{C_i^* \in C^*} \max_{C_j \in C} \delta(C_i^*, C_j) + \frac{1}{|C|}\sum_{C_j \in C} \max_{C_i^* \in C^*} \delta(C_j, C_i^*)$$

where function $\delta(.)$ measures the similarity between two communities. Thus, the value of final score ranges from 0 to 1 where 1 indicates perfect recovery of ground-truth communities and conversely worst recovery for 0.

5.3.3.3 Experimental Results

Comparison of accuracy. We compare the accuracy of the methods, i.e., Infomap, BigClam, CESNA and our proposed 3NCD in terms of average F1 score and Jaccard similarity over all datasets in each collection. The experimental results are presented in Table 30 and 31 where we allow each method to run within 24 hours, if running out of that amount of time, we ignore the results.

Table 30.: Accuracy of all the methods in terms of F1 score (Notions: T - Topology only, T&A - Topology + Attributes).

Methods	**Info**	**Facebook**	**Google+**	**Twitter**	**Average**
Infomap	T	0.1691	n/a	0.2117	0.1904
BigClam	T	0.4199	0.2475	0.2253	0.2975
CESNA	T&A	0.42106	0.2244	**0.2462**	0.29722
3NCD	T&A	**0.44075**	**0.2570**	0.2406	**0.3128**

From these two tables above, we see two most important characteristics. First, the methods that use both network links and node attributes, i.e., CESNA and 3NCD, perform much better than Infomap and BigClam using only network topology. In particular with F1 score, CESNA achieves 147% while 3NCD gives 158% relative improvements over

Table 31.: Accuracy of the methods in terms of Jaccard similarity.

Methods	**Info**	**Facebook**	**Google+**	**Twitter**	**Average**
Infomap	T	0.1063	n/a	0.1461	0.1264
BigClam	T	0.3016	0.1509	0.1448	0.1991
CESNA	T&A	0.3022	0.1428	**0.1568**	0.2006
3NCD	T&A	**0.3208**	**0.1712**	0.1565	**0.2162**

Infomap on Facebook collection. On Twitter, these improvements are 14% and 9%, respectively. Because Infomap ran out of time when running on Google+, we do not make any comparison on this collection. On average, CESNA and 3NCD are 56% and 63% better than Infomap. A similar pattern is witnessed in terms of Jaccard similarity with 58.7% and 71% improvements for CESNA and 3NCD, respectively, over Infomap's performance on average.

The second equally crucial characteristic is that 3NCD algorithm significantly outperforms CESNA method in most of the cases. In Table 30 with F1 score, 3NCD beats CESNA by a margin of 5% and 15% on Facebook and Google+ collections, respectively. Only on Twitter datasets, CESNA shows a little better performance of 5%. However, on average, 3NCD gives 5% relative gain over the competitive method. On the other hand, in terms of Jaccard similarity, the gaps between the 3NCD and CESNA are more striking. On Facebook and Google+, the 3NCD again overtakes CESNA by 7% and 21% gains, respectively. The results on Twitter for two methods can be seen equal or 0.2% better for CESNA and on average the 3NCD is relatively superior by 8%.

At this point, we can use the statistics of the data to have a more general statement on the performance of those methods. The average size of each datasets on Twitter collection is fairly small, i.e., 125 nodes per dataset while for Facebook, it is roughly 409 nodes and more than 2,500 in Google+ collection. As a result, CESNA shows a little better

performance on small datasets. However, on the larger (Facebook) or much larger ones (in case of Google+), 3NCD is significantly stronger and appears to be more appropriate.

Comparison of running time. We, next, compare the running time of the proposed, 3NCD, algorithm with other methods. The results are demonstrated in Fig. 41. Based on the results, 3NCD is the fastest algorithm among the considered ones. For Google+, which is the largest datasets, 3NCD is four times faster than CESNA, 279.34 (secs) for the former and 1187.4 (secs) for the later. Infomap was unable to finish running on Google+ on time. Similarly, 3NCD runtime leads on Facebook dataset that is three times faster than CESNA and 8 times faster than Infomap. Twitter contains only small datasets and certainly, the running time of all the methods are not much different from each other. Bigclam and 3NCD seems to have comparable running time in all the experiments.

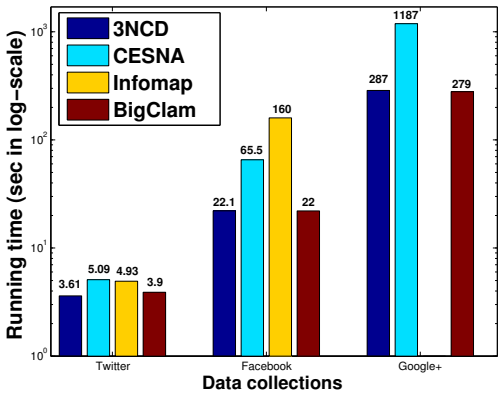


Fig. 41.: Running time of the methods on three dataset collections. No runtime for Infomap on Google+ dataset due to running out of 24 hours.

Comparison on partially observed networks. Since real-world data are always noisy or the network may miss some edges due to errors, e.g., in collecting processes. In this experiment, we will remove some portion of edges from the network and test how the accuracy of the methods is affected. Fig. 42 shows the results when we remove from

10% to 80% of the total edges in Facebook, Twitter and Google+ collections, respectively from left to right.

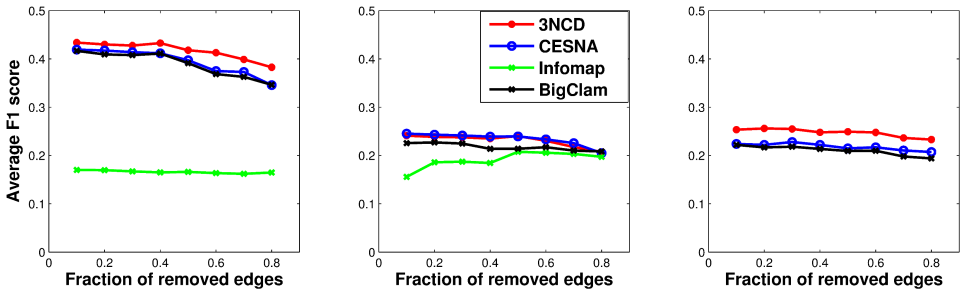


Fig. 42.: Accuracy when removing some portion of edges on Facebook, Twitter and Google+ datasets, respectively from left to right.

From the Fig. 42, we can see that in the cases of missing edges, 3NCD method admit consistently better performance than CESNA on Facebook and Google+ datasets while these two methods perform similarly on Twitter. We ignore Infomap on Google+ since it is incapable of running on Google+. On Facebook and Twitter, both 3NCD and CESNA have much higher results than Infomap. On the other hand, BigClam's results are very close to that of CESNA.

5.3.4 Conclusion

In this paper, we have proposed 3NCD - an accurate and fast NMF-based algorithm for detecting overlapping communities in social networks. The proposed method combines information from both network links and node attributes to a single non-negative matrix factorization model. The proofs for convergence to a stationary solution of the update rules are provided. By intensive experiments, we show that 3NCD algorithm is simultaneously more accurate and faster than the current state-of-the-art community detection methods.

CHAPTER 6

SECURITY AND PRIVACY

Security and privacy are two dominant concerns in cyber-world as the tremendous growth of the Internet and in there, protecting information means live or dead for any individual, organization, nation or even the whole world economy. There has been an enormous body of works on detection, protection and prevention cyber-attacks, however, as the diversity and fleet development of technologies, new and more sophisticated attacks are created every day aiming towards businesses and organizations.

In the literature, many research works have focused on the practical aspects of those attacks by monitoring and experimenting. In contrast, we desire to study how far the attacks can go or their theoretical limits. In the following section, we present our first attempt in studying the preliminary step, *target reconnaissance*, of all the attacks towards a target, e.g., organization, institution.

6.1 Target Reconnaissance Strategy via Social Networks

Summary of contributions:

- We introduce a model of adaptive targeted crawling in OSNs with intelligent attackers. The model takes into consideration the privacy setting in OSNs which grants the users privileges to decide who can see their profiles and the availability of public sources to the attackers.

- We formulate the crawling problem in OSNs as an adaptive targeted maximization problem and prove that it is NP-hard. Based on the submodular function maximization framework, we propose a provable $(1 - 1/e)$-approximate greedy policy.

- We carry thorough simulations in moderate-size networks in comparison with various naive node-ranking policies to show the superior performance of the greedy policy.

6.1.1 Problem Formulation and NP-hardness

In this section, we formally define our model of the network crawling and the Adaptive Targeted Crawling Maximization (ATCM) as a stochastic adaptive optimization problem. We then prove the NP-hardness of the problem by relating it with the classical NP-hard Maximum Coverage (MC).

6.1.1.1 Model and Problem Formulation

We model our network crawling problem as follows: the target is the set of users in an online network where each user has his/her profile information and connections (friendships) to other users. As a general privacy setting, a user can only see the profile information and connection list of his friends and cannot see those who are two or more hops away[1]. We model a crawler as an online user in the same networking environment. He does not have the complete picture of the network topology (who are friends with whom) but he knows the probability of any two nodes being friends. These friendship probabilities can be estimated based on link prediction methods [40, 39].

The crawler wants to gather as much information from the targeted group as possible. The only way he is allowed to take is friending the users in the group and if successful, he can crawl the information of those successfully friended and also their friends. Each user in the targeted network has different probability of accepting the friend request from the crawler. To maximize the information crawled, the crawler may want to send friend requests to all users, however, he will easily get detected by any network monitoring ser-

[1]We can easily extend this model to allow each user having his/her own privacy setting of profile/connections without changing the characteristics of the problem.

vice/manager. The best strategy for the crawler is to mimic the normal behavior by sending friend request to one user, observing the response and then sending to another user. Thus, the central concern is who the crawler should select in each step to maximize the information gained subject to a time limit or limited number of steps.

We also incorporate two types of benefit for each user in the targeted group: 1) friending benefit $B^f(v)$ obtained when the user accepts the friend request of the crawler and 2) the information benefit $B^i(v)$ which is the profile if the user is just a friend of the successfully friended users. Apparently, $B^f(v) \geq B^i(v)$ since friending benefit also contains information benefit. These benefits can be obtained from the importance of users in the targeted organization.

From the model, we abstract our targeted set of users in the online network as a stochastic directed graph $G = (V, E, w)$ where $V = \{v_1, ..., v_n, s\}$ is the set of $n + 1$ nodes (users): n targeted nodes $\{v_1, ..., v_n\}$ and s representing the crawler who initially has no connections to other users, E is the set of m directed edges with their corresponding probabilities of being present $w(e) \in (0, 1], e \in E$. Denote $P_s(v), v \in \{v_1, ..., v_n\}$ the probability of user v accepting friend request from s.

Based on the abstraction, we define the set of *states* that a node can be in as O which consists of: 1) accept (1) friend request if receiving one and reveal all the connections to other people that he has; 2) reject (0) friend request and conceal his connections. Since each edge in the graph has certain probability of being present, the first case actually composes of a family of states where each is a possible subset of edges and happens with some probability. In the original stochastic graph, the nodes are not in any of those states and when we select a node to send friend request, the state of that node is revealed. Similarly to the state of nodes, each edge can also be in one of three states $\{0, 1, ?\}$ where 0 means the edge is not present, 1 means it exists and $?$ means unrevealed since the origin node of the edge rejects the friend request.

We define a *realization* ϕ to be a function of the nodes to their states when all the nodes are revealed and in some state, $\phi(v) = o$ where $o \in O$. Since if the state of a node v is revealed, the connections from that node are also exposed, we use the notion $\phi((v,u))$ to refer to the state of edge (v,u). We require each realization to be consistent meaning that each edge must be in only one of the states $\{0,1,?\}$. Thus, there are many possible realizations which follow a probability distribution $P[\phi]$. We denote Φ to be a random realization and $P[\phi] = P[\Phi = \phi]$. When only some fraction of nodes are realized, we define a *partial realization* ψ to be a function of these realized nodes to their states, $\psi(v) = o$, and domain of ψ to be $dom(\psi) = \{v|\exists o \in O : \psi(v) = o\}$. A partial realization ψ is consistent with a realization ϕ if they are equal everywhere in the domain of ψ. In this case, we write $\phi \sim \psi$. If ψ and ψ' are both consistent with some ϕ and $dom(\psi) \subseteq dom(\psi')$, we say ψ is a *subrealization* of ψ'. Equivalently, ψ is a subrealization of ψ' if and only if $\psi \subseteq \psi'$.

Recall that our adaptive crawling problem asks for a strategy which selects a node to send friend request given the observations (states) of all the previous requests. We formally encode the selection strategy as a policy π, which is a function from a set of partial realization to V. Thus, we can denote the domain of π as $dom(\pi)$ which includes the set of partial realizations of the policy. If the current partial realization is not in $dom(\pi)$ then the policy terminates. The domain of a policy is usually closed under subrealization. That is, if $\psi' \in dom(\pi)$ and ψ is a subrealization of ψ', then $\psi \in dom(\pi)$. Given a policy π and a realization ϕ, we denote $E(\pi,\phi)$ the set of nodes selected by π under realization ϕ and compute the corresponding utility (total benefit) of π on ϕ as follows,

$$f(\pi,\phi) = \sum_{v \in E(\pi,\phi,1)} B^f(v) + \sum_{u \in E(\pi,\phi,v,1)} B^i(u)$$

where $E(\pi,\phi,1) = \{v|E(\pi,\phi), \phi(v) = accept\}$ and $E(\pi,\phi,v,1) = \{u|\exists v : \phi(v,u) =$

$1, v \in E(\pi, \phi, 1)\}$. Thus, the expected utility of a policy π is,

$$f_{avg}(\pi) = \mathbb{E}[f(\pi, \Phi), \Phi] \tag{6.1}$$

where the expectation is taken with respect to $P(\phi)$. We then define our adaptive crawling maximization (ATCM) as follows.

Definition 19 (Adaptive Targeted Crawling Maximization)**.** *Given a network $G = (V, E, w)$, where V is the set of user accounts, E is set of possible connections between users, each edge $(u, v) \in E$ exists with probability $w((u, v))$, for each node v, we have the probability $P_s(v)$ of v accepting our friend request and a budget k which is the number of nodes selected by the policy. The goal of Adaptive Crawling Maximization (ATCM) problem is to find a policy π that maximizes the expected utility $f_{avg}(\pi)$ with $|E(\pi, \phi)| \leq k$ for all ϕ.*

The above definition states our formulation for the adaptive targeted crawling maximization as a stochastic adaptive optimization. The striking difference between this type of problems and the others, termed one-step optimizations, is that the later find a solution (set of nodes) at a single step based entirely on the stochastic data without considering any observation. In contrast, stochastic adaptive optimizations find a policy which is a strategy to select an item (node) at any step i given the observations of what happen after selecting $(i - 1)$ previous items. Thus, the solution is a long term policy.

6.1.1.2 NP-hardness

This subsection provides the proof of the problem being NP-hard. We will start from the Maximum Coverage (MC) which is typical NP-hard problem and design a polynomial time reduction from it to our ATCM problem. The decision version of MC is defined in the following,

Definition 20 (Maximum Coverage (MC))**.** *Given a collection of possibly overlapping sets*

$S = S_1, S_2, ..., S_m$ *and two integers,* k *and* Q*, whether there exists a subcollection* $S' \subseteq S$ *such that* $|S'| \leq k$ *and* $|\bigcup_{S_i \in S'} S_i| \geq Q$*.*

Our hardness results is summarized in the following.

Theorem 30. *Adaptive Crawling Maximization (ATCM) is at least as hard as MC that implies the NP-hardness property of ATCM problem.*

Proof. First, the decision version of the Adaptive Crawling Maximization is formulated as follows: Given a graph G, a number k and a threshold D, whether there is a policy π such that $|E(\pi, \phi)| \leq k$ for all ϕ and $f_{avg}(\pi) \geq D$.

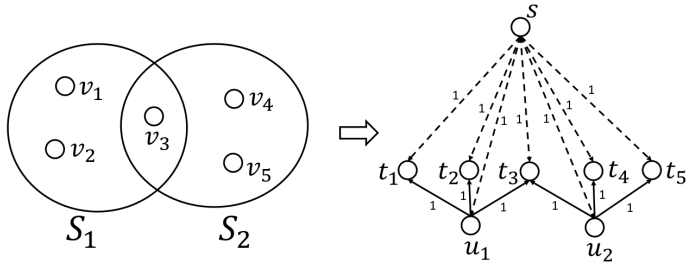


Fig. 43.: Reduction from Maximum Coverage to ATCM

Given an instance of Maximum Coverage problem, we reduce it to an instance of the above decision version of ATCM as follows: Denote $V = \bigcup_{S_i \in S} S_i$, then for each element $v_j \in V$, we create a node t_j in the ATCM instance. For each set $S_i \in S$, we also create a node u_i. If $v_j \in S_i$, a directed edge from u_i to t_j is generated with probability of existence being 1. In the ATCM instance, our source node s has the probability of successfully friending every node being 1. The reduction is illustrated in Fig. 43. Thus, the corresponding instance of ATCM problem is actually deterministic that we know all the connections in the graph and every node will accept friend request if receiving one. We set the friending and information benefits all to 1 and $D = Q + k$, where Q is from MC instance and k is the same as in the original problem. Obviously the reduction has

polynomial complexity.

Now, we prove that the MC instance is an YES if and only if ATCM instance is.

($\rightarrow$) Assume that $S' \subseteq S$ such that $|S'| \leq k$ and $|\bigcup_{S_i \in S} S_i| \geq Q$, then by setting the policy π to select nodes in $U = \{u_i | S_i \in S'\}$ in any order, we achieve a policy that $E(\pi, \phi) \leq k$ (ϕ is the deterministic graph) and $f_{avg}(\pi) \geq k + Q$. That indicates ATCM instance is an YES.

($\leftarrow$) Conversely, if ATCM instance is an YES, that means there is a policy π such that $E(\pi, \phi) \leq k$ and $f_{avg}(\pi) \geq k + Q$, we prove that the original MC is also an YES instance. Observe that if a node t_j is selected by π and π does not select any node u_i for which (u_i, t_j) is an edge in the graph, then we can instead change the selection of t_j to one of the nodes u_i that is connected to t_i (notice that there is at least one such u_i since u_i corresponds to a set, t_j corresponds to elements in MC and an element must be covered by at least a set). Thus, if π selects a node t_j, we can construct another policy π' that selects only nodes in $\{u_1, ..., u_m\}$ without decreasing the utility. For the MC instance, if we select the subcollection $S' = S_i | u_i \in E(\pi', \phi)$, then $|S'| \leq k$ and $|\bigcup_{S_i \in S'} S_i| = f_{avg}(\phi) - k \geq Q$.

Thus, we complete the proof. □

Therefore, the problem of finding the optimal policy π^* is NP-hard meaning we cannot find the solution in polynomial time unless the conjecture P=NP is proved. Our attention is shifted to finding a near optimal policy of π^*. There is a special class of stochastic adaptive optimization problems studied in [42] in which the utility function satisfies two properties: *Adaptive Monotonicity* and *Adaptive Submodularity*. Instances in this class of problems admit an $(1 - 1/e)$ adaptive greedy policy π meaning $f_{avg}(\pi) \geq (1 - 1/e) f_{avg}(\pi^*)$ where e is the base of natural logarithm. In the next section, we will represent our greedy policy that achieves the $(1 - 1/e)$ near optimality and prove that by showing the satisfaction of our utility function in Eq. 6.1 with adaptive monotonicity and submodularity.

6.1.2 Adaptive Greedy Policy and Guarantee

In this section, we describe in details our adaptive greedy policy which selects the node with maximum marginal gain at each step and prove that this policy is at least $(1 - 1/e)$ as good as the optimal policy. In other words, the greedy policy obtains an approximation factor $(1-1/e)$. The approximation guarantee is proved based on the monotone submodular property of the expected utility function as shown in Subsection 6.1.2.2.

6.1.2.1 Adaptive Greedy Policy

Algorithm 26: Adaptive Greedy Policy for ATCM

Input: Graph G, budget k.

Output: A set of nodes $A \in V$ with size k.

$A \leftarrow \emptyset; \psi \leftarrow \emptyset$

for *i=1 to* k **do**

  foreach $u \in V \backslash A$ **do**

    $\Delta(u|\psi) = \mathbb{E}[f(A \cup \{u\}, \Phi) - f(A, \Phi), \Phi \sim \psi]$

  end

  Select $u^* \in \arg\max_u \Delta(u|\psi)$

  Set $A \leftarrow A \cup \{u^*\}$

  Send friend request to u^* and observe $\Phi(u^*)$

  Set $\psi \leftarrow \psi \cup \Phi(u^*)$

end

Return A

The adaptive greedy policy bases mainly on the concept of conditional expected marginal gain which is defined as,

Definition 21 (Conditional Expected Marginal Gain)**.** *Given a partial realization ψ and a*

node v, the expected marginal gain of v conditioned on having observed ψ is

$$\Delta(v|\psi) = \mathbb{E}[f(dom(\psi) \cup \{v\}, \Phi) - f(dom(\psi), \Phi)|\Phi \sim \psi]$$

Informally speaking, the conditional expected marginal gain of a node v is total benefit of selecting that node with respect to all the previously selected nodes (observation ψ) and the expectation is taken over all realizations that are consistent with known observation.

The detailed description of the policy is illustrated in Alg. 26. The whole idea of adaptive greedy policy π is that, at each step $i \leq k$, π selects the next node with the maximum expected marginal gain conditioned on the last $(i-1)$ observations. The policy iterates through k rounds: at round i, it computes the expected marginal gain of nodes that have not been selected and selects the one that locally maximizes this measure. At the end of round i, we send the friend request to the newly selected node and observe the outcome which consists of the response to the request and if it is an acceptance, then, the states of the edges from that node are also revealed. These observations are updated into the current partial realization ψ.

6.1.2.2 Approximation Guarantee

In order to prove the $(1-1/e)$-approximation guarantee of the adaptive greedy policy, we relate the ATCM problem to the maximization of adaptive submodular functions which admits an $(1-1/e)$ natural greedy policy compared to the optimal policy as studied in [42]. Thus, we will show that our objective utility function possesses the two properties: adaptive monotonicity and adaptive submodularity. Recall from [42] that these two properties are specified as follows.

Definition 22 (Adaptive Monotonicity)**.** *A set function $f(.)$ is adaptive monotone with respect to the distribution $P(\phi)$ of realization if the conditional expected marginal gain of*

any node is nonnegative, i.e., for all ψ with $P[\Phi \sim \psi] > 0$ and all $v \in V$, we have

$$\Delta(v|\psi) \geq 0. \tag{6.2}$$

Definition 23 (Adaptive Submodularity)**.** *A set function $f(.)$ is adaptive submodular with respect to the distribution $P[\phi]$ of realizations if the conditional expected marginal gain of any fixed node does not increase as more nodes are selected and their states are observed. Formally, f is adaptive submodular w.r.t. $P[\phi]$ if for all ψ and ψ' such that ψ is a subrealization of ψ' and for all $v \in V \backslash dom(\psi')$, we have*

$$\Delta(v|\psi) \geq \Delta(v|\psi'). \tag{6.3}$$

Our result is summarized in the following theorem.

Theorem 31. *The adaptive greedy policy for our ATCM problem is $(1-1/e)$-approximate.*

Proof. We will consecutively show the satisfaction of the expected utility function $f_{avg}()$ with the definition of adaptive monotonicity and submodularity in Def. 22 and Def. 23.

The monotonicity of the expected utility function $f()$ is easy to verify since, under any realization ϕ, the marginal gain of adding node v into $dom(\psi)$ is either the benefit of friending v and information from his newly revealed friends if v accepts the request or 0 if v rejects. In both cases, the marginal gain is nonnegative. The conditional expected marginal gain is just the weighted combination of marginal gains in each realization and, thus, also nonnegative.

For the adaptive submodularity property, we need to prove that given two partial realizations ψ and ψ' in which ψ is a subrealization of ψ' and a node $v \in V \backslash dom(\psi')$, then

$\Delta(v,\psi) \geq \Delta(v,\psi')$. Based on the fact that ψ is a subrealization of ψ' and Def. 21, we have

$$\begin{aligned}\Delta(v|\psi) &= \mathbb{E}[f(dom(\psi)\cup\{v\},\Phi) - f(dom(\psi),\Phi)|\Phi\sim\psi] \\ &= \sum_{\phi\sim\psi}\Delta(v|\psi,\phi)P[\phi|\phi\sim\psi] \end{aligned} \tag{6.4}$$

where $\Delta(v|\psi,\phi) = f(dom(\psi)\cup\{v\},\phi) - f(dom(\psi),\phi)$. Thus,

$$\begin{aligned}\Delta(v|\psi) &= \sum_{\phi\sim\psi}\Delta(v|\psi,\phi)P[\phi|\phi\sim\psi] \\ &= \sum_{\phi'\sim\psi'}\sum_{\phi\sim\kappa}\Delta(v|\psi,\phi)P[\phi|\phi\sim\kappa]P[\phi'|\phi'\sim\psi'] \end{aligned}$$

where $\kappa = \psi\cup\phi'\backslash\psi'$. Now, given ψ as a subrealization of ψ' and $\phi\sim\psi, \phi'\sim\psi'$ are two fixed realizations sharing $\psi\cup\phi'\backslash\psi'$, we prove that $\Delta(v|\psi,\phi) \geq \Delta(v|\psi',\phi')$ where $v\in V\backslash\psi'$. That is because the utility of ψ given ϕ must be less or equal to that of ψ' given ϕ', thus the added node $v\in V\backslash\psi'$ ($\phi(v) = \phi'(v)$ due to sharing $\psi\cup\phi'\backslash\psi'$ which contains v) cannot bring more benefit to ψ' as it does to ψ. Applying $\Delta(v|\psi,\phi) \geq \Delta(v|\psi',\phi')$ to Eq. 6.5, we obtain

$$\begin{aligned}\Delta(v|\psi) &= \sum_{\phi'\sim\psi'}\sum_{\phi\sim\kappa}\Delta(v|\psi,\phi)P[\phi|\phi\sim\kappa]P[\phi'|\phi'\sim\psi'] \\ &\geq \sum_{\phi'\sim\psi'}\sum_{\phi\sim\kappa}\Delta(v|\psi',\phi')P[\phi|\phi\sim\kappa]P[\phi'|\phi'\sim\psi'] \\ &= \sum_{\phi'\sim\psi'}\Delta(v|\psi',\phi')P[\phi'|\phi'\sim\psi']\sum_{\phi\sim\kappa}P[\phi|\phi\sim\kappa] \\ &= \Delta(v|\psi')\sum_{\phi\sim\kappa}P[\phi|\phi\sim\kappa] \end{aligned} \tag{6.5}$$

Since $\sum_{\phi\sim\kappa}P[\phi|\phi\sim\kappa] = 1$, we finally achieve,

$$\Delta(v|\psi) \geq \Delta(v|\psi') \tag{6.6}$$

which completes the proof of adaptive submodularity.

Thus, the utility function $f_{avg}(.)$ is both adaptive monotone and submodular. Based on the results of [42] that the greedy policy admits an $(1-1/e)$-approximation if the objective function is adaptive monotone and submodular, we conclude that our greedy policy is $(1-1/e)$-approximate. □

Fast greedy computation. In the computation of adaptive greedy policy in Alg. 26, at each step, it has to compute the expected marginal gain for all the nodes that have not been selected. However, the marginal gain of most of the nodes do not change in consecutive steps and thus it has a considerable waste of recomputation. Therefore, a better strategy is to compute the utility of all the nodes and put them in a priority queue. Then, in each step, we pop the node with maximum expected marginal gain and after having the observation, we update the marginal gain of other nodes w.r.t. the observation. This implementation also gives fast query for node with maximum expected marginal gain since it maintains and updates the priority queue along with new observation.

6.1.3 Simulation

This section provides our simulation results in which we simulate the greedy policy against a randomized policy. We show that the adaptive greedy policy achieves drastically better results in terms of utility than the randomized and is scalable in terms of running time and memory used.

Table 32.: Datasets' statistical summary

Name	Type	#Node	#Edges
Enron-email	Communication Network	37k	184k
Epinions	Social Network	76k	509k
Slashdot	Social Network	77k	905k
Twitter	Social Network	81k	1,768k

6.1.3.1 Simulation Settings

Network Data: We select a set of four moderate-size networks[2] with different characteristics to run our simulation. The statistical description of those networks are given in Table 32. Since the data contain only the nodes and edges without any extra information to compute the probabilities needed, for simulation purposes, we randomize these numbers. In particular, for each edge in the network, we choose a random number in $(0, 1]$ as the probability of existence and for each node, we also draw a random number in $(0, 1]$ as the probability of that node accepting the friend request.

Comparing Methods: To have a better view of the performance measured by the utility (total benefit), we compare the adaptive greedy policy with three baseline policies: Random which selects a random node in the network at each time step and send a friend request, Degree which selects nodes with highest degrees, and Pagerank which ranks the nodes based on pagerank and select nodes with largest values.

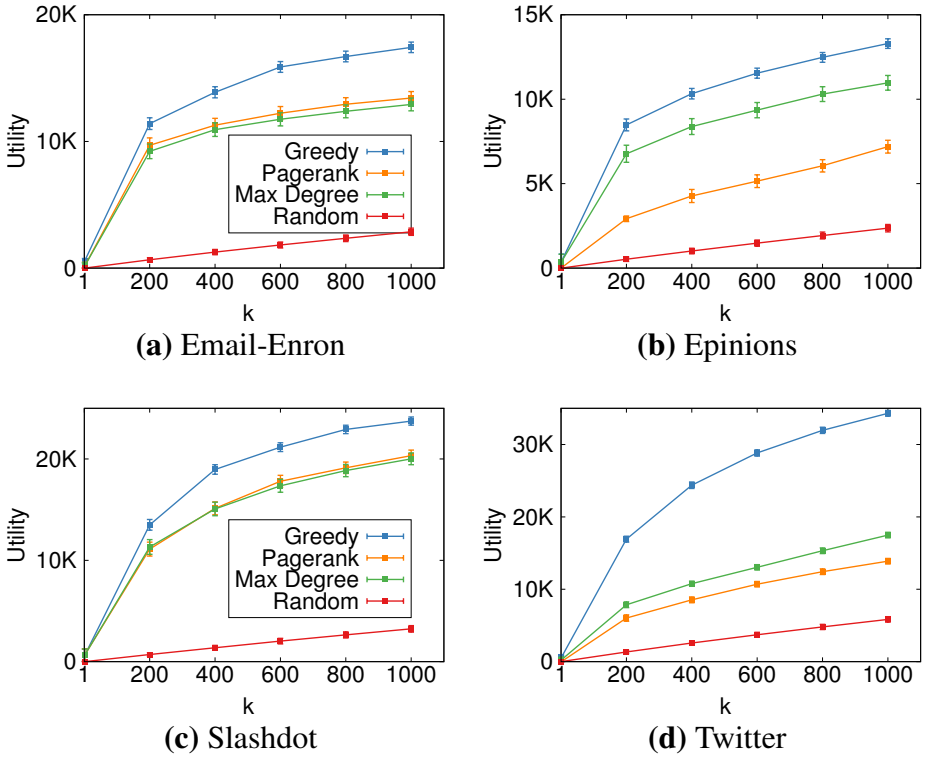
Running Environment: We write a C++ program to simulate each adaptive policies. The simulation is run on a Linux machine with a 3.0GHz 8 core Xeon CPU and 16GB RAM.

6.1.3.2 Simulation Results

Comparison of solution quality. In this simulation, we consider the whole network as out target and run the adaptive policies on the four networks in Table 32. We run each simulation 10000 times and record the average and standard deviation of the utility. The results are illustrated in Fig. 44.

Comparing between the two policies in Fig. 44, it is clear that the adaptive greedy policy considerably and consistently outperforms the other policies. For any value of k,

[2]Taken from: https://snap.stanford.edu/data/



(a) Email-Enron **(b)** Epinions

(c) Slashdot **(d)** Twitter

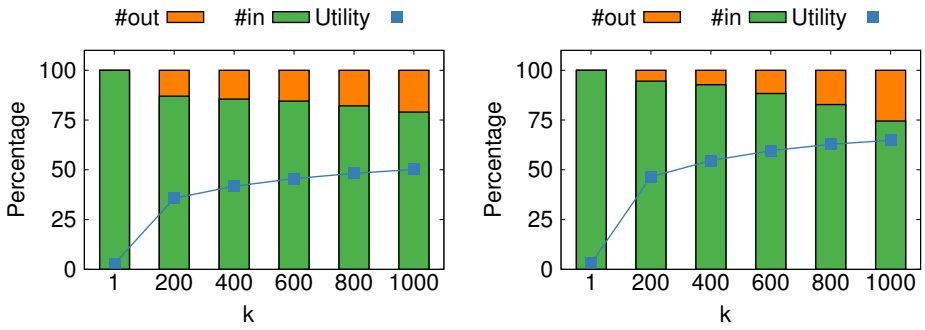
Fig. 44.: Comparison between policies on various networks.

the adaptive greedy policy achieves up to several orders of magnitudes better results than the others. This verifies our good theoretical guarantee on the solution quality of the adaptive greedy policy in the network crawling problem. Pagerank and Degree have similar behaviors since they both base on ranking nodes.

The second observation from Fig. 44 is that, for the group of adaptive greedy, pagerank and degree policies, the first few selected nodes bring in the greatest utility increases as opposed to less increase when k gets larger. This is understandable due to our greedy strategy which always selects the nodes with largest conditional expected marginal gains or selecting the nodes with highest degree/ranking first. Thus, the first few nodes carry the largest amount of gain. In contrary, the adaptive randomized policy behaves linearly with the value of k. That is the increases in utility linearly depending on the value of k. This observation may be explained by the randomness of node selection without any

consideration of the marginal gain in selecting nodes.

Targeted group in network. The previous simulations assume that we only know the targeted part of the network. Here, we consider the set of targeted nodes among the whole network and see how the adaptive greedy policy selecting nodes inside/outside the set of interest. We first find all the communities in the weighted directed graph by Infomap[100] which is one of the best known method for community detection. Then, we select two communities with 21.7k and 9.3k nodes as the targets and run the adaptive greedy policy.



(a) Targeted group 1 **(b)** Targeted group 2

Fig. 45.: Greedy policy on two communities (Lines indicate the utility percentage w.r.t. the total and bars describe the percentage of selected nodes inside and outside targeted set).

The results are shown in Fig. 45. We see that the expected utility is consistent with previous simulations. However, it is interesting that there is a small portion of the selected nodes coming from outside of the interest groups. This portion gets larger when the number of steps k increases. This may be attributed to the decreases in conditional marginal gain when k gets bigger and at some point, selecting nodes outside the target gives higher utility gain.

6.1.4 Conclusion

We propose the adaptive network crawling problem which mimics the normal uses and avoids being detected by selecting a node at time to send friend request and maximizing the benefit. We formulate the problem as a stochastic adaptive maximization. We prove that the cast optimization problem is adaptive monotone and submodular and thus admits an $(1 - 1/e)$-approximate policy. The good quality is verified by our thorough simulations on various network data.

CHAPTER 7

CONCLUSION AND FUTURE WORKS

7.1 Conclusion

In this dissertation, we propose near-optimal approximation algorithms for various important graph problems on real-world billion-scale networks in three main areas:

- Information Diffusion: Influence Maximization and more practical variations, Cascade size estimation, infection sources identification.
- Community Detection across multiple networks and on multi-attributed networks.
- Security on Social Networks: theoretical attacking models and near-optimal strategy.

7.2 Future works

More practical considerations of influence models and social streams are two very potential directions that form my future work. My thesis have mainly focused on boosting the algorithmic performance while assuming a complete diffusion model and network data. However, learning a influence from observed data is remained as an important problem. Furthermore, instead of taking an intermediate step in learning the model, optimizing the influence directly on a stream of social events which reflect the influence cascade is also very potential and worth studying in my future plan.

Appendix A

LIST OF PUBLICATIONS BY THE AUTHOR

Patents

[1] *Importance Sketching of Influence dynamics in massive-scale networks*. Thang N. Dinh and Hung T. Nguyen (Filed Feb 21, 2018, US1818941)

Journal Papers

[1] Hung T. Nguyen, My T. Thai and Thang N. Dinh. *A Billion-scale Approximation Algorithm for Maximizing Benefit in Viral Marketing*, IEEE/ACM Transactions on Networking (**ToN**), 2017

[2] Hung T. Nguyen, Preetam Ghosh, Michael L. Mayo and Thang N. Dinh. *Social Influence Spectrum at Scale: Near-optimal Solutions for Multiple Budgets at Once*, ACM Transactions on Information Systems (**TOIS**), 2017

[3] Hung T. Nguyen, Tri P. Nguyen, Tam Vu, and Thang N. Dinh. *Outward Influence and Cascade Size Estimation in Billion-scale Networks*, Proceedings of the ACM on Measurement and Analysis of Computing Systems (**POMACS**), 2017

[4] Hung T. Nguyen, Nam P. Nguyen, Tam Vu, Huan X. Hoang and Thang N. Dinh. *Breaking Bad: Finding Triangle-breaking Points in Large Networks*, IEEE The Multidisciplinary Open Access Journal (**IEEE Access**), 2017

Conference Papers

[5] Hung T. Nguyen, Tri P. Nguyen, Hai N. Pham, and Thang N. Dinh. *Importance Sketching of Influence Dynamics in Billion-scale Networks*, in Proceedings of the IEEE International Conference on Data Mining (**ICDM**), 2017, acceptance rate 9.25% **(Best Papers invited to KAIS)**

[6] Hung T. Nguyen, Tri P. Nguyen, Tam Vu, and Thang N. Dinh. *Outward Influence and Cascade Size Estimation in Billion-scale Networks*, in ACM International Conference on Measurement and Modeling of Computer Systems (**SIGMETRICS**), 2017, acceptance rate 13.4%

[7] Hung T. Nguyen, My T. Thai and Thang N. Dinh. *Stop-and-Stare: Optimal Sampling Algorithms for Viral Marketing in Billion-scale Networks*, in Proceedings of the 2016 ACM International Conference on Management of Data (**SIGMOD**), 2016 (acceptance rate 19%)

[8] Hung T. Nguyen, My T. Thai and Thang N. Dinh. *Cost-aware Targeted Viral Marketing in Billion-scale Networks*, in Proceedings of the IEEE International Conference on Computer Communications (**INFOCOM**), 2016 (acceptance rate 18.3%)

[9] Hung T. Nguyen, P. Ghosh, M. Mayo and Thang N. Dinh. *Multiple Infection Sources Identification with Provable Guarantees*, in 25th ACM International Conference on Information and Knowledge Management (**CIKM**), 2016 (acceptance rate 17.6%)

[10] Thang N. Dinh, Hung T. Nguyen, P. Ghosh and M. Mayo. *Social Influence Spectrum with Guarantees: Computing More in Less Time*, International Conference on Computational Social Networks (**CSoNet**), 2015 **(Best Paper Award)**

[11] Hung T. Nguyen and Thang N. Dinh. *Unveiling The Structure of Multi-attributed*

Networks via Joint Non-negative Matrix Factorization, In Proceedings of the IEEE Military Communications Conference (**MILCOM**), 2015

[12] Hung T. Nguyen and Thang N. Dinh. *Targeted Cyber-attacks: Unveiling Target Reconnaissance Strategy via Social Networks*, In Proceedings of the IEEE International Conference on Computer Communications, Security and Privacy in BigData Workshop, (**INFOCOM BigSecurity**), 2016

[13] Hung T. Nguyen, Thang N. Dinh and Tam Vu. *Community Detection in Multiplex Social Networks*, IEEE Workshop on Inter-Dependent Networks, (**INFOCOM WIDN**), 2015

[14] Hoang X. Huan, Tuyet T. A. Duong, Ha T. T. Doan, and Hung T. Nguyen. *An Efficient Ant Colony Algorithm for DNA Motif Finding*, in International Conference on Knowledge and Systems Engineering (**KSE**), Springer 2015.

Papers under Submission and Manuscripts

[15] Hung T. Nguyen, Kevin V. Bender and Thang N. Dinh. *Unveiling the Structure of Complex Social Networks: Fast and Accurate GPU-based Algorithms*

[16] Hung T. Nguyen, My T. Thai, Thang N. Dinh, *Approximate k-Cover in Hypergraphs: Bounds, Efficient Algorithms, and Applications*, in submission to conference

[17] Hung T. Nguyen, Alberto Cano, Tam Vu, and Thang N. Dinh. *Blocking Self-avoiding Walks Stops Cyber-epidemics: A Scalable GPU-based Approach*, major revision to IEEE Transactions on Knowledge and Data Engineering (**TKDE**).

[18] Hung T. Nguyen, My T. Thai and Thang N. Dinh. *Provably Good and Efficient Algorithms for Viral Marking in Billion-scale Networks*, in submission to IEEE Transactions on Knowledge and Data Engineering (**TKDE**)

[19] Tianyi Pan, Hung T. Nguyen, Thang N. Dinh and My T. Thai. *Ctrl+Break is OK: Anytime Approximation Algorithm for Influence Maximization*, in submission to IEEE Transactions on Computational Social Systems (**TCSS**)

[20] Hung T. Nguyen, and Thang N. Dinh. *Organizational Vulnerability Measure under Targeted Cyber-attacks on Social Networks*, in submission to IEEE Transactions on Information Forensics and Security (**TIFS**)

[21] Hung T. Nguyen, Tri P. Nguyen, Thang N. Dinh, *Towards Optimal Strategy for Adaptive Probing in Incomplete Networks*, in submission to conference

REFERENCES

[1] http://www.pcworld.com/article/163920/swine_flu_twitter.html.

[2] http://fox13now.com/2013/04/24/the-power-of-one-wrong-tweet/.

[3] http://www.businessinsider.com/reddit-falsely\protect\discretionary{\char\hyphenchar\font}{}{}accuses-sunil-tripathi-of-boston-bombing-2013-7.

[4] http://newsroom.fb.com/company-info/. Updated: 7.1.2014.

[5] V. Agarwal et al. "Scalable graph exploration on multicore processors". In: *SC*. IEEE. 2010, pp. 1–11.

[6] R.M. Anderson and R.M. May. *Infectious diseases of humans: dynamics and control*. Oxford science publications. Oxford University Press, 1992.

[7] Cigdem Aslay et al. "Online Topic-aware Influence Maximization Queries." In: *EDBT*. 2014, pp. 295–306.

[8] D. A. Bader and K. Madduri. "Gtgraph: A synthetic graph generator suite". In: *Atlanta, GA, February* (2006).

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

[10] Vincent D Blondel et al. "Fast unfolding of communities in large networks". In: *Journal of Statistical Mechanics: Theory and Experiment* 2008.10 (2008), P10008.

[11] C. Borgs et al. “Maximizing Social Influence in Nearly Optimal Time”. In: *SODA*. Portland, Oregon: SIAM, 2014, pp. 946–957. ISBN: 978-1-611973-38-9.

[12] A. Campan. “A clustering approach for data and structural anonymity in social networks”. In: *PinKDD* (2008), p. 54.

[13] Robert D Carr et al. “On the red-blue set cover problem”. In: *SODA*. Citeseer. 2000, pp. 345–353.

[14] M. Cha, A. Mislove, and K. P. Gummadi. “A measurement-driven analysis of information propagation in the flickr social network”. In: *WWW ’09*. Madrid, Spain: ACM, 2009, pp. 721–730. ISBN: 978-1-60558-487-4. DOI: 10.1145/1526709.1526806.

[15] Soumen Chakrabarti. *Mining the Web: Discovering knowledge from hypertext data*. Elsevier, 2002.

[16] Soumen Chakrabarti, Martin Van den Berg, and Byron Dom. “Focused crawling: a new approach to topic-specific Web resource discovery”. In: *Computer Networks* 31.11 (1999), pp. 1623–1640.

[17] Duen Horng Chau et al. “Parallel crawling for online social networks”. In: *Proceedings of the 16th WWW*. ACM. 2007, pp. 1283–1284.

[18] S. Chen et al. “Online topic-aware influence maximization”. In: *VLDB* (2015), pp. 666–677.

[19] W. Chen, C. Wang, and Y. Wang. “Scalable influence maximization for prevalent viral marketing in large-scale social networks”. In: *KDD*. ACM, 2010, pp. 1029–1038.

[20] W. Chen, Y. Wang, and S. Yang. “Efficient influence maximization in social networks”. In: *KDD*. ACM, 2009, pp. 199–208.

[21] Wei Chen, Laks VS Lakshmanan, and Carlos Castillo. “Information and influence propagation in social networks”. In: *Synthesis Lectures on Data Management* 5.4 (2013), pp. 1–177.

[22] Zhen Chen, Kai Zhu, and Lei Ying. “Detecting multiple information sources in networks under the SIR model”. In: *CISS 48th*. IEEE. 2014, pp. 1–4.

[23] Junghoo Cho, Hector Garcia-Molina, and Lawrence Page. “Efficient crawling through URL ordering”. In: (1998).

[24] Fan Chung and Linyuan Lu. “Concentration inequalities and martingale inequalities: a survey”. In: *Internet Mathematics* 3.1 (2006), pp. 79–127.

[25] E. Cohen et al. “Sketch-based influence maximization and computation: Scaling up with guarantees”. In: *CIKM*. ACM. 2014, pp. 629–638.

[26] P. Dagum et al. “An Optimal Algorithm for Monte Carlo Estimation”. In: *SICOMP* (2000), pp. 1484–1496.

[27] D. J. Daley, J. Gani, and J. M. Gani. *Epidemic modelling: an introduction*. Vol. 15. Cambridge University Press, 2001.

[28] L. Danon et al. “Comparing community structure identification”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2005.09 (2005), P09008.

[29] T. N. Dinh and M. T. Thai. “Assessing attack vulnerability in networks with uncertainty”. In: *INFOCOM*. IEEE. 2015, pp. 2380–2388.

[30] Thang N Dinh, Ying Xuan, and My T Thai. “Towards social-aware routing in dynamic communication networks”. In: *IPCCC*. 2009, pp. 161–168.

[31] Wenxiang Dong, Wenyi Zhang, and Chee Wei Tan. “Rooting out the rumor culprit from suspects”. In: *ISIT*. IEEE. 2013, pp. 2671–2675.

[32] N. Du et al. “Scalable influence estimation in continuous-time diffusion networks”. In: *NIPS*. 2013, pp. 3147–3155.

[33] Aviad Elyashar et al. “Homing socialbots: intrusion on a specific organization’s employee using Socialbots”. In: *Proceedings of the 2013 IEEE/ACM ASONAM*. ACM. 2013, pp. 1358–1365.

[34] AV Esquivel and M Rosvall. “Compression of flow can reveal overlapping modular organization in networks, 2011”. In: *arXiv preprint arXiv:1105.0812* ().

[35] Martin Ester, Hans-Peter Kriegel, and Matthias Schubert. “Accurate and efficient crawling for relevant websites”. In: VLDB Endowment. 2004, pp. 396–407.

[36] Mehrdad Farajtabar et al. “Back to the Past: Source Identification in Diffusion Networks from Partially Observed Cascades”. In: *arXiv:1501.06582* (2015).

[37] U. Feige. “A threshold of ln n for approximating set cover”. In: *JACM* 45.4 (1998), pp. 634–652. ISSN: 0004-5411. DOI: http://doi.acm.org/10.1145/285055.285059.

[38] Michael Fire, Roy Goldschmidt, and Yuval Elovici. “Online Social Networks: Threats and Solutions”. In: *Communications Surveys & Tutorials, IEEE* 16.4 (2014), pp. 2019–2036.

[39] Michael Fire, Rami Puzis, and Yuval Elovici. “Link prediction in highly fractional data sets”. In: *Handbook of computational approaches to counterterrorism*. Springer, 2013, pp. 283–300.

[40] Michael Fire et al. “Link prediction in social networks using computationally efficient topological features”. In: *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*. IEEE. 2011, pp. 73–80.

[41] Andrew V Goldberg and Robert E Tarjan. “Expected performance of Dijkstrafffdfffdfffds shortest path algorithm”. In: *NEC Research Institute Report* (1996).

[42] Daniel Golovin and Andreas Krause. “Adaptive submodularity: Theory and applications in active learning and stochastic optimization”. In: *Journal of Artificial Intelligence Research* (2011), pp. 427–486.

[43] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan. “Learning Influence Probabilities in Social Networks”. In: *WSDM*. ACM, 2010, pp. 241–250.

[44] A. Goyal, W. Lu, and L. V. S. Lakshmanan. “Celf++: optimizing the greedy algorithm for influence maximization in social networks”. In: *WWW*. ACM. 2011, pp. 47–48.

[45] A. Goyal, W. Lu, and L. V. S. Lakshmanan. “Simpath: An efficient algorithm for influence maximization under the linear threshold model”. In: *ICDM*. IEEE. 2011, pp. 211–220.

[46] A. Goyal, W. Lu, and LV Lakshmanan. “Celf++: optimizing the greedy algorithm for influence maximization in social networks”. In: *WWW*. ACM. 2011, pp. 47–48.

[47] A. Goyal, W. Lu, and LV Lakshmanan. “Simpath: An efficient algorithm for influence maximization under the linear threshold model”. In: *ICDM*. IEEE. 2011, pp. 211–220.

[48] P. Gupta et al. “Wtf: The who to follow service at twitter”. In: *WWW*. ACM. 2013, pp. 505–514.

[49] S. Hong et al. “Accelerating CUDA graph algorithms at maximum warp”. In: *SIGPLAN Notices*. ACM. 2011, pp. 267–276.

[50] Darko Hric, Richard K Darst, and Santo Fortunato. “Community detection in networks: Structural communities versus ground truth”. In: *Physical Review E* 90.6 (2014), p. 062805.

[51] Y. M. Ioannides and L. L. Datcher. “Job information networks, neighborhood effects, and inequality”. In: *Journal of economic literature* (2004), pp. 1056–1093.

[52] K. Jung, W. Heo, and W. Chen. “Irie: Scalable and robust influence maximization in social networks”. In: *ICDM*. IEEE. 2012, pp. 918–923.

[53] Nikhil Karamchandani and Massimo Franceschetti. “Rumor source detection under probabilistic sampling”. In: *ISIT*. IEEE. 2013, pp. 2184–2188.

[54] D. Kempe, J. Kleinberg, and E. Tardos. “Influential nodes in a diffusion model for social networks”. In: *ICALP*. 2005, pp. 1127–1138.

[55] D. Kempe, J. Kleinberg, and É. Tardos. “Maximizing the spread of influence through a social network”. In: *KDD*. ACM New York, NY, USA. 2003, pp. 137–146.

[56] S. Khuller, A. Moss, and J. S. Naor. “The budgeted maximum coverage problem”. In: *Inform. Process. Lett.* (1999), pp. 39–45.

[57] S. Khuller, A. Moss, and JS Naor. “The budgeted maximum coverage problem”. In: *Inform. Process. Lett.* 70.1 (1999), pp. 39–45.

[58] Masahiro Kimura, Kazumi Saito, and Hiroshi Motoda. “Blocking Links to Minimize Contamination Spread in a Social Network”. In: *TKDD* 3.2 (Apr. 2009), 9:1–9:23. ISSN: 1556-4681. DOI: 10.1145/1514888.1514892. URL: http://doi.acm.org/10.1145/1514888.1514892.

[59] Mikko Kivelä et al. “Multilayer networks”. In: *arXiv arXiv:1309.7233* (2013).

[60] B. Klimt and Y. Yang. “Introducing the Enron corpus”. In: *First Conference on Email and Anti-Spam (CEAS)*. 2004.

[61] Christos Koufogiannakis and Neal E Young. “Greedy Δ-approximation algorithm for covering with arbitrary constraints and submodular cost”. In: *Algorithmica* 66.1 (2013), pp. 113–152.

[62] A. Krause, A. Singh, and C. Guestrin. “Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies”. In: *JMLR* (2008), pp. 235–284.

[63] K. Kutzkov et al. “Strip: stream learning of influence probabilities”. In: *KDD*. ACM. 2013, pp. 275–283.

[64] Haewoon Kwak et al. “What is Twitter, a social network or a news media?” In: *Proceedings of the 19th WWW*. ACM. 2010, pp. 591–600.

[65] H. Kwak et al. “What is Twitter, a social network or a news media?” In: *WWW*. ACM. 2010, pp. 591–600.

[66] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. “Benchmark graphs for testing community detection algorithms”. In: *Phys. Rev. E* 78.4 (2008), p. 046110.

[67] Theodoros Lappas et al. “Finding effectors in social networks”. In: *16th SIGKDD*. ACM. 2010, pp. 1059–1068.

[68] Daniel D Lee and H Sebastian Seung. “Algorithms for non-negative matrix factorization”. In: *Adv. in neu. info. proc. sys.* 2001, pp. 556–562.

[69] J. Leskovec, J. Kleinberg, and C. Faloutsos. “Graphs over time: densification laws, shrinking diameters and possible explanations”. In: *KDD*. ACM. 2005, p. 187.

[70] J. Leskovec et al. “Cost-effective outbreak detection in networks”. In: *KDD*. ACM, 2007, pp. 420–429.

[71] Y. Li, D. Zhang, and K. Tan. “Real-time targeted influence maximization for online advertisements”. In: *VLDB* (2015), pp. 1070–1081.

[72] J. Lin and M. Schatz. “Design patterns for efficient graph algorithms in MapReduce”. In: *MLG*. ACM. 2010, pp. 78–85.

[73] Yu-Ru Lin et al. “Metafac: community discovery via relational hypergraph factorization”. In: *Proc. of the 15th ACM SIGKDD*. ACM. 2009.

[74] L. Liu et al. “Mining topic-level influence in heterogeneous networks”. In: *Proc. of the 19th ACM CIKM*. ACM. 2010, pp. 199–208.

[75] Andrey Y Lokhov et al. “Inferring the origin of an epidemic with a dynamic message-passing algorithm”. In: *Physical Review E* 90.1 (2014), p. 012801.

[76] C. Long and R. CW Wong. “Minimizing Seed Set for Viral Marketing”. In: *ICDM*. IEEE, 2011, pp. 427–436.

[77] B. Lucier, J. Oren, and Y. Singer. “Influence at scale: Distributed computation of complex contagion in networks”. In: *KDD*. ACM. 2015, pp. 735–744.

[78] Wuqiong Luo, Wee Peng Tay, and Mei Leng. “Identifying infection sources and regions in large networks”. In: *Signal Processing, IEEE Tran. on* 61.11 (2013), pp. 2850–2865.

[79] M. Mitzenmacher and E. Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[80] *Multiple Infection Sources Identification with Provable Guarantees*. `https://www.dropbox.com/s/96gpow3pfcp2h0p/cikm16_infection_extension.pdf?dl=0`.

[81] G. L. Nemhauser and L. A. Wolsey. “Maximizing submodular set functions: formulations and analysis of algorithms”. In: *North-Holland Mathematics Studies* (1981), pp. 279–301.

[82] Mark EJ Newman and Michelle Girvan. “Finding and evaluating community structure in networks”. In: *Phys. Rev. E* 69.2 (2004), p. 026113.

[83] D. T. Nguyen et al. “Least Cost Influence in Multiplex Social Networks: Model Representation and Analysis”. In: *ICDM*. 2013, pp. 567–576.

[84] H. T. Nguyen, M. T. Thai, and T. N. Dinh. “Stop-and-Stare: Optimal Sampling Algorithms for Viral Marketing in Billion-scale Networks”. In: *SIGMOD*. ACM, 2016, pp. 695–710.

[85] H. T. Nguyen et al. “Importance Sketching of Influence Dynamics in Billion-scale Networks”. In: *ICDM*. IEEE. 2017, pp. 1–1.

[86] H. T. Nguyen et al. “Multiple Infection Sources Identification with Provable Guarantees”. In: *CIKM*. ACM. 2016, pp. 1663–1672.

[87] H. Nguyen and R. Zheng. “On budgeted influence maximization in social networks”. In: *JSAC* (2013), pp. 1084–1094.

[88] Hung T Nguyen, Thang N Dinh, and My T Thai. “Cost-aware Targeted Viral Marketing in Billion-scale Networks”. In: *INFOCOM, 2016 Proceedings IEEE*. IEEE. 2016.

[89] Hung T Nguyen, My T Thai, and Thang N Dinh. “Cost-aware Targeted Viral Marketing in Billion-scale Networks”. In: *INFOCOM*. IEEE. 2013, pp. 55–59.

[90] Nam P. Nguyen, Guanhua Yan, and My T. Thai. “Analysis of Misinformation Containment in Online Social Networks”. In: *Comput. Netw.* 57.10 (July 2013), pp. 2133–2146. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2013.04.002. URL: http://dx.doi.org/10.1016/j.comnet.2013.04.002.

[91] Nam P Nguyen et al. “Adaptive algorithms for detecting community structure in dynamic social networks”. In: *INFOCOM*. IEEE. 2011, pp. 2282–2290.

[92] N. Ohsaka et al. “Dynamic influence analysis in evolving networks”. In: *VLDB* (2016), pp. 1077–1088.

[93] N. Ohsaka et al. “Fast and accurate influence maximization on large networks with pruned monte-carlo simulations”. In: *AAAI*. 2014, pp. 138–144.

[94] Abigail Paradise, Asaf Shabtai, and Rami Puzis. “Hunting Organization-Targeted Socialbots”. In: *Proceedings of the 2015 IEEE/ACM ASONAM*. ACM. 2015, pp. 537–540.

[95] Romualdo Pastor-Satorras and Alessandro Vespignani. “Epidemic spreading in scale-free networks”. In: *Phys. Rev. Let.* 86.14 (2001), p. 3200.

[96] Canh V Pham, Huan X Hoang, and Manh M Vu. “Preventing and Detecting Infiltration on Online Social Networks”. In: *Computational Social Networks*. Springer, 2015, pp. 60–73.

[97] B Aditya Prakash, Jilles Vreeken, and Christos Faloutsos. “Spotting culprits in epidemics: How many and which ones?” In: *ICDM*. IEEE. 2012, pp. 11–20.

[98] V. M. Preciado et al. “Optimal vaccine allocation to control epidemic outbreaks in arbitrary networks”. In: *CDC*. IEEE. 2013, pp. 7486–7491.

[99] Meisam Razaviyayn, Mingyi Hong, and Zhi-Quan Luo. “A unified convergence analysis of block successive minimization methods for nonsmooth optimization”. In: *SIAM Journal on Optimization* 23.2 (2013), pp. 1126–1153.

[100] Martin Rosvall and Carl T. Bergstrom. “Maps of random walks on complex networks reveal community structure”. In: *Proceedings of the National Academy of Sciences* 105.4 (2008), pp. 1118–1123. DOI: 10.1073/pnas.0706851105. eprint: http://www.pnas.org/content/105/4/1118.full.pdf.

[101] Devavrat Shah and Tauhid Zaman. “Rumor centrality: a universal source detector”. In: *SIGMETRICS Performance Evaluation*. Vol. 40. 1. ACM. 2012, pp. 199–210.

[102] Devavrat Shah and Tauhid Zaman. “Rumors in a Network: Who’s the Culprit?” In: *Information Theory, IEEE Tran. on* 57.8 (2011), pp. 5163–5181.

[103] *Stanford network analysis project*. http://snap.stanford.edu..

[104] Jie Tang et al. “Social influence analysis in large-scale networks”. In: *Proceedings of the 15th ACM SIGKDD*. ACM. 2009, pp. 807–816.

[105] Y. Tang, Y. Shi, and X. Xiao. “Influence Maximization in Near-Linear Time: A Martingale Approach”. In: *SIGMOD*. ACM. 2015, pp. 1539–1554.

[106] Y. Tang, X. Xiao, and Y. Shi. “Influence maximization: Near-optimal time complexity meets practical efficiency”. In: *SIGMOD*. ACM. 2014, pp. 75–86.

[107] V.V. Vazirani. *Approximation Algorithms*. Springer, 2001. ISBN: 9783540653677. URL: http://books.google.com/books?id=EILqAmzKgYIC.

[108] A. Wald. *Sequential Analysis*. John Wiley and Sons, 1947.

[109] Alastair J. Walker. “An Efficient Method for Generating Discrete Random Variables with General Distributions”. In: *ACM Trans. Math. Softw.* 3.3 (Sept. 1977), pp. 253–256. ISSN: 0098-3500.

[110] Feng Wang et al. “On positive influence dominating sets in social networks”. In: *Theoretical Computer Science* 412.3 (2011), pp. 265–269. ISSN: 0304-3975. DOI: DOI:10.1016/j.tcs.2009.10.001. URL: http://www.sciencedirect.com/science/article/B6V1G-4XDMHP1-1/2/3c4d1030936af86874d67e2c62b858bd.

[111] Y. Wang et al. “Community-based greedy algorithm for mining top-k influential nodes in mobile social networks”. In: *KDD*. ACM. 2010, pp. 1039–1048.

[112] D. J. Watts and S. H. Strogatz. “Collective dynamics of ’small-world’ networks.” In: *Nature* 393.6684 (1998). ISSN: 0028-0836. DOI: 10.1038/30918.

[113] Jaewon Yang and Jure Leskovec. “Overlapping community detection at scale: a nonnegative matrix factorization approach”. In: *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM. 2013, pp. 587–596.

[114] Jaewon Yang, Julian McAuley, and Jure Leskovec. “Community detection in networks with node attributes”. In: *Data Mining (ICDM), 2013 IEEE 13th International Conference on*. IEEE. 2013, pp. 1151–1156.

[115] H. Zhang, T. N. Dinh, and M. T. Thai. “Maximizing the Spread of Positive Influence in Online Social Networks”. In: *ICDCS*. 2013, pp. 317–326.

VITA

Hung T. Nguyen graduated summa cum laude with a Bachelor of Science in Information Technology from University of Engineering and Technology (UET), Vietnam National University (VNU) in 2014. Immediately after receiving his BS, he started his PhD study in Computer Science at Virginia Commonwealth University under the direction of Dr. Thang N. Dinh. His research interests include approximation algorithms on billion-scale networks, graph mining, combinatorial optimizations. He has served as TPC member of the 13rd international conference on Wireless Algorithms, Systems and Applications (WASA) 2018 and reviewers for numerous conferences and journals.