University of California
Santa Barbara

Scalable Algorithms for Network Design

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Computer Science

by

Sourav Medya

Committee in charge:

Professor Ambuj Singh, Chair
Prithwish Basu, PhD, Raytheon
Charu Aggarwal, PhD, IBM
Professor Xifeng Yan
Professor Amr El Abbadi

September 2019

The Dissertation of Sourav Medya is approved.

Prithwish Basu, PhD, Raytheon

Charu Aggarwal, PhD, IBM

Professor Xifeng Yan

Professor Amr El Abbadi

Professor Ambuj Singh, Committee Chair

July 2019

Scalable Algorithms for Network Design

To my parents and all of them who are passionate about science

Acknowledgements

My journey in the PhD has been enriching and enjoyable in many ways. I had the opportunity to meet some great people who had a significant contribution in my life.

First of all, I would like to express my gratitude to my advisor, Prof. Ambuj Singh for his encouragement about research and fostering in my personal growth. I would also like to thank all the members of my committee: Prof. Amr El Abbadi, Prof. Xifeng Yan, Dr. Charu Aggarwal, and Dr. Prithwish Basu for their valuable advice.

I wish to express my gratitude to masters advisor Prof. Yadati Narahari for showing me the path to pursue a PhD. My gratitude further extends to my internship advisor Dr. Lucy Cherkasova and collaborator Dr. Ananthram Swami for valuable research discussions and career guidance. I would also like to thank Professors Subhash Suri, Teofilo F. Gonzalez and Omer Egecioglu for being great course instructors and helping me to develop essentials skills for research.

I am very grateful to my friend, Dr. Arlei Silva. I thank him for many endless research discussions, plenty of motivational talks, and also making my PhD life extremely memorable. His idealistic research vision and assiduous attitude have increased my love and motivation towards science. I am honored to have him as my friend and collaborator.

I had the privilege to work with some great people: Dr. Sayan Ranu, Dr. Palash Dey, Dr. Petko Bogdanov, Jithin Vachery, Tiyani Ma, Zexi Huang, Mert Kosan, Akash Mittal, Anuj Dhawan, and Sahil Manchanda. I had the pleasure to spent time with some awesome colleagues in the Dyanmo lab: Wei, Xuan-Hong, Victor, Hongyuan, Haraldur, Alex, Omid, Yuanshun, Anh, Rachel, Yuning, Furkan, Sikun, Bo, Minh, Nazli, Kyoungmin, Aniruddha, Richika, Ashwini, Chandana, and Nikhil.

My PhD life was very cheerful because of a large group of awesome friends in Santa Barbara: Ebrahim, Soupitak da, Ranajay da, Anish da, Pritam da, Harris, Alam da, Rima di, Nadira di, Amrita di, Swagata, Soumyashree, Shoron, Seemanta, Souradeep, Chandi da, Srabanti, Kartick, Shantonu da, Ranita di, Anindya da, Tanmoy, Anchal, Era and Risha. I could not expect more affection. From delicious food to timely help, they have been always there for me. Thanks to all of them for creating plenty of beautiful memories of my PhD life. Thanks to my friends Joel and Rauri for the fun times while volunteering at Habitat.

I am grateful to my loving family. Thanks to my father Ganesh Medya and my mother Pratima Medya for their endless love. I would also like to thank my brother Saikat and sister-in-law Gargi for their immense belief in me. Thanks to my other family members: Chotokaku, Mejomoni, Chotomoni, Sejomoni, Raja, Jeet, Sarojmama, Mesomosai (boro, mejo), Mamoni, Sagnik, Boromoni, Apu Mesomosai, Kakolimasi for their valuable support. Without them, I will not be able to make this far.

I would like to thank my wife Swatilekha for her unconditional love. She has been very supportive through my PhD and even before that. She believed in me and constantly encouraged me in my academic journey. I consider myself extremely lucky to share my life with her.

My research has been largely supported by the US Army Research Lab and NSF.

Curriculum Vitæ
Sourav Medya

Education

2019	Ph.D. in Computer Science University of California, Santa Barbara, USA
2012	Master of Engineering in Computer Science and Engineering Indian Institute of Science, Bangalore, India
2010	Bachelor of Technology in Computer Science and Engineering West Bengal University of Technology, Kolkata, India

Publications

1. **Sourav Medya**, Arlei Silva, Ambuj Singh
Influence Minimization Under Budget and Matroid Constraints [Link]
Under Review, 2019

2. **Sourav Medya**, Tiyani Ma, Arlei Silva, Ambuj Singh
K-Core Minimization: A Game Theoretic Approach[Link]
Under Review, 2019

3. Akash Mittal, Anuj Dhawan, **Sourav Medya**, Sayan Ranu, Ambuj Singh
Learning Heuristics over Large Graphs via Deep Reinforcement Learning[Link]
Under Review, 2019

4. Palash Dey, **Sourav Medya**
Covert Networks: How Hard is It to Hide?[Link]
AAMAS - International Conference on Autonomous Agents and Multiagent Systems, 2019

5. **Sourav Medya**, Jithin Vachery, Sayan Ranu, Ambuj Singh
Noticeable Network Delay Minimization via Node Upgrades[Link]
VLDB - International Conference on Very Large Data Bases, 2018

6. **Sourav Medya**, Petko Bogdanov, Ambuj Singh
Making a Small World Smaller: Path Optimization in Networks [Link]
IEEE Transactions on Knowledge and Data Engineering (TKDE), 2018

7. **Sourav Medya**, Arlei Silva, Ambuj Singh, Prithwish Basu, Ananthram Swami
Group Centrality Maximization via Network Design [Link]
SIAM International Conference on Data Mining (SDM), 2018

8. **Sourav Medya**, Ludmila Cherkasova, Ambuj Singh
Predictive Modeling and Scalability Analysis for Large Graph Analytics [Link]
IFIP/IEEE International Symposium on Integrated Network Management (IM), 2017

9. **Sourav Medya**, Petko Bogdanov, Ambuj Singh
   Towards Scalable Network Delay Minimization [Link]
   IEEE International Conference on Data Mining (ICDM), 2016
10. **Sourav Medya**, Ludmila Cherkasova, Guilherme Magalhaes, Kivanc Ozonat, Chaitra Padmanabha, Jiban Sarma, Imran Sheikh
    Towards Performance and Scalability Analysis of Distributed Memory Programs on Large-Scale Clusters [Link]
    ACM/SPEC International Conference on Performance Engineering (ICPE), 2016

Reserach Internships

- Qatar Computing Research Institute (QCRI) 6/2016 - 9/2016
  Mentors: Dr. Nan Tang and Dr. Saravanan Thirumuruganathan
  Research Topic: Queries in Data Streams
- Hewlett Packard Labs, Palo Alto 6/2015 - 9/2015
  Mentor: Dr. Lucy (Ludmila) Cherkasova
  Research Topic: Scalability Analysis of Distributed Memory Programs

Teaching Experience

- Guest Lecturer: Data Structure and Algorithms (Two lectures), UCSB Spring, 2018
- Project Mentor: IGERT Bootcamp, UCSB Summer, 2017
- Teaching Assistant: Introduction to Programming, UCSB Spring, 2017
- Teaching Assistant: Data Structure and Algorithms, UCSB Fall, 2013
- Teaching Assistant: Introduction to Programming, UCSB Fall, 2012
- Project Mentor: Game Theory Course, IISc Spring, 2012
- Teaching Assistant: C Programming, IISc Fall, 2011

Research Mentorship

- Anuj Dhawan and Akash Mittal, Undergraduate student, IIT, Delhi, India
  Learning Combinatorial Graph Algorithms via Network Embedding 2018
- Tiyani Ma, Undergraduate student, University of California Santa Barbara
  Network Stability 2017
- Ali Hajimirza (University of Oklahoma) and Jason White (Cal State)
  Network Summarization: A Comparison of Methods Summer, 2014
- Kara Goodman (Cal State) and Austen Piers (UCSB)
  Multiscale Modeling of Biological Networks Summer, 2014

Service

- **Reviewer** :

TKDD - ACM Transactions on Knowledge Discovery from Data	2017 - 2018
TKDE - IEEE Transactions on Knowledge and Data Engineering	2015 - 2016

- **External Reviewer** :

VLDB - ACM International Conference on Very Large Data Bases	2019
WWW - ACM International World Wide Web Conference	2017, 2019
KDD - ACM Conference on Knowledge Discovery and Data Mining	2015 - 2018
SDM - SIAM International Conference on Data Mining	2017
AAAI -AAAI conference on Artificial Intelligence	2016
WSDM -ACM International Conference on Web Search and Data Mining	2016
ICDM - IEEE International Conference on Data Mining	2016

- **Member** :

Project Mentor **IGERT** Bootcamp, UCSB	2017
Organizer of **IGERT** Bootcamp, UCSB	2014

- **Committee** :

Graduate Student Workshop, UCSB	2013
Open Day (Research Showcase), IISc	2011 - 2012

Abstract

Scalable Algorithms for Network Design

by

Sourav Medya

Networks (or graphs) are a powerful tool to model complex systems such as social networks, transportation networks, and the Web. Network design problems, including planning, implementing and augmenting networks for desirable properties, arise naturally in many applications: How to improve commute time in traffic network? How to contain fake news in social networks? How to preserve a species by conserving important properties of an ecosystem? How to promote healthier behaviour among individuals via their social interactions? However, characterizing the combinatorial effect of these network modifications leads to challenging optimization problems. From a theoretical standpoint, different from their search counterparts (e.g. computing shortest path), the design problems (e.g. optimizing shortest paths) are computationally hard.

My thesis focuses on the development of algorithms for solving large-scale real-world problems using network design. In this thesis, I will discuss a few network design problems and their solutions. In the first part, I will describe design problems to optimize structural properties of a network. More specifically, I will focus on shortest path distance optimization and improvement of node centrality. In the second part, I will show how network design can be used in security related applications via leader hiding problem. Lastly, network modification will be used to improve or control network processes. In particular I will describe influence minimization and resilience maximization in networks via making optimal changes. I will present fast algorithms for these problems using continuous optimization, randomized algorithms and game theoretic techniques.

0.1 Permissions and Attributions

The major portion of the materials described in this thesis have either been published by the author of the thesis or are currently in the process of submission. The author has major contributions in developing the research of the published and submitted works mentioned below.

1. The content of chapter 2 has been previously published as:

   Medya, S., Vachery, J. Ranu, S., & Singh, A. (2018). Noticeable network delay minimization via node upgrades. Proceedings of the VLDB Endowment, 11(9), 988-1001.

   DOI: 10.14778/3213880.3213889

2. The content of chapter 3 has been previously published as:

   Medya, S., Silva, A., Singh, A., Basu, P., & Swami, A. (2018, May). Group centrality maximization via network design. In Proceedings of the 2018 SIAM International Conference on Data Mining (pp. 126-134). Society for Industrial and Applied Mathematics.

   DOI: [10.1137/1.9781611975321.14]

3. The content of chapter 4 has been previously published as:

   Dey, P., & Medya, S. (2019, May). Covert Networks: How Hard is It to Hide?. In Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (pp. 628-637). International Foundation for Autonomous Agents and Multiagent Systems.

   Paper: [Link]

4. Finally, large portions of the content of Chapter 5 and Chapter 6 are currently in submission.

Contents

Curriculum Vitae **vii**

Abstract **x**

0.1 Permissions and Attributions xi

List of Figures **xvi**

List of Tables **xix**

1 Introduction **1**

1.1 Network Design 1

1.2 Related Work 2

1.3 Overview of the thesis 4

1.3.1 Other Work 7

2 Delay Minimization **9**

2.1 Introduction 10

2.2 Related Work 14

2.3 Problem Definition 15

2.3.1 Preliminaries 15

2.3.2 Hardness and Approximability 19

2.4 Algorithms 21

2.4.1 Optimal Solution 22

2.4.2 Restricted Path Optimization Problem (RPOP) 25

2.4.3 Greedy Selection of Nodes 28

2.4.4 Sampling 30

2.5 Experiments 37

2.5.1 Experimental Setup 37

2.5.2 GSN and optimal MIP 40

2.5.3 ISS vs GSN 41

2.5.4 Comparison with scalable baselines 42

2.5.5 Impact of Parameters on Performance 44

2.5.6 Total Improvement 46
2.5.7 Tightness of RPOP 47
2.5.8 Experiments on Synthetic data 48
2.5.9 Edge Delays 52
2.6 Conclusion 52

3 Centrality Maximization 54
3.1 Introduction 55
3.2 Related Work 57
3.3 Problem Definition 58
3.4 Hardness and Inapproximability 60
3.5 Algorithms 62
3.5.1 Greedy Algorithm 62
3.5.2 Sampling Algorithm 63
3.6 Analysis 64
3.6.1 Constrained Problem 65
3.6.2 Analysis of Greedy Algorithm 65
3.6.3 Analysis of Sampling Algorithm 67
3.7 Experimental Results 72
3.7.1 GES: RCCO vs CCO 74
3.7.2 BUS vs. GES 74
3.7.3 Results for Large Graphs 75
3.7.4 Parameter Sensitivity 75
3.7.5 Impact on Other Metrics: 76
3.8 Conclusions 77

4 Hiding in Covert Networks 79
4.1 Introduction 80
4.1.1 Contribution 82
4.2 Related Work 83
4.3 Preliminaries 85
4.3.1 Network Centrality 85
4.3.2 Problem Definition 86
4.4 Results for Degree Centrality 86
4.5 Results for Core Centrality 93
4.6 Captain Networks 98
4.6.1 For Multiple Leaders 98
4.6.2 For Single Leader 99
4.7 Simulation Results 100
4.7.1 Evaluation of 2-Approximation Algorithm 100
4.7.2 Captain Networks and Core Centrality 102
4.8 Conclusion and Future Work 103

5 Core Resilience **105**
5.1 Introduction . . . 106
5.2 Related Work . . . 109
5.2.1 K-core computation and applications: . . . 109
5.2.2 Network Resilience/Robustness . . . 110
5.2.3 Resilience of k-core . . . 110
5.2.4 Shapley Value (SV) and combinatorial problems . . . 111
5.3 Problem Definition . . . 111
5.3.1 Hardness and Approximability . . . 113
5.3.2 Parameterized Complexity . . . 117
5.4 Algorithms . . . 119
5.4.1 Baseline: Greedy Cut . . . 119
5.4.2 Shapley Value Based Algorithm . . . 120
5.4.3 Optimizations for GC and SV . . . 127
5.5 Experiments . . . 127
5.5.1 Experimental Setup . . . 128
5.5.2 Quality Evaluation . . . 130
5.5.3 Running Time . . . 131
5.5.4 Application: k-core Resilience . . . 133
5.5.5 K-core Minimization on the Karate Network . . . 134
5.6 Conclusion . . . 135

6 Controlling Influence **138**
6.1 Introduction . . . 139
6.2 Related Work . . . 141
6.3 Influence Limitation . . . 142
6.3.1 Credit Distribution Model . . . 142
6.3.2 Problem Definitions . . . 144
6.4 Submodularity . . . 149
6.5 Curvature and APX-hardness . . . 152
6.6 Method: BIL . . . 155
6.7 Method: ILM . . . 158
6.7.1 Continuous Relaxation . . . 159
6.7.2 Rounding . . . 162
6.7.3 Generalizations . . . 164
6.8 Experimental Results . . . 164
6.8.1 Experiments: BIL . . . 167
6.8.2 Experiments: ILM . . . 169
6.9 Conclusions . . . 172

7 Conclusion **174**

Bibliography **178**

List of Figures

2.1 Figure shows how the percentage of improvement is skewed over the number of pairs. 13

2.2 Example of Path Optimization Problem. We want to optimize the set of pairs $\{(a,d),(d,f)\}$ with a budget of two nodes from the candidates $\Gamma = \{b,c,e,g\}$. 18

2.3 The figures shows the transformed graph G' as a representation of nodes u and v and the edge (u,v) in G. The values show the delays of the edges. 22

2.4 The distribution of (a) node-pair flows and (b) node delays in three real road network datasets. 31

2.5 Comparison between MIP vs GSN: Normalized Flow Improvement for (a-b) SF, (c-d) BJ. 39

2.6 (a) Approximation quality of ISS against GSN. (b-d)Running times of ISS and GSN. 40

2.7 (NY) The flow improvement (FI) by varying (a) budget, (b) the number of samples, and (c) β (%). 42

2.8 (a-c) The running times of ISS and USS by varying (a) (All) the size of graphs, (b) (NY) the budget, (c) (NY) the number of samples. 42

2.9 The total improvement (TI) on (a) BJ and (c) NY. 46

2.10 (a) Tightness of RPOP and (b) The likelihood to have more than one improved node in the shortest path between a randomly selected node pair against β (in percentage). 47

2.11 Comparison of *Flow Improvement* FI(N) between ISS and other baselines varying budget (k) (a-c) Synthetic Network Structure: on (a) BA, (b) WS, and (c) ER graph generation models; (d) when the delays are on the edges. 49

3.1 Example of Coverage Centrality Optimization problem. We want to optimize the centrality of $\{d,f\}$ with a budget of one edge from the candidates $\{(d,a),(d,b),(f,b)\}$. The coverage centrality of $\{d,f\}$ is 0 in the initial graph (a) and 3 in the modified graph (b). Node d belongs to the shortest paths between (a,e), (a,c) and (a,f) in (b). 59

3.2 BUS vs. Greedy: Improvement in coverage centrality produced by different algorithms. 73
3.3 Comparison with baselines on the EE dataset varying (b) the number of samples and (c) the budget. 73

4.1 Example construction for hardness from Set Cover where $\mathcal{U} = \{u_1, u_2, u_3, u_4\}, S = \{S_1, S_2, S_3\}, S_1 = \{u_1, u_2\}, S_2 = \{u_2\}, S_3 = \{u_3, u_4\}$. The red nodes are the leaders and the blue nodes are the followers. 97
4.2 Number of edges added (b) by different algorithms: LB implies a loose lower bound, HLDA is our algorithm that gives 2-approximation and Random denotes a random edge addition algorithm. Clearly, in both networks while varying edge density (average degree of nodes), the number of edge addition by our algorithm HLDA is almost same as that of LB. 101
4.3 Summary of the difference in core centralities between a leader and a captain in a given captain network (with 550 vertices) by varying number of captains in each group (p) and leaders (h). 102

5.1 K-core minimization for an illustrative example: (a) Initial graph, where all the vertices are in the 3-core; (b) Removing e_1 causes all the vertices to leave the 3-core; (c) Removing e_2 causes only six vertices to leave the 3-core. 107
5.2 Examples of (a) a graph G; (b) its 2-core; and (c) its 3-core structures. . 113
5.3 Example of the changes in the core structure via deletion of an edge: (a) All the nodes are in the 3-core. (b) In the modified graph, the nodes $\{a, b, c, d\}$ are in the 2-core. 113
5.4 Example construction for hardness reduction from SK where $U = \{u_1, u_2, u_3\}, S = \{S_1, S_2, S_3\}, S_1 = \{u_1, u_2\}, S_2 = \{u_1, u_3\}, S_3 = \{u_2\}$. 116
5.5 Example construction for parameterized hardness from Set Cover where $U = \{u_1, u_2, u_3, u_4\}, S = \{S_1, S_2, S_3\}, S_1 = \{u_1, u_2\}, S_2 = \{u_2, u_3\}, S_3 = \{u_3, u_4\}$. 119
5.6 K-core minimization (DN(%)) for different algorithms varying (a-d) the number of edges in the budget; (e-f) the core parameter k; (g-h) and the sampling error ε. The Shapley Value based Cut (SV) algorithm outperforms the best baseline (LD) by up to 6 times. On the other hand, the Greedy approach (GC) achieves worse results than the baselines, with the exception of RD, in most of the settings. SV error increases smoothly with ε and LD becomes a good alternative for large values of k. 129
5.7 Running times by SV using FB while varying (a) the sampling error ε and (b) the core parameter k; and (c-d) impact of pruning for GC and SV algorithms using three datasets. SV is efficient even for small values of sampling error and its running time decreases with k. GC is up to one order of magnitude faster with pruning, while SV is up to 50% faster. . 132

5.8 K-core ($k = 3$) minimization on the Newman's Karate network: (a) $b = 5$ and (b) $b = 10$. Unfilled circle nodes are not in the 3-core of the original network. After removal of b dashed (red) edges, filled (blue) circle nodes remain in the 3-core and unfilled (red) square nodes are removed from the 3-core. 133
5.9 Core resilience for four different networks: (a) DB (co-authorship), (b) WS (Webgraph), (c) FB (social), (d) ER (random). ER and DB are the most and least stable networks, respectively. Tipping points are found for ER and DB. 133
5.10 Core resilience (a, b) and other robustness metrics, clustering coefficient (c, d) and efficiency (e, f) [1], for the Human and Yeast protein-protein interaction networks. 137

6.1 Illustrative example of a social graph and CDM with the corresponding credits over the edges. 146
6.2 We perform our methods for BIL (a) and ILM (b) on the Newman's Karate network with $|X| = 5, k = 9$. Square (red) nodes are in the target set, $|X|$, and dotted (red) edges are in the solution set. The edges are incident to few nodes in the solution for BIL, being strongly biased towards a small set of nodes. For ILM, we have considered $b = 2$, which leads to a solution with more uniform set of edges. 148
6.3 This illustrates a counter example in Theorem 34. 152
6.4 [BIL] (a, c, e) Decrease in Influence (DI) produced by different algorithms. (b, d, f) DI produced by different algorithms varying the size of the target set, X with $k = 30$. 165
6.5 [BIL] Comparison of our greedy algorithm and simulation based baselines varying number of simulations: (a-b) Quality and (c-d) Running time. . 167
6.6 [ILM] Decrease in Influence produced by different algorithms on (a-b) CA, (c-d) FXS, and (e-f) FCS. Our algorithm, CG outperforms the baselines by up to 20%. 169
6.7 [ILM] Decrease in Influence (DI) produced by different algorithms varying the size of the target set, X when $b = 2$. 173

List of Tables

2.1 Table shows the improvement (%) of pairs (%) for budget 8 and 10. . . . 13
2.2 Dataset description and statistics. 38
2.3 (NY) Comparison of *Flow Improvement* between ISS and other baselines on NY against varying budget (k) and β. Each cell reports the relative Improved Flow w.r.t. ISS, i.e. $\frac{FI_{ISS}}{FI_X}$, where X is the method used in that particular cell. When X does not produce any FI ($FI_X = 0$), $\frac{FI_{ISS}}{FI_X} = \infty$. 43
2.4 (BJ) Comparison of *Flow Improvement* between ISS and other baselines on BJ against varying budget (k) and β. 43
2.5 (Synthetic Delay:) Comparison of *Flow Improvement* between ISS and other baselines on NY against varying budget (k) and type of synthetic delays. 48
2.6 (Synthetic Flows/Importance:) Comparison of *Flow Improvement* between ISS and other baselines on NY against varying budget (k) and type of synthetic flows/importance. 48

3.1 Frequently used symbols 59
3.2 Summary of the probabilistic approximations. 70
3.3 Dataset description and statistics. 72
3.4 he ratio between the improvement in coverage produced by GES for CCO and RCCO. 72
3.5 CG data: Comparison of our sampling algorithm (BUS) and the baselines, including our Greedy (GES) approach, using the CG dataset and varying the budget k. We evaluate the coverage of BUS relative to the baselines—i.e. how many times more new pairs are covered by BUS compared to the baseline. 72
3.6 EU data: Comparison of our sampling algorithm (BUS) and the baselines using the EU dataset. 74
3.7 Coverage centrality of BUS relative to baselines. 76
3.8 Improvement of other metrics after adding the edges found by BUS: the numbers are improvement in percentage with respect to the value for the initial graph. 77

5.1 Frequently used symbols 112
5.2 Dataset descriptions and statistics. The value of k_{max} (or degeneracy) is the largest k among all the values of k for which there is a k-core in the graph. 128
5.3 SV (approximate) vs. optimal algorithm using two datasets and a small candidate set size ($|\Gamma| = 50$), and $k = 5$. 130

6.1 Frequently used symbols 144
6.2 Statistics of the datasets. We generate synthetic actions via IC model for CA, EE and CY datasets. 164
6.3 [BIL] Running times (in seconds) of Greedy varying number of tuples. The number of tuples and actions are in thousands. 166
6.4 [BIL] Running Times of Greedy varying graph size for $|X| = 30$ and $k = 30$. 168
6.5 [ILM] Running Times (in min.) of CG varying number of tuples for $|X| = 20$ and $b = 2$. 170
6.6 [ILM] Decrease in Influence (%) in FCS by Continuous Greedy (CG) vs GRR varying the number of tuples. 170
6.7 [ILM] Decrease in Influence (%) in FXS by Continuous Greedy (CG) vs GRR varying the number of tuples. The number of tuples and actions are in thousands. 171
6.8 [ILM] Running times (in minutes) of CG varying number of tuples for $|X| = 20$ and $b = 2$ on FXS. 172
6.9 [ILM] Running Time (Scalability) in seconds of CG and Decrease in Influence (percentage) by CG and GRR varying graph size for $|X| = 20$, $b = 2$ and the number of edges removed is 20. 172

Chapter 1

Introduction

1.1 Network Design

The availability of data generated from computer systems is changing computer science, providing an opportunity to build smart data-driven systems. This thesis focuses on complex systems that can be modeled via an abstract framework known as graphs or networks. For instance, social networks (e.g. Facebook, Twitter) model interpersonal interactions and protein networks model interactions between protein structures. In particular, via leveraging structure and dynamics of these, this thesis is intended to improve real-world and large-scale networks, contributing to the combinatorial optimization, and the field of network and data science as a whole.

Network design problems include planning, implementing and augmenting networks for desirable properties, have a wide range of applications in communication, transportation and social networks as well as computational sustainability [2, 3, 4]: How to contain fake news in social networks (**viral marketing**[4, 5, 6])? How to select a set of land parcels for conservation to ensure species viability (**computational sustainability** **[7]**)? How to modify a social network to promote healthier behaviors among individuals (**health care** [8])?

The problems become very challenging because of the rapidly growing sizes of real-world networks, leading to the need for scalable, data-driven approaches. In particular, some network design problems involve local changes to an existing large network such as adding/modifying links or nodes as a means to improve its global properties [6, 9, 7, 10, 3, 4].

From a theoretical standpoint, different from search versions (e.g. computing shortest path), the design ones (e.g. optimizing shortest path) are computationally harder[11]. This thesis is motivated by algorithmic challenges as well as real-world applications of network design. Network design problems become harder due to the combinatorial nature. These problems discussed in this thesis cover a large range of the spectrum of intractability. The k-core minimization problem [12] is NP-hard to approximate within any constant, fixed parameter intractable, and W[2]-hard. For centrality [13] and shortest path [14] optimization, the problems cannot be approximated within a constant factor grater than $1-1/e$. Additionally, while the approximations of these two remain open, the influence minimization [15] under budget constraint can be solved within a tight constant factor approximation of $1 - 1/e$.

1.2 Related Work

The majority of work on network design aim to solve various objectives via modifying the network structure and node or edge attributes. The problems differ in the upgrade models and the objective functions. Paik et al. [16] first introduced a set of design problems in which vertex upgrades improve the delays of adjacent edges. Afterwards, the problems regarding improving shortest path distances, centrality of nodes, and controlling influence have received a significant amount of attention.

Improving Shortest Paths: Network design problems to improve several global objectives (vertex eccentricity, diameter, all-pairs shortest paths etc.) by addition of edges have been addressed in the past literature [10, 17, 18, 19, 20]. Meyerson et al. [10] designed approximation algorithms for single source and all pairs shortest path minimization. Demaine et al. [19] proposed a constant factor approximation algorithm to minimize the diameter of a network by adding shortcut edges. Prior work has also studied the eccentricity minimization problem in a composite network [20]. Lin et al. [3] also proposed the shortest path improvement problem where the weights are associated with undirected edges. Nikos et al. proposed a novel procedure to maximize the expected decrease in shortest path distances from a given node to the remaining nodes via edge addition [21]. While all of the above problems consider edge improvement, the node version version of the problem is also studied in [7].

Computing centrality and related design problems. A significant amount of related work study the computationally complexity of various centrality measures and improving them. An efficient algorithm to compute the betweenness centrality of a vertex in a network was proposed by Brandes [22]. More recently, Riondato et al. [23] introduced an approach to compute the top-k vertices according to the betweenness centrality using VC-dimension theory. Yoshida [24] studied similar problems for both the betweenness and coverage centrality measures in a group setting. Mahmoody et al. subsequently improved the performance of the above algorithms using a novel sampling scheme [25]. There is an active line of research to optimize the centrality of one node as well as of a set of nodes [26, 27, 28, 29]. There are other work [26, 30] that proposed greedy algorithms and randomized algorithms to increase different types of centrality of certain vertices.

Controlling or Boosting Influence: The influence boosting or limitation problems via network modifications are orthogonal to the influence maximization task [31].

In these modification problems, the objective is to optimize (maximize or minimize) the content spread via structural or attribute-level change in the network. Previous work addressed the influence limitation problem in the SIR model [9, 32, 33]. The objective is to optimize specific network properties in order to boost or contain the content/virus spread. For instance, Tong et al. proposed methods to add (delete) edges to maximize (minimize) the eigenvalue of the adjacency matrix. The influence spread optimization problem also has been studied under the IC model via network design [34, 35, 36, 6, 37] and injecting an opposite campaign [38, 39]. The same problem via edge addition and deletion, respectively, were also studied under the Linear Threshold (LT) model by Khalil et al. [4]. The influence minimization problem was also studied under a few variants of LT model [5, 40]. Kuhlman et al. [5] solved the minimization problem via edge removal heuristics in a simpler version of LT. In [40, 41], the authors explored influence blocking maximization problem in a variant of LT model via node deletion and also showed supermodular property. In summary, the approaches for optimizing influence (propagation) are mostly based on the well-known diffusion models such as SIR, LT and IC. However, Chapter 6 addresses the influence minimization problem based on available cascade information.

1.3 Overview of the thesis

This thesis consists of five major chapters. The first two explore network design problems based on structural properties. The third one is focused on security in covert networks, and the last two chapters study how design controls network processes.

Chapter 2 studies the design problem to improve flows significantly in networks. Reduction of delays in end-to-end data flow is an important network optimization task. Reduced delays enable shorter travel times for vehicles in road networks, faster information flow in social networks, and increased rate of packets in communication networks. This chapter proposes a network design problem where the goal is to perform k network upgrades such that it maximizes the number of flows in the network with a *noticeable* reduction in delay. The problem is NP-hard, APX-hard, and non-submodular. The key idea to overcome these computational challenges is to design an *importance sampling* based algorithm with provable quality guarantees.

Central nodes in social networks can be influential, driving opinions and spreading news or rumors. In hyperlinked environments, such as the Web, where users navigate via clicks, central content receives high traffic, becoming target for advertising campaigns. Controlling nodes' centrality via network updates is a challenging task. Performing minimal modifications to a network to achieve a desired property falls under the umbrella of network design problems. Chapter 3 is focused on improving group (coverage and betweenness) centrality, which is a function of the shortest paths passing through a set of nodes, by adding edges to the network. The core idea is to apply a greedy algorithm, and even faster sampling algorithms, for group centrality maximization with theoretical quality guarantees under realistic constraints.

Chapter 4 is dedicated to solve design problems in covert networks. Covert networks are social networks that often consist of harmful users. There are various popular measures to quantify how influential or central any vertex is in a network. As expected, strategic and influential miscreants in covert networks would try to hide herself from being detected via these measures by introducing new edges. The corresponding computational problem, called the hiding leader problem is NP-complete for the degree and closeness centrality measures.This chapter studies the popular core centrality measure and show that the problem is NP-complete even when the core centrality of every leader is only 3. On the contrary, this work proves that the problem becomes polynomial time solvable for the degree centrality measure if the degree of every leader is bounded above by any constant. This chapter also shows that, although classical complexity theoretic framework fails to shed any light on relative difficulty of Hiding Leader for different centrality measures, the problem is significantly "harder" for the core centrality measure than the degree centrality one.

K-cores are dense patterns that have applications in community detection, network visualization and protein function prediction. However, k-cores can be quite unstable to network modifications, which motivates the question: *How resilient is the k-core structure of a network, such as the Web or Facebook, to edge deletions?* We investigate this question from an algorithmic perspective in Chapter 5. More specifically, we study *the problem of computing a small set of edges for which the removal minimizes the k-core structure of a network.* Besides studying comprehensive characterization of the hardness of the k-core minimization problem (KCM), we propose a novel algorithm inspired by Shapley value that is able to leverage the strong interdependencies in the effects of edge removals in the search space. In experiments, we illustrate how KCM can be applied in the analysis of the k-core resilience of networks.

The problem of influence limitation is discussed in Chapter 6. The spread of fake news is a classical example of the abuse of social networks by "bad actors". In this work, we study how to limit the influence of a target group in a social network via the removal of a few users/links. The idea is to control the diffusion processes while minimizing the amount of disturbance in the network structure. We formulate of the influence limitation problem is in a data-driven fashion and consider two types of constraints over edge removals, a budget constraint and also a, more general, set of matroid constraints. These problems lead to interesting challenges in terms of algorithm design.

1.3.1 Other Work

There are other problems I have studied during my PhD. A few of them are network design problems and I will not describe them in details in this thesis. The following is a summary of these problems.

Network Design Problem [11, 42]

One of the design problems is to improve shortest paths in infrastructure (e.g., airport, traffic), social, and technological networks. While the problem is intractable, we provide a probabilistic approximation guarantee for a restricted problem formulation using VC dimension theory. We also design randomized algorithms that provide scalability and probabilistic approximations for different versions of the problem.

Machine Learning Models for Scalability Analysis [43, 44]

Many high-performance computing (HPC) and modern large graph processing applications belong to a class of scale-out applications, where the application dataset is partitioned and processed by a cluster of machines. Assessing the application scalability is one of the primary goals during such application implementation. In my summer internship, we have worked on the problem of assessing and predicting the scalability of a distributed memory program in a large-scale cluster. We have introduced a novel regression-based approach to do so. Our solution involves profiling data from traditional experiments in a small-sized cluster and an additional set of similar experiments performed with an "interconnect bandwidth throttling" tool, which exposes the bandwidth impact on the performance. These measurements are used in creating an ensemble of analytical models. We predict scalability using linear regression models on important parameters such as the number of cluster nodes and application processes, the dataset size, and the interconnect bandwidth. The accuracy of our approach is demonstrated using the popular Graph500 benchmark. Our main contribution is to apply machine learning methods towards scalability prediction in large clusters.

Solving Combinatorial Problems using Machine Learning [45]

Optimization problems on graphs appear routinely in various applications such as viral marketing in social networks [31], computational sustainability [7], and health-care [8]. These optimization problems are often combinatorial in nature, which results in NP-hardness. Therefore, designing an exact algorithm is infeasible and polynomial-time algorithms, with or without approximation guarantees, are often desired and used in practice [46, 47, 14]. Furthermore, these graphs are often dynamic in nature and the approximation algorithms need to be run repeatedly at regular intervals. Since real-world graphs may contain millions of nodes and edges, this entire process becomes tedious and time-consuming. For instance, advertising through social networks is a common practice today and needs to solved repeatedly due to the networks being dynamic in nature. Furthermore, even the greedy approximation algorithm has been shown to not scale on large networks [48]. We design a deep reinforcement learning framework called GCOMB to mimic the greedy algorithm. The proposed method utilizes Graph Convolutional Network (GCN) to generate node embeddings that predicts the potential nodes in the solution set. These embeddings enable an efficient training process to learn the greedy policy via Q-learning. This work establishes GCMOB produces high quality results while being faster than the greedy algorithm. GCOMB is also robust and scalable to large dynamic networks.

Chapter 2

Delay Minimization

In several domains, the flow of data is governed by an underlying network. Reduction of delays in end-to-end data flow is an important network optimization task. Reduced delays enable shorter travel times for vehicles in road networks, faster information flow in social networks, and increased rate of packets in communication networks. While techniques for network delay minimization have been proposed, they fail to provide any *noticeable* reduction in individual data flows. Furthermore, they treat all nodes as equally important, which is often not the case in real-world networks. In this chapter, we incorporate these practical aspects and propose a network design problem [14] where the goal is to perform k network upgrades such that it maximizes the number of flows in the network with a *noticeable* reduction in delay. We show that the problem is NP-hard, APX-hard, and non-submodular. We overcome these computational challenges by designing an *importance sampling* based algorithm with provable quality guarantees. Through extensive experiments on real and synthetic data sets, we establish that importance sampling imparts up to 1000 times speed-up over the greedy approach, and provides up to 70 times the improvement achieved by the state-of-the-art technique.

2.1 Introduction

Many applications generate data that flow through a network. Examples include trajectories of vehicles in road networks [49, 50, 51], flow of packets in communication networks [7], and electricity distribution in power grids [52]. Naturally, the quality of flow is governed by the network properties. In this chapter, we study the problem of network optimization to minimize delay in data flow.

To provide a concrete example, consider a road network and trajectories of vehicles that ply through this network. In a road network, each edge corresponds to a road segment and a node represents an intersection. Naturally, optimizing the network for smooth flow of vehicles is of critical importance. The quality of a traffic system is enhanced if the commuting time is as low as possible. Therefore, an important question in optimizing a road network is as follows. *Given the trajectories and the number of people navigating through these, how do we optimize the network to reduce the commuting time (delay) for the maximum number of people?* Optimizing the network in this example could mean widening a road (edge), installing better signalling procedure at an intersection (node) or deployment of traffic police at critical locations (edge or node) to better regulate vehicles.

An important consideration in this optimization problem is the budget. More specifically, we never have infinite resources to widen as many roads as needed. Rather, we can introduce only k changes where k is decided based on the resource constraints. Consequently, the goal is to intelligently perform k network optimizations such that the overall path delays are minimized.

Applications: A prominent application of the proposed problem is in transportation infrastructure management for traffic congestion minimization. Metropolitan cities worldwide are facing severe traffic congestion, which increases pollution, fuel usage, and unproductive travel time [53]. For example, it is estimated that drivers in the City of

Chicago cumulatively suffered 302 million hours of travel delay with a total congestion cost of $\$7,222$ million in 2014 [54]. The cause of traffic delays could arise from various factors such as overshooting road capacities, poor road conditions, and sub-optimal signalling infrastructure. It is therefore of practical importance to analyze road network data, identify bottleneck points, and propose systematic upgrades to these bottlenecks so that they work in cohesion to reduce congestion.

There are several applications beyond traffic networks as well. For instance, given a communication network, end-to-end delays of data packet flow can be improved via upgrading network devices [7]. The delays of sending packets (data) arise in the device level. Improving or upgrading the devices, such as upgrading a switch, enable a better and faster communication system. As another application scenario, consider an airport network [11] generated by a particular carrier. In the network, airports represent nodes and the edges correspond to flights offered by the carrier between endpoint cities. Based on information of the past flights one can associate airports with airline-caused delays such as security check delay, luggage handling, etc. An important question for a carrier is how to minimize overall travel time by improving the available infrastructure (e.g. luggage handling).

The importance of this problem has been recognized in the data mining literature [3, 10]. The problem is modeled as follows. The delay in each node or edge of the network is quantified using a weight. The length of a path is the summation of its constituent node and edge weights. A *network upgrade* involves reducing the weight of a node or an edge to a small value $\alpha \geq 0$. The total delay in the *flow* from node u to v is therefore the length of shortest path from u to v. Given a budget k, the goal is to perform k *network upgrades* such that the sum (or average) of all pairs shortest path distances in the network is minimized. Existing techniques, however, ignore two key practical aspects.

1. Noticeable impact: Existing techniques [11, 7] focus on minimizing the sum of the shortest path distances across all pairs of nodes in a network. This formulation leads to negligible reduction in the delay between most of the individual pairs. Consequently, none of the stakeholders (such as vehicles in road networks) witness any noticeable difference in their experiences and may not be satisfied even in the improved network.

To substantiate our claim that existing techniques fail to provide noticeable reduction on individual pairwise delays, we execute the state-of-the-art algorithm [11] on the Los Angeles road network [11] and present the results in Table 2.1 and Fig. 2.1. Fig. 2.1 presents the distribution of improvement (%) over all node pairs and Table 2.1 presents the cumulative distribution of improvement (%). As shown in Table 2.1, more than 92% of the total node pairs do not improve at all. Furthermore, less than 3% of the total pairs have improvement more than 5%. To mitigate the above outlined issue: in this chapter, we ask a more practical question:*How should we perform k node upgrades such that the maximum number of shortest paths have a* significant reduction *in their lengths?* We call a reduction in length *significant* if the reduced length is at least β% shorter than its original length, where β is a user-provided input parameter.

2. Node pair importance: Existing techniques assume that the shortest path between any pair of nodes is equally important. In reality, this is often not the case. In road networks for example, arterial roads connecting prominent places such as the downtown, office and residential districts, airport, etc., have far higher traffic than other roads. Consequently, having a noticeable reduction in the delay in these arterial roads is of higher importance than other less frequently travelled roads.

Our main contributions are as follows:

- We propose and formalize a novel and practical *Path Optimization Problem (POP)* to minimize delays in a network. The proposed problem incorporates practical aspects such as *importance* of node pairs and *noticeable* improvement in quality.

Improvement (%)	$k = 8$	$k = 10$
$= 0$	92.22	92.16
> 0	7.78	7.84
$>= 2$	6.73	6.78
$>= 5$	2.44	2.45
$>= 10$	0.43	0.42

Table 2.1: Table shows the improvement (%) of pairs (%) for budget 8 and 10.

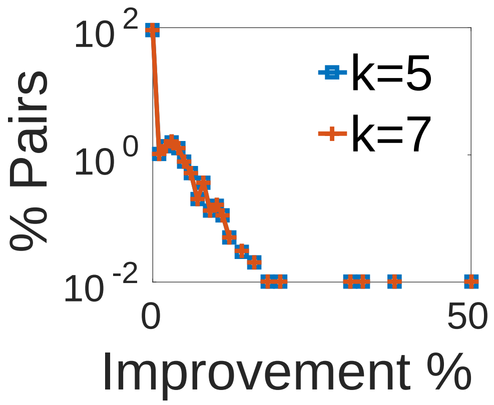


Figure 2.1: Figure shows how the percentage of improvement is skewed over the number of pairs.

- We show that POP is NP-hard as well as APX-hard. We formulate an optimal mixed integer programming (MIP) formulation.
- To overcome the hardness of POP, we propose an iterative greedy algorithm that produces comparable results to optimal MIP. To further enhance scalability, we propose an *importance sampling* algorithm with probabilistic approximation guarantees.
- We perform extensive benchmarking on real-world road networks and establish that importance sampling obtains a speed-up of three orders of magnitude over greedy and 70 times better than the state-of-the-art algorithm.

2.2 Related Work

The majority of work on network design target various objectives via modifying the network structure and attributes [6, 55, 56]. The problems differ in the upgrade models and the objective functions. Paik et al. [16] first introduced a set of design problems in which vertex upgrades improve the delays of adjacent edges. Later, Krumke et al. [57] generalized this model assuming varying costs for vertex/edge upgrades and proposed algorithms to minimize the cost of the minimum spanning trees. Lin et al. [3] also proposed the shortest path improvement problem where the weights are associated with undirected edges. In the above problems, the upgrade models are different and cannot be used to solve out problem.

The closest works to our problem are [7] and [11]. Though they consider a variation of POP, our formulation is different in the sense that we target *significant* improvement of the *important* paths. Both these techniques do not capture the intricacies of significant improvement and node pair importance and thus, as shown in our comparative evaluation, the performance suffers (Section 2.5).

Network design problems to improve several global objectives (vertex eccentricity, diameter, all-pairs shortest paths etc.) by *addition of edges* have been addressed in the past literature [10, 17, 18, 19, 20]. Meyerson et al. [10] designed approximation algorithms for single source and all pairs shortest path minimization. Demaine et al. [19] proposed a constant factor approximation algorithm to minimize the diameter of a network by adding shortcut edges. Prior work has also studied the eccentricity minimization problem in a composite network [20]. All of the above problems, however, consider structural modification (addition of new edges), and hence are complementary to our setting.

Other related problems involve efficient computation of network centrality. In [23], the authors compute top-k nodes based on betweenness centrality via sampling. The group betweenness problem has been solved in almost linear time [24, 25] by a probabilistic approximation algorithm. The design problems to improve the centralities of nodes had been studied in recent past [13, 26, 27, 58]. In these works, the shortest path based centralities has been improved via edge addition. These words differ from the proposed problem in both the upgrade model as well as the objective function.

2.3 Problem Definition

In this section, we first define the concepts central to our problem and then analyze the problem complexity.

2.3.1 Preliminaries

The path optimization problem ingests two sources of data as inputs: a network and a collection of *network flows.*

Definition 1 (Network) *A network is modeled as a graph (directed or undirected) $G(V, E, L)$, where V and E are sets of nodes and edges respectively and L is function $L : V \to \mathbb{R}_{>0}$ over V. This function specify the delays (weights) $l_v := L(v)$ of individual nodes.*

In several domains, objects flow through a network. For example, in transportation networks, vehicles move through a road network. Similarly, in a communication network, data packets move through a sequence of connected routers. We capture the flow of such objects through the idea of network flows.

Definition 2 (Network Flow) *A network flow represents the flow of an object through a network. Mathematically, each flow corresponds to some path $P_{v_1,v_r} = (v_1, v_2, ..., v_r)$ from the source node v_1 to destination v_r.*

In a transportation network, the set of network flows would be a set of vehicular trajectories, and in a communication network, the network flows would constitute the routes taken by data packets

Definition 3 (Path Delay) *The* delay *of a path is defined as the cumulative delays (weights) of the nodes along the path, excluding that of the destination node. More formally, if $P_{v_1,v_r} = (v_1, v_2, ..., v_r)$ is a path from node v_1 to v_r, its delay is defined as $\Sigma_{i=1}^{r-1} l_{v_i}$.*

Weight at the destination node in a path is excluded since the destination node typically does not add any delays in case of the targeted applications (e.g., commuting time in the traffic network). The *shortest path* between two nodes s and t is the one with the minimum delay among all paths connecting these two nodes and its delay is denoted as $d(s,t)$. By convention, $d(s,s) = 0$ for all $s \in V$. Furthermore, if there does not exist a path between two nodes u and v, then $d(u,v) = \infty$. We next define the concept of *improving* a node.

Definition 4 (Node Improvement) *A node is improved by reducing the node delay l_v to a small value $\alpha \geq 0$. The nodes (a fixed/budget k number of nodes) to be improved are chosen from a given candidate set of nodes Γ.*

For simplicity, in the rest of the chapter, we assume $\alpha = 0$. The proposed techniques and proofs hold for any value of α.

The reduction of delays in few nodes (call it *solution set S*) may *significantly* reduce the delay of the shortest paths. We quantify significant improvement in path delay through the definition of *β-improvement.*

Definition 5 (β-Improvement) *A node pair (s,t) is β-improved if $\frac{d(s,t)-d(s,t;S)}{d(s,t)} \geq \beta$, where $d(s,t)$ is the original shortest path and $d(s,t;S)$ is the updated shortest path after improving nodes in solution set S.*

We use Λ_S to denote the set of β-improved node pairs.

As discussed in Sec. 2.1, not all node pairs are equally important. For instance, if the traffic flow between two prominent regions is much higher than the flow between two less-visited regions, then it is more important to improve the path between the prominent regions. This aspect is modeled through *node pair flow.*

Definition 6 (Node pair flow:) *Let $\mathbb{F}$ be the collection of all network flows in the network. The flow associated with a node pair (u, v), denoted by $\xi_{u,v}$, is the proportion of flows originating at u and terminating at v. Mathematically,*

$$\xi_{u,v} = \frac{\big|\{f \in \mathbb{F} \mid f \textit{ starts at } u \textit{ and ends at } v\}\big|}{|\mathbb{F}|} \tag{2.1}$$

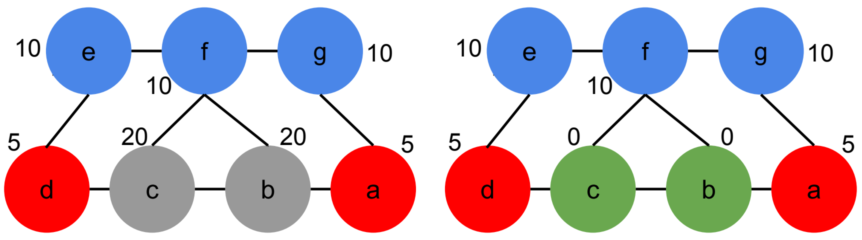
In a road network, the flow between a node pair (u, v) can be computed by counting the number of vehicles that started at u and ended at v. Similarly, in a communication network, the flow corresponds to the number of packets flowing from u to v. The quality of a solution set S is quantified as the total flow $f(S)$ among β-improved node pairs. Mathematically,

$$f(S) = \sum_{(u,v)\in\Lambda_S} \xi_{u,v} \tag{2.2}$$

The path optimization problem is defined as follows.

Problem 1 *(Path Optimization Problem (POP)) Given a network $G = (V, E, L)$, the set of flows $\mathbb{F}$, the improvement parameter β, a candidate set of nodes Γ that can be improved, and a budget k, find a solution set $S \subset \Gamma$ such that $|S| = k$ and $f(S)$ is maximized.*

Example 1 *Figure 2.2 shows a possible solution of size $k = 2$. Initially, the delays of the green, red, blue and grey nodes are $0, 5, 10,$ 20 respectively. Let the candidate set*



(a) Initial graph (b) Modified graph

Figure 2.2: Example of Path Optimization Problem. We want to optimize the set of pairs $\{(a, d), (d, f)\}$ with a budget of two nodes from the candidates $\Gamma = \{b, c, e, g\}$.

$\Gamma = \{b, c, e, g\}$ *and* $\beta = 60\%$. *Furthermore, let the collection of network flows* $\mathbb{F}$ *be such that* $|\mathbb{F}| = 150$ *and* $\xi_{a,d} = \frac{100}{150}, \xi_{d,f} = \frac{50}{150}$, *and the flow between all other pairs is 0. Figure 2.2b shows an optimal solution, where the modified graph has the delays of* b *and* c *are reduced to* 0. *Improving* b *and* c *results in* $f(\{b, c\}) = \frac{150}{150} = 1$ *since it produces at least* 60% *improved shortest path between* (a, d) $(35$ *becomes* $5)$ *and between* (d, f) $(15$ *becomes* $5)$.

Edge Upgrades: We limit ourselves to node upgrades since this is a more generic formulation and any network with delays on edges can easily be mapped to an equivalent network with delays on nodes. More specifically, an edge (u, v) with delay l_e is partitioned into two edges (u, e) and (e, v), where e is a new node inserted into the network and the delay on node e is l_e. By setting Γ to include only the newly added nodes corresponding to the edges, we solve the problem for edge upgrades. If delays are both on edges and nodes, Γ may include all candidate nodes and edge-converted-nodes.

Practical Implications: In a practical scenario, the cause of a delay can be multiple. For example, in a road network, it could arise due to reaching the capacity of a road segment, poor road quality such as potholes, or sub-optimal signalling infrastructure. Reducing delays in such cases could therefore mean widening a road (edge), improving the quality of a road (edge), or upgrading signalling infrastructure at an intersection (node). In our problem, however, we do not deal with the exact cause of the delay or the upgrade mechanism to remove this delay. These domain-specific aspects are abstracted out and the problem only identifies the delay causing entities, and the impact of removing these delays.

2.3.2 Hardness and Approximability

Theorem 1 *POP (decision version) is NP-hard.*

Proof: Consider an instance of the NP-complete Set Cover problem, defined by a collection of subsets $S = \{S_1, S_2, ..., S_m\}$ for a universal set of items $U = \{u_1, u_2, ..., u_n\}$. The problem is to decide whether there exist k subsets whose union is U. To define a corresponding POP instance, we construct an undirected graph with $m + n + 1$ nodes in V: there are nodes i and j corresponding to each set S_i and each element u_j respectively, and an undirected edge (i, j) whenever $u_j \in S_i$. Node a is added to the graph. Node a is connected to i for all $S_i \in S$. The reduction clearly takes polynomial time. The candidate set $\Gamma = \{i|S_i \in S\}$, $\xi_{a,j} = 1/n$ for each node j corresponding to $u_j \in U$, and all remaining node pairs have 0 flow between them. All weights are equal (let us say d) and $\beta = \frac{d}{2d} = \frac{1}{2}$.

A set $S' \subset \Gamma = S$, with $|S'| \leq k$ is a set cover iff $f(S')$ becomes 1. Assume that S' is a set cover and weights are reduced to $\alpha = 0$ for every node in S'. Then $f(S')$ becomes 1 as the nodes in U are of distance d from a. Note, in the initial graph, the distances between a and nodes in U are $2d$.

On the other hand, assume that the $f(S')$ becomes 1 after reducing the weights of nodes in any set $S' \subset S$ with $|S'| \leq k$. The only way to have the distance between (a, u) $(u \in U)$, improved by $\beta = \frac{d}{2d}$ or $\frac{1}{2}$ is by making S' a set cover. ■

Theorem 2 *POP is APX-hard. More specifically, it is NP-hard to approximate POP within a factor of* $(1-\frac{1}{e})$.

Proof: For reduction, we use the Maximum Coverage (MSC) problem. Given a collection of subsets $S_1, S_2, ..., S_m$ for a universal set of items $U = \{u_1, u_2, ..., u_n\}$, the problem is to choose at most k sets to cover as many elements as possible. We give an L-reduction [59] from the MSC problem with parameters x and y. Our reduction is such that following two equations are satisfied:

$$OPT(I_{POP}) \leq xOPT(I_{MSC}) \tag{2.3}$$

$$OPT(I_{MSC}) - s(T^M) \leq y(OPT(I_{POP}) - s(T^P)) \tag{2.4}$$

where I_{MSC} and I_{POP} are problem instances, and $OPT(Y)$ is the optimal value for instance Y. $s(T^M)$ and $s(T^P)$ denote any solution of the MSC and POP instances respectively. If the conditions hold and POP has an ε approximation, then MSC has an $(1-xy(1-\varepsilon))$ approximation. However, MSC is NP-hard to approximate within a factor greater than $(1-\frac{1}{e})$. It follows that $(1-xy(1-\varepsilon)) < (1-\frac{1}{e})$, or, $\varepsilon < (1-\frac{1}{xye})$ [26]. So, if the conditions are satisfied, POP is NP-hard to approximate within a factor greater than $(1-\frac{1}{xye})$.

We apply the same construction as in Theorem 1. However, we use MSC problem for reduction. Let the solution of I_{POP} be $s(T^P)$. That implies that the number of β-improved pairs increases by $s(T^P) \cdot n$. Note that, by construction, $s(T^P) \cdot n = s(T^M)$. So, it follows that both the conditions are satisfied when $x = 1/n$ and $y = n$. Thus, POP is NP-hard to approximate within a factor greater than $(1-\frac{1}{e})$. ■

We next investigate the existence of submodularity property. A function $f(.)$ is submodular if the marginal gain from adding an element to a set S is at least as high as the marginal gain from adding it to a superset of S. Mathematically, it satisfies:

$$f(S \cup \{o\}) - f(S) \geq f(T \cup \{o\}) - f(T) \tag{2.5}$$

for all elements o and all pairs of sets $S \subseteq T$. For submodular and monotone functions, the greedy algorithm of iteratively adding the element with the maximum marginal gain approximates the optimal solution within a factor of $(1 - \frac{1}{e})$[60]. The next theorem shows that the optimization function related to POP is monotone but does not have submodular property.

Theorem 3 *The objective function $f(.)$ is monotone but not submodular.*

Proof: Monotone: Follows from the definition of a shortest path. Let the reduction of delays in S result in the set of β-improved node pairs, Λ_S. Reducing the delay of one more node $v \in \Gamma$ cannot decrease $\sum_{(u,v)\in\Lambda_S} \xi_{u,v}$, i.e., $f(S)$. Thus, $f(.)$ is monotone.

Non-submodular: To prove non-submodularity, we consider the simple example of a chain graph G of four nodes with unit delays: node x_1 is connected to x_2, x_2 to x_3 and x_3 to x_4 by edges. The intuition is the following: a super-set of nodes as solution might force the shortest paths improved by β along with the newly added vertex, whereas, a sub-set of nodes is not sufficient to improve by β. Let us set $A = \phi, B = \{x_2\}, \beta = \frac{2}{3}, \alpha = 0, \Gamma = V$. Only the node pair $\{(x_1, x_4)\}$ has a flow of 1. In our example, $f(B \cup \{x_3\}) = 1$, $f(B) = 0$ as $\beta = \frac{2}{3}$. $f(A \cup \{x_3\}) = 0$, $f(A) = 0$. So, $f(B \cup \{x_3\}) - f(B) > f(A \cup \{x_3\}) - f(A)$. So, $f(\cdot)$ is not submodular. ■

2.4 Algorithms

POP is not only NP-hard, but also hard to approximate. Furthermore, due to lack of submodularity, it is hard to even decide on an optimization strategy. We therefore

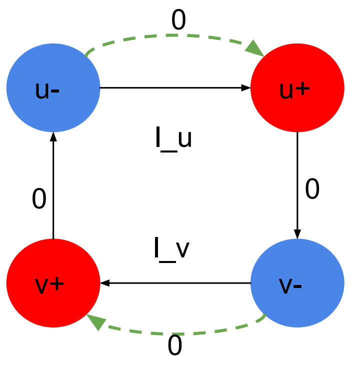


Figure 2.3: The figures shows the transformed graph G' as a representation of nodes u and v and the edge (u, v) in G. The values show the delays of the edges.

start with the optimal solution to POP through *mixed integer programming (MIP)*. Next, we improve efficiency through a greedy approach. Finally, we further expedite greedy through importance sampling with probabilistic guarantees on the approximation error, which allows us to scale on real networks containing more than half a million nodes.

2.4.1 Optimal Solution

We formulate the POP problem as a mixed integer program (MIP), in order to obtain the optimal solution. We use a multi-commodity flow formulation[7] to compute the shortest path delay between a node pair $p = (u, v) \in \mathscr{P}$ (for simplicity, $\mathscr{P} = V \times V$). To apply MIP on a given graph G, we first convert it to a directed graph G' as follows: a node v is replaced by two nodes v^- and v^+ with two additional parallel edges from v^- to v^+ with delays l_v (original node edge, e_v) and 0 (upgraded node edge e'_v) respectively. If an edge (u, v) is present in the original graph, there are two edges (u^+, v^-) and (v^+, u^-) with delays 0 in G'. Figure 2.3 shows an example of this transformation procedure for an edge $e = (u, v)$. The variables used in the MIP formulation are as follows:

- x_v: a flag for whether node v is to be upgraded.
- x_p: a variable that indicates a pair p is β-improved

- $d'(p)$:the effective shortest path delay between nodes in pair p
- Δ_p: difference between effective improvement and β in pair p.
- *budget*: the total number of upgraded nodes
- g_{pv}: a continuous variable that indicates the flow of the commodity p on edge e_v.
- g'_{pv}: continuous variable that indicates the flow of the commodity p on edge e'_v.
- h_{pe}: continuous variable that indicates whether edge e is chosen to be on the shortest path for the pair p.

In an integral solution, g_{pv} and g'_{pv} denote whether the original node and upgraded node respectively are chosen to be on the shortest path between the pair p in an integral solution. We use $\delta^-(v^-)$ and $\delta^+(v^+)$ to denote the set of incoming and outgoing edges respectively. $d(p)$ denotes the original shortest path distance (we assume this as constant as it is given) in the initial graph for pair p. M is a large positive constant. The full MIP formulation is as follows:

$$\max \left\{ \sum_{p \in \mathscr{P}} \xi_p \right\} \text{ such that,}$$

$$g_{ps} + g'_{ps} = 1, g_{pt} + g'_{pt} = 1 \qquad \forall p = (s,t) \in \mathscr{P} \tag{2.6}$$

$$\sum_{e \in \delta^-(s^-)} h_{pe} = 0 \qquad \forall p = (s,t) \in \mathscr{P} \tag{2.7}$$

$$\sum_{e \in \delta^+(s^+)} h_{pe} = g_{ps} + g'_{ps} \qquad \forall p = (s,t) \in \mathscr{P} \tag{2.8}$$

$$\sum_{e \in \delta^-(t^-)} h_{pe} = g_{pt} + g'_{pt} \qquad \forall p = (s,t) \in \mathscr{P} \tag{2.9}$$

$$\sum_{e \in \delta^+(t^+)} h_{pe} = 0 \qquad \forall p = (s,t) \in \mathscr{P} \tag{2.10}$$

$$\sum_{e \in \delta^+(v^+)} h_{pe} = g_{pv} + g'_{pv} \qquad \forall p = (s,t) \in \mathscr{P}, \forall v \neq s,t \in V \tag{2.11}$$

$$\sum_{e \in \delta^-(v^-)} h_{pe} = g_{pv} + g'_{pv} \qquad \forall p = (s,t) \in \mathscr{P}, \forall v \neq s,t \in V \tag{2.12}$$

$$g'_{pv} \leq x_v, g_{pv} \leq 1 - x_v \qquad \forall p = (s,t) \in \mathscr{P}, \forall v \neq s,t \in V \tag{2.13}$$

$$d'(p) = \sum_{v \in V} l_v \cdot g_{pv} \qquad \forall p \in \mathscr{P} \tag{2.14}$$

$$\Delta_p = \frac{d(p) - d'(p)}{d(p)} - \beta \qquad \forall p \in \mathscr{P} \tag{2.15}$$

$$budget = \sum_{v \in V} x_v, \qquad budget \leq k, x_v \in \{0,1\} \forall v \in V \tag{2.16}$$

$$\Delta_p \geq -M(1 - x_p), \qquad \Delta_p < M x_p, \forall p \in \mathscr{P} \tag{2.17}$$

$$x_p \in \{0,1\} \qquad \forall p \in \mathscr{P} \tag{2.18}$$

$$h_{pe}, g_{pv}, g'_{pv} \geq 0 \qquad \forall p \in \mathscr{P}, e \in E, v \in V \tag{2.19}$$

The constraints in MIP formulation for POP are shown in Eqs. (2.6-2.19). The constraints as Eqs. (2.6-2.12) are used to model the shortest path delay of each terminal pair as multi-commodity flow. Constraints (2.6-2.10) enforce the nodes s and t to be the source and sink respectively with one unit of flow in each terminal pair (s,t). The next two constraints (2.11,2.12) ensure the flow conservation through the rest of the nodes.

Constraint 2.13 enforces that the upgraded node edge e'_v will carry the flow instead of the original node edge e_v, when the node v is upgraded. The original node edge e_v carries the flow when the node v is not upgraded. Constraint 2.14 computes the total effective delay. Constraint 2.15 computes the difference between effective fractional improvement of each pair and β. Constraint 2.16 computes the total budget and sets the maximum as k. It also ensures that the upgrade decision variables for nodes are binary. Constraints 2.18 and 2.19 ensure that improvement decision variables for pairs and flow variables are binary and non-negative respectively. Constraint 2.17 ensures that x_p is 1 when Δ_p is non-negative.

While the above MIP formulation allows us to compute the optimal solution, it is not scalable to large networks. This motivates us to design approximation algorithms for POP. For any approximation algorithm, it is desirable to provide theoretical guarantees on the approximation error. However, since POP is APX-hard, it is NP-hard to even approximate within a factor of $1-\frac{1}{e}$. Owing to this property, we first consider a restricted version of POP and develop a greedy algorithm. Next, we generalize this greedy algorithm for POP and provide probabilistic error bounds.

2.4.2 Restricted Path Optimization Problem (RPOP)

RPOP introduces the restriction that *each node pair can be β-improved by only one node*. In other words, there cannot be two β-improved nodes in the shortest path between a node pair. We next show that RPOP is a lower bound of POP.

Theorem 4 *Let $f^r(S)$ be the total flow from all pairs that satisfy the RPOP constraint. For any solution set S, $f^r(S) \leq f(S)$.*

Proof. Let P^r and P be the set of all β-improved node pairs under $RPOP$ and POP for a solution set S respectively. Since $P^r \subseteq P$,

$$f^r(S) = \sum_{\forall(u,v)\in P^r} \xi_{u,v} \leq f(S) = \sum_{\forall(u,v)\in P} \xi_{u,v} \quad \Box$$

Since RPOP is a lower bound of POP, intuitively, a good solution to RPOP is likely to yield a good solution to POP as well if the objective functions yield values that are close to each other. With this intuition, we next show that RPOP is monotone and submodular.

Theorem 5 *The objective function $f^r(.)$ is monotone and submodular.*

Proof: Monotone: The proof is similar as in Theorem 3.
Submodular: We consider improvement (delay reduction) of two sets of nodes, V_a and V_b where $V_a \subset V_b$, and show that $f^r(V_a \cup \{v\}) - f^r(V_a) \geq f^r(V_b \cup \{v\}) - f^r(V_b)$ for any node $v \in \Gamma$ such that $v \notin V_a$ and $v \notin V_b$. Let $F(A)$ be the set of node pairs (s,t) which are β-improved by a vertex $v \in A$. Then $f^r(.)$ is submodular if $F(V_b \cup \{v\}) \setminus F(V_b) \subseteq F(V_a \cup \{v\}) \setminus F(V_a)$. To prove this claim, we use the described constraint. Therefore, each pair $(s,t) \in F(V_b)$ is β-improved by only one node in V_b. As $V_a \subset V_b$, adding v to V_a will β-improve some of the pairs which are already β-improved by $V_b \setminus V_a$. Then, for any newly β-improved pair $(s,t) \in F(V_b \cup \{v\}) \setminus F(V_b)$, it must hold that $(s,t) \in F(V_a \cup \{v\}) \setminus F(V_a)$. ■

Theorem 6 *RPOP is APX-hard, i.e., RPOP cannot be approximated within a factor greater than $(1 - \frac{1}{e})$.*

Proof: The proof directly follows from Theorem 2 as the construction respects the described constraint. ■

Since the objective function $f^r(.)$ is submodular and monotone, the greedy algorithm of iteratively adding the node with the maximum marginal gain on Eq. 2.2 approximates RPOP within a factor of $(1-\frac{1}{e})$ [60]. This result, in conjunction with Theorem 6, allows us to conclude that greedy is the best possible polynomial-time algorithm for RPOP.

The above conclusion motivates us to propose a greedy heuristic (GSN, Algorithm 1) for optimizing POP as well. More specifically, we know from Theorem 4 that RPOP is a lower bound for POP. Furthermore, from an intuitive point of view, in most real life networks the number of node pairs is much larger than the budget k. Thus, the likelihood of having more than one improved node in the shortest path between a randomly selected node pair is low. Consequently, it is likely that the lower bound is tight.

However, an important question remains. *Can we provide bounds on the quality of the greedy solution for POP?* We answer this question through the *sandwich theorem [61].* The idea of the sandwich theorem is as follows: First, run the greedy algorithm on the actual function ($f(.)$) and its lower bound ($f^r(.)$). Let S' and S^r be the produced solution sets respectively. If $S = \arg\max_{S\in\{S',S^r\}} f(S)$, $f(S)$ has the following lower bound:

Theorem 7 ***Sandwich Theorem:*** *If f^r is a lower bound of f and f^r is monotone and submodular, then*

$$f(S) \geq \mathscr{C} \cdot (1-\frac{1}{e}).f(S^*) \tag{2.20}$$

where $\mathscr{C} = \frac{f^r(S^*)}{f(S^*)}$ and S^* is optimal solution under cardinality constraint for function $f(.)$.

Proof. $f(S) \geq f^r(S) \geq (1-\frac{1}{e})f^r(S^*)$

$$\geq \mathscr{C} \cdot (1-\frac{1}{e}).f(S^*) \text{ Since } f^r(S^*) = \mathscr{C} f(S^*)\square$$

Theorem 7 says that the performance of greedy on POP is directly proportional to the

Algorithm 1 Greedy Selection of Nodes (GSN)

```
Require: Network $G = (V, E, L)$ with vertex delays $l(v)$, network flows $\mathbb{F}$, Budget $k$,
    candidate set $\Gamma$, $\beta$.
Ensure: A subset of $k$ nodes
 1: $S \leftarrow \emptyset$
 2: Compute the shortest path between all nodes that is part of some node pair with
    non-zero flow to all other nodes in the network. Store the corresponding shortest
    path delays in distance matrices $A, B$
 3: while $|S| < k$ do
 4:     for $v \in \Gamma \setminus S$ do
 5:         $\lambda_v \leftarrow \sum_{(x,y)\in\Lambda_s} \xi_{x,y}$ where $\Lambda_s$ is the set of new $\beta$-improved pairs when $l_v = 0$
            (Use $A$ to compute $\beta$-improvement and $B$ to compute marginal gain)
 6:     end for
 7:     $v' \leftarrow \arg\max_{v\in\Gamma\setminus S}\{\lambda_v\}$ and then set $l(v')$ as 0
 8:     $S \leftarrow S \cup \{v'\}$
 9:     Update $d(s,t)$ in $B$ as $l(v')$ becomes 0
10: end while
11: return $S$
```

tightness of RPOP. More specifically, $\mathscr{C} = \frac{f^r(S^*)}{f(S^*)}$ quantifies the tightness of RPOP. The closer the ratio is to 1, the better is the approximation quality. Our empirical evaluation in Section 2.5 reveals that $\mathscr{C}$ typically lies in the range $[0.5, 1]$.

2.4.3 Greedy Selection of Nodes

Algorithm 1, called *GSN*, outlines the pseudocode of the greedy algorithm for POP.

In each iteration, it selects the node that produces the maximum marginal gain on the total flow from β-improved pairs given the current solution set, S (step 7). To enable this operation, first, GSN pre-computes the shortest path delays between any node that is part of at least one node pair with a positive flow to all other nodes in the network and stores them in matrices A and B (step 2). Note, we ignore shortest paths between nodes u and v, if neither u nor v is part of some node pair with a positive flow, since such paths do not contribute to the flow improvement. If the network is undirected, computing only the upper half of the distance matrix is enough since distances are symmetric. Stored path delays in A remain unchanged through out the algorithm. On the other hand, B stores the updated shortest path delays following the node upgrades made in S. A is used to determine if a pair has been β-improved and B is used to compute the marginal gain of a node upgrade. S is populated iteratively (3-9) and B is updated in each of these iterations (line 9). Finally, after k iterations, the solution set S is returned (line 11).

Example 2 *Figure 2.2 shows a possible solution of size $k = 2$ for a small network. The settings are already described before. In the first step, GSN can choose either node c or e as both of them individually β-improve (the initial shortest path delay, 15 becomes 5) the pair (d, f) and thus improves flow of $\frac{50}{150}$. If node c is chosen, then in second iteration, GSN will choose node b as it β-improves (the initial distance, 35 becomes 5) the pair (a, d) and thus flow of $\frac{100}{150}$.*

Cost Analysis

Computation Cost: The most important steps in GSN are lines $2, 5$ and 9. In the worst case, line 2 computes all-pairs-shortest-paths in time $O(n^2 \log n)$, where n is the total number of vertices. Next, it chooses, among the candidate nodes Γ, the one that maximizes the number of β-improved pairs (line 5), which takes $O(|\Gamma|n^2)$ time. As the shortest path distances are stored, the computation of β-improved pairs takes $O(n^2)$ for each candidate node. After choosing the best node and improving its delay, the shortest path distances are updated in B (line 9) consuming $O(n^2)$ time. Therefore, the total running time of GSN is $O(n^2 \log n + k|\Gamma|n^2)$.

Memory footprint: Since in the worst case we need to store the distance matrix for all pairs shortest path delays, the memory footprint is $O(n^2)$.

Both the computation cost and the memory footprint are prohibitively large for large networks. Clearly, a more efficient approach than greedy (Alg. 1) is required. Towards that goal, we expedite greedy through sampling and provide theoretical guarantees on the sampling quality.

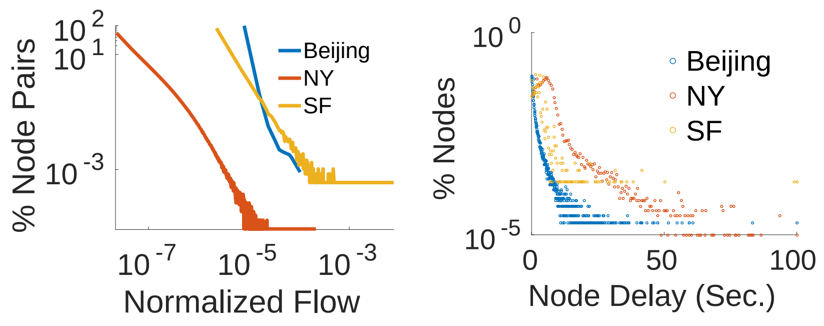
2.4.4 Sampling

As in any sampling algorithm, the goal is to carefully select a subset of the data and compute the answer set by only analyzing this subset. In our particular case, the goal is to sample a subset of node pairs, and execute the greedy algorithm only on the sampled subset. The sampling algorithm is effective if the computed answer set is as good as the answer set that would be computed if the entire dataset is processed. The key step towards ensuring this quality control is to choose a subset that is representative of the entire data for the task in hand. Towards that end, we first consider the naïve approach of uniform sampling.

Uniform Sampling

As the name suggests, in this procedure, we sample uniformly with replacement a set of ordered node pairs $\mathcal{U}$ from the set of all node pairs in $V \times V$. Although the approach is simple, this sampling algorithm under-utilizes the information hidden in node pair flows. More specifically, not all node pairs are of equal importance. In our optimization function (Eq. 2.2), improving the shortest paths on important node pairs has a much more profound impact than improving pairs that have negligible flow between them.

To better understand the impact of ignoring node pair importance, in Figure 2.4a, we plot their distribution on three real road networks of Beijing, New York, and San Francisco. The flow of a node pair (a, b) is the proportion of taxi trips from location a



(a) Flow (b) Delay

Figure 2.4: The distribution of (a) node-pair flows and (b) node delays in three real road network datasets.

to b. As can be seen, the distribution follows a power law, which means a small set of node pairs are highly more popular than the remaining majority. Since a small minority of the node pairs contribute majority of the node flows, uniform sampling is unlikely to have a large enough sample of these important node pairs. Consequently, the estimates computed from the sampled sets may suffer. To overcome this drawback, we propose *Importance Sampling*.

Importance sampling

Importance sampling [62, 63] samples elements proportional to their importance. When applied to POP, importance sampling samples node pairs proportional to their flow value. Consequently, the sampling is biased towards node pairs that are more relevant to compute the given estimate. The formal definition is as follows.

Definition 7 (Importance Sampling) *In importance sampling, a node pair* (s, t) *is selected (with replacement) in the sampled set* $\mathcal{I}$ *with probability* $\hat{p}_{s,t} = \xi_{s,t}$.

Next, we show that importance sampling is an unbiased estimator of the entire set. A sampling procedure is *unbiased* if it is possible to estimate the mean of the entire set from the sampled set. More formally, $\mathbb{E}[\hat{\mu}(I)] = \mu_\xi = \frac{\sum \xi_{s,t}}{n(n-1)} = \frac{1}{n(n-1)}$, where $\hat{\mu}(\mathcal{I})$

is the mean estimator. We define $\hat{\mu}(\mathcal{I})$ as the *weighted average* over the samples in $\mathcal{I}$: $\hat{\mu}(\mathcal{I}) = \frac{1}{\sum_{(s,t)\in\mathcal{I}} \hat{w}_{s,t}} \sum_{(s,t)\in\mathcal{I}} \hat{w}_{s,t} \cdot \xi_{s,t}$, where $\hat{w}_{s,t} = \frac{1}{\hat{p}_{s,t}}$. Next, we show $\hat{\mu}(\mathcal{I})$ is an unbiased estimator of μ_ξ.

Lemma 1 *$\hat{\mu}(\mathcal{I})$ is an unbiased estimate of μ_ξ when $\hat{w}_{s,t} = \frac{1}{\hat{p}_{s,t}}$, i.e., $\mathbb{E}[\hat{\mu}(\mathcal{I})] = \mu_\xi$.*

Proof: $\mathbb{E}[\hat{\mu}(\mathcal{I})] = \frac{1}{\mathbb{E}[\sum_{(s,t)\in\mathcal{I}} \hat{w}_{s,t}]} \cdot \mathbb{E}\left[\sum_{(s,t)\in\mathcal{I}} \hat{w}_{s,t} \cdot \xi_{s,t}\right]$

If we simplify the first term, we obtain

$$\mathbb{E}\left[\sum_{(s,t)\in\mathcal{I}} \hat{w}_{s,t}\right] = |\mathcal{I}|.\mathbb{E}[\hat{w}_{s,t}] = |\mathcal{I}|. \sum_{\forall(s,t)\in V\times V} \hat{w}_{s,t} \cdot \xi_{s,t} = n(n-1)|\mathcal{I}|$$

where, $n = |V|$. From the second term, we get,

$$\mathbb{E}\left[\sum_{(s,t)\in\mathcal{I}} \hat{w}_{s,t} \cdot \xi_{s,t}\right] = |\mathcal{I}|\mathbb{E}[\hat{w}_{s,t} \cdot \xi_{s,t}] = |\mathcal{I}| \sum_{\forall(s,t)\in V\times V} \hat{p}_{s,t}(\hat{w}_{s,t} \cdot \xi_{s,t}) = |\mathcal{I}|$$

Combining these two, $\mathbb{E}[\hat{\mu}(\mathcal{I})] = \frac{|\mathcal{I}|}{n(n-1)|\mathcal{I}|} = \mu_\xi$. ■

Armed with an unbiased estimator, we show that by carefully choosing the size of the sample set, importance sampling provides an accurate estimation of the marginal gain of adding a node to the solution set (line 5 in Alg. 1). More formally, we prove the following.

Theorem 8 *Let S be the solution set in the current iteration and let $\mathcal{I}$ be the set of sampled node pairs. In this setting, let v_g be the node with the highest marginal gain on the entire set of node pairs and v_s be the highest one when only considering the sampled set $\mathcal{I}$. The difference in the flow is bounded as follows: $Pr\left[|f(S \cup \{v_g\}) - f(S \cup \{v_s\})| < \varepsilon\right] > 1 - \frac{1}{n^2}$, where $|\mathcal{I}|$ is $O(\frac{c \cdot \log n}{\varepsilon^2})$, $c = (\frac{\xi_m}{\mu_\xi})^2$, ε is the error bound, and ξ_m and μ_ξ are the maximum and average flow of all pairs of nodes respectively.*

PROOF. Let $\mu_g = \frac{f(S \cup \{v_g\})}{n(n-1)}$ and $\mu_s = \frac{f(S \cup \{v_s\})}{n(n-1)}$ denote the corresponding means and Y_g and Y_s be the corresponding *expected* means from the samples.

The samples can be viewed as random variables associated with the selection of a pair of nodes. More specifically, the random variable, X_i, is the flow associated with the i-th pair of vertices in the importance sample $\mathcal{I}$. Since the samples provide an unbiased estimate (Lemma 1) and are i.i.d., we can apply *Hoeffding's inequality* [64] to bound the error of the mean estimates:

$$Pr[|Y_g - \mu_g| \geq \theta] \leq \delta \tag{2.21}$$

where $\delta = 2\exp\left(-\frac{2|\mathcal{I}|^2\theta^2}{\mathcal{T}}\right)$, $\mathcal{T} = \sum_{i=1}^{|\mathcal{I}|}(b_i - a_i)^2$, where each X_i is strictly bounded by the intervals $[a_i, b_i]$. Similarly,

$$Pr[|Y_s - \mu_s| \geq \theta] \leq \delta \tag{2.22}$$

Applying union bound,

$$Pr[(|Y_g - \mu_g| \geq \theta) \cup (|Y_s - \mu_s| \geq \theta)] \leq 2\delta \tag{2.23}$$

By construction, $\mu_g \geq \mu_s$ as GSN selects the best next node at each step. On the other hand, if importance sampling selects v_s, it must be that $Y_s \geq Y_g$. As, the sampled best node is probabilistic, we need to apply *union bound* over n possible nodes. As a consequence, $Pr[|\mu_g - \mu_s| \geq 2\theta] \leq 2n\delta$ and $Pr[|\mu_g - \mu_s| < 2\theta] > 1 - 2n\delta$.

$$
\begin{aligned}
&\text{Now, } Pr\left[|f(S \cup \{v_g\}) - f(S \cup \{v_s\})| < \varepsilon\right] \\
=&Pr[|\mu_a - \mu_g| < \frac{\varepsilon}{n(n-1)}] \\
>&1 - 4n \exp\left(-\frac{2|\mathcal{I}|^2 \left(\frac{\varepsilon}{2n(n-1)}\right)^2}{\mathcal{T}}\right)
\end{aligned}
\tag{2.24}
$$

Since the average flow $\mu_\xi = \frac{1}{n(n-1)}$ and $\xi_m \geq b_i$, $a_i \geq 0$, we get $\mathcal{T} \leq |\mathcal{I}| \cdot \xi_m^2$. Combining these factors, we get,

$$
\begin{aligned}
&1 - 4n \exp\left(-\frac{2|\mathcal{I}|^2 \left(\frac{\varepsilon}{2n(n-1)}\right)^2}{\mathcal{T}}\right) \\
>&1 - 4n \exp\left(-\frac{2|\mathcal{I}|^2 \left(\frac{\varepsilon \cdot \mu_\xi}{2}\right)^2}{|\mathcal{I}| \cdot \xi_m^2}\right)
\end{aligned}
\tag{2.25}
$$

Combining Eq. 2.24 and Eq. 2.25, we get

$$
\begin{aligned}
&Pr\left[|f(S \cup \{v_g\}) - f(S \cup \{v_s\})| < \varepsilon\right] \\
>&1 - 4n \exp\left(-\frac{|\mathcal{I}| \left(\varepsilon . \mu_\xi\right)^2}{2\xi_m^2}\right)
\end{aligned}
\tag{2.26}
$$

By setting the number of samples $|\mathcal{I}| = \frac{2\xi_m^2 \cdot \log(4n^3)}{(\varepsilon \cdot \mu_\xi)^2}$, we have

$$
Pr\left[|f(S \cup \{v_g\}) - f(S \cup \{v_s\})| < \varepsilon\right] > 1 - \frac{1}{n^2}
$$

Remarks: Theorem 8 shows the number of samples required to keep the approximation error compared to the greedy approach in Alg. 1, sufficiently low. The key results

Algorithm 2 Importance Sampling's Selection (ISS)

```
Require: Network $G = (V, E, L)$, Approximation error $\varepsilon$, Sampling factor $c$, Budget $k$, candidate node set $\Gamma$, network flows $\mathbb{F}$, $\beta$.
Ensure: A subset of $k$ nodes
 1: Choose $O(c(\log n)/\varepsilon^2)$ sample pairs of vertices in $\mathcal{I}$ via importance sampling
 2: $A \leftarrow$ A distance matrix containing the distance of all nodes appearing in the sampled set $|\mathcal{I}|$ to all other nodes in the network.
 3: $S \leftarrow \Phi$
 4: while $|S| < k$ do
 5:     for $(s, t) \in \mathcal{I}$ do
 6:         $B \leftarrow$ Re-compute the distances in matrix A after considering the upgraded nodes in $S$.
 7:     end for
 8:     for $v \in \Gamma$ do
 9:         $\lambda_v \leftarrow \sum_{(x,y)\in\Lambda_s} \xi_{x,y}$ where $\Lambda_s$ is the set of new $\beta$-improved pairs in $\mathcal{I}$ when $l(v) = 0$ (Use $A$ to compute $\beta$-improvement and $B$ to compute marginal gain)
10:     end for
11:     $v' \leftarrow \arg\max_{v\in\Gamma\setminus S}\{\lambda_v\}$, $l(v') \leftarrow 0$
12:     $S \leftarrow S \cup \{v'\}$
13: end while
14: Return $S$
```

are as follows.

- $|\mathcal{I}|$, which is the sample size, is a function of the error ε and inversely proportional to ε. Thus, with higher number of samples, our estimates get more accurate.
- The sample size grows logarithmically with the network size.

Efficient greedy through Importance Sampling

Armed with Theorem 8, we next describe how Importance Sampling is used to speed up greedy. We call this algorithm *Importance Sampling's Selection (ISS)*. Alg. 2 presents the pseudocode. In simple terms, Theorem 8 is used to first decide the sample size (line 1), and the Greedy selection of node upgrades is performed by analyzing the impact of an upgrade only on the sampled set of node pairs. As in greedy(Alg. 1), the algorithm runs for k iterations and in each iteration, the best node from the candidate set is selected based on the sampled node pairs (lines 4-13). Instead of computing the entire distance matrix, we compute the distance from all nodes in the network to only those nodes that appears in at least one sampled pair (lines 2 and 6). These distances provide the β-improved pairs in the sampled set $\mathcal{I}$ of pairs of nodes. The remaining operations remain same as in greedy (Alg. 1).

Computation Cost: First, we sample $|\mathcal{I}|$ node pairs based on importance sampling. Recall from Def. 6, that the importance of a node pair (u, v) is the proportion of flows from u to v. Let d be the total number of network flows (Def. 2) in the dataset and each flow is assigned an ID from 1 to d. To sample $|\mathcal{I}|$ node pairs proportional to their importance, we generate $|\mathcal{I}|$ random IDs in the range $[1, d]$ and extract the corresponding flows. The origin and destination nodes of each of these sampled flows form a node pair in our sample set. The sampling procedure consumes $O(|\mathcal{I}|)$ time. As defined in Def. 7, the sampling is performed *with replacement*. Hence, the number of times a node pair appears in $\mathcal{I}$ is proportional to its importance.

The costliest steps of our algorithm are lines 2, 5-7 and 8-12. In line 2, we compute a distance matrix A, which stores the initial shortest path distances between all nodes that appear in some pair in $|\mathcal{I}|$ to all other nodes in the network. This computation consumes $O(|\mathcal{I}|n \log n)$ time since the shortest path is computed between each node of a sampled pair to all other nodes. In steps 5-7, ISS recomputes the shortest paths based on the node upgrades made so far. This operation again consumes $O(|\mathcal{I}|n \log n)$ time. Next, the algorithm estimates the additional number of β-improved pairs after removing the delay of each of the nodes in the candidate set Γ. (steps 8-10). As the shortest path distances of all sampled nodes are stored in A, the computation of β-improved pairs takes $O(|\mathcal{I}|)$ for each candidate node upgrade. Therefore, the best node to be upgraded is selected in $O(|\Gamma||\mathcal{I}|)$ time (step 11-12). This entire process from line 5-12 is then repeated k times to select the k nodes upgrade to be performed. Combining all these factors, the total running time of ISS is $O(k|\mathcal{I}|n(\log n) + k|\Gamma||\mathcal{I}|)$. Since $|\mathcal{I}|$ is a logarithmic function of n (i.e., $|V|$), the complexity of ISS reduces to $O(kn(\log^2 n) + k|\Gamma|\log n)$. Recall that the computation cost of greedy (Alg. 1) is $O(n^2 \log n + k|\Gamma|n^2)$. Consequently, we get a dramatic reduction in running time.

Storage Cost: In addition to the network ($O(|V| + |E|)$ space), ISS stores $O(|\mathcal{I}|)$ node pairs and their importance. The matrices A and B consume $O(|\mathcal{I}||V|)$ space. Since $|\mathcal{I}|$ is a logarithmic function of n (i.e., $|V|$), the space complexity is bounded by $O(|\mathcal{I}| + |\mathcal{I}||V| + |V| + |E|) = O(n \log n + m)$.

2.5 Experiments

In this section, we benchmark the proposed algorithms and evaluate their approximation quality and scalability.

2.5.1 Experimental Setup

All experiments are performed using Java on an Intel(R) Xeon(R) E5-2609 8-core machine with 2.5 GHz CPU and 512 GB RAM running Linux Ubuntu 14.04.

Baselines

We denote our importance sampling algorithm as *ISS*, the optimal algorithm based on mixed integer programming as *MIP-OPT*, and the greedy algorithm as *GSN*. The other baselines are as follows: (1) **USS (Uniform Sampling's Selection):** We adapt the state-of-the-art method [11] of uniform sampling and apply towards our problem. (2) **High-Cen [24]:** We choose the top-k most central nodes to improve. The central nodes are present on the maximum number of distinct shortest paths and could potentially β-improve a large number of node pairs. However, this method does not capture the

Name	**#Trajectories**	**Type**	$\lvert V \rvert$	$\lvert E \rvert$
Bejing (BJ)	123K	Directed	623.9K	672.2K
San Fransisco (SF)	442K	Undirected	5.08K	41.7K
New York (NY)	49247K	Directed	72.7K	169.8K

Table 2.2: Dataset description and statistics.

dependency among node upgrades. (3) **High-Delay:** This baseline selects the top-k nodes with highest delays.

Datasets:

Table 2.2 summarizes the real-world datasets. Each dataset contains the road network of a city and is extracted from *OpenStreetMap*[1]. Each node corresponds to a region and an edge denotes a street connecting these regions. For network flows, we use the cab trajectory data from each of the cities listed in Table 3.3.

(1) **Beijing (BJ)[65]:** We have trajectories of cabs, over a period of 1 week. Each trajectory contains the sequence of nodes visited in a trip and the timestamps at which the nodes were visited.

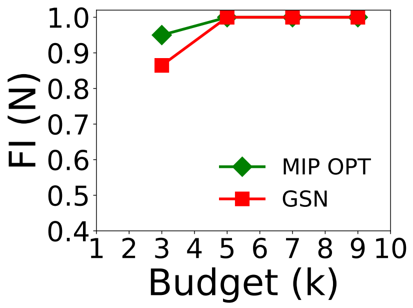
(2) **San Francisco (SF)[66]:** The dataset has been collected over a duration of one month and contains taxi pick-up and drop-off information from taxis in San Francisco.

(3) **New York (NY) [67]:** This is the largest publicly available taxi dataset. It was collected over a period of four years ranging from 2009 to 2013. It contains records of yellow and green taxis in the city of New York. We use the trajectories from January, 2013 to March, 2013 to find the importance of the node pairs.

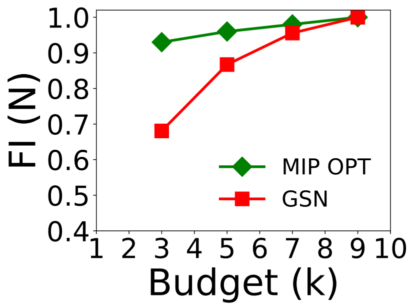
To compute the delay in a node, we first partition the cab trajectories based on their starting timestamp into four windows of six hours each viz. 00:00 to 06:00, 06:00 to 12:00, and so on. Next, for each edge, we compute the average time taken to cross it in each of the four time slots. The delay in an edge is quantified as the difference between the

[1]`http://openstreetmap.org/`

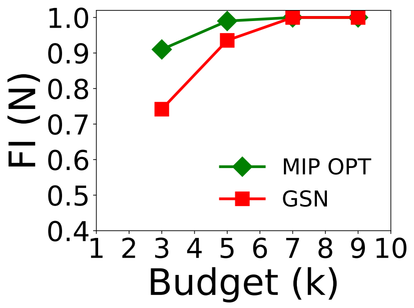
maximum and the minimum average times across the four windows. Finally, the delay in a node is set to the sum of the delays in all of its incoming edges.



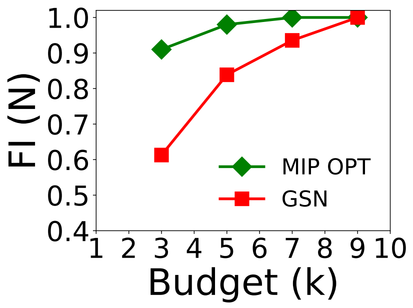
(a) SF ($\beta = 5\%$)



(b) SF ($\beta = 10\%$)



(c) BJ ($\beta = 5\%$)



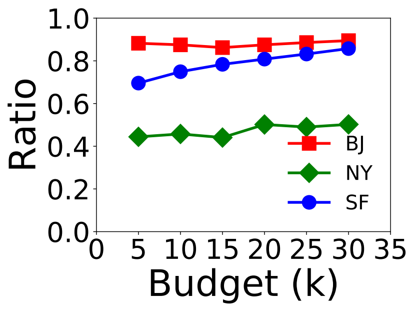
(d) BJ ($\beta = 10\%$)

Figure 2.5: Comparison between MIP vs GSN: Normalized Flow Improvement for (a-b) SF, (c-d) BJ.

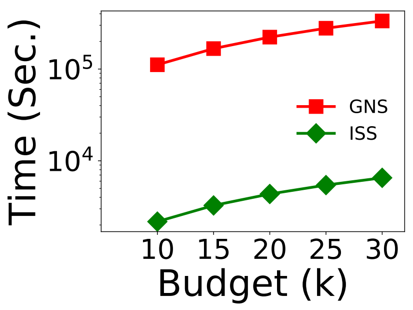
Performance metric and parameters

Performance Metric: The quality of a solution set S in a network N is defined in Eq. 2.2. We call this metric the *flow improvement* due to S and is denoted as $FI(N)$. From our formulation of node-pair importance in Def. 6, the total flow in a network is 1.

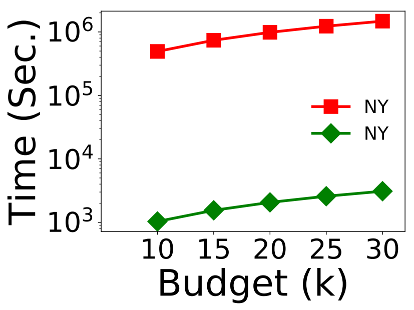
The solution set S by any algorithm is evaluated based on $f(S) = \sum_{(u,v)\in\Lambda_S} \xi_{u,v}$ where Λ_S denotes the set of β-improved node pairs. For large graphs, this evaluation is time consuming as it involves all-pair-shortest-paths computation. To mitigate this scalability bottleneck, we evaluate the algorithms based on randomly chosen sampled pairs Q. More



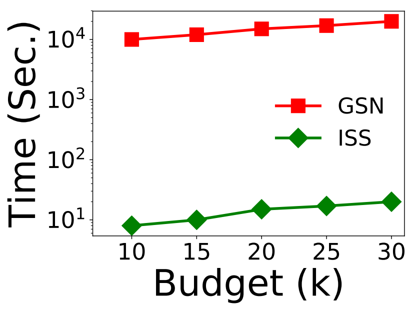
(a) Ratio



(b) Running Time in BJ



(c) Running Time in NY



(d) Running Time in SF

Figure 2.6: (a) Approximation quality of ISS against GSN. (b-d)Running times of ISS and GSN.

specifically, the metric is $f(S) = \sum_{(u,v)\in\Lambda^*_S} \xi_{u,v}$, where Λ^*_S denotes the set of β-improved node pairs in Q. The size of the sample set Q is set to $50,000$.

Default Parameters: We set Γ, which is the candidate set of nodes that can be improved, to V, i.e., the set of all nodes. Unless specifically mentioned, the default value for β is 0.1 or 10%, and the default size of sample set used in ISS (and USS) is $15\log(n)$. Note that, we use the number samples in the form of $c\log(n)$ where c controls the error ε (mentioned in Theorem 8).

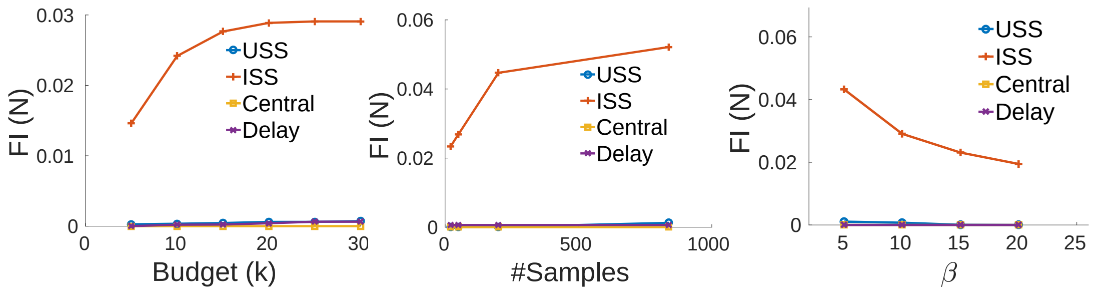
2.5.2 GSN and optimal MIP

First, we compare the performance of GSN with the optimal solution. As described in Sec. 2.4.1, the optimal solution is computed using mixed integer programming (MIP). We implement MIP using CPLEX and validate on the SF and BJ datasets. However, we extract a sub-network containing only 1000 nodes from these datasets since MIP consumes exorbitantly high running times on larger networks. Figure 2.5 presents the results as we vary β and the budget k. Across both datasets, GNS produces results that are close to optimal. More specifically, beyond $k = 5$, GSN is at most 10% away from the flow improvement achieved by the optimal algorithm. This result validates our intuition that greedy is an effective heuristic for the proposed problem. Moreover, GSN takes only a few seconds (< 100 seconds) to produce the nearly optimal results whereas MIP takes more than a day's time to terminate. A pattern consistent across both datasets is that as β increases, the gap between GSN and optimal increases as well. This trend is a direct consequence of the fact that when β increases, only a group of node upgrades can β-improve a path and hence higher is the need to fully search the combinatorial space and identify the best group.

2.5.3 ISS vs GSN

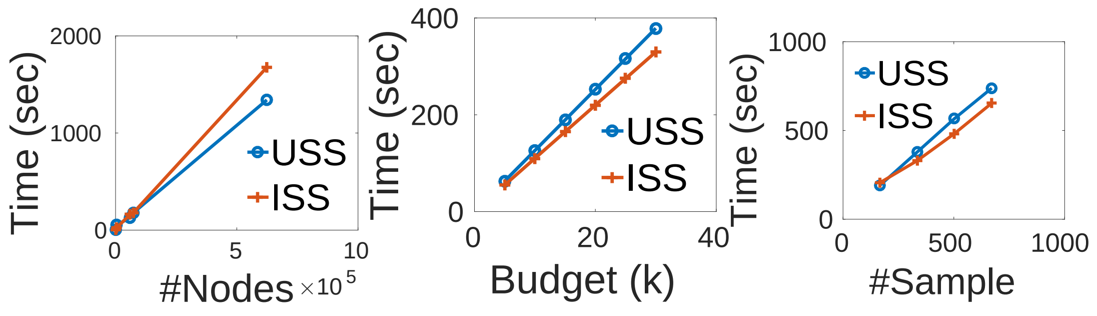
We next compare the performance of GSN with ISS. In Fig. 2.6a, we plot the approximation quality of ISS with respect to GSN. The approximation quality is the ratio between the flow improvements produced by ISS and GSN. A high ratio indicates that ISS produces similar quality results as that of GSN. As can be seen, the ratio is the highest in BJ, followed by SF and finally NY. This result follows directly from the distribution of node pair importances. Specifically, it is evident from Fig. 2.4a, that the node-pair importances are most skewed in BJ, followed by SF, and finally NY. When the distribution is skewed towards a small number of important node pairs, importance sampling is better able to estimate the marginal gain of a node upgrade from just the sampled collection of node pairs. This results in the trend visible in Fig. 2.6a.

Next, we analyze the running times of GSN and ISS. Figs. 2.6b-2.6d present the



(a) Varying Budget (b) Varying #Sample (c) Varying β (%)

Figure 2.7: (NY) The flow improvement (FI) by varying (a) budget, (b) the number of samples, and (c) β (%).



(a) Scalability (b) Vs Budget (c) Vs #Sample

Figure 2.8: (a-c) The running times of ISS and USS by varying (a) (All) the size of graphs, (b) (NY) the budget, (c) (NY) the number of samples.

results. ISS is up to three orders of magnitudes faster than GSN. GSN finds it most difficult to scale in the NY dataset, where it consumes around 17 days (410 hours) to terminate. Although, NY is a smaller network than BJ, GSN consumes more time since the number of node pairs with non-zero importance is much higher in NY and therefore more shortest path computations are necessary while calculating the marginal gain of an upgrade.

2.5.4 Comparison with scalable baselines

Next, we compare the performance of ISS with the baseline algorithms listed in Sec. 2.5.1. We omit GSN from further experiments since it fails to scale.

Tables 2.3 and 2.4 present the results for NY and BJ respectively. To highlight the

	$\beta = 5\%$			$\beta = 10\%$		
k	**High-Cen**	**USS**	**High-Delay**	**High-Cen**	**USS**	**High-Delay**
5	∞	27.5	350	∞	50.3	2085
10	36000	37.5	110	∞	69.6	115
15	40000	40	80	∞	46	120
20	42000	42	56.5	20624	47.2	72
25	43000	43	54.8	20772	47.5	46
30	21500	43	54	21000	40.9	46

Table 2.3: (NY) Comparison of *Flow Improvement* between ISS and other baselines on NY against varying budget (k) and β. Each cell reports the relative Improved Flow w.r.t. ISS, i.e. $\frac{FI_{ISS}}{FI_X}$, where X is the method used in that particular cell. When X does not produce any FI ($FI_X = 0$), $\frac{FI_{ISS}}{FI_X} = \infty$.

	$\beta = 5\%$			$\beta = 10\%$		
k	**High-Cen**	**USS**	**High-Delay**	**High-Cen**	**USS**	**High-Delay**
5	4.0	25.1	∞	∞	∞	∞
10	9.1	10.3	∞	∞	∞	∞
15	5.1	12.2	∞	9.1	∞	∞
20	2.5	14.8	∞	6.2	∞	∞
25	1.8	16.8	∞	3.1	∞	∞
30	1.7	17.7	∞	2.4	∞	∞

Table 2.4: (BJ) Comparison of *Flow Improvement* between ISS and other baselines on BJ against varying budget (k) and β.

efficacy of ISS more prominently, we report the *performance improvement* of ISS in terms of ratio. More specifically, each cell reports the ratio

$$\text{Performance improvement}(X) = \frac{FI_{ISS}}{FI_X} \tag{2.27}$$

where X is the method corresponding to the cell's column and FI_X is the flow improvement achieved by that method.

Across both NY and BJ, ISS drastically outperforms all baselines. Some baselines fail to provide any flow improvement at all and therefore the improvement ratio in those cases is ∞. This impressive performance of ISS stems from two factors. First, unlike ISS, both

High-Cen and High-Delay are oblivious to the dependencies between the nodes selected for reduction. Second, although this problem does not exist in USS, USS samples node pairs uniformly and therefore fails to capture an adequate representation of the important node pairs. Consequently the performance suffers.

A consistent trend we see across both NY and BJ is that as β increases the performance gap between ISS and other baselines increases. This is another natural consequence of being oblivious to the node dependencies. At a higher β, it is even more important to know the other nodes in the solution set since such high improvement is typically possible only when all nodes in the solution set collectively bring down the shortest path delay by $\beta\%$.

2.5.5 Impact of Parameters on Performance

The key parameters impacting the performance are the budget k, the number of samples, and β. Note that the number of samples affects the error ε (Theorem 8). The running time is affected by the budget, the number of samples and the size of the graph. We study the impact of these parameters on the quality and running time.

Quality

First, we evaluate the quality in terms of Flow Improvement (FI) against varying budget. Figure 2.7a shows the results for NY. As the budget increases ISS shows growth in improving the flows and outperforms all other baselines convincingly.

Next, we vary the number of samples and observe its effects in Figure 2.7b. The budget (k) is set to 50. Though ISS is able to show a steady growth in quality, others struggle to make a noticeable impact. This behavior is a direct consequence of not being sensitive to the importance of node pairs. More specifically, all techniques except ISS randomly sample node pairs. As shown in Fig. 2.4a, the node pair importance follows a power-law distribution and thus a small minority of the pairs contribute majority of the network flows. The chances of these important pairs to get randomly sampled is extremely low and thus the performance of the other baselines suffer.

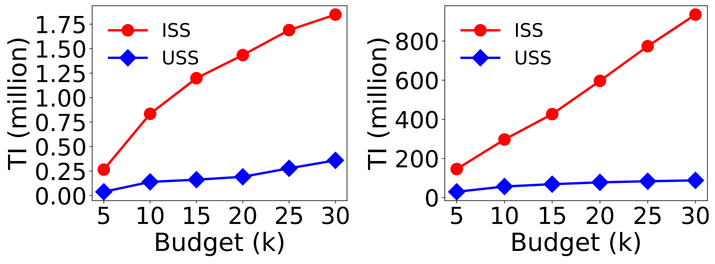
Finally, we show the quality of the algorithms varying β. With the increase of β, the possibility of improving a node pair reduces and thus the number of improved flows should reduce. Figure 2.7c validates this intuition.

Scalability

First, we study the growth rate of running time against the network size. In this experiment, we compute the running time for each of the datasets listed in Table 2.2 and verify how the running time grows with respect to the network size. Figure 2.8a presents the results. The running time grows at a rate that is slightly higher than linear with increase in network size. This result is consistent with the theoretical analysis of the computation cost (Section 2.4.4), where we show that the computation cost of ISS grows at $O(n \log^2 n)$ with respect to the network size. On the largest network of Beijing, ISS finishes with 30 minutes.

Next, we evaluate the growth rate of running time against budget on the NY dataset. Figure 2.8b presents the results. As expected from the theoretical analysis of computation cost in Section 2.4.4, the running time grows linearly and consumes less than 7 minutes across all values of k.

Finally, we analyze scalability against the number of samples on NY dataset and present the results in Fig. 2.8c. Here, we fix the budget as 30 and vary the number of samples. As expected, the running time grows linearly with the number of samples.



(a) BJ (b) NY

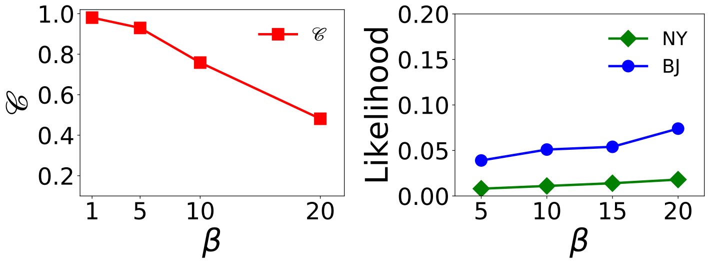
Figure 2.9: The total improvement (TI) on (a) BJ and (c) NY.

2.5.6 Total Improvement

In our analysis, we have argued that noticeable improvement for each individual pair of nodes is important. Does this focus on individual improvement compromise on the total improvement across the entire network? In the next experiment, we analyze this question by measuring the performance of ISS on the *total improvement (TI)* metric. TI is the total reduction in delay (in minutes) across all trajectories of the datasets. TI is defined as follows.

$$TI = \sum_{\forall P_{u,v} \in \mathbb{F}} \big(d(u,v) - d(u,v;S)\big) \tag{2.28}$$

Here, $\mathbb{F}$ is the set of all trajectories, $P_{u,v}$ denotes a trajectory that starts at node u and terminates at v. S is the set of upgraded nodes and $d(u,v)$ and $d(u,v;S)$ are the delays of the shortest paths from u to v before node upgrades and after node upgrades respectively. We compare the performance of ISS with the state-of-the-art method USS [11], which is designed specifically for total improvement. For ISS, we set $\beta = 0$ since the goal is to optimize total improvement. Fig. 2.9 presents the results in NY and BJ. As visible, ISS is up to 8 times better than USS. USS is blind to the idea of node pair importance and hence its performance suffers.



(a) $\mathscr{C}$ in BJ (b) Likelihood

Figure 2.10: (a) Tightness of RPOP and (b) The likelihood to have more than one improved node in the shortest path between a randomly selected node pair against β (in percentage).

2.5.7 Tightness of RPOP

Recall that our motivation to use greedy as the optimization strategy for POP emerged from the observation that RPOP is a lower bound of POP (Section 2.4.2) and greedy is the optimal polynomial-time algorithm for RPOP. Theorem 7 establishes an error bound on the performance of greedy on POP as a function of the tightness factor $\mathscr{C} = \frac{f^r(S)}{f(S)}$, between RPOP and POP. In the next, experiment, we analyze how the tightness varies with increase in β.

Fig. 2.10a presents the results in the BJ dataset at $k = 30$. $\mathscr{C}$ decreases with increase in β. Unsurprisingly, When β is high, it is hard for a single node upgrade to β-improve a path. Consequently, multiple node upgrades in a path are necessary to achieve β-improvement. Since RPOP allows only one node upgrade per path, $f^r(S)$ (RPOP) stays much lower than $f(S)$ (POP). As a result $\mathscr{C}$ decreases. Overall, $\mathscr{C}$ lies in the range $[0.5, 1]$.

We next analyze another assumption regarding RPOP. We claim in Section 2.4.2 that intuitively, in most real life networks the number of node pairs is much larger than the budget k. Thus, the likelihood of having more than one improved node in the shortest path between a randomly selected node pair is low. Consequently, POP should behave

	Gaussian		**Uniform**	
k	**High-Cen**	**USS**	**High-Cen**	**USS**
5	∞	∞	∞	∞
10	∞	829	1700	∞
15	∞	1400	5800	1600
20	∞	1600	6900	1900
25	∞	1700	8300	2200
30	∞	1900	9200	2500

Table 2.5: (Synthetic Delay:) Comparison of *Flow Improvement* between ISS and other baselines on NY against varying budget (k) and type of synthetic delays.

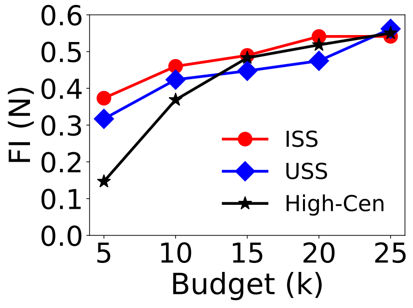
	Gaussian		**Uniform**	
k	**High-Cen**	**USS**	**High-Cen**	**USS**
5	∞	2.2	∞	2.2
10	∞	2.7	∞	2.6
15	∞	2.0	∞	2.0
20	∞	2.3	∞	2.3
25	∞	2.4	∞	2.3
30	∞	2.5	∞	2.3

Table 2.6: (Synthetic Flows/Importance:) Comparison of *Flow Improvement* between ISS and other baselines on NY against varying budget (k) and type of synthetic flows/importance.

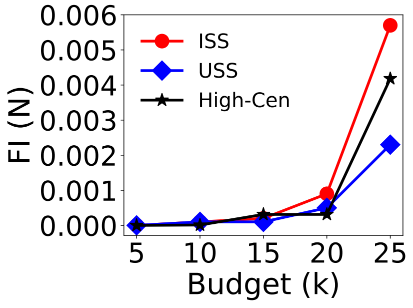
similarly to RPOP. We empirically evaluate this likelihood in real large datasets of BJ and NY. Fig. 2.10b shows that our assumption is indeed true and the likelihood is less than 0.1 (i.e., 1%) in BJ and 0.08 in NY. k is set to 30 in this experiment.

2.5.8 Experiments on Synthetic data

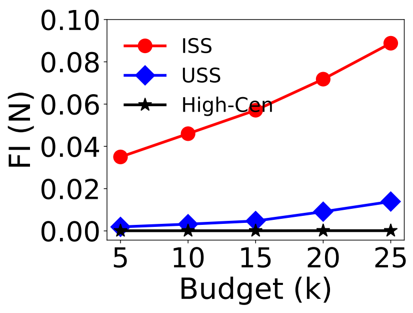
The three data properties that have a profound impact on the quality of the approximation algorithms are: 1) the delay distribution of nodes, 2) the flow/importance distribution of the node pairs, and 3) the graph structure itself. We systematically pick each of these properties and study its impact on the performance. Towards that end,



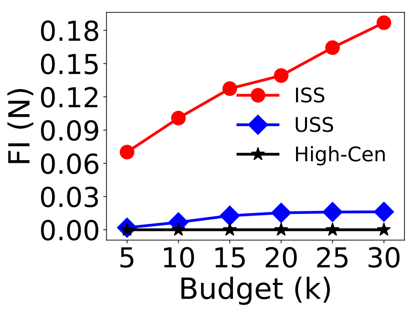
(a) BA



(b) WS



(c) ER



(d) Edge-NY

Figure 2.11: Comparison of *Flow Improvement* FI(N) between ISS and other baselines varying budget (k) (a-c) Synthetic Network Structure: on (a) BA, (b) WS, and (c) ER graph generation models; (d) when the delays are on the edges.

let X be the property under study. To isolate the impact of property X, we pick a real dataset and synthetically alter its property X while keeping the other two properties intact.

1) Synthetic Delay: As shown in Fig. 2.4b, the delay distributions in real transportation networks are highly skewed. Specifically, only a small minority of nodes face high delays. In this experiment, we benchmark the performance of ISS on uniform and standard normal distributions. Towards that end, we pick the NY dataset, and assign node delays synthetically from $\mathcal{U}(0,1)$ and $\mathcal{N}(0,1)$, while retaining the original node pair flows and network structure. In the case of $\mathcal{N}(0,1)$, it may generate negative numbers. Since negative delay is not feasible, we add the value of the minimum delay to all the delays to make them non-negative. Table 2.5 presents the Performance Improvement ratio (Eq. 2.27). Clearly, regardless of the distribution, ISS is significantly better than the competing baselines. This superior performance of ISS is a direct consequence of the other baselines being ignorant of node pair flows.

2) Synthetic Node Pair Flows: In this experiment, we alter the NY dataset by assigning the node pair flows synthetically. First, we assign flow values to node pairs from uniform distribution $\mathcal{U}(0,1)$ and standard normal distribution $\mathcal{N}(0,1)$. As in the case of synthetic node delays, we shift the normal distribution to exclusively non-negative values by adding the minimum of all flow values. Finally, for both Uniform and Normal, the flow values are normalized by dividing them with the sum of all node pair flows. Note that the NY dataset contains more than 5 billion node pairs and assigning flow values to all of them is not computationally feasible. In addition, to normalize the flow values, we also need to store them, and this memory requirement is prohibitively large. To mitigate this issue, we randomly pick 100 million unique node pairs and assign flow values to only these. The rest of the node pairs are assumed to have 0 importance. As a comparison, the real NY dataset has ≈ 11 million node pairs with positive flow.

Table 2.6 shows the performance improvement ratios (Eq. 2.27). and consistent with previous trends, ISS outperforms both baselines. However, the gap between ISS and USS is much smaller compared to their performance on the real unaltered datasets. This result is expected since in the real dataset the node-pair flows follow a power-law distribution (Fig. 2.4a). In a power law distribution, only a small set of node pairs contribute significantly to the flow improvement. These highly important pairs are also likely to be present in the importance sample set. Consequently, the estimate of a node upgrade computed from this sample set is accurate. In uniform or normal distributions, since a large number of node pairs contribute to the flow improvement, a small sample set is unable to capture the entire picture. As a result, the gap between ISS and USS reduces. We also observe that the performance of High-Cen is worse than both ISS and

USS. Since the flows of node pairs are synthetic, the likelihood that a central node will lie on the shortest path between an important pair of nodes is low. The effect is visible in Table 2.6.

3) Synthetic Network Structure: Finally, we investigate the impact of the network structure. We generate synthetic network structures from three well-studied models: (a) Barabasi-Albert (BA), (b) Watts-Strogatz (WS) and (c) Erdos-Renyi (ER).

As in the previous two experiments, we alter the NY dataset by synthetically constructing the network structure, while retaining the original node-pair and delay distributions. Specifically, we first construct a network through one of the graph generation models where the size of the network in terms of number of nodes is the same as the original NY network. Since the number of nodes is the same, we create a mapping from each node in the original structure to the synthetically generated one. Based on this mapping, we assign delays and node-pair importances in the synthetic dataset. Note that although the number of nodes is the same, the number of edges would be different.

Figs. 2.11a-2.11c present the results. Several key insights emerge from this experiment. First, ISS is the best performing technique across all models of graph structure. Second, all techniques achieve high flow improvement in BA (Fig. 2.11a). Since BA has scale-free property, there are few nodes of extreme high degree. These nodes typically also have high centrality and thus, reducing the delays in these central nodes bring a reduction in the shortest path delays among a large number of node pairs. Due to this same reason, all three techniques provide similar flow improvements in the BA model, particularly at a high value of k. In contrast, the flow improvement problem is most difficult in WS. WS generates small-world networks where the average shortest path delays are small (in terms of number of hops and not delays). The flow improvement problem is easier when many shortest paths go through a central node, since in such a case improving the central node improves many node pairs. When shortest paths are small, the likelihood of two shortest paths going through a common node is also small. Consequently, achieving a high flow improvement through a small number of node upgrades is difficult.

The gap between ISS and the baselines is most pronounced in ER. The structure of ER is random by construction. In this scenario, the flows of the node pairs and the delays of the nodes play a crucial role. Unlike USS and High-Cen, ISS takes into account these aspects and provides up to 5 times more flow improvement.

2.5.9 Edge Delays

Recall from Sec. 2.3, that the proposed algorithm can incorporate delays in either nodes, edges or a mixture of both. To showcase this ability, we next measure the performance of ISS with delays on edges. The delay on an edge is computed as discussed in Sec. 2.5.1. Fig. 2.11d presents the results. Consistent with previous results, ISS outperforms USS and High-Cen significantly. ISS is up to 36 times better than the next best baseline USS. High-Cen does not provide any substantial improvement as the notion of *edge centrality*, i.e., the number of node pairs with at least one shortest path going through the edge, is an even weaker indicator of a good upgrade than node centrality. Consequently, the performance suffers.

2.6 Conclusion

In this chapter, we studied and proposed solutions for a novel network design problem of delay minimization. Different from existing techniques, our formulation incorporated the practical considerations that the impact of delay minimization should be noticeable and favor important paths in a network. The proposed problem has diverse applications in a variety of domains including road, airline, power and communication networks. We showed that the problem is NP-hard as well as APX-hard. To overcome the exponential cost of the optimal solution, we proposed an importance sampling based algorithm with provable quality guarantees. Through extensive evaluation on multiple real-word traffic networks, we established that importance sampling is accurate and up to three orders of magnitude faster than the greedy approach. In addition, importance sampling produces flow improvement that is up to 70 times better than the state-of-the-art technique. Finally, our experiments on synthetic datasets established that ISS is robust to variation in network structure, delay distributions and node-pair importances.

Chapter 3

Centrality Maximization

Network centrality plays an important role in many applications. Central nodes in social networks can be influential, driving opinions and spreading news or rumors. In hyperlinked environments, such as the Web, where users navigate via clicks, central content receives high traffic, becoming target for advertising campaigns. While there is an extensive amount of work on centrality measures and their efficient computation, controlling nodes' centrality via network updates is a more recent and challenging task. Performing minimal modifications to a network to achieve a desired property falls under the umbrella of network design problems. This chapter is focused on improving group (coverage and betweenness) centrality [13], which is a function of the shortest paths passing through a set of nodes, by adding edges to the network. Several variations of the problem, which are NP-hard as well as APX-hard, are introduced. We present a greedy algorithm, and even faster sampling algorithms, for group centrality maximization with theoretical quality guarantees under realistic constraints. The experimental results show that our sampling algorithms outperform the best baseline solution in terms of centrality by up to 5 times while being 2-3 orders of magnitude faster than our greedy approach.

3.1 Introduction

Network design is a recent area of study focused on modifying or redesigning a network in order to achieve a desired property [2]. As networks become a popular framework for modeling complex systems (e.g. VLSI, transportation, communication, society), network design provides key controlling capabilities over these systems, especially when resources are constrained. Existing work has investigated the optimization of global network properties, such as minimum spanning tree [57], shortest-path distances [3, 7, 10], diameter [19], and information diffusion-related metrics [4, 9] via a few local (e.g. vertex, edge-level) upgrades. Due to the large scale of real networks, computing a global network property becomes time-intensive. For instance, computing all-pairs shortest paths in large networks is prohibitive. As a consequence, design problems are inherently challenging. Moreover, because of the combinatorial nature of these local modifications, network design problems are often NP-hard, and thus, require the development of efficient approximation algorithms.

We focus on a novel network design problem, that improves the *group centrality.* Given a node v, its coverage centrality is the number of distinct node pairs for which a shortest path passes through v, whereas its betweenness centrality is the sum of the fraction of shortest paths between all distinct pair of nodes passing through v. The centrality of a group X is a function of the shortest paths that go through members of X [24]. *Our goal is to maximize group centrality, for a target group of nodes, via a small number of edge additions.*

There are several applications for group centrality optimization. Broadly speaking, whenever computing the centrality of a single node, or a group of nodes, is a problem of interest, one might as well pose the question of how to improve the centrality of one or more nodes. For instance, in online advertising, links can be added to boost the traffic towards a target set of Web pages.

In a professional network, such as *LinkedIn*, the centrality of some users (e.g. employees of a given company) might be increased via connection recommendations/advertising. In military settings, where networks might include adversarial elements, inducing the flow of information towards key agents can enhance communication and decision making [20].

From a theoretical standpoint, for any objective function of interest, we can define a *search* and a corresponding *design* problem. In this chapter, we show that, different from its search version [24], group centrality maximization cannot be approximated by a simple greedy algorithm. Furthermore, we study several variations of the problem and show that, under two realistic constraints, the problem has a constant factor approximation algorithm. In fact, we are able to prove that our approximation for the constrained problem is *optimal*, in the sense that the best algorithm cannot achieve a better approximation than ours. In order to scale our greedy solution to large datasets, we also propose efficient sampling schemes, with approximation guarantees, for group centrality maximization.

The main contributions of this chapter are summarized as follows:

- We study a novel general network design problem, the group centrality optimization, and prove that it is NP-hard as well as APX-hard.
- We propose a greedy algorithm and faster sampling algorithms for group centrality maximization.
- We show the effectiveness of our algorithms on several datasets and also prove their theoretical guarantees for a constrained version of the problem.

3.2 Related Work

General network design problems: A set of design problems were introduced by Paik et al. [16]. They focused on vertex upgrades to improve the delays on adjacent edges. Krumke et al. [57] generalized this model and proposed minimizing the cost of the minimum spanning tree with varying upgrade costs for vertices/edges. Lin et al. [3] also proposed a shortest path optimization problem via improving edge weights under a budget constraint. Dilkina et al. [7] and Medya et al. [11] studied the same under node improvement.

Design problems via edge addition: Meyerson et al. [10] proposed approximation algorithms for single-source and all-pairs shortest paths minimization. Faster algorithms for the same were designed by Parotisidis et al. [18]. Demaine et al. [19] minimized the diameter of a network and the node eccentricity by adding shortcut edges with a constant factor approximation algorithm. Past research had also considered eccentricity minimization in a composite network [20]. However, all aforementioned problems are based on improving distances and hence are complementary to our objective.

Centrality computation and related optimization problems: The first efficient algorithm for betweenness centrality computation was proposed by Brandes [22]. Recently, [23] introduced an approach for computing the top-k nodes in terms of betweenness centrality via VC-dimension theory. Yoshida [24] studied similar problems —for both betweenness and coverage centrality— in the adaptive setting, where shortest paths already covered by selected nodes are not taken into account. Yoshida's algorithm was later improved using a different sampling scheme [25]. Here, we focus on the design version of the problem, where the goal is to optimize the coverage centrality of a target set of nodes by adding edges. Previous work has studied a constrained version of our problem where the target set size is one [26, 27, 28]. Note that, as the target set X can be chosen arbitrarily in our problem, our solutions and theoretical analysis differ significantly from theirs. In [21], the authors also assume a single target node while maximizing the expected decrease in shortest path distances to the remaining nodes via edge addition. Our work is the first to address the more general and challenging problem of maximizing the centrality of a group of nodes via budgeted edge additions.

3.3 Problem Definition

We assume $G(V, E)$ to be an undirected[1] graph with sets of vertices V and edges E. A shortest path between vertices s and t is a path with minimum distance (in hops) among all paths between s and t, with length $d(s,t)$. By convention, $d(s,s) = 0$, for all $s \in V$. Let P_{st} denote the set of vertices in the shortest paths (multiple ones might exist) between s and t where $s, t \notin P_{st}$. We want to maximize the centrality of the group of nodes X. We define Z as the set of candidate pairs of vertices, $Z \subseteq V \setminus X \times V \setminus X$, which we want to cover. The *coverage centrality* of a vertex is defined as:

$$C(v) = |\{(s,t) \in Z | v \in P_{st}, s \neq v, t \neq v\}| \tag{3.1}$$

$C(v)$ is the number of pairs of vertices with at least one shortest path going through (i.e. covered by) v. The *coverage centrality* of a set $X \subseteq V$ is defined as:

$$C(X) = |\{(s,t) \in Z | v \in P_{st}, v \in X \wedge s, t \notin X\}| \tag{3.2}$$

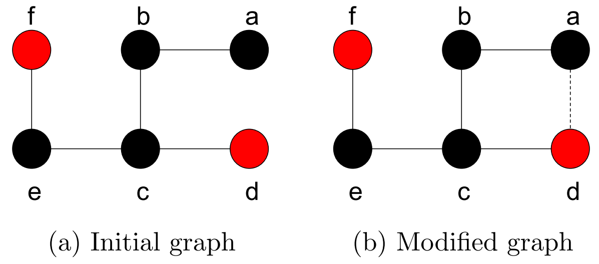
A set X covers a pair (s,t) iff $X \cap P_{st} \neq \varnothing$, i.e., at least one vertex in X is part of a shortest path from s to t. Our goal is to maximize the coverage centrality of X over a set of vertex pairs Z by adding edges from a set of candidate edges Γ to G. Let G_m

[1]We discuss how our methods can be generalized to directed networks in [68].

Symbols	Definitions and Descriptions
$d(s,t)$	Shortest path (s.p.) distance between s and t
n	Number of nodes in the graph
m	Number of edges in the graph
$G(V,E)$	Given graph (vertex set V and edge set E)
X	Target set of nodes
$C(v), C(X)$	Coverage centrality of node v, node set X
Γ	Candidate set of edges
k	budget
P_{st}	The set of nodes on the s.p.s between s and t
G_m, C_m	Modified graph and modified centrality
Z	Pairs of vertices to be covered
m_u	Number of uncovered pairs, $\|Z\|$

Table 3.1: Frequently used symbols

denote the modified graph after adding edges $E_s \subseteq \Gamma$, $G_m = (V, E \cup E_s)$. We define the coverage centrality of X (over pairs in Z) in the modified graph G_m as $C_m(X)$.



(a) Initial graph (b) Modified graph

Figure 3.1: Example of Coverage Centrality Optimization problem. We want to optimize the centrality of $\{d, f\}$ with a budget of one edge from the candidates $\{(d, a), (d, b), (f, b)\}$. The coverage centrality of $\{d, f\}$ is 0 in the initial graph (a) and 3 in the modified graph (b). Node d belongs to the shortest paths between (a, e), (a, c) and (a, f) in (b).

Problem 2 ***Coverage Centrality Optimization (CCO):*** *Given a network $G = (V, E)$, a set of vertices $X \subset V$, a candidate set of edges Γ, a set of vertex pairs Z and a budget k, find a set of edges $E_s \subseteq \Gamma$, such that $|E_s| = k$ and $C_m(X)$ is maximized.*

For simplicity, in the rest of the chapter, we assume $Z = V \setminus X \times V \setminus X$ unless stated otherwise. Thus,

$$C(X) = |\{(s,t) \in V \setminus X \times V \setminus X | v \in P_{st}, v \in X, s < t\}| \tag{3.3}$$

where $s < t$ implies ordered pairs of vertices. Fig. 3.1 shows a solution for the CCO problem with budget $k = 1$ for an example network where the target set $X = \{d, f\}$ and the candidate set $\Gamma = \{(d,a),(d,b),(f,b)\}$.

Similarly, we can also formulate the group betweenness centrality optimization problem. Given a vertex set $X \subseteq V$, its *group betweenness centrality* is defined as:

$$B(X) = \sum_{s,t \in V \setminus X} \frac{\sigma_{s,t}(X)}{\sigma_{s,t}} \tag{3.4}$$

where $\sigma_{s,t}$ is the number of shortest paths between s and t, $\sigma_{s,t}(X)$ is the number of shortest paths between s and t passing through X. We define the group betweenness centrality of X in the modified graph G_m as $B_m(X)$.

Problem 3 ***Betweenness Centrality Optimization (BCO):*** *Given a network $G = (V, E)$, a node set $X \subset V$, a candidate edge set Γ, a set of node pairs Z and a budget k, find a set of edges $E_s \subseteq \Gamma$, such that $|E_s| \leq k$ and $B_m(X)$ is maximized.*

While we focus on the CCO problem, the results described here can be easily mapped to BCO.

3.4 Hardness and Inapproximability

This section provides complexity analysis of the CCO problem. We show that CCO is NP-hard as well as APX-hard. More specifically, CCO cannot be approximated within a factor grater than $(1 - \frac{1}{e})$.

Theorem 9 *The CCO problem is NP-hard.*

The proof is in [68]. While computing an optimal solution for CCO is infeasible in practice, a natural question is whether it has a polynomial-time approximation. The next theorem shows that CCO is also NP-hard to approximate within a factor greater than $(1 - \frac{1}{e})$. Interestingly, different from its search counterpart [24], CCO is not submodular (see [68]). These two results provide strong evidence that, for group centrality, network design is strictly harder than search.

Theorem 10 *CCO cannot be approximated within a factor greater than* $(1 - \frac{1}{e})$.

Proof: We give an L-reduction [59] from the maximum coverage (MSC) problem with parameters x and y. Given a collection of subsets $S_1, S_2, ..., S_m$ for a universal set of items $U = \{u_1, u_2, ..., u_n\}$, the MSC problem is to choose at most k sets to cover as many elements as possible. Our reduction is such that following two equations are satisfied:

$$OPT(I_{CCO}) \leq xOPT(I_{MSC})$$

$$OPT(I_{MSC}) - s(T^M) \leq y(OPT(I_{CCO}) - s(T^C))$$

where I_{MSC} and I_{CCO} are problem instances, and $OPT(Y)$ is the optimal value for instance Y. $s(T^M)$ and $s(T^C)$ denote any solution of the MSC and CCO instances, respectively. If the conditions hold and CCO has an α approximation, then MSC has an $(1 - xy(1 - \alpha))$ approximation. However, MSC is NP-hard to approximate within a factor greater than $(1 - \frac{1}{e})$. It follows that $(1 - xy(1 - \alpha)) < (1 - \frac{1}{e})$, or, $\alpha < (1 - \frac{1}{xye})$. So, if the conditions are satisfied, CCO is NP-hard to approximate within a factor greater than $(1 - \frac{1}{xye})$.

We use the same construction as in Theorem 9. For CCO, the set Z contains pairs in the form (b, u), $u \in U$. Let the solution of I_{CCO} be $s(T^C)$. The centrality of node

Algorithm 3 Greedy Edge Set (GES)

```
Require: Network $G = (V, E)$, target node set $X$, Candidate set of edges $\Gamma$, Budget $k$
Ensure: A subset $E_s$ from $\Gamma$ of $k$ edges
 1: $E_s \leftarrow \emptyset$
 2: Compute all-pairs shortest path distances
 3: while $|E_s| \leq k$ do
 4:     for $e \in \Gamma \setminus E_s$ do
 5:         $Count(e) \leftarrow$# new covered pairs after adding $e$
 6:     end for
 7:     $e^* \leftarrow \arg\max_{e \in \Gamma \setminus E_s} \{Count(e)\}$
 8:     $E_s \leftarrow E_s \cup e^*$ and $E \leftarrow E \cup e^*$
 9:     Update the shortest path distances
10: end while
11: return $E_s$
```

a will increase by $s(T^C)$ to cover the pairs in Z. Note that $s(T^C) = 2s(T^M)$ from the construction (as the graph is undirected, the covered pair is unordered). It follows that both the conditions are satisfied when $x = 2$ and $y = \frac{1}{2}$. So, CCO is NP-hard to approximate within a factor grater than $(1 - \frac{1}{e})$. $\square$ ■

Theorem 10 shows that there is no polynomial-time approximation better than $(1-\frac{1}{e})$ for CCO. Given such an inapproximability result, we propose an efficient greedy heuristic for our problem in the next section.

3.5 Algorithms

3.5.1 Greedy Algorithm

Algorithm 3 (GES) selects the best edge to be added in each of k iterations, where k is the budget. Its most important steps are 2 and 7. In step 2, it computes all-pairs shortest paths in time $O(n(m + n))$. Next, it chooses, among the candidate edges Γ, the one that maximizes the marginal coverage centrality gain of X (step 7), which takes $O(|\Gamma|n^2)$ time. After adding the best edge, shortest path distances are updated. Then, the algorithm checks the pairwise distances in $O(n^2)$ time (step 9). The total running time of GES is $O(n(m+n) + k|\Gamma|n^2)$.

We illustrate the execution of GES on the graph from Figure 3.1a for a budget $k = 2$, a candidate set of edges $\Gamma = \{(d,a), (d,b), (f,b)\}$, and a target set $X = \{d, f\}$. Initially, adding $(d,a), (d,b)$ and (f,b) increases the centrality of X by 3, 0, and 2, respectively, and thus (d,a) is chosen. In the second iteration, (d,b) and (f,b) increase the centrality of X by 0 and 1, respectively, and (f,b) is chosen.

Algorithm 4 Best Edge via Uniform Sampling (BUS)

```
Require: Network $G = (V, E)$, target node set $X$, Candidate set of edges $\Gamma$, Budget $k$
Ensure: A subset $\gamma$ from $\Gamma$ of $k$ edges
 1: $Q$ is a set of $q$ pairs of vertices chosen randomly, with replacement, from $M_u$
 2: $E_s \leftarrow \emptyset$
 3: while $|E_s| \leq k$ do
 4:     for $(s,t) \in Q$ do
 5:         Compute and store shortest path distances $d(s,v)$ and $d(t,v)$ for all $v \in V$
 6:     end for
 7:     for $e \in \Gamma \setminus E_s$ do
 8:         $Count(e) \leftarrow$ # new covered pairs in $Q$ after adding $e$
 9:     end for
10:     $e^* \leftarrow \arg\max_{e \in \Gamma \setminus E_s} \{Count(e)\}$
11:     $E_s \leftarrow E_s \cup e^*$ and $E \leftarrow E \cup e^*$
12: end while
13: Return $E_s$
```

3.5.2 Sampling Algorithm

The execution time of GES increases with $|\Gamma|$ and m. In particular, if $m = O(n^2)$ and $|\Gamma| = O(n)$, the complexity reaches $O(n^3)$, which is prohibitive for large graphs. To address this challenge, we propose a sampling algorithm that is nearly optimal, regarding each greedy edge choice, with probabilistic guarantees (see Section 3.6.3). Instead of selecting edges based on all the uncovered pairs of vertices, our scheme does it based on a small number of sampled uncovered pairs. This strategy allows the selection of edges with probabilistic guarantees using a small number of samples, thus ensuring scalability to large graphs. We show that the error in estimating the improvement in coverage based on the samples is small.

Algorithm 4 (Best Edge via Uniform Sampling, or BUS) is a sampling scheme to select the best edge to be added in each of the k iterations based on sampled uncovered node pairs (q and M_u are the number of samples and the set of uncovered pairs respectively). For each pair of samples, we compute the distances from each node in the pair to all others. These distances are used to estimate the true number of covered pairs after an edge addition. In Sec. 3.6.3, we provide a theoretical analysis of the approximation achieved by BUS.

In terms of time complexity, steps 4-6, where BUS performs shortest-path computations, take $O(q(n+m))$ time. Next, the algorithm estimates the additional number of shortest pairs covered by X after adding each of the edges based on the samples (steps 7-9) in $O(|\Gamma|q^2)$ time. Given such an estimate, the algorithm chooses the best edge to be added (step 10). The total running time of BUS is $O(kq(m+n)+k|\Gamma|q^2)$.

3.6 Analysis

In the previous section, we described a greedy heuristic and an efficient sampling algorithm to approximate the greedy approach. Next, we show that, under some realistic assumptions, the described greedy algorithm provides a constant-factor approximation for a modified version of CCO. More specifically, our approximation guarantees are based on the addition of two extra constraints to the general CCO described in Section 3.3.

3.6.1 Constrained Problem

The extra constraints, S^1 and S^2, considered are the following: (1) S^1: We assume that edges are added from the target set X to the remaining nodes, i.e. edges in a given candidate set Γ have the form (a, b) where $a \in X$ and $b \in V \setminus X$ [26, 28]; and (2) S^2: Each pair (s, t) can be covered by at most one single newly added edge [6, 10].

S^1 is a reasonable assumption in many applications. For instance, in online advertising, adding links to a third-party page gives away control over the navigation, which is undesirable. S^2 is motivated by the fact that, in real-life graphs, centrality follows a skewed distribution (e.g. power-law), and thus most of the new pairs will have shortest paths through a single edge in Γ. Generalizing our methods to the case where shortest paths are covered by any fixed number of edges in Γ is straightforward. In our experiments (see Section 3.7.1), we show that solutions for the constrained and general problem are often close. Moreover, both constraints have been considered in previous work [6, 10]. Next, we show that COO under S^1 and S^2, or RCCO (Restricted CCO), for short, is still NP-hard.

Corollary 11 *RCCO is NP-hard.*

Proof: Follows directly from Theorem 9, as our construction respects both the constraints. □ ■

3.6.2 Analysis of Greedy Algorithm

The next theorem shows that RCCO's optimization function is monotone and submodular. As a consequence, the greedy algorithm described in Section 3.5.1 leads to a well-known constant factor approximation of $(1 - 1/e)$ [60].

Theorem 12 *The objective function $f(E_s) = C_m(X)$ in RCCO is monotone and submodular.*

Proof: Monotonicity: Follows from the definition of a shortest path. Adding an edge $(u, v) \in E_s$ cannot increase $d(s, t)$ for any (s, t) already covered by X. Since $u \in X$ for any $(u, v) \in E_s$, the coverage $C_m(X)$ is also non-decreasing.

Submodularity: We consider addition of two sets of edges, E_a and E_b where $E_a \subset E_b$, and show that $f(E_a \cup \{e\}) - f(E_a) \geq f(E_b \cup \{e\}) - f(E_b)$ for any edge $e \in \Gamma$ such that $e \notin E_a$ and $e \notin E_b$. Let $F(A)$ be the set of node pairs (s, t) which are covered by an edge $e \in A$ ($|F(E_s)| = C_m(X)$). Then $f(.)$ is submodular if $F(E_b \cup \{e\}) \setminus F(E_b) \subseteq F(E_a \cup \{e\}) \setminus F(E_a)$. To prove this claim, we make use of S^B. Therefore, each pair $(s, t) \in F(E_b)$ is covered by only one edge in E_b. As $E_a \subset E_b$, adding e to E_a will cover some of the pairs which are already covered by $E_b \setminus E_a$. Then, for any newly covered pair $(s, t) \in F(E_b \cup \{e\}) \setminus F(E_b)$, it must hold that $(s, t) \in F(E_a \cup \{e\}) \setminus F(E_a)$. $\square$ ■

Based on Theorem 12, if OPT is the optimal solution for an instance of the RCCO problem, GES will return a set of edges E_s such that $f(E_s) \geq (1 - 1/e)OPT$. The existence of such an approximation algorithm shows that the constraints S^1 and S^2 make the CCO problem easier, compared to its general version. On the other hand, whether GES is a good algorithm for the modified CCO (RCCO) remains an open question. In order to show that our algorithm is optimal, in the sense that the best algorithm for this problem cannot achieve a better approximation from those of GES, we also prove an inapproximability result for the constrained problem.

Corollary 13 *RCCO cannot be approximated within a factor greater than* $(1 - \frac{1}{e})$.

Proof: Follows directly from Thm. 10, as the construction applied in the proof respects both the constraints. ■

Corollary 13 certifies that GES achieves the best approximation possible for constrained CCO (RCCO).

3.6.3 Analysis of Sampling Algorithm

In Section 3.5.2, we presented BUS, a fast sampling algorithm for the general CCO problem. Here, we study the quality of the approximation provided by BUS as a function of the number of sampled node pairs. The analysis will assume the constrained version of CCO (RCCO), but the general case will also be discussed.

Let us assume that X covers a set M_c of pairs of nodes. The set of remaining pairs is $M_u = \{(s,t)|s \in V, t \in V, s \neq t, X \cap P_{st} = \emptyset\}$ and $m_u = |M_u| = n(n-1)/2 - |M_c|$. We sample, uniformly with replacement, a set of ordered pairs Q ($|Q| = q$) from M_u. Let $g^q(.)$ denote the number of *new* pairs covered by the candidate edges based on the samples Q. For an edge set $\gamma \subset \Gamma$, X_i is a random variable that denotes whether the ith sampled pair is covered by any edge in γ. In other words, $X_i = 1$ if the pair is covered and 0, otherwise. Each pair is chosen with probability $\frac{1}{m_u}$. Also, let us define $f^q = \frac{m_u}{q} g^q$ as the estimated coverage.

Lemma 2 *Given q sampled node pairs from M_u:*

$$E(g^q(\gamma)) = \frac{q}{m_u} f(\gamma)$$

Proof: From the samples, we get $g^q(\gamma) = \Sigma_{i=1}^{q} X_i$. By the linearity and additive rule, $E(g^q(\gamma)) = \Sigma_{i=1}^{q} E(X_i) = q.E(X_i)$. As the probability $P(X_i) = \frac{f(\gamma)}{m_u}$ and X_is are i.i.d., $E(g^q(\gamma)) = \frac{q}{m_u} f(\gamma)$. $\square$ ■

Lemma 3 *Given ε ($0 < \varepsilon < 1$), a positive integer l, a budget k, and a sample of* *independent uncovered node pairs* $Q, |Q| = q$*, where* $q(\varepsilon) \geq \frac{3m_u(l+k)log(|\Gamma|)}{\varepsilon^2 \cdot OPT}$*; then:*

$$Pr(|f^q(\gamma) - f(\gamma)| < \varepsilon \cdot OPT) \geq 1 - 2|\Gamma|^{-l}$$

For all $\gamma \subset \Gamma$*,* $|\gamma| \leq k$*, where* OPT *denotes the optimal coverage* ($OPT = Max\{f(\gamma)|\gamma \subset \Gamma, |\gamma| \leq k\}$)*.*

Proof: Using Lemma 2:

$$Pr(|f^q(\gamma) - f(\gamma)| \geq \delta \cdot f(\gamma))$$
$$Pr\Big(|\frac{q}{m_u}f^q(\gamma) - \frac{q}{m_u}f(\gamma)| \geq \frac{q}{m_u} \cdot \delta \cdot f(\gamma)\Big)$$
$$Pr\Big(|g^q(\gamma) - \frac{q}{m_u}f(\gamma)| \geq \frac{q}{m_u} \cdot \delta f(\gamma)\Big)$$
$$Pr(|g^q(\gamma) - E(g^q(\gamma))| \geq \delta E(g^q(\gamma)))$$

As samples are independent, the Chernoff bound gives:

$$Pr\Big(|g^q(\gamma) - \frac{q}{m_u}f(\gamma)| \geq \frac{q}{m_u}\delta f(\gamma)\Big) \leq 2\exp\Big(-\frac{\delta^2}{3}\frac{q}{m_u}f(\gamma)\Big)$$

Substituting $\delta = \frac{\varepsilon OPT}{f(\gamma)}$ and q:

$$Pr(|f^q(\gamma) - f(\gamma)| \geq \varepsilon \cdot OPT) \leq 2\exp\Big(-\frac{OPT}{f(\gamma)}(l+k)log(\Gamma)\Big)$$

Using the fact that $OPT \geq f(\gamma)$:

$$Pr(|f^q(\gamma) - f(\gamma)| \geq \varepsilon \cdot OPT) \leq 2|\Gamma|^{-(l+k)}$$

Applying the union bound over all possible size-k subsets of $\gamma \subset \Gamma$ (there are $|\Gamma|^k$)

we conclude that:

$$Pr(|f^q(\gamma) - f(\gamma)| \geq \varepsilon \cdot OPT) < 2|\Gamma|^{-l}, \forall \gamma \subset \Gamma$$

$$Pr(|f^q(\gamma) - f(\gamma)| < \varepsilon \cdot OPT) \geq 1 - 2|\Gamma|^{-l}, \forall \gamma \subset \Gamma\square$$

■

Now, we prove our main theorem which shows an approximation bound of $(1 - \frac{1}{e} - \varepsilon)$ by Algorithm 4 whenever the number of samples is at least $q(\varepsilon/2) = \frac{12m_u(l+k)log(|\Gamma|)}{\varepsilon^2 \cdot OPT}$ (l and ε are as in Lemma 3).

Theorem 14 *Algorithm 4 ensures $f(\gamma) \geq (1 - \frac{1}{e} - \varepsilon)OPT$ with high probability $(1 - \frac{2}{|\Gamma|^l})$ if at least $q(\varepsilon/2)$ samples are considered.*

Proof: $f(.)$ is monotonic and submodular (Thm. 12) and one can prove the same for $f^q(.)$. Given the following:

1. From Lemma 3, the number of samples is at least $q(\varepsilon/2)$. So, with probability $1 - \frac{2}{|\Gamma|^l}$, $f(\gamma) \geq f^q(\gamma) - \frac{\varepsilon}{2}OPT$;

2. $f^q(\gamma) \geq (1 - \frac{1}{e})f^q(\gamma*)$, $\gamma* = \arg\max_{\gamma' \subset \Gamma, |\gamma'| \leq k} f^q(\gamma')$ (submodularity property of $f^q(.)$);

3. $f^q(\gamma*) \geq f^q(\bar{\gamma})$, $\bar{\gamma} = \arg\max_{\gamma' \subset \Gamma, |\gamma'| \leq k} f(\gamma')$ (Note that, $OPT = f(\bar{\gamma})$)

Thm.	**#Samples**	**Approximations**
Thm. 14	$O(\frac{m_u klog(\|\Gamma\|)}{\varepsilon^2.OPT})$	$f(\gamma) > (1-\frac{1}{e}-\varepsilon)OPT$
Cor. 16	$O(\frac{klog(\|\Gamma\|)}{\varepsilon^2})$	$f(\gamma) > (1-\frac{1}{e})OPT - \varepsilon.m_u$

Table 3.2: Summary of the probabilistic approximations.

We can prove with probability $1-\frac{2}{|\Gamma|^l}$ that:

$$\begin{aligned} f(\gamma) &\geq f^q(\gamma) - \frac{\varepsilon}{2}OPT \\ &\geq \Big(1-\frac{1}{e}\Big)f^q(\gamma*) - \frac{\varepsilon}{2}OPT \\ &\geq \Big(1-\frac{1}{e}\Big)f^q(\bar{\gamma}) - \frac{\varepsilon}{2}OPT \\ &\geq \Big(1-\frac{1}{e}\Big)\Big(f(\bar{\gamma}) - \frac{\varepsilon}{2}OPT\Big) - \frac{\varepsilon}{2}OPT \\ &> \Big(1-\frac{1}{e}-\varepsilon\Big)OPT\square \end{aligned}$$

■

Choosing #Samples: While we are able to achieve a good probabilistic approximation with respect to the optimal value OPT, deciding the number of samples is not straightforward. In practice, we do not know the value of OPT beforehand, which affects the number of samples needed. However, notice that OPT is bounded by the number of uncovered pairs m_u. Moreover, the number of samples $q(\varepsilon/2)$ depends on the ratio $\frac{m_u}{OPT}$. Increasing this ratio while keeping the quality constant requires more samples. If OPT (which depends on X) is close to the number of uncovered pairs m_u, we need fewer samples to achieve the bound. In the experiments, we assume this ratio to be constant. Next, we propose another approximation scheme where we can reduce the number of samples by avoiding the term OPT in the sample size while waiving the assumption involving constants.

Let M_u and m_u be the set and number of uncovered pairs by X, respectively, in the initial graph. Moreover, we define $\bar{q}(\varepsilon)$ so that:

$$\bar{q}(\varepsilon) \geq \frac{3(l+k)log(|\Gamma|)}{\varepsilon^2}$$

Corollary 15 *Given ε $(0 < \varepsilon < 1)$, a positive integer l, a budget k, and a sample of independent uncovered node pairs $Q, |Q| = \bar{q}(\varepsilon)$, then:*

$$Pr(|f^q(\gamma) - f(\gamma)| < \varepsilon \cdot m_u) \geq 1 - 2|\Gamma|^{-l}, \forall \gamma \subset \Gamma, |\gamma| \leq k$$

The proof is given in [68]. Next, we provide an approximation bound by our sampling scheme for at least $\bar{q}(\varepsilon/2) = \frac{12(l+k)log(|\Gamma|)}{\varepsilon^2}$ samples.

Corollary 16 *Algorithm 4 ensures $f(\gamma) \geq (1 - \frac{1}{e})OPT - \varepsilon.m_u$ with high probability $(1 - \frac{2}{|\Gamma|^l})$ if at least $\bar{q}(\varepsilon/2)$ samples are used.*

This proof is also in [68]. Table 3.2 summarizes the number of samples and corresponding bounds for Algorithm 4. Theorem 14 ensures higher quality with higher number of samples than Corollary 16. On the other hand, Corollary 16 does not assume anything about the ratio $\frac{m_u}{OPT}$. The results reflect a trade-off between number of samples and accuracy.

Theorem 14 and Corollary 16 assume that a greedy approach achieves a constant-factor approximation of $(1 - 1/e)$, which holds only for the RCCO problem (see Sections 3.6.1 and 3.6.2). As a consequence, in the case of the general problem, the guarantees discussed in this Section apply only for each iteration of our sampling algorithm, but not for the final results. In other words, BUS provides theoretical quality guarantees that each edge selected in an iteration of the algorithm achieves a coverage within bounded distance from the optimal edge. Nonetheless, experimental results show, in practice, BUS is also effective in the general setting.

Dataset Name	$\lvert V\rvert$	$\lvert E\rvert$
Network Science Coauthorship (NS)	0.3k	1k
email-Eu-core (EU)	1k	25k
ca-GrQc (CG)	5K	14K
email-Enron (EE)	36K	183K
loc-Brightkite (LB)	58K	214K
loc-Gowalla (LG)	196K	950K
web-Stanford (WS)	280K	2.3M
DBLP (DB)	1.1M	5M

Table 3.3: Dataset description and statistics.

	Ratio		
Data	$k = 5$	$k = 10$	$k = 15$
NS	1.02	1.14	1.17
EU	1.0	1.1	1.08
Synthetic	1.0	1.0	1.0

Table 3.4: he ratio between the improvement in coverage produced by GES for CCO and RCCO.

3.7 Experimental Results

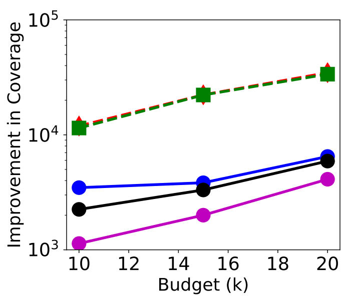
	Coverage of BUS				Time [sec.]			Sample
k	GES	High-ACC	High-Deg	Random	GES	High-ACC	BUS	BUS
10	0.95	2.46	5.41	14.45	$> 7K$	157.1	5.1	2560
15	0.97	2.92	7.29	9.98	$> 7K$	156.9	10.1	3840
20	0.98	2.78	9.96	9.59	$> 7K$	157.2	18.2	5120

Table 3.5: CG data: Comparison of our sampling algorithm (BUS) and the baselines, including our Greedy (GES) approach, using the CG dataset and varying the budget k. We evaluate the coverage of BUS relative to the baselines—i.e. how many times more new pairs are covered by BUS compared to the baseline.

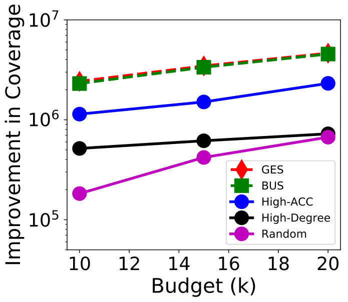
Experimental Setup: We evaluate our algorithms on real-world networks. All experiments were conducted on a 3.30GHz Intel Core i7 machine with 30 GB RAM and Ubuntu. Algorithms were implemented in Java.

Dataset: All datasets applied are available online[2]. Table 3.3 shows dataset statistics.

[2]Datasets are from (1) `https://snap.stanford.edu`, (2) `http://dblp.uni-trier.de`, (3)

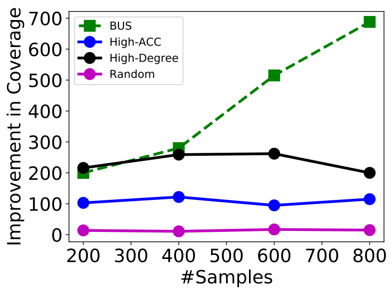


(a) Quality on EU

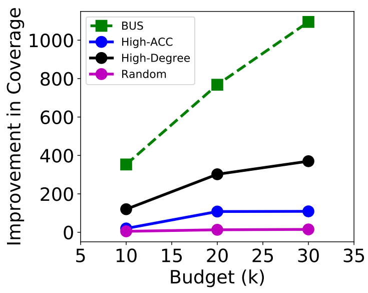


(b) Quality on CG

Figure 3.2: BUS vs. Greedy: Improvement in coverage centrality produced by different algorithms.



(a) Fixed Budget



(b) Fixed #Sample

Figure 3.3: Comparison with baselines on the EE dataset varying (b) the number of samples and (c) the budget.

The graphs are undirected and we consider the largest connected component in our experiments. The datasets are from different categories: EE and EU are constructed from email communication; NS, CG and DB are collaboration networks; LB and LG are OSNs and WS is a webgraph.

Other Settings: We set the candidate edge set Γ as those edges from X to the remaining vertices that are absent in the initial graph (i.e. $\Gamma = \{(u, v)|u \in X \wedge v \in V \setminus X \wedge (u, v) \notin E\}$). The set of target nodes X is randomly selected from the set of all nodes. Results reported are averages over 10 repetitions.

Baselines: We consider three baselines in our experiments: 1) **High-ACC [24, 25]:**

`http://www-personal.umich.edu/~mejn/netdata/` and the code is available at `http://cs.ucsb.edu/~medya/CODE/SDM18/`.

	Coverage of BUS				Time [sec.]			Samples
k	GES	High-ACC	High-Deg	Random	GES	High-ACC	BUS	BUS
10	0.96	3.3	5.1	10.1	271	2.3	1.8	2093
15	0.97	5.8	6.7	11.1	423	2.4	3.4	3139
20	0.97	5.2	5.7	8.2	531	2.5	4.5	4186

Table 3.6: EU data: Comparison of our sampling algorithm (BUS) and the baselines using the EU dataset.

Finds the top k central nodes based on *maximum adaptive centrality coverage* and adds edges between target nodes X and the set of top-k central nodes; 2) **High-Degree:** Selects edges between the target nodes X and the top k high degree nodes; 3) **Random:** Randomly chooses k edges from Γ. We compare our sampling algorithm (BUS) against our Greedy solution (GES) and show that BUS is more efficient while producing similar results in terms of quality.

Performance Metric: The quality of a solution set (a set of edges produced by the algorithm) is the number of newly covered pairs by the target set of nodes after addition of these edges to the intial graph. We call it *improvement in coverage.*

3.7.1 GES: RCCO vs CCO

We compare coverage centrality optimization (CCO) and its restricted version (RCCO) by applying GES to two small real (NS and EU) and one synthetic (Barabasi) network ($|V| = 2k$, $|E| = 10k$). The target set size $|X|$ is set to 5. Table 3.4 shows the ratio between results for CCO and RCCO varying the budget k. The results, close to 1, support the RCCO assumptions discussed in Section 3.6.1.

3.7.2 BUS vs. GES

We use only the smallest dataset (CG) in this experiment, as the GES algorithm is not scalable—it requires the computation of all-pairs shortest paths. For BUS, we set the error $\varepsilon = 0.3$. First, we evaluate the effect of sampling on quality, which we theoretically analyzed in Theorem 14 and Corollary 16.

Fig. 3.2 shows the number of new pairs covered by the algorithms. Table 3.5 and 3.6 show the running times and the quality of BUS relative to the baselines—i.e. how many times more pairs are covered by BUS compared to a given baseline on CG and EU data, respectively. BUS and GES produce results at least 2 times better than the baselines. Moreover, BUS achieves results comparable to GES while being 2-3 orders of magnitude faster.

3.7.3 Results for Large Graphs

We compare our sampling algorithm against the baseline methods on large graphs (EE, LB, LG, WS and DB). Due to the high cost of computing all-pairs shortest paths, we estimate the centrality based on $10K$ randomly selected pairs. For High-ACC, we also use sampling for adaptive coverage centrality computation [24, 25] and the same number of samples is used. The budget and target set sizes are set as 20 and 5, respectively.

Table 3.7 shows the results, where the quality is relative to BUS results. BUS takes a few minutes ($8, 15, 17, 45, 85$ minutes for EE, LB, WS, LG and DB respectively) to run and significantly outperforms the baselines. This happens as the existing approaches do not take into account the dependencies between the edges selected. BUS selects the edges sequentially, considering the effect of edges selected in previous steps.

3.7.4 Parameter Sensitivity

The main parameters of BUS are the budget and the number of samples—both affect the error ε, as discussed in Thm. 14 and Cor. 16. We study the impact of these two parameters on performance. Again, we estimate coverage using $10K$ randomly selected pairs of nodes.

Fig. 3.3a shows the results on EE data for budget 20 and target set size 5. With 600 samples, BUS produces results at least 2 times better than the baselines. Next, we fix the number of samples and vary the budget. Figure 3.3b shows the results with $10K$ samples and 5 target nodes. BUS produces results at least 2.5 times better than the baselines. Moreover, BUS takes only 30 seconds to run with budget of 30 and 1000 samples. We find that the running time grows linearly with the budget for a fixed number of samples. These results validate the running time analysis from Sec. 3.5.2.

	BUS (relative to baselines)			**# Samples**
Data	High-ACC	High-Degree	Random	BUS
EE	4.88	2.74	51	6462
LB	3.3	2.3	33.8	6796
LG	3.3	4.2	62	4255
WS	1.89	1.95	4.8	2000
DB	2.5	1.6	5	875

Table 3.7: Coverage centrality of BUS relative to baselines.

3.7.5 Impact on Other Metrics:

While this chapter is focused on optimizing Coverage Centrality, it is interesting to analyze how our methods affect other relevant metrics. Here, we look at the following ones: 1) influence, 2) average shortest-path distance, and 3) closeness centrality. The idea is to assess how BUS improves the influence of the target nodes, decreases the distances from the target to the remaining nodes, and increases the closeness centrality of these nodes as new edges are added to the graph. For influence analysis, we consider the popular independent cascade model [31] with edge probabilities of 0.1. In all the experiments, we fix the number of sampled pairs at 1000 and choose 10 nodes, uniformly at random, as the target set X. The metrics are computed before and after the addition

	Influence			Distance			Closeness		
k	EE	LB	LG	EE	LB	LG	EE	LB	LG
25	57.7	12.2	10.7	2.7	1.2	2.2	2.0	2.0	1.0
50	96.8	17.5	92.7	3.8	3.5	3.3	4.9	3.9	4.0
75	134.3	29.1	45.9	5.2	2.1	2.3	5.9	2.3	1.9

Table 3.8: Improvement of other metrics after adding the edges found by BUS: the numbers are improvement in percentage with respect to the value for the initial graph.

of edges and presented as the relative improvement (in percentage). Because target nodes are chosen at random, increasing the budget does not necessarily lead to an increase in the metrics considered.

Results are presented in Table 3.8. There is a significant improvement of the three metrics as the budget (k) increases. For influence, the number of seed nodes is small, and thus the relative improvement for increasing k is large. The improvement of the other metrics is also significant. For instance, in EE, the decrease in distance is nearly 5%, which is approximately $72K$, for a budget of 75.

3.8 Conclusions

We studied several variations of a novel network design problem, *group centrality optimization*. This problem has applications in a variety of domains including social, collaboration, and communication networks. From a theoretical perspective, we have shown that these variations of the problem are NP-hard as well as APX-hard. Moreover, we have proposed a greedy algorithm, and even faster sampling algorithms, for group centrality optimization. Our algorithms provide theoretical quality guarantees under realistic assumptions and also outperform the baseline methods by up to 5 times on several datasets. From a broader point of view, we believe that this chapter highlights interesting properties of network design problems compared to their, more well-studied, search counterparts.

Chapter 4

Hiding in Covert Networks

Covert networks are social networks that often consist of harmful users. Social Network Analysis (SNA) has played an important role in reducing criminal activities (e.g., counter terrorism) via detecting the influential users in such networks. There are various popular measures to quantify how influential or central any vertex is in a network. As expected, strategic and influential miscreants in covert networks would try to hide herself and her partners (called *leaders*) from being detected via these measures by introducing new edges. Waniek et al. [58] show that the corresponding computational problem, called Hiding Leader, is NP-complete for the degree and closeness centrality measures. We study the popular core centrality measure and show that the problem [69] is NP-complete even when the core centrality of every leader is only 3. On the contrary, we prove that the problem becomes polynomial time solvable for the degree centrality measure if the degree of every leader is bounded above by any constant. We then focus on the optimization version of the problem and show that the Hiding Leader problem admits a 2 factor approximation algorithm for the degree centrality measure. We complement it by proving that one cannot hope to have any $(2-\varepsilon)$ factor approximation algorithm for any constant $\varepsilon > 0$ unless there is a $\varepsilon/2$ factor polynomial time algorithm for the

Densest k-Subgraph problem which would be considered a significant breakthrough. We empirically establish that our 2 factor approximation algorithm frequently finds out a near optimal solution. On the contrary, for the core centrality measure, we show that the Hiding Leader problem does not admit any $(1-\alpha)\ln n$ factor approximation algorithm for any constant $\alpha \in (0,1)$ unless $\mathsf{P} = \mathsf{NP}$ even when the core centrality of every leader is only 3. Hence, our work shows that, although classical complexity theoretic framework fails to shed any light on relative difficulty of Hiding Leader for different centrality measures, the problem is significantly "harder" for the core centrality measure than the degree centrality one.

4.1 Introduction

Social network analysis (SNA) has played a pivotal role in many applications in multi-agent systems and artificial intelligence [70, 71, 72, 73]. One of the most successful applications of SNA is in counter-terrorism via analyzing *covert networks* [74, 75, 76, 77]. Covert network loosely refers to network of criminals, terrorists, illegal activities, etc. Security personnel regularly use various SNA tools to understand criminal behavior, catch their leaders, and effectively dismantle such networks [78, 79, 80].

Centrality measure is one of the most useful tools that SNA provides to analyze covert networks. It assigns scores to the vertices based on their relative *influence or importance* in the network [81]; depending on the centrality measure, higher scores may correspond to important vertices and important vertices are expected to be more central. One of the simplest and oldest such centrality measures is the *degree centrality* which ranks vertices according to their degree [82]. Other important examples include *closeness centrality* and *betweenness centrality* that are measures based on shortest paths [83]. Another centrality measure is the *core centrality* [84] which ranks the vertices based on their *core number*.

Intuitively speaking, if a vertex has a high core number, then it is part of some dense cohesive community within the network. Formally, a *k-core* is an induced subgraph of the network where the minimum degree of the vertices is at least k. The core number of a vertex is the highest integer k such that the vertex is part of some *k-core*. Therefore, the core centrality can be more revealing about the position of a node than its degree centrality—while degree centrality only concerns about the degree of a vertex, the core centrality elegantly takes into consideration the degrees of the neighbors as well as the vertex. These two measures are also related in the sense that the core centrality of any vertex is at most its degree centrality. Due to its sophisticated nature, the core centrality has been extensively used in the study of covert networks [85, 86, 87] as well as in other important tasks such as viral marketing and social engagement [88, 89] in social networks.

In this chapter, our goal is to study the centrality measure based secrecy in covert networks. Indeed, understanding covert networks remains a challenging task mainly due to incompleteness and dynamic evolution of the data as well as the strategic nature of the users [90, 91, 92, 93, 94, 95]. Since the criminals often possess technical expertise [96, 97, 98, 99, 100], we are interested in the evolution of terrorist networks under a framework of strategic users [58]: *How is the network designed to hide the central or influential users aka the leaders?*

Waniek et al. [58] first propose the Hiding Leader problem which incorporates the viewpoint of the leaders of a criminal organization. It also explicitly models knowledge of the criminals about SNA tools that are used to detect them and thus help in dismantling their organization. Intuitively, the input in the Hiding Leader problem is a network with a subset of vertices marked as leaders. The goal is to add fewest edges to ensure that various SNA tools do not rank any leader high based on centrality measures thereby capturing the *efficiency vs secrecy dilemma* that the criminals are believed to possess [85, 101, 102]. Waniek et al. show promising results that the Hiding Leader problem is computationally intractable even for the simplest degree centrality measure.

4.1.1 Contribution

In this chapter, we study the Hiding Leader problem for the core centrality measure and show that the degree centrality measure is much more computationally vulnerable than the core centrality measure although the Hiding Leader problem is NP-complete for both of them. We reinforce our above claim further through extensive empirical evaluations. Our specific contributions for the Hiding Leader problem are as follows.

- We show that the Hiding Leader problem for degree centrality is polynomial time solvable if the degree of every leader is bounded by some constant [Theorem 17].

- We present a 2 factor approximation algorithm for the Hiding Leader problem for degree centrality which optimizes the number of edges added [Theorem 18]. We complement this by proving that, if there exists a $(2 - \varepsilon)$ factor approximation algorithm for the above problem for any constant $0 < \varepsilon < 1$, then there exists a $^{\varepsilon}/_{2}$ factor approximation algorithm for the Densest k-Subgraph problem [Theorem 19] which would be considered a substantial breakthrough. To the best of our knowledge, the state of the art algorithm for the Densest k-Subgraph problem achieves an approximation ratio of $\tilde{\mathcal{O}}(n^{1/4})$ only [103].

- For the core centrality measure, we show that the Hiding Leader problem is NP-complete even if the core centrality of every leader is exactly 3 [Theorem 20]. We prove that our result is almost tight in the sense that the Hiding Leader problem is polynomial time solvable if the core centrality of every leader is at most 1 [Theorem 1]. Moreover, we also prove that there does not exist any $(1 - \alpha) \ln n$

factor approximation algorithm for any constant $\alpha \in (0, 1)$ which optimizes the number of edges that one needs to add even when the core centrality of every leader is 3 [Corollary 21].

- We show that a construction of a network by Waniek et al. [58], called "captain network" there, hides the leaders with respect to the core centrality measure also.
- We empirically evaluate our 2-approximation algorithm for the degree centrality measure in synthetic networks. We observe that our algorithm almost always produces near optimal results in practice. In the experimental results, we also show the extent in which a leader can hide in the captain network with respect to core centrality.

4.2 Related Work

Waniek et al. first proposed and studied the Hiding Leader problem [58, 104]. They proved that the problem is NP-complete for both the degree and closeness centrality measures. They also proposed a procedure to design a captain (covert) network from scratch which not only hides the leaders based on the degree, closeness, and betweenness centrality measures, but also keeps the influence of the leaders high in the network. In this chapter, we provide two approximability results for degree centrality and core centrality respectively. We also show the problem is harder in the case of core centrality. Liu et al. [105] studied another related problem to make the degree of each node in the network beyond a given constant by adding minimal edges.

Other problems that align with privacy issues in social networks were studied before [106, 107]. In [106], the authors showed how an adversary exploits online social networks to find the private information about users. Altshuler et al. [107] discussed the threat of malware targeted at extracting information in a real-world social network.

Computing centrality and related problems. A significant amount of related work study the computationally complexity of various centrality measures. Brandes [22] first proposed an efficient algorithm to compute the betweenness centrality of a vertex in a network. More recently, Riondato et al. [23] introduced an approach to compute the top-k vertices according to the betweenness centrality using VC-dimension theory. Yoshida [24] studied similar problems for both the betweenness and coverage centrality measures in a group setting. Mahmoody et al. subsequently improved the performance of the above algorithms using a novel sampling scheme [25]. There is an active line of research to optimize the centrality of one node as well as of a set of nodes [26, 27, 28, 13]. Nikos et al. proposed a novel procedure to maximize the expected decrease in shortest path distances from a given node to the remaining nodes via edge addition [21]. Crescenzi et al. [26] proposed greedy algorithms to increase centrality of certain vertices and show effectiveness of their approach through extensive simulation. Kilberg [108] and others studied behavioral models to understand why certain network topologies are common in covert networks [98, 109, 110]. Enders and Su [111] and others develop models to explain various properties like efficiency vs secrecy dilemma etc. of covert networks [112, 113, 114, 115]. Other important direction includes quantifying the influence of vertices; most prominent among them include *Independent Cascade* model [116], *Linear Threshold* model [31], Bass model [117, 118], etc.

Other network design problems. We also provide a few details about previous work on other network modification (design) problems. A set of design problems were introduced in [16]. Lin et al. [3] addressed a shortest path optimization problem via improving edge weights on undirected graphs. The node version of this problem was also studied [7, 14, 42]. Meyerson et al. [10] proposed approximation algorithms for single-source and all-pair shortest paths minimization. Faster algorithms for some of these problems were also presented in [17, 18]. emaine et al. [19] minimized the diameter of a network by adding shortcut edges.

4.3 Preliminaries

For a positive integer ℓ, we denote the set $\{1, 2, \ldots, \ell\}$ by $[\ell]$. A network or graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a tuple consisting of a finite set $\mathcal{V}$ (or $\mathcal{V}[\mathcal{G}]$) of n vertices and a set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ of edges (also denoted by $\mathcal{E}[\mathcal{G}]$). A network is called *undirected* if we have $(x, y) \in \mathcal{E}$ whenever we have $(y, x) \in \mathcal{E}$ for any $x, y \in \mathcal{V}$ with $x \neq y$. A *self loop* is an edge of the form (x, x) for some $x \in \mathcal{V}$. In this chapter, we focus on undirected networks without any self loop. The *degree* of a vertex x is the number of edges incident on it which is $|\{e \in \mathcal{E} : x \in e\}|$. A *subgraph* of a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a network $\mathcal{H} = (\mathcal{U}, \mathcal{F})$ such that $\mathcal{U} \subseteq \mathcal{V}$ and $\mathcal{F} \subseteq \mathcal{E} \cap (\mathcal{U} \times \mathcal{U})$. For a positive integer k, a subgraph $\mathcal{H}$ of a network $\mathcal{G}$ is called a *k-core* if the degree of every vertex in $\mathcal{H}$ at least k. The *core number* of a vertex x in a network is the largest integer k such that x belongs to a k-core.

4.3.1 Network Centrality

Let $\mathcal{G}$ be any network. Bavelas [81] introduces the notion of centrality of vertices. Intuitively, centrality measures try to capture the importance of a vertex in a network. Shaw [82] proposes the *degree centrality* measure which has turned out to be one of the most useful measures. The degree centrality of a vertex x in $\mathcal{G}$ is the degree $\deg_{\mathcal{G}}(x)$ of x in the network, that is $|\{y \in \mathcal{V}[\mathcal{G}] : \{x, y\} \in \mathcal{E}[\mathcal{G}]\}|$.

Seidman [84] introduces the idea of *core centrality* which is particularly useful for finding network cohesion. For an integer k, a *k-core* is a subgraph $\mathcal{H}$ of $\mathcal{G}$ such that the degree of every vertex in $\mathcal{H}$ is at least k. The core number of a vertex x in $\mathcal{G}$ is the largest k such that x belongs to a k-core, that is $\max\{k \in \mathbb{N} : \exists \mathcal{H} \subseteq \mathcal{G}, \deg_{\mathcal{H}}(x) \geq$

$k \forall x \in \mathcal{V}[\mathcal{H}]\}$. The core centrality of a vertex x in $\mathcal{G}$ is its core number in the network. Other popular network centrality measures includes closeness centrality [83], betweenness centrality [119, 120], etc.

4.3.2 Problem Definition

Intuitively, the input in the Hiding Leader problem is a network with a subset of vertices marked as leaders (and the other vertices are followers), a budget b which is the maximum number of edges that we can add in the network, and a target d which is the minimum number of followers whose centrality must be at least as high as the centrality of any leader in the resulting network (after addition of the new edges). We now define our problem formally. In Definition 8, $c(\cdot, \cdot)$ denote either degree centrality or core centrality.

Definition 8 (Hiding Leader (HL)) *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a subset $\mathcal{L} \subseteq \mathcal{V}$ of leader vertices, an integer b denoting the maximum number of edges that we are allowed to add in $\mathcal{G}$, an integer d denoting the number of follower vertices in $\mathcal{F} = \mathcal{V} \setminus \mathcal{L}$ whose final centrality should be at least as high as any leader, the goal is to compute if there exists a subset $\mathcal{W} \subseteq \mathcal{F} \times \mathcal{F}$ of edges between followers such that the conditions below hold.*

1. $|\mathcal{W}| \leq b$
2. $\exists_{\mathcal{F}' \subseteq \mathcal{F}} |\mathcal{F}'| \geq d$ *such that* $c(\mathcal{G}', f) \geq c(\mathcal{G}', l), \forall f \in \mathcal{F}', l \in \mathcal{L}$ *where* $\mathcal{G}' = (\mathcal{V}, \mathcal{E} \cup \mathcal{W})$.

4.4 Results for Degree Centrality

We present our algorithmic and hardness results for the Hiding Leader problem for the degree centrality measure in this section. We begin with presenting our polynomial time algorithm for the Hiding Leader problem for the degree centrality measure when the degree of every leader in the network is bounded above by any constant. On a high level, our algorithm makes greedy choices as long as it can and uses local search technique when "stuck."

Theorem 17 *There exists a polynomial time algorithm for the* Hiding Leader *problem for degree centrality if the degree of every leader is bounded by any constant.*

Proof: Let $\mathcal{G}$ be the input graph and k the highest degree of any leader; that is $k = \max\{\deg_{\mathcal{G}}(l) : l \in L\}$. We are given that k is a constant. If the number of followers is at most $2k$, then there are at most $\binom{2k}{2}$ (which is a constant) new edges that we can add and we try all possible subsets of it of cardinality at most b. The number of such subsets is at most $2^{\binom{2k}{2}}$ which is a constant and thus we can output correctly in polynomial time. So let us assume that the number of followers is at least $2k + 1$. Similarly we can also assume without loss of generality that the budget b is at least $4k^2$ since otherwise we will try to add all possible b new edges (there are only $\mathcal{O}(n^{2b}) = n^{\mathcal{O}(1)}$ possibilities since $k = \mathcal{O}(1)$) and thus we can output correctly in polynomial time.

Suppose there are already d' number of followers in $\mathcal{G}$ whose degrees are at least k. If $d' \geq d$, we output yes. Otherwise let us assume without loss of generality that $d' < k$. Let $\mathcal{X} \subseteq \mathcal{F}$ with $|\mathcal{X}| = d - d'$ be the set of top $d - d'$ highest degree followers in the network whose degrees are less than k. Intuitively, our algorithm greedily adds new edges between two vertices in $\mathcal{X}$ whose degrees are less than k until it is stuck and removes some edge it had added before to make progress. Concretely, our algorithm works as follows. To distinguish existing (old) edges from newly added edges (by the algorithm) in $\mathcal{G}$, we color the existing (old) edges as red and whenever we add a new edge, we color it green. To begin with, all the edges in $\mathcal{G}$ are colored red and there is no green edge. We apply the following step $(\star)$ as long as we can. If the number of green edges in $\mathcal{G}$ is less than b and there exist two vertices $x, y \in \mathcal{X}$ such that the degrees of both the vertices are less than k and there is no edge between them, then we add an edge $\{x, y\}$ in $\mathcal{G}$ and color it green. Such a pair of vertices (if exists) can be found in $\mathcal{O}(n^2)$. If such a pair of vertices does not exist in $\mathcal{G}$, then one of the following four cases must hold.

Case 1*: The degree of every vertex in $\mathcal{X}$ is at least k.* In this case, we output YES.

Case 2*: The number of green edges in $\mathcal{G}$ is b.* In this case, we output YES if the degree of every vertex in $\mathcal{X}$ is at least k; otherwise we output NO.

Case 3*: There exists exactly one vertex $x \in \mathcal{X}$ with degree less than k.* If the degree of x is $k-1$, then we add an edge between x and any vertex $y \in F$ such that there is no edge between x and y in $\mathcal{G}$ and color it green. If the number of green edges in $\mathcal{G}$ is at most b, then we output YES; otherwise we output NO. Otherwise we assume that the degree of x is less than $k-1$. If the number of green edges in $\mathcal{G}$ is at most $3k^2$, then we can add k new green edges on x and answer YES since we have already assumed that $b \geq 4k^2$. So let us assume without loss of generality that the number of green edges in $\mathcal{G}$ is more than $3k^2$. Let $\{u, v\}$ be a green edge such that there is no edge between x and u and between x and v in $\mathcal{G}$. Such a green edge $\{u, v\}$ always exists in $\mathcal{G}$ since the degree of x is less than $k-1$ in $\mathcal{G}$ and there are more than $3k^2$ green edges in $\mathcal{G}$. Moreover such an edge can be found in polynomial time by simply checking all the green edges. We now remove the green edge $\{u, v\}$ from $\mathcal{G}$, add two edges $\{x, u\}$ and $\{x, v\}$ in $\mathcal{G}$, color both of them green, and continue (return to the step $(\star)$).

Case 4*: For every pair of vertices $x, y \in \mathcal{X}, x \neq y$ with degree less than k for both the vertices, there is an edge between them.* Let $\mathcal{Z} \subseteq \mathcal{X}$ be the set of vertices in $\mathcal{X}$ with degree less than k. Since there exists an edge between every pair of vertices in $\mathcal{Z}$ in this case and the degree of every vertex in $\mathcal{Z}$ is less than k, we have $|\mathcal{Z}| \leq k$. If the number of green edges in $\mathcal{G}$ is at most $3k^2$, then we can add k new green edges on every vertex in $\mathcal{Z}$ and answer YES since we have $|\mathcal{Z}| \leq k$ and $b \geq 4k^2$. So let us assume that the number of green edges in $\mathcal{G}$ is more than $3k^2$. Let $a, b \in \mathcal{X}$ be any two vertices. Let $\mathcal{N}(a)$ and $\mathcal{N}(b)$ denote the set of neighbors of a and b in $\mathcal{X}$. Since the degrees of both a and b are less than k, we have $|\mathcal{N}(a)| < k$ and $|\mathcal{N}(b)| < k$. Since the degree of every vertex in $\mathcal{X}$ is at most k and the number of green edges in $\mathcal{G}$ is at least $3k^2$, there exists at least one green edge $\{u, v\}$ in $\mathcal{G}$ which does not incident on any vertex in $\mathcal{N}(a) \cup \mathcal{N}(b) \cup \{a, b\}$ (there can be at most $2k^2$ green edges incident on any vertex in this set). Moreover such an edge can be found in polynomial time by simply checking all the green edges. We now remove the green edge $\{u, v\}$ from $\mathcal{G}$, add two edges $\{a, u\}$ and $\{b, v\}$ in $\mathcal{G}$, color both of them green, and continue (return to the step $(\star)$).

The algorithm always terminates in polynomial time since in every iteration, it adds a green edge and at most b ($< n^2$) green edges could be added – in cases 3 and 4, we have added two green edges and removed only one green edge; the algorithm terminates in cases 1 and 2. Also, whenever the algorithm outputs YES, adding green edges to the graph makes the degree of at least d followers in the network at least k. Hence, if the algorithm outputs YES, the instance is indeed a YES instance. So, let us assume that the algorithm outputs NO. Except (in case 3) when there exists exactly one vertex x in the network with degree less than k and the degree of x is $k - 1$, whenever we add one green edge in total (which is the same as adding two green edges and removing one green edge in cases 3 and 4), the sum of the degrees of all the vertices in $\mathcal{X}$ increases by 2. Hence the number of green edges added in the graph is at most ALG $= \lceil \sum_{x \in \mathcal{X}} {}^{(k - deg(x))}/2 \rceil$. Since the algorithm outputs NO, we have ALG $> b$. We observe that since any edge increases the degree of at most 2 vertices, when the algorithm outputs NO, the instance is indeed a NO instance. Hence the algorithm is correct. ■

We now present a simple 2 factor polynomial time approximation algorithm for the HIDING LEADER problem for degree centrality.

Theorem 18 *There exists a polynomial time algorithm (HLDA) for approximating the budget b in* Hiding Leader *within a factor of 2 for degree centrality.*

Proof: Let $\mathcal{F}' \subseteq \mathcal{F}$ be the set of followers in $\mathcal{F}$ whose degree centrality is at least the degree centrality of every vertex in $\mathcal{L}$. Let $|\mathcal{F}'| = d'$. If $d' \geq d$, then we output an empty set of edges. Let $x_i \in \mathcal{F}, i \in [d-d']$ be the $(d-d')$ followers with highest degree centrality among the vertices in $\mathcal{F} \setminus \mathcal{F}'$. We keep on adding edges with at least one end point in $\{x_i : i \in [d-d']\}$ until the degree centrality of every $x_i, i \in [d-d']$ is at least the degree of every vertex in $\mathcal{L}$ in the resulting graph. When we have d followers with degree at least the degree of every vertex in $\mathcal{L}$, we output the set of edges that we have added. The algorithm adds at most $\sum_{i=1}^{d-d'}(d - \deg(x_i))$ many edges where where $\deg(x_i)$ is the degree of the vertex x_i in the input graph. Since any new edge can increase the sum of the degrees of the followers by at most 2, we have OPT$\geq \sum_{i=1}^{d-d'} {}^{(d-\deg(x_i))}/_{2} \geq {}^{\text{ALG}}/_{2}$ by the choice of $\mathcal{F}'$. Hence our algorithm approximates b by a factor of 2. ■

We now complement our approximation algorithm in Theorem 18 by proving that if there exists a polynomial time approximation algorithm for the Hiding Leader problem for degree centrality with approximation factor $(2-\varepsilon)$ for any constant $\varepsilon > 0$, then there exists a constant factor polynomial time approximation algorithm for the Densest k-Subgraph problem. In the Densest k-Subgraph problem, the input is a graph $\mathcal{G}$ and an integer k and we need to find a subgraph $\mathcal{H}$ of $\mathcal{G}$ on k vertices with highest density. The density of a graph on n vertices is the number of edges in it divided by $\binom{n}{2}$. To the best of our knowledge, we do not know whether there exists any polynomial time algorithm which can distinguish a graph containing a clique of size k from a graph where the density of every sub-graph of size k is at most $({}^{\varepsilon}/_{2})$ (any ${}^{\varepsilon}/_{2}$ factor approximation algorithm for the Densest k-Subgraph problem would be able to distinguish). In fact, none of the known algorithms can distinguish even for some sub-constant values for ε

(see [121] and references therein). We now show that if there exists a $(2-\varepsilon)$ factor approximation algorithm for the Hiding Leader problem for any constant $0 < \varepsilon < 1$, then there exists an $\varepsilon/2$ factor approximation algorithm with the same running time (of the Hiding Leader algorithm).

Theorem 19 *Suppose there exists a $(2-\varepsilon)$ factor polynomial time approximation algorithm for the* Hiding Leader *problem for degree centrality for some constant ε. Then there exists a polynomial time algorithm for distinguishing a graph containing a clique of size k from a graph where the density of every sub-graph of size k is at most $\varepsilon/2$.*

Proof: Let $\mathcal{G}$ be any graph which satisfies either (exactly) one of the following properties.

1. **Completeness:** There exists a clique of size k in $\mathcal{G}$.
2. **Soundness:** The density of any subgraph of $\mathcal{G}$ of size k is at most $\varepsilon/2$.

From $\mathcal{G}$ we construct an instance of Hiding Leader. Intuitively, we introduce a vertex a_v corresponding to every vertex $v \in \mathcal{V}[\mathcal{G}]$ in $\mathcal{G}$ and the edge set among those vertices is the complement of the corresponding edge set in $\mathcal{G}$. To ensure that, in the resulting graph, the degree of every vertex a_v for $v \in \mathcal{V}[\mathcal{G}]$ is n, we add appropriate number of edges between a_v and some auxiliary vertices $d_{(v,\ell)}$ for every $v \in \mathcal{V}[\mathcal{G}], \ell \in [n]$; we ensure that the degree of any such auxiliary vertex is at most 1 which will guarantee that these auxiliary vertices are never part of any optimal solution. Finally we add a clique on a set $\{x_i : i \in [n+k]\}$ of leader vertices so that the degree centrality of every leader is $n+k-1$. Formally, the instance $(\mathcal{H}, \mathcal{L}, d)$ of Hiding Leader is defined as follows.

$$\begin{aligned}
\mathcal{V}[\mathcal{H}] &= \{a_v, d_{(v,\ell)} : v \in \mathcal{V}[\mathcal{G}], \ell \in [n]\} \cup \{x_i : i \in [n+k]\} \\
\mathcal{L} &= \{x_i : i \in [n+k]\} \\
\mathcal{E}[\mathcal{H}] &= \{\{a_u, a_v\} : (u,v) \notin \mathcal{E}[\mathcal{G}]\} \\
&\quad \cup \{\{x_i, x_j\} : 1 \le i < j \le n+k\} \\
&\quad \cup \{\{a_v, d_{(v,\ell)}\} : v \in \mathcal{V}[\mathcal{G}], \ell \in [\deg_{\mathcal{G}}(v)+1]\} \\
d &= k
\end{aligned}$$

We now use the $(2-\varepsilon)$ factor approximation algorithm for the Hiding Leader problem for degree centrality which outputs that there is a way to add b_{ALG} number of edges so that there exist at least d followers in $\mathcal{H}$ whose degree is at least the degree of any leader. Let b_{OPT} denotes the minimum number of edges that one needs to add to ensure that there exist at least d followers in $\mathcal{H}$ whose degrees are at least the degree of every leader. Then we have $b_{ALG} \le (2-\varepsilon)b_{OPT}$. We output that the graph $\mathcal{G}$ contains a k-clique if $b_{ALG} \le (2-\varepsilon)\binom{k}{2}$. Otherwise, we output that the density of any subgraph of $\mathcal{G}$ on k vertices is at most $\varepsilon/2$. We now prove correctness of our algorithm. We first observe that the degree of every leader in $\mathcal{H}$ is $n+k-1$, the degree of $a_v \in \mathcal{V}[\mathcal{H}]$ for every $v \in \mathcal{V}[\mathcal{G}]$ is n, and the degree of every other vertex is at most 1.

1. **Completeness:** Let $\mathcal{W} \subseteq \mathcal{V}[\mathcal{G}]$ with $|\mathcal{W}| = k$ be a clique in $\mathcal{G}$. Let us consider the subset $\mathcal{X} = \{a_v : v \in \mathcal{W}\} \subseteq \mathcal{V}[\mathcal{H}] \setminus \mathcal{L}$. By construction, $\mathcal{X}$ forms an independent set in $\mathcal{H}$ and the degree of every vertex in $\mathcal{H}$ is n. Since, adding all the edges in $\{\{a_u, a_v\} : u, v \in \mathcal{W}, u \ne v\}$ in $\mathcal{H}$ makes the degree of every vertex in $\mathcal{X}$ in the resulting graph $n+k-1$, we have $b_{OPT} \le \binom{k}{2}$. Hence, we have $b_{ALG} \le (2-\varepsilon)b_{OPT} \le (2-\varepsilon)\binom{k}{2}$.

2. **Soundness:** In this case, the density of any subgraph of $\mathcal{G}$ on k vertices is at most $\varepsilon/2$. Let $\mathcal{Y} \subseteq \mathcal{V}[\mathcal{H}] \setminus \mathcal{L}$ with $|\mathcal{Y}| = k$ be a set of any k followers. By the construction of $\mathcal{H}$, we have $|\mathcal{E}[\mathcal{Y}[\mathcal{H}]]| \geq (1 - (\varepsilon/2))\binom{k}{2}$. Hence, the minimum number of edges one needs to add to make the degree of every vertex in $\mathcal{Y}$ at least $n + k - 1$ is at least $k(k-1) - (\varepsilon/2)\binom{k}{2} = (2 - (\varepsilon/2))\binom{k}{2}$. In particular, we have $b_{ALG} \geq b_{OPT} \geq (2 - (\varepsilon/2))\binom{k}{2} > (2 - \varepsilon)\binom{k}{2}$.

This concludes the proof of the statement. ■

4.5 Results for Core Centrality

We present our results for the Hiding Leader problem for the core centrality measure in this section. Unlike in degree centrality case, the problem becomes NP-complete even when the core centrality of every leader is only 3. This is almost tight as we prove that the problem is polynomial time solvable if the core centrality of every leader is at most 1.

In Theorem 20 below, we prove that the Hiding Leader problem is NP-complete even when the core centrality of every leader is 3. We reduce the Set Cover problem to the Hiding Leader problem there. In the Set Cover problem, the input is a universe $\mathcal{U} = \{u_1, u_2, ..., u_n\}$, a collection $\mathcal{S} = \{S_1, S_2, ..., S_m\}$ of subsets of $\mathcal{U}$, and an integer t and we need to compute if there exist at most t sets in $\mathcal{S}$, union of which results in $\mathcal{U}$. It is well known that the Set Cover problem is NP-complete [122].

Theorem 20 *The* Hiding Leader *problem for the core centrality measure is* NP-*complete even when the core centrality of every leader is* 3.

Proof: The Hiding Leader problem for the core centrality measure is clearly in NP. Note that computing core centrality of a node takes polynomial time [89]. To prove

NP-hardness, we reduce from the Set Cover problem. Let $(\mathcal{U} = \{u_1, u_2, ..., u_n\}, \mathcal{S} = \{S_1, S_2, ..., S_m\}, t)$ be an instance of the Set Cover problem. To define a corresponding Hiding Leader problem instance, we construct the graph $\mathcal{G}$ as follows.

Intuitively, for each subset $S_i \in \mathcal{S}$, we create a path of n vertices $X_{i,1}, X_{i,2}, \cdots, X_{i,n}$ in $\mathcal{G}$; $(X_{i,2}, X_{i,3}), \cdots, (X_{i,n-1}, X_{i,n}), (X_{i,n}, X_{i,1})$ are the edges of the above path. We also add 5 vertices $W_{i,1}$ to $W_{i,5}$ with eight edges where the four vertices in $\{W_{i,\ell} : 2 \leq \ell \leq 5\}$ form a clique with six edges; the other two edges are $(W_{i,1}, W_{i,2})$ and $(W_{i,1}, W_{i,5})$. For each $u_j \in \mathcal{U}$, we add a set of 5 vertices $\{Z_{j,\ell} : 1 \leq \ell \leq 5\}$ with eight edges where the four vertices (leaders) $\{Z_{i,\ell} : 2 \leq \ell \leq 5\}$ form a clique with six edges; the other two edges are $(Z_{j,1}, Z_{j,2})$ and $(Z_{j,1}, Z_{j,5})$. We also have an edge $(X_{i,j}, Z_{j,1})$ for every $u_j \in S_i$. We allow to add t new edges and demand that the core centrality of at least $4m+n(t+1)+t$ followers should be at least as high as the core centrality of every leader. Figure 4.1 illustrates the structure of our construction for sets $S_1 = \{u_1, u_2\}, S_2 = \{u_2\}, S_3 = \{u_3, u_4\}$. We now formally describe our Hiding Leader instance.

$$
\begin{aligned}
V[\mathcal{G}] &= \{X_{i,j} : S_i \in \mathcal{S}, u_j \in \mathcal{U}\} \cup V_1\\
\mathcal{E}[\mathcal{G}] &= E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5 \cup E_6 \cup E_7 \cup E_8\\
V_1 &= \{W_{i,p} : S_i \in \mathcal{S}, p \in [5]\} \cup \{Z_{j,p} : u_j \in \mathcal{U}, p \in [5]\}\\
E_1 &= \{(X_{i,j}, X_{i,j+1}) : j \in [n-1] \setminus \{1\}, i \in [m]\}\\
E_2 &= \{(X_{i,n}, X_{i,1}) : i \in [m]\}\\
E_3 &= \{(W_{i,1}, W_{i,p}) | i \in [m], p = 2, 5\}\\
E_4 &= \{(W_{i,1}, X_{i,j}) | u_j \notin S_i, i \in [m], j \in [n]\}\\
E_5 &= \{(Z_{j,1}, Z_{j,p}) | j \in [n], p = 2, 5\}\\
E_6 &= \{(X_{i,j}, Z_{j,1}) | u_j \in S_i, i \in [m], j \in [n]\}\\
E_7 &= \{(W_{i,p}, W_{i,q}) | i \in [m], 2 \leq p < q \leq 5\}\\
E_8 &= \{(Z_{j,p}, Z_{j,q}) | j \in [n], 2 \leq p < q \leq 5\}\\
\mathcal{L} &= \{Z_{j,p} | j \in [n], p = 2, 3, 4, 5\}\\
b &= t, d = 4m + n(t+1) + t
\end{aligned}
$$

We now claim that these two instances are equivalent. In one direction, let us assume that the Set Cover instance is a yes instance. By renaming, let us assume that the collection $\{S_1, \ldots, S_t\}$ forms a valid set cover of the instance. We add the edges in the set $\mathcal{E}' = \{(X_{i,1}, X_{i,2}) : i \in [t]\}$ in the graph $\mathcal{G}$. Let the resulting graph be $\mathcal{H}$. We claim that the core centrality of every vertex in $\{Z_{i,1} : i \in [n]\} \cup \{X_{i,j} : i \in [t], j \in [n]\} \cup \{W_{i,1} : i \in [t]\} \cup \{W_{i,j} : i \in [m], j \in [4]\}$ is 3 in $\mathcal{H}$. We first observe that the core centrality of every leader remains 3 even after adding the edges in $\mathcal{E}'$. Also, for any $i \in [m]$, if the edge $(X_{i,1}, X_{i,2})$ is added in the graph, the core centrality of the $n+1$ vertices $X_{i,t}, t \in [n]$ and

$W_{i,1}$ become 3. Hence after addition of the edges in $\mathcal{E}'$ in $\mathcal{G}$, the core centrality of every vertex in $\{X_{i,j} : i \in [t], j \in [n]\} \cup \{W_{i,1} : i \in [t]\}$ becomes 3. Since $\{S_1, \ldots, S_t\}$ forms a set cover for $\mathcal{U}$, the core centrality of every vertex in $\{Z_{i,1} : i \in [n]\}$ becomes 3. Lastly, the core centrality of every vertex in $\{W_{i,j} : i \in [m], j \in [4]\}$ was already 3 in $\mathcal{G}$ and since addition of edges never decreases the core centrality of any vertex, the core centrality of these vertices are at least 3 in $\mathcal{H}$. Hence the Hiding Leader instance is a yes instance.

For the other direction, let us assume that there exists a set $\mathcal{E}'$ of edges such that in the graph $\mathcal{H} = (\mathcal{V}[\mathcal{G}], \mathcal{E}[\mathcal{G}] \cup \mathcal{E}')$, the core centrality of at least d vertices in $\mathcal{V}[\mathcal{G}] \setminus \mathcal{L}$ is at least 3; let the set of followers with core centrality at least 3 in $\mathcal{H}$ be $\mathcal{Y} \subseteq \mathcal{V}[\mathcal{G}] \setminus \mathcal{L}$. Since adding edges in the graph never decreases the core centrality of any vertex, we have $\{W_{i,j} : i \in [m], j \in [4]\} \subseteq \mathcal{Y}$. Let us consider the following subset $\mathcal{J} \subseteq [m]$ defined as: $\mathcal{J} = \{j \in [m] : \exists 1 \leq i < k \leq n \text{ with } (X_{j,i}, X_{j,k}) \in \mathcal{F}\}$. Since $|\mathcal{F}| \leq b$ and $b = t$, we have $|\mathcal{J}| \leq t$. We claim that $\{S_j : j \in \mathcal{J}\}$ forms a set cover for $\mathcal{U}$. Suppose not, then at most $n - 1$ vertices in $\{Z_{i,1} : i \in [n]\}$ can belong to $\mathcal{Y}$ since $Z_{\ell,1}$ does not belong to $\mathcal{Y}$ if u_ℓ is uncovered. Also, any vertex in $\{X_{i,\ell}, W_{i,1} : i \in [m] \setminus \mathcal{J}, \ell \in [n]\}$ does not belong to $\mathcal{Y}$. Hence, we have $|\mathcal{Y}| \leq 4m + t(n + 1) + n - 1 < d$ which contradicts our assumption that $\mathcal{F}$ forms a valid solution for the Hiding Leader instance. Hence the Set Cover instance is a yes instance. ■

Theorem 20 along with well known inapproximability result for the Set Cover problem immediately give us the following result.

Corollary 21 *There does not exists any polynomial time algorithm for approximating the number of edges one needs to add in the* Hiding Leader *problem for core centrality within an approximation ratio of* $(1 - \alpha) \ln n$ *for any constant* α *assuming* $\mathsf{P} \neq \mathsf{NP}$ *even when the core centrality of every leader is* 3.

Proof: The result follows from the observation that the reduction in the proof of

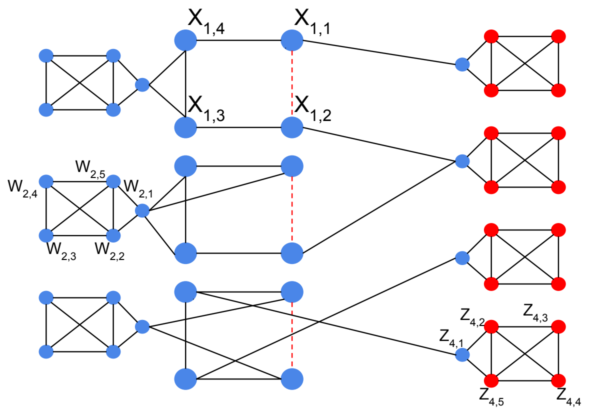


Figure 4.1: Example construction for hardness from Set Cover where $\mathcal{U} = \{u_1, u_2, u_3, u_4\}, S = \{S_1, S_2, S_3\}, S_1 = \{u_1, u_2\}, S_2 = \{u_2\}, S_3 = \{u_3, u_4\}$. The red nodes are the leaders and the blue nodes are the followers.

Theorem 20 is approximation preserving and the $(1-\alpha)\ln n$ inapproximability result for the SET COVER problem for any constant α assuming $\mathsf{P} \neq \mathsf{NP}$ [123]. ■

We now show that the hardness result in Theorem 20 is almost tight in the sense that if the core centrality of every leader is at most 1 in the network, then the corresponding HIDING LEADER problem is polynomial time solvable.

Proposition 1 *There exists polynomial time algorithm for the* HIDING LEADER *problem for core centrality if the core centrality of every leader in the network is at most* 1.

Proof:

We observe that if the degree of any vertex x is at least 1, then its core centrality is at least 1. Let $\mathcal{F}' \subseteq \mathcal{F}$ be the subset of followers whose core centrality is at least 1; say $|\mathcal{F}'| = d'$. Hence the degree of every vertex in $\mathcal{F} \setminus \mathcal{F}'$ is 0. We add $\lceil {}^{(d-d')}/_{2} \rceil$ new edges such that the degree of at least $d - d'$ vertices in $\mathcal{F} \setminus \mathcal{F}'$ becomes at least 1 in the resulting graph. We output YES if $\lceil {}^{(d-d')}/_{2} \rceil \geq b$; otherwise we output NO. Since any optimal solution must add at least $\lceil {}^{(d-d')}/_{2} \rceil$ edges, our algorithm is correct.

■

4.6 Captain Networks

In this section, we show the "captain network", originally proposed by Waniek et al. [58], also ensures that the core centrality of any leader is at most the core centrality of any captain. They propose two constructions; one for single leader and another for multiple leaders.

4.6.1 For Multiple Leaders

We first describe the construction in [58]. The set $\mathcal{L}$ of leaders forms a clique. Each leader $l_i \in \mathcal{L}$ has a corresponding group of p captains $\mathcal{C}_i = \{C_{i,1}, C_{i,2}, \cdots, C_{i,p}\}$ and l_i is connected to all vertices in $\mathcal{C}_i$. Assuming that $|\mathcal{L}| = h \geq 2$, there are h such sets of captains $\{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_h\}$. All vertices in the captain sets are connected as a complete h-partite graph. A captain $C_{i,j}$ serves two things: 1) It helps to hide the leader by being higher or of same centrality than the leader with maximum centrality. 2) It spreads the influence from the leader to the rest of the network. The remaining vertices $\mathcal{X} = \{X_1, X_2, \cdots, X_m\}$ are each connected to one captain from each group C_i. The follower set in the network is $\mathcal{F} = \mathcal{X} \cup \mathcal{C}_1 \cup \mathcal{C}_2 \cup \cdots \cup \mathcal{C}_h$. Let us call the resulting graph $\mathcal{G}_M$. We now show that the core centrality of every leader in $\mathcal{G}_M$ is at most the core centrality of every captain.

Theorem 22 *Given a captain network $\mathcal{G}_M$, let $r = \lfloor \frac{m}{p} \rfloor$ denotes the minimum number of connections that a captain $C_{i,j}$ has with vertices from X. Assuming we have at least 2 leaders and $p > 1$, the core centralities of the captains are either greater or same as the leaders.*

Proof: In $\mathcal{G}_M$, the vertices in $\mathcal{X}$ do not contribute in the core centrality of either the leaders or the captains. We observe that the degree of any vertex in X is h. So their core centrality can be at most h. We claim that the captains and the leaders are in higher core than h. Consider a induced subgraph $G' \subset \mathcal{G}_M$ that includes only the leaders, the captains and the edges between them. In G', the degree of any captain $C_{i,j}$ is $p(h-1)+1$; on the other hand, the degree of any leader l_i is $h-1+p$. So, in G', all the captains and the leaders are at least in $d_{min} = min\{d(G, c_{i,j}), d(G, l_i)\}$-core. This comes from the fact that all the nodes (captains and leaders) have a minimum degree of d_{min} and thus they are at least in d_{min}-core. Note that, $d(G, c_{i,j}) - d(G, l_i) = p(h-1) + 1 - (h-1+p) = (h-2)(p-1) \geq 0$ as $h \geq 2$ and $p > 1$. This implies the captains have higher degree than the leaders in G'. So, the captains have at least the same core-centrality as the leaders. Our claim is proved. Additionally, any vertex in X is in h-core and $h < d_{min}$ assuming $p > 1$. ■

Corollary 23 *Given the same captain network G, assuming $h > 2$ and $p > 1$, the core centrality of all the captains is strictly larger than the leaders.*

Proof: The key idea is that the captains will form a core only among themselves and that will be higher core than the leaders. Now, for any captain $c_{i,j}$ the degree among themselves is $p(h-1)$. Now that, $p(h-1) - (h-1+p) = (h-2)(p-1) - 1 > 0$ or $(h-2)(p-1) > 1$ is possible when $h > 2$ and $p > 1$. So, the captains have larger core-centrality than the leaders. ■

4.6.2 For Single Leader

We show that in the construction from [58], the core centralities of the leaders and the captains remain same when $h = 1$. A single leader l ($h = 1$) has two sets of p captains $C_1 = \{C_{1,1}, C_{1,2}, \cdots, C_{1,p}\}$ and similarly it has C_2. All captain vertices are connected as a complete bipartite graph. Each remaining vertex in $X = \{x_1, x_2, \cdots, x_m\}$ is connected to one captain from each group C_1 and C_2. The follower set is $F = \{X \cup C_1 \cup C_2\}$.

Corollary 24 *Given the captain network described above, let $r = \lfloor \frac{m}{p} \rfloor$ denote the minimal number of connections that a captain, $c_{i,j}$ has with vertices from X. Assuming $h = 1$, the core centralities of all the captains are same as the leader.*

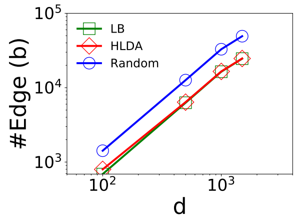
Proof: The proof follows from that of Theorem 22. The leader has degree $2p$ where as the captains have degree $1 + p + r$. But the vertices in X has only degree 2. So the leader and the captains will be in the higher core and it will be $min\{2p, p+1\}$. Assuming $p \geq 2$ all the captain vertices and the leader will be in $p+1$-core. If $p = 1$, all the vertices in the network will be in 2-core. ■

4.7 Simulation Results

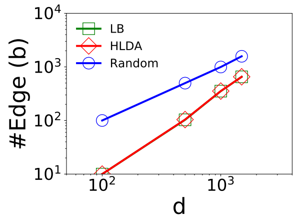
In this section, we evaluate the performance of our 2 approximation algorithm in Theorem 18 using synthetic networks. For brevity, let us call our algorithm in Theorem 18 as HLDA and called the lower bound used in Theorem 18 as LB. We also show how well the leaders can be hidden in the captain network via the core centrality measure. Solutions were implemented in Java and experiments conducted on 3.30 GHz Intel cores with 30 GB RAM.

4.7.1 Evaluation of 2-Approximation Algorithm

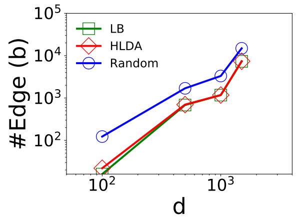
Settings: We generate synthetic network structures from two well-studied models: (a) Barabasi-Albert (BA) [124] and (b) Watts-Strogatz (WS) [125]. While both have "small-world" property, WS do not have a scale-free degree distribution. We generate both the datasets of 70 thousands vertices for three different edge densities: average degree of vertices as 2, 4 and 10. In the experiments we choose 20 leaders ($|\mathcal{L}| = h = 20$) randomly from the top 100 high degree vertices.



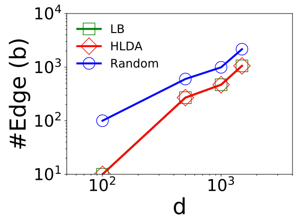
(a) BA (avg. degree = 2)



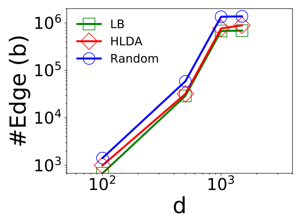
(b) WS (avg. degree = 2)



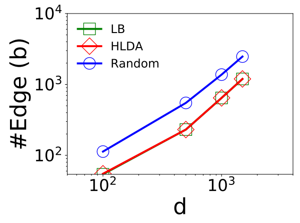
(c) BA (avg. degree = 4)



(d) WS (avg. degree = 4)



(e) BA (avg. degree = 10)



(f) WS (avg. degree = 10)

Figure 4.2: Number of edges added (b) by different algorithms: LB implies a loose lower bound, HLDA is our algorithm that gives 2-approximation and Random denotes a random edge addition algorithm. Clearly, in both networks while varying edge density (average degree of nodes), the number of edge addition by our algorithm HLDA is almost same as that of LB.

Baselines: We compare our algorithm (HLDA) with two baselines. Our first baseline is the lower bound used in Theorem 18 which we call LB. Our second baseline is Random which denotes the number of random edges one needs to add to achieve the goal. The performance metric of the algorithms is the number of edges being added to satisfy the degree centrality requirement for d followers. Hence, the quality is better when the number of edges is lower.

Results: Theorem 18 shows that our algorithm (HLDA) proposed for degree centrality gives a 2-approximation. However in practice it gives near optimal results. Figure 4.2 shows the results varying d on four datasets. Note that, the axes are in logarithmic scale. In all six datasets, the number of solution edges of HLDA is similar to LB. However, Random cannot produce high quality results. Comparing the datasets (BA and WS), the algorithms (HLDA and LB) need higher number of edges in BA as the chosen leaders (randomly chosen from 100 top degree nodes) have much higher degree than the

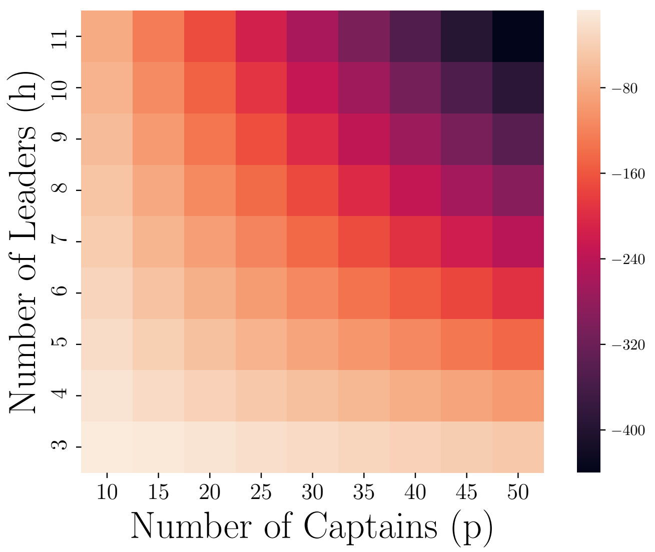


Figure 4.3: Summary of the difference in core centralities between a leader and a captain in a given captain network (with 550 vertices) by varying number of captains in each group (p) and leaders (h).

followers due to the scale-free degree distribution.

4.7.2 Captain Networks and Core Centrality

In section 4.6, we prove that the core centrality of the leaders can be hidden by the captains in the captain networks [58] (Theorem 22 and Corollary 23). We empirically evaluate the core centralities of the the leaders and the captains by varying two parameters: the number of captains (p) in each group $\mathcal{C}_i$ and the number of leaders (h) for network with multiple leaders.

Figure 4.3 presents the results for a captain network with 550 vertices. For every pair of two parameters (p and h), we compute the maximum difference in core centrality between any leader and any captain. The intensity of the color signifies that the difference is higher. Higher difference also implies higher disguise for the leaders. Low values of p result into lower disguise for a leader. On the other hand, a high value of p (large number of captains in each group) with high values of h produces the maximum amount of disguise. But for a high value of p, if the number of leaders are low, i.e., low h, the amount of disguise for the leaders decreases.

We summarize our experimental findings as follows.

- HLDA produces near optimal results in practice, where as, Random cannot produce high quality results. HLDA and LB need more edges to satisfy the degree requirements for d followers in BA due to the scale-free degree distribution.
- A captain network with large number of leaders (large h) and a large number of captains in each group (high value of p) produces the maximum amount of disguise.
- A low value of p, i.e., a small number of captains in each group yields lower disguise for core centrality which is not true for other centralities such as degree, closeness, and betweenness [58].

4.8 Conclusion and Future Work

We have shown that the Hiding Leader problem for the core centrality measure is NP-hard to approximate with a factor of $(1-\alpha)\ln n$ for any constant $\alpha > 0$ for optimizing the number of edges one needs to add even when the core centrality of every leader is only 3. On the other hand, we prove that the Hiding Leader leader problem for degree centrality is polynomial time solvable if the degree of every leader is $\mathcal{O}(1)$. Moreover, we also provide a 2 factor polynomial time approximation algorithm for the Hiding Leader problem for optimizing the number of edges one needs to add to hide all the leaders. Hence, our results prove that, although classical complexity theoretic framework fails to compare relative difficulty of hiding leaders with respect to various centrality measures [58], hiding leaders may be significantly harder for the core centrality than the degree centrality. We complement our 2 factor approximation algorithm for the Hiding

Leader problem for degree centrality by proving that if there exists a $(2-\varepsilon)$ factor approximation algorithm for the Hiding Leader problem for degree centrality for any constant $0 < \varepsilon < 1$, then there exists a $\varepsilon/2$ factor approximation algorithm for the famous Densest k-Subgraph problem which would be considered a major break through. The current best polynomial time algorithm for the Densest k-Subgraph problem achieves an approximation ratio of only $\tilde{\mathcal{O}}(n^{1/4})$ [103]. We have also empirically evaluated our approximation algorithm which shows that our algorithm produces an optimal solution for most of the cases. We have also shown that the captain networks proposed in [58] can hide the leaders with respect to core centrality.

Chapter 5

Core Resilience

K-cores are maximal induced subgraphs where all vertices have degree at least k. These dense patterns have applications in community detection, network visualization and protein function prediction. However, k-cores can be quite unstable to network modifications, which motivates the question: *How resilient is the k-core structure of a network, such as the Web or Facebook, to edge deletions?* We investigate this question from an algorithmic perspective. More specifically, we study *the problem of computing a small set of edges for which the removal minimizes the k-core structure of a network* [12].

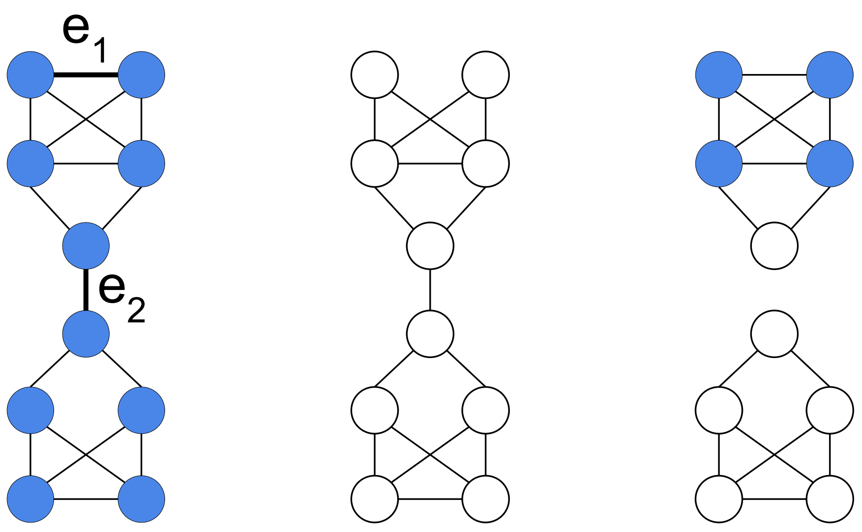
This chapter provides a comprehensive characterization of the hardness of the k-core minimization problem (KCM), including innaproximability and fixed-parameter intractability. Motivated by such a challenge in terms of algorithm design, we propose a novel algorithm inspired by Shapley value—a cooperative game-theoretic concept— that is able to leverage the strong interdependencies in the effects of edge removals in the search space. As computing Shapley values is also NP-hard, we efficiently approximate them using a randomized algorithm with probabilistic guarantees. Our experiments, using several real datasets, show that the proposed algorithm outperforms competing solutions in terms of k-core minimization while being able to handle large graphs. Moreover, we illustrate how KCM can be applied in the analysis of the k-core resilience of networks.

5.1 Introduction

K-cores play an important role in revealing the higher-order organization of networks. A k-core [84] is a maximal induced subgraph where all vertices have internal degree of at least k. These cohesive subgraphs have been applied to model users' engagement and viral marketing in social networks [89, 88]. Other applications include anomaly detection [126], community discovery [127], protein function prediction [128], and visualization [129, 130]. However, the k-core structure can be quite unstable under network modification. For instance, removing only a few edges from the graph might lead to the collapse of its core structure. This motivates the k-core minimization problem: *Given a graph G and constant k, find a small set of b edges for which the removal minimizes the size of the k-core structure [131].*

We motivate k-core minimization using the following applications: (1) *Monitoring:* Given an infrastructure or technological network, which edges should be monitored for attacks [132, 133]? (2) *Defense:* Which communication channels should be blocked in a terrorist network in order to destabilize its activities [134, 135]? and (3) *Design:* How to prevent unraveling in a social or biological network by strengthening connections between nodes [89, 136]?

Consider a specific application of k-cores to online social networks (OSNs). OSN users tend to perform activities (e.g., joining a group, playing a game) if enough of their friends do the same [137]. Thus, strengthening critical links between users is key to the long-term popularity, and even survival, of the network [138]. This scenario can be modeled using k-cores. Initially, everyone is engaged in the k-core. Removal of a few



(a) Initial G (b) Modification G' (c) Modification G''

Figure 5.1: K-core minimization for an illustrative example: (a) Initial graph, where all the vertices are in the 3-core; (b) Removing e_1 causes all the vertices to leave the 3-core; (c) Removing e_2 causes only six vertices to leave the 3-core.

links (e.g., unfriending, unfollowing) might not only cause a couple of users to leave the network but produce a mass exodus due to cascading effects. This process can help us to understand the decline and death of OSNs such as Friendster [139].

K-core minimization (KCM) can be motivated both from the perspective of a centralized agent who protects the structure of a network or an adversary that aims to disrupt it. Moreover, our problem can also be applied to measure network resilience [133].

We illustrate KCM in Figure 5.1. An initial graph G (Figure 5.1a), where all vertices are in the 3-core, is modified by the removal of a single edge. Graphs G' (Figure 5.1b) and G'' (Figure 5.1b) are the result of removing e_1 and e_2, respectively. While the removal of e_1 brings all the vertices into a 2-core, deleting e_2 has a smaller effect—four vertices remain in the 3-core. Our goal is to identify a small set of edges removal of which minimizes the size of the k-core.

From a theoretical standpoint, for any objective function of interest, we can define a *search* (e.g. the k-core decomposition) and a corresponding *modification* problem, such as k-core minimization. In this chapter, we show that, different from its search version [140], KCM is NP-hard. Furthermore, there is no polynomial time algorithm that achieves a constant-factor approximation for our problem. Intuitively, the main challenge stems from the strong combinatorial nature of the effects of edge removals. While removing a single edge may have no immediate effect, the deletion of a small number of edges might cause the collapse of the k-core structure. This behavior differs from more popular problems in graph combinatorial optimization, such as submodular optimization, where a simple greedy algorithm provides constant-factor approximation guarantees.

The algorithm for k-core minimization proposed in this chapter applies the concept of *Shapley values* (SVs), which, in the context of cooperative game theory, measure the contribution of players in coalitions [141]. Our algorithm selects edges with largest Shapley value to account for the joint effect (or cooperation) of multiple edges. Since computing SVs is NP-hard, we approximate them in polynomial time via a randomized algorithm with quality guarantees.

Recent papers have introduced the KCM problem [131] and its vertex version [142], where the goal is to delete a few vertices such that the k-core structure is minimized. However, our work provides a stronger theoretical analysis and more effective algorithms that can be applied to both problems. In particular, we show that our algorithm outperforms the greedy approach proposed in [131].

Our main contributions are summarized as follows:

- We study the k-core minimization (KCM) problem, which consists of finding a small set of edges, removal of which minimizes the size of the k-core structure of a network.

- We show that KCM is NP-hard, even to approximate by a constant for $k \geq 3$.

We also discuss the parameterized complexity of KCM and show the problem is $W[2]$-hard for the same values of k.

- Given the above inapproximability result, we propose a randomized Shapley Value based algorithm that efficiently accounts for the interdependence among the candidate edges for removal.
- We show that our algorithm is both accurate and efficient using several datasets. Moreover, we illustrate how KCM can be applied to profile the structural resilience of real networks.

5.2 Related Work

5.2.1 K-core computation and applications:

A k-core decomposition algorithm was first introduced by Seidman [84]. A more efficient solution—with time complexity $O(|E|)$—was presented by Batagelj et al. [140] and its distributed version was proposed in [143]. Sariyuce et al. [144] proposed algorithms k-core decomposition in streaming data. For the case of uncertain graphs, where edges have probabilities, Bonchi et al. [145] introduced efficient algorithms for the problem. The k-core decomposition has been used in many applications. K-cores are often applied in the analysis and visualization of large scale complex networks [129]. Other applications include clustering and community detection [146], characterizing the Internet topology [130], and analyzing the structure of software systems [147]. In social networks, k-cores are usually associated with models for user engagement. Bhawalkar et al. [89] studied the problem of increasing the size of k-core by anchoring a few vertices initially outside of the k-core. Chitnis et al. [148] proved stronger inapproximation results for the anchoring problem. Malliaros et al. [149] investigated user engagement dynamics via k-core decomposition.

5.2.2 Network Resilience/Robustness

Understanding the behavior of a complex system (e.g. the Internet, the power grid) under different types of attacks and failures has been a popular topic of study in network science [150, 151, 152]. This line of work is mostly focused on non-trivial properties of network models, such as critical thresholds and phase transitions, assuming random or simple targeted modifications. Najjar et al. [153] and Smith et al. [154] apply graph theory to evaluate the resilience of computer systems, specially communication networks. An overview of different graph metrics for assessing robustness/resilience is given by [1]. Malliaros et al. [155] proposed an efficient algorithm for computing network robustness based on spectral graph theory. The appropriate model for assessing network resilience and robustness depends on the application scenario and comparing different such models is not the focus of our work.

5.2.3 Resilience of k-core

Adiga et al. [156] studied the stability of high cores in noisy networks. Laishram et al. [133] recently introduced a notion of resilience in terms of the stability of k-cores against deletion of random nodes/edges. If the rank correlation of core numbers before and after the removal is high, the network is core-resilient. They also provided an algorithm to increase resilience via edge addition. Notice that this is different from our problem, as we search for edges that can destroy the stability of the k-core. Another related paper is the work by Zhang et al. [142]. Their goal is to find b vertices such that their deletion reduces the k-core maximally. The k-core minimization problem via edge deletion has been recently proposed by Zhu et al. [131]. However, here we provide stronger inapproximability results and a more effective algorithm for the problem, as shown in our experiments.

5.2.4 Shapley Value (SV) and combinatorial problems

A Shapley value based algorithm was previously introduced for influence maximization (IM) [157]. However, IM can be approximated within a constant-factor by a simple greedy algorithm due to the submodular property [31]. In this work, we use Shapley value to account for the joint effect of multiple edges in the solution of the KCM problem, for which we have shown stronger inapproximability results.

5.3 Problem Definition

We assume $G(V, E)$ to be an undirected and unweighted graph with sets of vertices V ($|V| = n$) and edges E ($|E| = m$). Let $d(G, u)$ denote the degree of vertex u in G. An induced subgraph, $H = (V_H, E_H) \subset G$ is the following: if $u, v \in V_H$ and $(u, v) \in E$ then $(u, v) \in E_H$. The k-core [84] of a network is defined below.

Definition 9 ***k-Core:*** *The k-core of a graph G, denoted by $C_k(G) = (V_k(G), E_k(G))$, is defined as a maximal induced subgraph that has vertices with degree at least k.*

Figure 5.2 shows an example. The graphs in Figures 5.2b and 5.2c are the 2-core and the 3-core, respectively, of the initial graph in Figure 5.2a. Note that, $C_{k+1}(G)$ is a subgraph of $C_k(G)$. Let $\mathcal{E}_G(v)$ denote the core number of the node v in G. If $v \in V_k(G)$ and $v \notin V_{k+1}(G)$ then $\mathcal{E}_G(v) = k$. K-core decomposition can be performed in time $O(m)$ by recursively removing vertices with degree lower than k [140].

Let $G^B = (V, E \setminus B)$ be the modified graph after deleting a set B with b edges. Deleting an edge reduces the degree of two vertices and possibly their core numbers. The

Symbols	**Definitions and Descriptions**
$G(V, E)$	Given graph (vertex set V and edge set E)
n	Number of nodes in the graph
m	Number of edges in the graph
$C_k(G) = (V_k(G), E_k(G))$	The k-core of graph G
$N_k(G)$	$\lvert V_k(G)\rvert$, #nodes in the k-core of G
$M_k(G)$	$\lvert E_k(G)\rvert$, #edges in the k-core of G
Γ	Candidate set of edges
b	Budget
$\mathscr{V}(P)$	The value of a coalition P
Φ_e	The Shapley value of an edge e
$P_e(\pi)$	Set of edges before e in permutation π

Table 5.1: Frequently used symbols

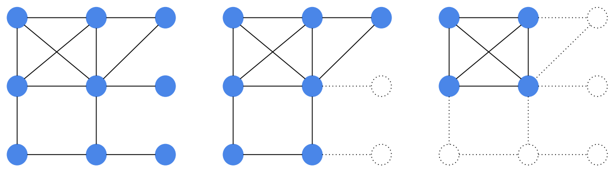
reduction in core number might propagate to other vertices. For instance, the vertices in a simple cycle are in the 2-core but deleting any edge from the graph moves all the vertices to the 1-core. Let $N_k(G) = |V_k(G)|$ and $M_k(G) = |E_k(G)|$ be the number of nodes and edges respectively in $C_k(G)$.

Definition 10 ***Reduced** k**-Core:*** *A reduced k-core, $C_k(G^B)$ is the k-core in G^B, where $G^B = (V, E \setminus B)$.*

Example 3 *Figures 5.3a and 5.3b show an initial graph, G and modified graph G^B (where $B = \{(a, c)\}$) respectively. In G, all the nodes are in the 3-core. Deleting (a, c) brings the vertices a and c to the 2-core and thus b and d also go to the 2-core.*

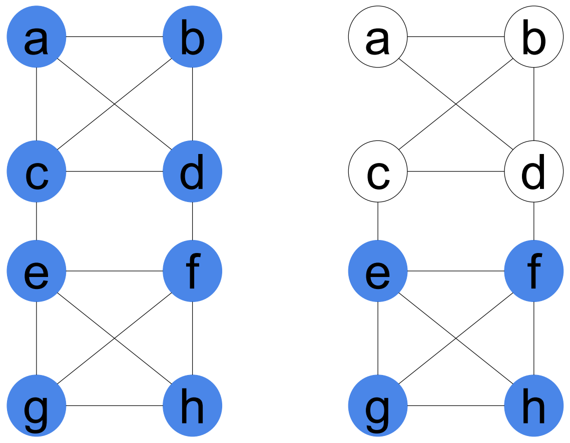
Definition 11 *K**-Core Minimization (KCM):*** *Given a candidate edge set Γ, find the set, $B \subset \Gamma$ of b edges to be removed such that $C_k(G^B)$ is minimized, or, $f_k(B) = N_k(G) - N_k(G^B)$ is maximized.*

Example 4 *Figures 5.3a shows an initial graph, G, where all the nodes are in the 3-core. Deleting (a, c) and (e, g) brings all the vertices to the 2-core, whereas deleting (e, c) and (d, f) has no effect on the 3-core structure (assuming b=2).*



(a) Initial graph, G (b) The 2-core of G (c) The 3-core of G

Figure 5.2: Examples of (a) a graph G; (b) its 2-core; and (c) its 3-core structures.



(a) Initial, G (b) Modified, G^B

Figure 5.3: Example of the changes in the core structure via deletion of an edge: (a) All the nodes are in the 3-core. (b) In the modified graph, the nodes $\{a, b, c, d\}$ are in the 2-core.

Clearly, the importance of the edges varies in affecting the k-core upon their removal. Next, we discuss strong inapproximability results for the KCM problem along with its parameterized complexity.

5.3.1 Hardness and Approximability

The hardness of the KCM problem stems from two major facts: 1) There is a combinatorial number of choices of edges from the candidate set, and 2) there might be strong dependencies in the effects of edge removals (e.g. no effect for a single edge but cascading effects for subsets of edges). We show that KCM is NP-hard to approximate within any constant factor for $k \geq 3$.

Theorem 25 *The KCM problem is NP-hard for $k = 1$ and $k = 2$.*

Proof: First, we sketch the proof for $k = 1$. Consider an instance of the NP-hard 2-MINSAT [158] problem which is defined by a set $U = \{u_1, u_2, ..., u_{m'}\}$ of m' variables and collection $C' = \{c_1, c_2, ..., c_{n'}\}$ of n' clauses. Each clause $c \in C'$ has two literals ($|c| = 2$). So, each $c_i \in C'$ is of the form $z_{i1} \vee z_{i2}$ where z_{ij} is a literal and is either a variable or its negation. The problem is to decide whether there exists a truth assignment in U that satisfies no more than $n^* < n'$ clauses in C. To define a corresponding KCM instance, we construct the graph G' as follows. We create a set of n' vertices $X_c = \{v_i| \ c_i \in C'\}$. For each variable $u_i \in U$, we create two vertices: one for the variable (w_{i1}) and another for its negation (w_{i2}). Thus, a total of $2m'$ vertices, $Y_u = \{w_{11}, w_{12}, w_{21}, w_{22} \ , \ldots, w_{m'1}, w_{m'2}\}$ are produced. Moreover, whenever the literal $u_i \vee \bar{u}_j \in c_t$, we add two edges, (v_t, w_{i1}) and (v_t, w_{j2}) to G'.

For $k = 1$, KCM aims to maximize the number of isolated vertices (0-core, k=0) via removing b edges. An edge in the KCM instance (K_I) is a vertex v_i in G'. Each vertex is connected to exactly two vertices (end points of the edge in K_I) in Y_u. Satisfying a clause is equivalent to removing the corresponding vertex (deleting the edge in K_I) from G'. A vertex in Y_u will be isolated when all associated clauses (or vertices) in X_c are satisfied (removed). If there is a truth assignment which satisfies no more than $b = n^*$ clauses in 2-MINSAT, that implies m' vertices can be isolated in G' by removing $\leq b$ vertices (or deleting $\leq b$ edges in KCM). If there is none, then m' vertices cannot be isolated by removing $\leq b$ edges in KCM.

To prove for $k = 2$, we can transform [142] the $k = 1$ version of the KCM to the $k = 2$ one. ■

Notice that a proof for Theorem 25 is also given by [131]. However, our proof applies a different construction and was developed independently from (and simultaneously with) this previous work.

Theorem 26 *The KCM problem is NP-hard and it is also NP-hard to approximate within a constant-factor for all $k \geq 3$.*

Proof: We sketch the proof for $k=3$ (similar for $k>3$).

Let $SK(U, \mathcal{S}, P, W, q)$ be an instance of the Set Union Knapsack Problem [159], where $U = \{u_1, \dots u_{n'}\}$ is a set of items, $\mathcal{S} = \{S_1, \dots S_{m'}\}$ is a set of subsets ($S_i \subseteq U$), $p : \mathcal{S} \to \mathbb{R}_+$ is a subset profit function, $w : U \to \mathbb{R}_+$ is an item weight function, and $q \in \mathbb{R}_+$ is the budget. For a subset $\mathcal{A} \subseteq \mathcal{S}$, the weighted union of set $\mathcal{A}$ is $W(\mathcal{A}) = \sum_{e \in \cup_{t \in \mathcal{A}} S_t} w_e$ and $P(\mathcal{A}) = \sum_{t \in \mathcal{A}} p_t$. The problem is to find a subset $\mathcal{A}^* \subseteq S$ such that $W(\mathcal{A}^*) \leq q$ and $P(\mathcal{A}^*)$ is maximized. SK is NP-hard to approximate within a constant factor [160].

We reduce a version of SK with equal profits and weights (also NP-hard to approximate) to the KCM problem. The graph G' is constructed as follows. For each $u_j \in U$, we create a cycle of m' vertices $Y_{j,1}, Y_{j,2}, \dots, Y_{j,m'}$ in V and add $(Y_{j,1}, Y_{j,2})$, $(Y_{j,2}, Y_{j,3}), \dots, (Y_{j,m'-1}, Y_{j,m'}), (Y_{j,m'}, Y_{j,1})$ as edges between them. We also add 5 vertices $Z_{j,1}$ to $Z_{j,5}$ with eight edges where the four vertices $Z_{j,2}$ to $Z_{j,5}$ form a clique with six edges. The other two edges are $(Z_{j,1}, Z_{j,2})$ and $(Z_{j,1}, Z_{j,5})$. Moreover, for each subset S_i we create a set of $O((m')^3)$ vertices (sets $X_{i,*}$ are red rectangles in Figure 5.4), such that each node has exactly degree 3, and add one more node $X_{i,1}$ with two edges incident to two vertices in $X_{i,*}$ from $X_{i,1}$. In the edge set E, an edge $(X_{i,1}, Y_{j,i})$ will be added if $u_j \in S_i$. Additionally, if $u_j \notin S_i$, the edge $(Y_{j,i}, Z_{j,1})$ will be added to E. Figure 5.4 illustrates our construction for a set $S_1 = \{u_1, u_2\}, S_2 = \{u_1, u_3\}, S_3 = \{u_2\}$.

In KCM, the number of edges to be removed is the budget, b. The candidate set

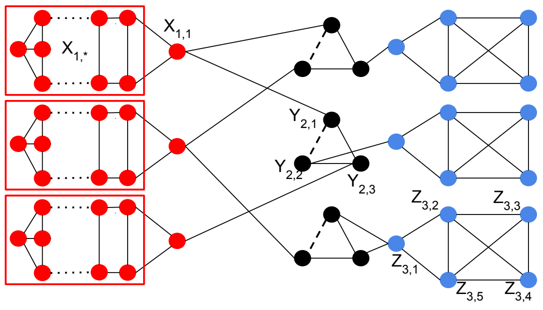


Figure 5.4: Example construction for hardness reduction from SK where $U = \{u_1, u_2, u_3\}, S = \{S_1, S_2, S_3\}, S_1 = \{u_1, u_2\}, S_2 = \{u_1, u_3\}, S_3 = \{u_2\}$.

of edges, Γ is the set of all the edges with form $(Y_{j,1}, Y_{j,2})$. Initially all the nodes in G' are in the 3-core. Our claim is, for any solution $\mathcal{A}$ of an instance of SK there is a corresponding solution set of edges, B (where $|B| = b$) in G' of the KCM problem, such that $f_3(B) = P(\mathcal{A}) + b(m' + 1)$ if the edges in $\mathcal{A}$ are removed.

The m' nodes in any Y_j and the node $Z_{j,1}$ will be in the 2-core if the edge $(Y_{j,1}, Y_{j,2)}$ gets removed. So, removal of any b edges from Γ enforces $b(m' + 1)$ nodes to go to the 2-core. But the node $X_{i,1}$ and each node in $X_{i,*}$ ($O((m')^3)$ nodes) will be in the 2-core iff all its neighbours in $Y_{j,i}$s go to the 2-core after the removal of b edges in Γ. Thus, an optimal solution B^* will be $f_3(B^*) = P(\mathcal{A}^*) + b(m' + 1)$ where $\mathcal{A}^*$ is the optimal solution for SUKP. For any non-optimal solution B, $f_3(B) = P(\mathcal{A}) + b(m' + 1)$ where $\mathcal{A}$ is also non-optimal solution for SUKP. As $P(\mathcal{A}^*)$ is at least $O((m')^3)$ by construction (i.e. $P(\mathcal{A}^*) \gg b(m' + 1)$), and $\frac{P(\mathcal{A}^*)}{P(\mathcal{A})}$ cannot be within a constant factor, $\frac{f_3(B^*)}{f_3(B)}$ will also not be within any constant factor. ■

Theorem 26 shows that there is no polynomial-time constant-factor approximation for KCM when $k \geq 3$. This contrasts with well-known NP-hard graph combinatorial problems in the literature [31]. In the next section, we explore the hardness of our problem further in terms of exact exponential algorithms with respect to the parameters.

5.3.2 Parameterized Complexity

There are several NP-hard problems with exact solutions via algorithms that run in exponential time in the size of the parameter. For instance, the NP-hard Vertex Cover can be solved via an exhaustive search algorithm in time $2^{b_1}{n_1}^{O(1)}$ [161], where b_1 and n_1 are budget and the size of the graph instance respectively. Vertex cover is therefore fixed-parameter tractable (FPT) [162], and if we are only interested in small b_1, we can solve the problem in polynomial time. We investigate whether the KCM problem is also in the FPT class.

A parameterized problem instance is comprised of an instance X in the usual sense, and a parameter b. A problem with parameter b is called fixed parameter tractable (FPT) [162] if it is solvable in time $g(b) \times p(|X|)$, where g is an arbitrary function of b and p is a polynomial in the input size $|X|$. Just as in NP-hardness, there exists a hierarchy of complexity classes above FPT. Being hard for one of these classes is an evidence that the problem is unlikely to be FPT. Indeed, assuming the Exponential Time Hypothesis, a problem which is $W[1]$-hard does not belong to FPT. The main classes in this hierarchy are: FPT$\subseteq W[1] \subseteq W[2] \subseteq \dots W[P] \subseteq XP$. Generally speaking, the problem is harder when it belongs to a higher $W[.]$-hard class in terms of the parameterized complexity. For instance, *dominating set* is in $W[2]$ and is considered to be harder than *maximum independent set*, which is in $W[1]$.

Definition 12 ***Parameterized Reduction [162]:*** *Let P_1 and P_2 be parameterized problems. A parameterized reduction from P_1 to P_2 is an algorithm that, given an instance (X_1, b_1) of P_1, outputs an instance (X_2, b_2) of P_2 such that: (1) (X_1, b_1) is a yes-instance of P_1 iff (X_2, b_2) is a yes-instance of P_2; (2) $b_2 \leq h(b_1)$ for some computable (possibly exponential) function h; and (3) the running time of the algorithm is $g(b_1) \cdot |X|^{O(1)}$ for a computable function g.*

Theorem 27 *The KCM problem is not in FPT, in fact, it is in $W[2]$ parameterized by b for $k \geq 3$.*

Proof: We show a parameterized reduction from the Set Cover problem. The Set Cover problem is known to be $W[2]$-hard [145]. We sketch the proof for $k = 3$. A similar construction can be applied for $k > 3$.

Consider an instance of the $W[2]$-hard Set Cover [145] problem, defined by a collection of subsets $S = \{S_1, S_2, ..., S_m\}$ from a universal set of items $U = \{u_1, u_2, ..., u_n\}$. The problem is to decide whether there exist b subsets whose union is U. We define a corresponding KCM instance (graph G) as follows.

For each $S_i \in S$ we create a cycle of n vertices $X_{i,1}, X_{i,2}, \cdots, X_{i,n}$ in V with edges $(X_{i,1}, X_{i,2}), (X_{i,2}, X_{i,3}), \cdots, (X_{i,n}, X_{i,1})$. We also add 5 vertices $W_{i,1}$ to $W_{i,5}$ with cliques of four vertices $W_{i,2}$ to $W_{i,5}$ and add two more edges, $(W_{i,1}, W_{i,2}), (W_{i,1}, W_{i,5})$. Moreover, for each $u_j \in U$, we create a cycle of m vertices $Y_{j,1}, Y_{j,2}, \cdots, Y_{j,m}$ with edges $(Y_{j,1}, Y_{j,2}), \cdots, (Y_{j,m-1}, Y_{j,m}), (Y_{j,m}, Y_{j,1})$. We add 5 vertices $Z_{j,1}$ to $Z_{j,5}$ with cliques of four vertices $Z_{j,2}$ to $Z_{j,5}$ and two more edges: $(Z_{j,1}, Z_{j,2}), (Z_{j,1}, Z_{j,5})$. Furthermore, edge $(X_{i,j}, Y_{j,i})$ will be added to E if $u_j \in S_i$. Additionally, if $u_j \notin S_i$, edges $(W_{i,1}, X_{i,j})$ and $(Y_{j,i}, Z_{j,1})$ will be added to E. Clearly the reduction is in FPT. The candidate set, $\Gamma = \{(X_{i,1}, X_{i,2}) | \forall i = 1, 2, ..., m\}$. Fig. 5.5 illustrates our construction for sets $S_1 = \{u_1, u_2\}, S_2 = \{u_2, u_3\}$, and $S_3 = \{u_3, u_4\}$.

Initially all nodes are in the 3-core. We claim that a set $S' \subset S$, with $|S'| \leq b$, is a cover iff $f_3(B) = b(n+1) + n(m+1)$ where $B = \{(X_{i,1}, X_{i,2}) | S_i \in S'\}$. For any i, if $(X_{i,1}, X_{i,2})$ is removed, the n nodes $\{X_{i,1}, ..., X_{i,n}\}$ and node $W_{i,1}$ go to the 2-core. If $u_j \in S_i$, then $m+1$ nodes $\{Y_{j,1}, ..., X_{j,m}\}$ and $Z_{j,1}$ go to 2-core after $(X_{i,1}, X_{i,2})$ is removed. If S' is a set cover, all the u_js will be in some $S_i \in S'$ and $n(m+1)$ nodes will go into 2-core; so $f_3(B) = b(n+1) + n(m+1)$—any b edges from Γ would remove

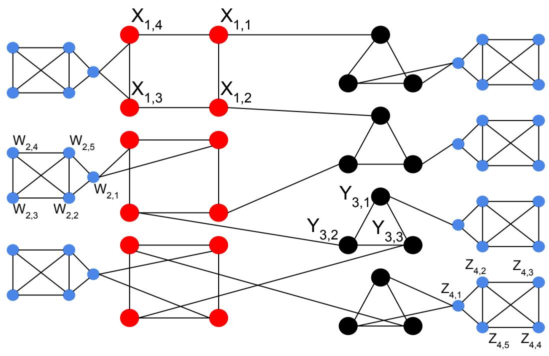


Figure 5.5: Example construction for parameterized hardness from Set Cover where $U = \{u_1, u_2, u_3, u_4\}, S = \{S_1, S_2, S_3\}, S_1 = \{u_1, u_2\}, S_2 = \{u_2, u_3\}, S_3 = \{u_3, u_4\}$.

$b(n+1)$ nodes. On the other hand, assume that $f_3(B) = b(n+1)+n(m+1)$ after removing edges in $B = \{(X_{i,1}, X_{i,2}) | S_i \in S'\}$. The only way to have $m+1$ nodes removed from corresponding u_j is if $u_j \in X_i$ and $X_i \in X$. Thus, $n(m+1)$ nodes will be removed, making S' a set cover. ■

Motivated by these strong hardness and inapproximability results, we next consider some practical heuristics for the KCM problem.

5.4 Algorithms

According to Theorems 26 and 27, an optimal solution or a constant-factor approximation for k-core minimization requires enumerating all possible size-b subsets from the candidate edge set, assuming $P \neq NP$. In this section, we propose efficient heuristics for the KCM problem.

5.4.1 Baseline: Greedy Cut

For KCM, only the current k-core of the graph, $\mathscr{G}(V_k, E_k) = C_k(G)$ ($|V_k| = N_k$,$|E_k| = M_k$), has to be taken into account. Remaining nodes will already be in a lower-than-k-core and can be removed. We define a vulnerable set $VS_k(e, \mathscr{G})$ as those nodes that would be demoted to a lower-than-k-core if edge e is deleted from the current core graph $\mathscr{G}$. Algorithm 5 (GC) is a greedy approach for selecting an edge set B ($|B| = b$) that maximizes the k-core reduction, $f_k(B)$. In each step, it chooses the edge that maximizes $|VS_k(e, \mathscr{G})|$ (step 3-4) among the candidate edges Γ. The specific procedure for computing $VS_k(e, \mathscr{G})$ (step 3), *LocalUpdate* and their running times ($O(M_k + N_k)$) are described below. The overall running time of GC is $O(b|\Gamma|(M_k + N_k))$.

Algorithm 6. This procedure computes the vulnerable set—i.e., the set of nodes that will leave the k-core upon deletion of the edge e from $\mathscr{G}$. The size of the set is essentially the marginal gain of deleting e. If $e = (u, v)$ is deleted, u will be removed iff $d(\mathscr{G}, u) = k$ (the same for v). This triggers a cascade of node removals from the k-core (with the associated edges). Let $vul(w)$ be the set of nodes already removed from $\mathscr{G}$ that are neighbours of node w. We observe that w will be removed if $d(\mathscr{G}, w) - |vul(w)| < k$. Note that the procedure is similar to Algorithm 2 (*LocalUpdate*), having $O(M_k + N_k)$ running time.

Local Update (Algorithm 7). After the removal of the edge e^* in each step, the current graph $\mathscr{G}$ is updated (step 9). Recomputing the k cores in $\mathscr{G}$ would take $O(M_k)$ time. Instead, a more efficient approach is to update only the affected region after deleting the e^*. If an edge $e^* = (u, v)$ is deleted, u will be removed if $d(\mathscr{G}, u) = k$ (the same for v). This triggers a cascade of node removals (with the associated edges). Let $vul(w)$ be a set of nodes already removed from $\mathscr{G}$ that are neighbours of node w. We observe that w will be removed if $d(\mathscr{G}, w) - |vul(w)| < k$.

5.4.2 Shapley Value Based Algorithm

The greedy algorithm discussed in the last section is unaware of some dependencies between the candidates in the solution set. For instance, in Figure 5.3a, all the edges

Algorithm 5 Greedy Cut (GC)

```
Require: $G, k, b$
Ensure: $B$: Set of edges to delete
1: $B \leftarrow \emptyset, max \leftarrow -\infty, \mathscr{G} \leftarrow C_k(G)$
2: while $|B| < b$ do
3:     $e^* \leftarrow \arg\max_{e \in \mathscr{G}(E_k) \setminus B} |computeVS(e = (u,v), \mathscr{G}, k)|$
4:     $B \leftarrow B \cup \{e^*\}$
5:     LocalUpdate($e, \mathscr{G}, k$)
6: end while
7: return $B$
```

have same importance (the value is 0) to destroy the 2-core structure. In this scenario, GC will choose an edge arbitrarily. However, removing an optimal set of seven edges can make the graph a tree (1-core). To capture these dependencies, we adopt a cooperative game theoretic concept named Shapley Value [141]. Our goal is to make a coalition of edges (players) and divide the total gain by this coalition equally among the edges inside it.

Shapley Value

The Shapley value of an edge e in the context of KCM is defined as follows. Let the value of a coalition P be $\mathscr{V}(P) = f_k(P) = N_k(G) - N_k(G^P)$. Given an edge $e \in \Gamma$ and a subset $P \subseteq \Gamma$ such that $e \notin P$, the marginal contribution of e to P is:

$$\mathscr{V}(P \cup \{e\}) - \mathscr{V}(P), \quad \forall P \subseteq \Gamma \tag{5.1}$$

Let Ω be the set of all $|\Gamma|!$ permutations of all the edges in Γ and $P_e(\pi)$ be the set of all the edges that appear before e in a permutation π. The Shapley value of e the average of its marginal contributions to the edge set that appears before e in all the permutations:

$$\Phi_e = \frac{1}{|\Gamma|!} \sum_{\pi \in \Omega} \mathscr{V}(P_e(\pi) \cup \{e\}) - \mathscr{V}(P_e(\pi)) \tag{5.2}$$

Algorithm 6 computeVS

```
Require: $e = (u, v), \mathscr{G}, k$
Ensure: $X$
 1: if $d(\mathscr{G}, u) = \mathcal{E}_{\mathscr{G}}(u)$ then
 2:     Queue $S \leftarrow S \cup \{u\}$, $X \leftarrow X \cup \{u\}$
 3: end if
 4: if $d(\mathscr{G}, v) = \mathcal{E}_{\mathscr{G}}(v)$ then
 5:     Queue $S \leftarrow S \cup \{v\}$, $X \leftarrow X \cup \{v\}$
 6: end if
 7: while $S \neq \emptyset$ do
 8:     Remove $y$ form $S$
 9:     for $w \in N(y)$ do
10:         $vul(w) \leftarrow \{z | z \in N(w) \cap X\}$
11:         if $w \notin X$ & $d(\mathscr{G}, w) - |vul(w)| < k$ then
12:             Add $w$ to $X$, $S$
13:         end if
14:     end for
15: end while
16: return $X$
```

Shapley values capture the importance of an edge inside a set (or coalition) of edges. However, computing Shapley value requires considering $O(|\Gamma|!)$ permutations. Next we show how to efficiently approximate the Shapley value for each edge via sampling.

Approximate Shapley Value Based Algorithm

Algorithm 8 (Shapley Value Based Cut, SV) selects the best b edges according to their approximate Shapley values based on a sampled set of permutations, S. For each permutation in S, we compute the marginal gains of all the edges. These marginal gains are normalized by the sample size, s. In terms of time complexity, steps 4-6 are the dominating steps and take $O(s|\Gamma|(N_k + M_k))$ time, where N_k and M_k are the number of nodes and edges in $C_k(G)$, respectively. Note that similar sampling based methods have been introduced for different applications [163, 164, 165].

Algorithm 7 LocalUpdate

```
Require: $e = (u, v), \mathscr{G}, k$
 1: Remove $(u, v)$, update $d(\mathscr{G}, u), d(\mathscr{G}, v)$, $X \leftarrow \Phi$, $Y \leftarrow \Phi$
 2: if $d(\mathscr{G}, u) < k$ then
 3:     Queue $Y \leftarrow Y \cup \{u\}$, $X \leftarrow X \cup \{u\}$
 4: end if
 5: if $d(\mathscr{G}, v) < k$ then
 6:     Queue $Y \leftarrow Y \cup \{v\}$, $X \leftarrow X \cup \{v\}$
 7: end if
 8: while $Y \neq \emptyset$ do
 9:     Remove $y$ form $Y$
10:     for $w \in N(y)$ do
11:         $vul(w) \leftarrow \{z | z \in N(w) \cap X\}$
12:         if $w \notin X$ & $d(\mathscr{G}, w) - |vul(w)| < k$ then
13:             Add $w$ to $X$, $S$
14:         end if
15:     end for
16:     if $d(\mathscr{G}, y) < k$ then
17:         Remove $y$ from $\mathscr{G}$
18:     end if
19: end while
```

Analysis

In the previous section, we presented a fast sampling algorithm (SV) for k-core minimization using Shapley values. Here, we study the quality of the approximation provided by SV as a function of the number of samples. We show that our algorithm is nearly optimal with respect to each Shapley value with high probability. More specifically, given $\varepsilon > 0$ and $\delta < 1$, SV takes $p(\frac{1}{\varepsilon}, \frac{1}{\delta})$ samples, where p is a polynomial in $\frac{1}{\varepsilon}, \frac{1}{\delta}$, to approximate the Shapley values within ε error with probability $1 - \delta$.

We sample uniformly with replacement, a set of permutations S ($|S| = s$) from the set of all permutations, Ω. Each permutation is chosen with probability $\frac{1}{|\Omega|}$. Let Φ'_e be the approximate Shapley value of e based on S. X_i is a random variable that denotes the marginal gain in the i-th sampled permutation. So, the estimated Shapley value is $\Phi'_e = \frac{1}{s}\sum_{i=1}^{s} X_i$. Note that $\mathbb{E}[\Phi'_e] = \Phi_e$.

Algorithm 8 Shapley Value Based Cut (SV)

```
Require: $G, k, b$
Ensure: $B$: Set of edges to delete
 1: Initialize all $\Phi'_e$ as 0, $\forall e \in \Gamma$
 2: Generate $S = O(\frac{\log \Gamma}{\varepsilon^2})$ random permutations of edges
 3: $B \leftarrow \emptyset, \mathscr{G} \leftarrow C_k(G)$
 4: for $\pi \in S$ do
 5:     for $e = (u, v) \in \Gamma$ do
 6:         $\Phi'_e \leftarrow \Phi'_e + (\mathscr{V}(P_e(\pi) \cup \{e\}) - \mathscr{V}(P_e(\pi)))$
 7:     end for
 8: end for
 9: $\Phi'_e \leftarrow \frac{\Phi'_e}{|S|}, \forall e \in \Gamma$
10: Select top $b$ $\Phi'_e$ edges from $B$
11: return $B$
```

Theorem 28 *Given ε ($0 < \varepsilon < 1$), a positive integer ℓ, and a sample of independent permutations $S, |S| = s$, where $s \geq \frac{(\ell+1)\log|\Gamma|}{2\varepsilon^2}$; then $\forall e \in \Gamma$:*

$$Pr(|\Phi'_e - \Phi_e| < \varepsilon \cdot N_k) \geq 1 - 2|\Gamma|^{-\ell}$$

where N_k denotes the number of nodes in $C_k(G)$.

Proof: We start by analyzing the Shapley value of one edge. Because the samples provide an unbiased estimate and are i.i.d., we can apply *Hoeffding's inequality* [64] to bound the error for edge e:

$$Pr[|\Phi'_e - \Phi_e| \geq \varepsilon \cdot Q_e] \leq \delta \tag{5.3}$$

where $\delta = 2\exp\left(-\frac{2s^2\varepsilon^2 Q_e^2}{\mathcal{R}}\right)$, $\mathcal{R} = \sum_{i=1}^{s}(b_i - a_i)^2$, and each X_i is strictly bounded by the intervals $[a_i, b_i]$. Let $Q_e = Max\{\mathscr{V}(P_e(\pi) \cup \{e\}) - \mathscr{V}(P_e(\pi)) | \pi \in \Omega\}$ be the maximum gain for e in any permutation. Then, $\mathcal{R} < sQ_e^2$, as for any X_i the minimum and maximum values are 0 and Q_e respectively. As a consequence:

$$\delta = 2\exp\left(-\frac{2s^2\varepsilon^2Q_e^2}{\mathcal{R}}\right) < 2\exp\left(-\frac{2s^2\varepsilon^2Q_e^2}{sQ_e^2}\right) = 2\exp\left(-2s\varepsilon^2\right)$$

Thus, the following holds for each edge e:

$$Pr[|\Phi'_e - \Phi_e| \geq \varepsilon \cdot Q_e] < 2\exp\left(-2s\varepsilon^2\right)$$

Using the above equation we compute a joint sample bound for all edges $e \in \Gamma$. Let $\Gamma = \{e_1, e_2, ..., e_{|\Gamma|}\}$ and E_i be the event that $|\Phi'_{e_i} - \Phi_{e_i}| \geq \varepsilon \cdot Q_{e_i}$. So, $Pr[E_i] = Pr[|\Phi'_{e_i} - \Phi_{e_i}| \geq \varepsilon \cdot Q_{e_i}] < 2\exp\left(-2s\varepsilon^2\right)$. Similarly, one can prove that $Pr[|\Phi'_{e_i} - \Phi_{e_i}| \geq \varepsilon \cdot N_k] \leq \delta'$, where $\delta' = 2\exp\left(-\frac{2s^2\varepsilon^2N_k^2}{\mathcal{R}}\right) < 2\exp\left(-2s\varepsilon^2\right)$, as $\mathcal{R} < sN_k^2$.

Applying union bound ($Pr(\cup_i E_i) \leq \sum_i Pr(E_i)$), for all edges in Γ, i.e., $\forall i \in \{1, 2, ...|\Gamma|\}$, we get that:

$$Pr[|\Phi'_{e_i} - \Phi_{e_i}| \geq \varepsilon \cdot N_k] < 2|\Gamma|\exp\left(-2s\varepsilon^2\right)$$

By choosing $s \geq \frac{(\ell+1)\log|\Gamma|}{2\varepsilon^2}$, $\forall i \in \{1, 2, ...|\Gamma|\}$,

$$Pr[|\Phi'_{e_i} - \Phi_{e_i}| \geq \varepsilon \cdot N_k] < \frac{2}{|\Gamma|^\ell}, \quad \text{or,}$$

$$Pr[|\Phi'_{e_i} - \Phi_{e_i}| < \varepsilon \cdot N_k) \geq 1 - 2|\Gamma|^{-\ell}$$

This ends the proof. ■

Next, we apply Theorem 28 to analyze the quality of a set B produced by Algorithm 8 (SV), compared with the result of an exact algorithm (without sampling). Let the exact Shapley values of top b edges be $\Phi_B^o = \{\Phi_{O1}, \Phi_{O2}, \Phi_{O3}, ..., \Phi_{Ob}\}$ where $\Phi_{O1} \geq \Phi_{O2} \geq ... \geq \Phi_{Ob}$. The set produced by Algorithm 8 (SV) has Shapley values, $\Phi_B^a = \{\Phi_{A1}, \Phi_{A2}, \Phi_{A3}, ..., \Phi_{Ab}\}$ where $\Phi_{A1} \geq \Phi_{A2} \geq ... \geq \Phi_{Ab}$. We can prove the following result regarding the SV algorithm.

Corollary 29 *For any* $i, \Phi_{Oi} \in \Phi_B^o$ *and* $\Phi_{Ai} \in \Phi_B^a$, ε $(0 < \varepsilon < 1)$, *positive integer* ℓ, *and a sample of independent permutations* $S, |S| = s$, *where* $s \geq \frac{(\ell+1)\log|\Gamma|}{2\varepsilon^2}$:

$$Pr(|\Phi_{Oi} - \Phi_{Ai}| < 2\varepsilon \cdot N_k) \geq 1 - 2|\Gamma|^{-\ell}$$

where N_k *denotes the number of nodes in* $C_k(G)$.

Proof: For all edges $e \in \Gamma$, Theorem 28 shows that $Pr(|\Phi'_e - \Phi_e| < \varepsilon \cdot N_k) \geq 1 - 2|\Gamma|^{-\ell}$. So, with probability $1-2|\Gamma|^{-\ell}$, $|\Phi'_{Oi} - \Phi_{Oi}| < \varepsilon \cdot N_k$ and $|\Phi'_{Ai} - \Phi_{Ai}| < \varepsilon \cdot N_k$. As $\Phi'_{Ai} > \Phi'_{Oi}$, $|\Phi_{Oi} - \Phi_{Ai}| < 2\varepsilon \cdot N_k$ with the same probability. ■

At this point, it is relevant to revisit the hardness of approximation result from Theorem 26 in the light of Corollary 29. First, SV does not directly minimize the KCM objective function (see Definition 11). Instead, it provides a score for each candidate edge e based on how different permutations of edges including e minimize the KCM objective under the assumption that such scores are divided fairly among the involved edges. Notice that such assumption is not part of the KCM problem, and thus Shapley values play the role of a heuristic. Corollary 29, which is a polynomial-time randomized approximation scheme (PRAS) type of guarantee instead of a constant-factor approximation, refers to the exact Shapley value of the top b edges, and not the KCM objective function. We evaluate how SV performs regarding the KCM objective in our experiments.

Generalizations

Sampling-based approximate Shapley values can also be applied to other relevant combinatorial problems on graphs for which the objective function is not submodular. Examples of these problems include k-core anchoring [89], influence minimization [34], and network design [7]).

5.4.3 Optimizations for GC and SV

Here, we discuss optimizations for the Greedy (GC) and Shapley Value based (SV) algorithms. We propose a general pruning technique to speed up both Algorithms 5 and 8 (GC and SV). For GC, in each step, all the candidate edges are evaluated (step 3). *How can we reduce the number of evaluations in a single step?* In SV, in a single permutation, marginal gains are computed for all the candidate edges (step 5). *How can we skip edges that have* 0 *marginal gain?*. We answer these questions by introducing the concept of *edge dominance*. Let $Z(e, \mathscr{G})$ be the set of vertices that would be removed if e is deleted from $\mathscr{G}$ due to the k-core constraint. If $e' = (u, v)$ has one of the end points u or v in $Z(e, \mathscr{G})$, then e' is dominated by e.

Observation 1 *If e' is dominated by e, then $Z(e', \mathscr{G}) \subseteq Z(e, \mathscr{G})$.*

In Algorithm 5 (GC), while evaluating each edge in the candidate set (step 3) if e' comes after e, we skip the evaluation of e', as $|X_e| \geq |X_{e'}|$ (Obs. 1). In Algorithm 8 (SV), while computing the marginal gain of each edge in a coalition for a particular permutation π, assume that e' appear after e. As $e \in P_{e'}(\pi)$ and using Observation 1, $\mathscr{V}(P_e(\pi) \cup \{e\}) - \mathscr{V}(P_e(\pi))) = 0$. Thus, the computation of the marginal gain of e' can be skipped. We evaluate the performance gains due to pruning in our experimental evaluation.

5.5 Experiments

In this section, we evaluate the proposed Shapley Value Based Cut (SV) algorithm for k-core minimization against baseline solutions. Sections 5.5.2 and 5.5.3 are focused on the quality results (k-core minimization) and the running time of the algorithms,

Dataset Name	$\|V\|$	$\|E\|$	k_{max}	Type
Yeast	1K	2.6K	6	Biological
Human	3.6K	8.9K	8	Biological
email-Enron (EE)	36K	183K	42	Email
Facebook (FB)	60K	1.5M	52	OSN
web-Stanford (WS)	280K	2.3M	70	Webgaph
DBLP (DB)	317K	1M	113	Co-authorship
com-Amazon (CA)	335K	926K	6	Co-purchasing
Erdos-Renyi (ER)	60K	800K	19	Synthetic

Table 5.2: Dataset descriptions and statistics. The value of k_{max} (or degeneracy) is the largest k among all the values of k for which there is a k-core in the graph.

respectively. Moreover, in Section 5.5.4, we show how k-core minimization can be applied in the analysis of the structural resilience of networks.

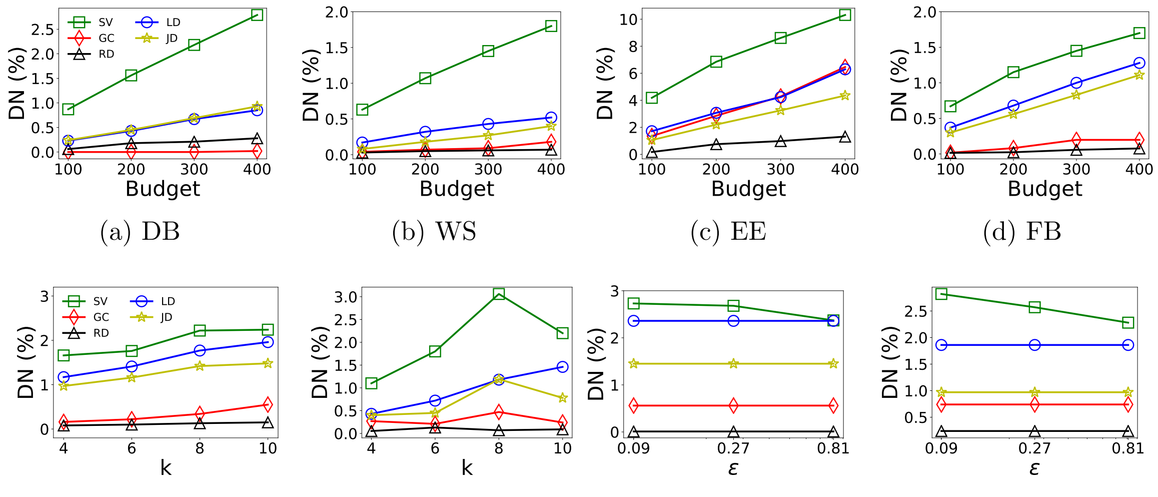
5.5.1 Experimental Setup

All the experiments were conducted on a 2.59 GHz Intel Core i7-4720HQ machine with 16 GB RAM running Windows 10. Algorithms were implemented in Java.

Datasets: The real datasets used in our experiments are available online and are mostly from SNAP[1]. The Human and Yeast datasets are available in [166]. In these datasets the nodes and the edges correspond to genes and interactions (protein- protein and genetic interactions) respectively. The Facebook dataset is from [167]. Table 5.2 shows dataset statistics, including the largest k-core (a.k.a. degeneracy). These are undirected and unweighted graphs from various applications: EE is from email communication; FB is an online social network, WS is a Web graph, DB is a collaboration network and CA is a product co-purchasing network. We also apply a random graph (ER) generated using the Erdos-Renyi model.

Algorithms: Our algorithm, *Shapley Value Based Cut (SV)* is described in Section 5.4.2. Besides the Greedy Cut (GC) algorithm [131] (Section 5.4.1), we also consider

[1]`https://snap.stanford.edu`



(a) DB (b) WS (c) EE (d) FB

(e) FB (f) WS (g) FB (h) WS

Figure 5.6: K-core minimization (DN(%)) for different algorithms varying (a-d) the number of edges in the budget; (e-f) the core parameter k; (g-h) and the sampling error ε. The Shapley Value based Cut (SV) algorithm outperforms the best baseline (LD) by up to 6 times. On the other hand, the Greedy approach (GC) achieves worse results than the baselines, with the exception of RD, in most of the settings. SV error increases smoothly with ε and LD becomes a good alternative for large values of k.

three more baselines in our experiments. *Low Jaccard Coefficient (JD)* removes the k edges with lowest Jaccard coefficient. Similarly, *Low-Degree (LD)* deletes k edges for which adjacent vertices have the lowest degree. We also apply *Random (RD)*, which simply deletes k edges from the candidate set Γ uniformly at random. Notice that while LD and JD are quite simple approaches for KCM, they often outperform GC.

Quality evaluation metric: We apply the percentage $DN(\%)$ of vertices from the initial graph G that leave the k-core after the deletion of a set of edges B (produced by a KCM algorithm):

$$DN(\%) = \frac{N_k(G) - N_k(G^B)}{N_k(G)} \times 100 \tag{5.4}$$

Default parameters: We set the candidate edge set Γ to those edges ($M_k(G)$) between vertices in the k-core $C_k(G)$. Unless stated otherwise, the value of the approximation parameter for SV (ε) is 0.05 and the number samples applied is $\frac{\log|\Gamma|}{\varepsilon^2}$ (see

	Human		**Yeast**	
	$b = 5$	$b = 10$	$b = 5$	$b = 10$
OPT	2.88	3.24	11.16	12.05
SV ($\varepsilon = .1$)	2.88	3.06	10.27	11.16
SV ($\varepsilon = .2$)	2.8	3.06	8.48	10.71

Table 5.3: SV (approximate) vs. optimal algorithm using two datasets and a small candidate set size ($|\Gamma| = 50$), and $k = 5$.

Theorem 28).

5.5.2 Quality Evaluation

KCM algorithms are compared in terms of quality (DN(%)) for varying budget (b), core value k, and the error of the sampling scheme applied by the SV algorithm (ε).

Varying budget (b): Figure 5.6 presents the k-core minimization results for $k = 5$—similar results were found for $k = 10$—using four different datasets. SV outperforms the best baseline by up to six times. This is due to the fact that our algorithm can capture strong dependencies among sets of edges that are effective at breaking the k-core structure. On the other hand, GC, which takes into account only marginal gains for individual edges, achieves worse results than simple baselines such as JD and LD. We also compare SV and the optimal algorithm in small graphs and show that SV produces near-optimal results.

SV and the optimal algorithm: In these experiments, we evaluate the approximation achieved by SV (Algorithm 8) compared to the optimal results using two small networks (Human and Yeast). The optimal set of $b = 5$ and $b = 10$ edges among a randomly chosen a set of 50 edges is selected as the candidate set Γ inside the k-core. An optimal solution is computed based on all possible sets with size b in Γ. Table 5.3 shows the $DN(\%)$ produced by the optimal solution (OPT) and SV. Notice that the SV algorithm produces near-optimal results.

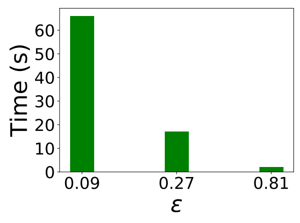
Varying core value (k): We evaluate the impact of k over quality for the algorithms using two datasets (FB and WS) in Figures 5.6e and 5.6f. The budget (b) is set to 400. As in the previous experiments, SV outperforms the competing approaches. However, notice that the gap between LD (the best baseline) and SV decreases as k increases. This is due to the fact that the number of samples decreases for higher k as the number of candidate edge also decreases, but it can be mended by a smaller ε. Also, a larger k will increase the level of dependency between candidate edges, which in turn makes it harder to isolate the impact of a single edge—e.g. independent edges are the easiest to evaluate. On the other hand, a large value of k leads to a less stable k-core structure that can often be broken by the removal of edges with low-degree endpoints. LD is a good alternative for such extreme scenarios. Similar results were found for other datasets.

Varying the sampling error (ε): The parameter ε controls the the sampling error of the SV algorithm according to Theorem 28. We show the effect of ε over the quality results for FB and WS in Figures 5.6g and 5.6h. The values of b and k are set to 400 and 12 respectively. The performance of the competing algorithms do not depend on such parameter and thus remain constant. As expected, DN(%) is inversely proportional to the value of ε for SV. The trade-off between ε and the running time of our algorithm enables both accurate and efficient selection of edges for k-core minimization.

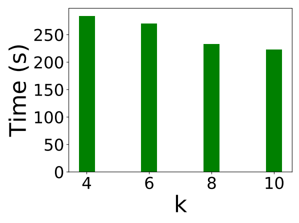
5.5.3 Running Time

Here, we evaluate the running time of the GC and SV algorithms. In particular, we are interested in measuring the performance gains due to the pruning strategies described above. LD and JD do not achieve good quality results in general, as discussed in the previous section, thus we omit them from this evaluation.

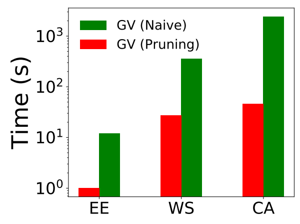
Running times for SV varying the sampling error (ε) and the core parameter (k) using



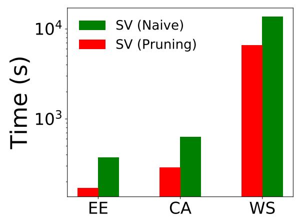
(a) Varying ε



(b) Varying k



(c) Pruning, GC

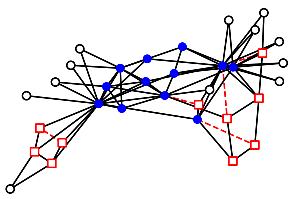


(d) Pruning, SV

Figure 5.7: Running times by SV using FB while varying (a) the sampling error ε and (b) the core parameter k; and (c-d) impact of pruning for GC and SV algorithms using three datasets. SV is efficient even for small values of sampling error and its running time decreases with k. GC is up to one order of magnitude faster with pruning, while SV is up to 50% faster.

the FB dataset are given in Figures 5.7a and 5.7b, respectively. Even for small error, the algorithm is able to process graphs with tens of thousands of vertices and millions of edges in, roughly, one minute. Running times decay as k increases due to two factors: (1) the size of the k-core structure decreases (2) pruning gets boosted by a less stable core structure.

In Figures 5.7c and 5.7d, we look further into the effect of pruning for GC and SV by comparing versions of the algorithms with and without pruning using three datasets. GC becomes one order of magnitude faster using pruning. Gains for SV are lower but still significant (up to 50%). We found in other experiments that the impact of pruning for SV increases with the budget, which is due to the larger number of permutations to be considered by the algorithm.


(a) $b = 5$

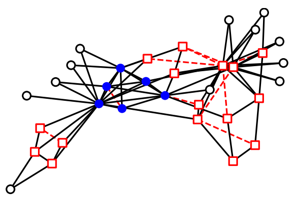
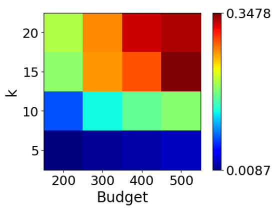
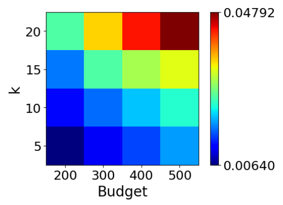

(b) $b = 10$

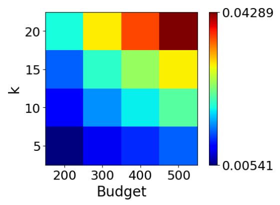
Figure 5.8: K-core ($k = 3$) minimization on the Newman's Karate network: (a) $b = 5$ and (b) $b = 10$. Unfilled circle nodes are not in the 3-core of the original network. After removal of b dashed (red) edges, filled (blue) circle nodes remain in the 3-core and unfilled (red) square nodes are removed from the 3-core.



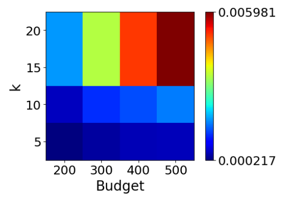
(a) DB



(b) WS



(c) FB



(d) ER

Figure 5.9: Core resilience for four different networks: (a) DB (co-authorship), (b) WS (Webgraph), (c) FB (social), (d) ER (random). ER and DB are the most and least stable networks, respectively. Tipping points are found for ER and DB.

5.5.4 Application: k-core Resilience

We show how KCM can be applied to profile the resilience or stability of real networks. A profile provides a visualization of the resilience of the k-core structure of a network for different combinations of k and budget. We apply $DN(\%)$ (Equation 5.4) as a measure of the percentage of the k-core removed by a certain amount of budget—relative to the immediately smaller budget value.

Figure 5.9 shows the results for co-authorship (DB), Web (WS), social network (FB) and a random (ER) graph. We also discuss profiles for Human and Yeast. Each cell corresponds to a given k-b combination and the color of cell (X, Y) shows the difference in $DN(\%)$ between $b{=}Y$ and $b{=}Y{-}100$ for $k{=}X$. As colors are relative, we also show the range of values associated to the the color scheme. We summarize our main findings as follows:

Stability: ER (Figure 5.9d) is the most stable graph, as can be noticed by the range of values in the profile. The majority of nodes in ER are in the 19-core. DB (Figure 5.9a) is the least stable, but only when $k > 5$, which is due to its large number of small cliques. The high-core structure of DB is quite unstable, with less than 1% of the network in the 20-core structure after the removal of 500 edges.

Tipping points: We also look at large effects of edge removals within small variations in budget—for a fixed value of k. Such a behavior is not noticed for FB and WS (Figures 5.9b and 5.9c, respectively), for which profiles are quite smooth. This is mostly due to the presence of fringe nodes at different levels of k-core structure. On the other hand, ER produced the most prominent tipping points ($k = 15$ and $k = 20$). This pattern is also found for DB.

5.5.5 K-core Minimization on the Karate Network

In Figure 5.8, we demonstrate the application of our algorithm for KCM using the popular Newman's Karate network with two different budget settings, $b = 5$ and $b = 10$, and k fixed to 3. Unfilled circles are nodes initially out of the 3-core. The dashed (red) edges are removed by our algorithm—often connecting fringe nodes. Filled (blue) circles and unfilled (red) squares represent nodes that remain and are removed from the 3-core, respectively, after edge removals.

K-core Resilience: Human vs Yeast

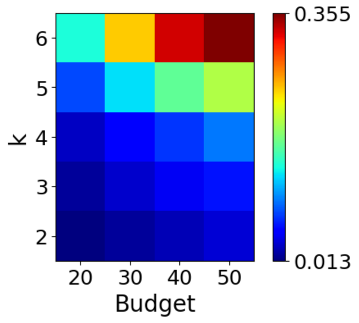
K-cores have been previously applied in the analysis of functional modules in protein-protein networks [129, 168]. Here, we compare the k-core stability of Human and Yeast (Figs. 5.10a, 5.10b). Human is shown to be more stable, as can be inferred from the range of values in the profile—1% to 35% for Human and 3.4% to 100% for Yeast.

Moreover, the profile for Human is smoother than Yeast. These results confirm our intuition that proteins have a more complex functional structure in Humans compared to other organisms [169]. We also show similar results for clustering coefficient and efficiency, which are other popular robustness measures for networks [1], within the same core set of vertices to facilitate the comparison. Both competing metrics fail to effectively assess robustness for varying values of k and budget. In particular, the clustering coefficient of the networks remain mostly unchanged after edge deletions. The effect of network efficiency minimization over the core of the network does not necessarily increase with the budget, which is counter-intuitive. More specifically, efficiency minimization often fails to break dense substructures of the network, even for large values of budget.

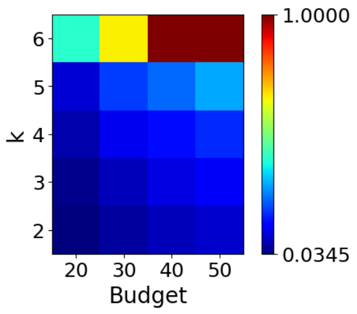
5.6 Conclusion

We have studied the k-core minimization (KCM) problem, which consists of finding a set of edges, removal of which minimizes the size of the k-core structure. KCM was shown to be NP-hard, even to approximate within any constant when $k \geq 3$. The problem is also not fixed-parameter tractable, meaning it cannot be solved efficiently even if the number of edges deleted is small. Given such inapproximability results, we have proposed an efficient randomized heuristic based on Shapley value to account for the interdependence in the impact of candidate edges. For the sake of comparison, we also evaluate a simpler greedy baseline, which cannot assess such strong dependencies in the effects of edge deletions.

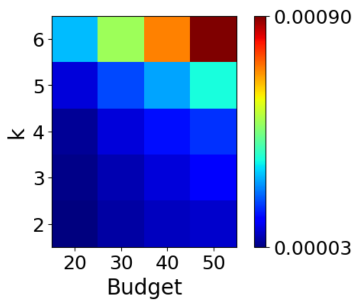
We have evaluated the algorithms using several real graphs and shown that our Shapley value based approach outperforms competing solutions in terms of quality. The proposed algorithm is also efficient, enabling its application to graphs with hundreds of thousands of vertices and millions of edges in time in the order of minutes using a desktop PC. We have also illustrated how KCM can be used for profiling the resilience of networks to edge deletions.



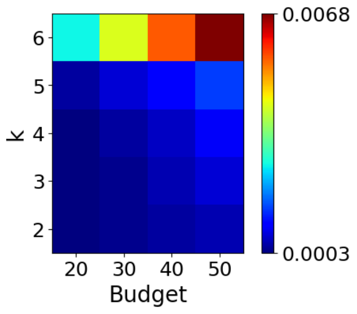
(a) Human (Core resilience)



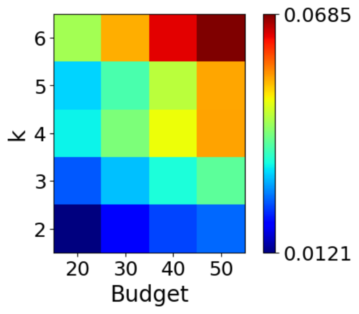
(b) Yeast (Core resilience)



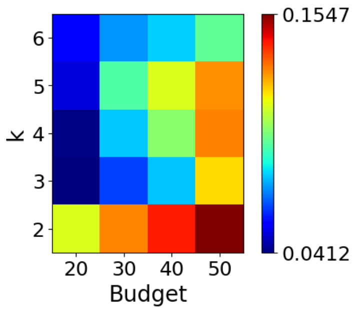
(c) Human (Clustering coefficient)



(d) Yeast (Clustering coefficient)



(e) Human (Efficiency)



(f) Yeast (Efficiency)

Figure 5.10: Core resilience (a, b) and other robustness metrics, clustering coefficient (c, d) and efficiency (e, f) [1], for the Human and Yeast protein-protein interaction networks.

Chapter 6

Controlling Influence

Online social networks have become major battlegrounds for political campaigns, viral marketing, and the dissemination of news. As a consequence, "bad actors" are increasingly exploiting these platforms, becoming a key challenge for their administrators, businesses and society in general. The spread of fake news is a classical example of the abuse of social networks by these actors. While some have advocated for stricter policies to control the spread of misinformation in social networks, this often happens in detriment of their democratic and organic structure.

In this chapter, we study how to limit the influence [15] of a target group in a social network via the removal of a few users/links. The idea is to control the diffusion processes while minimizing the amount of disturbance in the network structure. We formulate the influence limitation problem in a data-driven fashion, by taking into account past propagation traces. Moreover, we consider two types of constraints over edge removals, a budget constraint and also a, more general, set of matroid constraints. These problems lead to interesting challenges in terms of algorithm design. For instance, we are able to show that influence limitation is APX-hard and propose deterministic and probabilistic approximation algorithms for the budgeted and matroid version of the problem, respectively. Experiments show that the proposed approaches outperform several baselines.

6.1 Introduction

Online social networks, such as Facebook and Twitter, were popularized mostly as platforms for sharing entertaining content and maintaining friendship. However, they have been quickly transformed into major battlegrounds for political campaigns, viral marketing, and the dissemination of news. With this shift, the increase in the number of “bad actors”, such as tyrannical governments, spammers, and bullies exploiting these platforms has become a key challenge for their administrators, businesses and society.

A questionable approach to control the diffusion of misinformation in social platforms is via stricter laws and regulations by governments. This control often happens in detriment of the democratic and organic structure that are central to these platforms. Instead, a more sensible approach is to limit the impact of bad actors in the network while minimizing the disruption of its structure.

In this chapter we formalize *the influence limitation problem*. In particular, we focus on a setting where the network is modified via the removal (or blocking) of a few edges or nodes. These modifications can be implemented by social network administrators or induced by other organizations or governments via advertising campaigns. Although we focus on influence limitation, our problem is also relevant from the perspective of an agent that aims to maintain the influence of a set of users. Nodes/edges discovered by our algorithm are those that should be protected by such an agent. Similarly, while we focus on the edge version of our problem, the techniques discussed here also apply to the node version.

The problem of controlling or maintaining influence spread via structural changes in a network has attracted recent interest from the research community [9, 5, 4]. However, existing work assumes that the diffusion process follows classical models—e.g., Independent Cascade (IC), Linear Threshold (LT), and Susceptible-Infected-Recovered (SIR)—that require computationally-intensive simulations for validation. Instead, we propose a data-driven approach for influence minimization based on historical data [170].

Influence limitation (minimization) problems are often studied under budget constraints [9, 5], where a fixed number of nodes or edges can be blocked in the network. However, budget constraints have undesired effects in many settings. For instance, they might disconnect or disproportionately affect particular sub-networks. Such effects are in conflict with important modern issues in algorithm design, such as fairness [171]. We address these challenges by studying the influence limitation problem not only under a budget constraint but also under a more general set of matroid constraints [60, 172].

This work demonstrates how the formalization of the influence limitation problem under budget and matroid constraints leads to interesting challenges in terms of algorithm design. Different from the budget version, for which we propose a simple greedy algorithm, the matroid version requires a more sophisticated solution via continuous relaxation and rounding. Yet, we provide a theoretical analysis of the performance of both algorithms that is supported by the fact that the objective function of the influence limitation problem is submodular. Moreover, we provide strong inapproximability results for both versions of the problem.

Our contributions are summarized as follows:

- We investigate the novel data-driven influence limitation problem via node/edge removals. We show that the edge version is more general and covers the node version of the problem.

- We study our problem under both budget and matroid constraints, discussing how these affect algorithmic design.

- We show that the influence limitation problem is APX-hard and propose constant-factor approximations for both versions of the problem—deterministic and probabilistic aproximation for the budget and matroid version, respectively.
- We show that our methods outperform baseline solutions by up to 35% while scaling to large graphs.

6.2 Related Work

The influence boosting or limitation problems via network modifications are orthogonal to the classical influence maximization task [31]. In these modification problems, the objective is to optimize (maximize or minimize) the content spread via structural or attribute-level change in the network.

Previous work addressed the influence limitation problem in the SIR model [9, 32, 33]. The objective is to optimize specific network properties in order to boost or contain the content/virus spread. For instance, Tong et al. proposed methods to add (delete) edges to maximize (minimize) the eigenvalue of the adjacency matrix.

The influence spread optimization problem also has been studied under the IC model via network design [34, 35, 36, 6, 37] and injecting an opposite campaign [38, 39]. Bogunovic [35] addressed the minimization problem via node deletion. On the other hand, Sheldon et al. [36] studied the node addition problem and proposed expensive algorithms based on mixed integer programming. Kimura et al. [34] proposed greedy algorithms for the same. While Chaoji et al. [6] studied the problem of boosting the content spread via edge addition, Lin et al. [37] investigated the same via influencing initially uninfluenced users.

Boosting and controlling the influence via edge addition and deletion, respectively, were also studied under the Linear Threshold (LT) model by Khalil et al. [4]. They showed the supermodular property for the objective functions and then applied known approximation guarantees. The influence minimization problem was also studied under a few variants of LT model. [5, 40]. Kuhlman et al. [5] solved the minimization problem via edge removal heuristics in a simpler version of LT. In [40, 41], the authors explored influence blocking maximization problem in a variant of LT model via node deletion and also showed supermodular property.

In summary, the approaches for optimizing influence (propagation) are mostly based on the well-known diffusion models such as SIR, LT and IC. However, our work addresses the influence minimization problem based on available cascade information.

Optimization over matroids: Matroids have been quite popular for modelling combinatorial problems [60, 172]. Nemhauser [60] introduced a few optimization problems under matroids. Vondrak [173] addressed matroid optimization with a continuous greedy technique for submodular functions. Calinescu et al. [174] and Chekuri et al. [175] proposed rounding techniques for continuous relaxation of submodular functions under matroids.

6.3 Influence Limitation

We start with a description of Credit Distribution Model and formulate the influence limitation problems in Section 6.3.2.

6.3.1 Credit Distribution Model

The Credit Distribution Model (CDM) [170] estimates user influence directly from propagation traces. Its main advantages compared to classical influence models (e.g.

Independent Cascade and Linear Threshold [31, 176]) is that it does not depend on computationally intensive simulations while also relying less on the strong assumptions made by such models.

Let $G(V, E)$ and $\mathscr{L}(User, Action, Time)$ be a directed social graph and an action log respectively. A tuple (u, a, t) in action log $\mathscr{L}$ indicates that user u has performed action a at time t. Action $a \in \mathscr{A}$ propagates from node u to node v iff u and v are linked in social graph and u performed action a before v. This process defines a *propagation (action) graph* of a as a directed acyclic graph (DAG) $G(a) = (V(a), E(a))$. The action log $\mathscr{L}$ is thus a set of DAGs representing different actions' propagation traces. For a particular action a, a potential influencer of a node or user can be any of its in-neighbours. We denote $N_{in}(u, a) = \{v|(v, u) \in E(a)\}$ as the set of potential influencers of u for action a and $d_{in}(u, a) = |N_{in}(u, a)|$. When a user u performs action a, the *direct influence credit*, denoted by $\gamma_{(v,u)}(a)$, is given to all $v \in N_{in}(u, a)$. Intuitively the CDM distributes the influence credit backwards in the propagation graph $G(a)$ such that not only u gives credit to neighbours, but also in turn the neighbours pass on the credit to their predecessors. The total credit, $\Gamma_{v,u}(a)$ given to a user v for influencing u via action a from v to u in the propagation graph $G(a)$ is:

$$\Gamma_{v,u}(a) = \sum_{w \in N_{in}(u,a)} \Gamma_{v,w}(a).\gamma_{(w,u)}(a) \tag{6.1}$$

Similarly, one can define the credit for a set of nodes X,

$$\Gamma_{X,u}(a) = \begin{cases} 1 & if\ u \in X \\ \sum_{w \in N_{in}(u,a)} \Gamma_{X,w}(a).\gamma_{(w,u)}(a) & otherwise \end{cases}$$

Symbols	**Definitions and Descriptions**
$G(V, E)$	Given graph (vertex set V and edge set E)
X	Target set of source nodes
C	The set of candidate edges
k	Budget for BIL
$G(a) = (V(a), E(a))$	Action/propagation graph (DAG) for action a
$\Gamma_{v,u}(a)$	Credit of node v for influencing u in $G(a)$
$\Gamma_{X,u}(a)$	Credit given to set X for influencing u in $G(a)$
$\gamma_e(a) = \gamma_{(v,u)}(a)$	Direct credit for v to influence u via $e = (v, u)$
$u \overrightarrow{a} v$	It implies there is a path from u to v in $G(a)$
b	Maximum #edges removed from a node in ILM
$\vec{y}$	The vector with edge membership probabilities

Table 6.1: Frequently used symbols

The influence $\sigma_{cd}(X)$ is the credit given to X by all vertices:

$$\sigma_{cd}(G, X) = \sum_{u \in V} \frac{1}{|\mathscr{A}_u|} \sum_{a \in \mathscr{A}_u} \Gamma_{X,u}(a) \tag{6.2}$$

Influence probabilities $\gamma_{(u',v')}$ are computed using well-known techniques [177]. However, our theoretical results do not depend on how γ is computed. We study influence minimization in two settings: The first is budget constrained optimization, where a limit on the number of edges to be modified is set as a parameter. The second setting takes into account a more general class of constraints that can be expressed using the notion of an independent set.

6.3.2 Problem Definitions

Our goal is to remove a few edges $B \subset E$ such that the influence of a target set of users X is minimized according to the CDM. The credit of target user v for influencing user u in $G(a)$ is computed based on Equation 6.1. Consider $P(v, u)$ to be the set of paths from v to u where each path $p = \{e_1, e_2, ..., e_t\}$ is such that $e_1 = (v, v')$, $e_t = (u', u)$, and $e_i \in E(a)$ for all i and $u', v' \in V(a) - \{v, u\}$. We use $\gamma_{(w',w)}(a)$ or $\gamma_e(a)$ to represent the credit exclusively via edge $e = (w', w)$ for influencing w in $G(a)$. Therefore, Equation 6.1 can be written as:

$$\Gamma_{v,u}(a) = \sum_{p \in P(v,u)} \prod_{e \in p} \gamma_e(a) \tag{6.3}$$

A similar expression can be obtained for a target set X:

$$\Gamma_{X,u}(a) = \sum_{p \in P(X,u)} \prod_{e \in p} \gamma_e(a) \tag{6.4}$$

where $P(X, u)$ contains only the minimal paths from $v \in X$ to u—i.e. $\nexists p_i, p_j \in P(X, u)$ such that $p_i \subseteq p_j$.

We apply Equation 6.4 to quantify the change in credit for a target set of nodes X and a particular action a after the removal of edge e according to the credit distribution model:

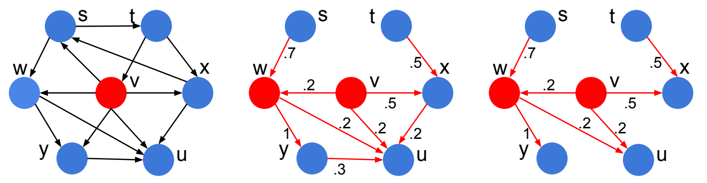
$$\delta_a(\{e\}) = \sum_{w \in V} \Big(\Gamma_{X,w}(a) - \sum_{\substack{p \in P(X,w) \\ e \notin p}} \prod_{e' \in p} \gamma_{e'}(a) \Big) \tag{6.5}$$

An edge deletion potentially blocks a few paths from v, reducing its credit (or influence). We use $G^m = (V, E - B)$ and $G^m(a)$ to denote the graph and the propagation graph for action a after the removal of edges in B, respectively. The following sections introduce the budget and matroid constrained versions of the influence limitation problem.

Budgeted Influence Limitation (BIL)

We start formalizing the budgeted version of our problem:

Problem 4 ***Budgeted Influence Limitation (BIL):*** *Given a directed graph $G(V, E)$, an action log $\mathscr{L}$, a candidate set of edges C, a given seed set X, and an integer $k < |C|$,*



(a) Social Graph, G (b) $G(a)$ (c) Modified: $G^m(a)$

Figure 6.1: Illustrative example of a social graph and CDM with the corresponding credits over the edges.

find a set $B \subset C \subset E$ of k edges such that $\sigma_{cd}(G^m, X)$ is minimized or, $\Delta(B) = \sigma_{cd}(G, X) - \sigma_{cd}(G^m, X)$ is maximized where $G^m = (V, E \setminus B)$.

Example 5 *In Figure 6.1, we assume the candidate set $C = \{(t, x), (y, u), (x, u)\}$, $k = 2$, and $X = \{w, v\}$. In $G(a)$ (Fig. 6.1b), $\sigma_{cd}(G, X) = 4.21$ and the deletion of $(t, x) \in C$ will not change the influence of X. The removal of (y, u) and (x, u) (Fig. 6.1c) will make $\sigma_{cd}(G^m(a), X) = \Gamma_{X,v} + \Gamma_{X,w} + \Gamma_{X,x} + \Gamma_{X,u} + \Gamma_{X,y} = 1 + 1 + 0.5 + (0.2 + 0.2) + 1 = 3.9$.*

Theorem 30 *The BIL problem is NP-hard.*

Proof: We prove the hardness result by reducing the known *Influence Maximization* (IM) problem [170] under the credit distribution model (CDM) to our problem, Budgeted Influence Limitation (BIL). Consider a problem instance I_{IM} [170], where graph $G = (V, E)$, $|V| = n, |E| = m$ and integer k are given. We create a corresponding BIL problem instance (I_{BIL}) as follows. The directed social graph is $G' = (V', E')$ where $V' = V \cup \{x\}$, x is an additional node. Let $C = \{(x, v) | v \in V\}$. In I_{BIL}, $E' = E \cup C$. We assume that the edges in C are present for every action in IM. C is also candidate set of edges. Let us assume the set S (of size k) has the maximum influence (σ^*). Now, it is easy to see that the maximum reduction of the influence of node x in BIL can be obtained if and only if the edges (k edges) between x and S are removed. ■

Node Version: The BIL problem setting generalizes the node version of the influence minimization problem. The construction is as follows: In the node version, a candidate node u is divided into two nodes u_{in} and u_{out} associated with the incoming and outgoing edges of u respectively. A directed edge from u_{in} to u_{out} will be added to the candidate edge set in BIL for the corresponding candidate node u in the node version.

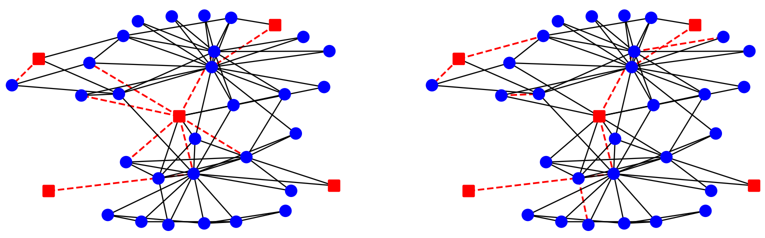
BIL assumes that any k edges in the candidate set can be removed. As a consequence, an optimal solution for BIL might make the network disconnected or disproportionately affect particular portions of the network. Fig. 6.2 exemplifies this issue using the Newman's karate[1] network. BIL modifications are strongly biased towards a small set of nodes. Next we present a different formulation for influence limitation that addresses some of these challenges.

Influence Limitation under Matroid (ILM)

Matroids are abstract objects that generalize the notion of linear independence to sets [175]. We apply matroids to characterize a class of constraints for influence limitation. To illustrate the expressive power of matroids as a general class of constraints for optimization problems defined over networks, we focus on a particular setting of influence minimization. More specifically, we upper bound the number of incoming edges that can be removed (or blocked) from each node in the network.

Problem 5 ***Influence Limitation under Matroid (ILM):***
Given a directed social graph $G(V, E)$, an action log $\mathscr{L}$, a candidate set of edges C, a given seed set X, and an integer b, find a set B (where $B \subset C \subset E$) such that at most b edges from B are incident (incoming) on any node in V and $\sigma_{cd}(G^m, X)$ is minimized where $G^m = (V, E - B)$ or, $\Delta(B) = \sigma_{cd}(G, X) - \sigma_{cd}(G^m, X)$ is maximized.

[1]http://www-personal.umich.edu/ mejn/netdata/



(a) BIL (b) ILM

Figure 6.2: We perform our methods for BIL (a) and ILM (b) on the Newman's Karate network with $|X| = 5, k = 9$. Square (red) nodes are in the target set, $|X|$, and dotted (red) edges are in the solution set. The edges are incident to few nodes in the solution for BIL, being strongly biased towards a small set of nodes. For ILM, we have considered $b = 2$, which leads to a solution with more uniform set of edges.

The effect of ILM enforces network modifications that are more uniformly distributed across the network. Notice that a valid solution for the budget constrained version (BIL) might not necessarily be a valid solution for ILM and vice-versa.

Theorem 31 *The ILM problem is NP-hard.*

It remains to show that ILM follows a matroid—any valid solution is a matroid. We will show that ILM follows a *partition matroid*, which is a matroid where the ground set C is partitioned into non-overlapping subsets $C_1, C_2, \cdots, C_l$ with associated integers $b_1, b_2, \cdots, b_l$ such that a set B is independent iff $|B \cap C_i| \leq b_i$.

Observation 2 *ILM follows a partition matroid.*

The key insight for this observation is that, for any incoming edge, the associated node is unique to the edge. As an example, if $e = (u, v)$ then the node v is unique to the edge e. Thus, the ground set C can be partitioned into edge sets $(C_1, C_2, ..., C_{|V|})$ based on the $|V|$ unique incidence edges associated with them. Any feasible solution B (edge set) is an independent set as $B \cap C_v \leq b$, where $v \in V$. Notice that the more general setting where a constant b_v is defined for each node in the network is also a partition matroid.

The BIL and ILM problems are APX-hard and cannot be approximated a factor greater than $(1 - \frac{1}{e})$. We prove these results in Section 6.5 using the notion of *curvature* [178].

6.4 Submodularity

A key feature in the design of efficient algorithms for influence limitation is *submodularity.* Intuitively, submodular functions are defined over sets and have the so called *diminishing returns property.* Besides its more usual application to the budgeted version of our problem, we also demonstrate the power of submodular optimization in the solution of its matroid constrained version.

To prove that the maximization function Δ associated to BIL and ILM is submodular, we analyze the effect of the removal of a candidate edge e over the credit of the target set X. Lemma 4 defines the change in credit $(\delta_a(\{e\}))$ in $G(a)$. The notation $u\overrightarrow{a}v$ denotes that there is a path from u to v in $G(a)$.

Lemma 4 *For an action a, with corresponding DAG $V(a)$, the change in credit after the removal of $e = (u, v)$ is as follows: $\delta_a(\{e\}) = \big(\Gamma_{X,u}(a) \cdot \gamma_{(u,v)}(a)\big) \cdot \sum_{w \in V} \Gamma_{v,w}(a)$.*

Proof: If w is not reachable from v, the proof becomes trivial. For $v\overrightarrow{a}w$ we use induction on length l. Let the set of reachable nodes via a path length of l from v in $G(a)$ be $R^a(v, l)$. We denote $N_{out}(u, a) = \{v|(u, v) \in E(a)\}$ and the decrease in credit contribution via the removal of the edge e by any arbitrary node w in $R^a(v, l)$ as $\delta_a^{l,w}(\{e\})$ and by all nodes in $R^a(v, l)$ as $\delta_a^l(\{e\})$.

Base case: when $l = 0$, $\sum_{w \in V} \Gamma_{v,w}(a, 0) = \Gamma_{v,v} = 1$. So, the statement is true for $l = 0$.

Induction step: Assume that the statement is true when restricted to path lengths l, for any arbitrary node w where $w \in R^a(v, l)$, i.e., $\delta_a^{l,w}(\{e\}) = \big(\Gamma_{X,u}(a).\gamma_{(u,v)}(a)\big).\Gamma_{v,w}(a, l)$

Notice that,

$$\begin{aligned}\delta_a^l(\{e\}) &= \big(\Gamma_{X,u}(a).\gamma_{(u,v)}(a)\big). \sum_{w \in R^a(v,l)} \Gamma_{v,w}(a)\\ &= \sum_{w \in R^a(v,l)} \delta_a^{l,w}(\{e\})\end{aligned}$$

We will prove that the statement remains true for paths of length $l + 1$ for nodes $w \in R^a(v, l + 1)$.

Now in RHS,

$$\begin{aligned}&\sum_{w \in R^a(v,l+1)} \big(\Gamma_{X,u}(a).\gamma_{(u,v)}(a)\big).\Gamma_{v,w}(a, l + 1)\\ &= \sum_{w \in R^a(v,l+1)} \big(\Gamma_{X,u}(a).\gamma_{(u,v)}(a)\big). \sum_{y \in N_{in}(w)} \Gamma_{v,y}(a, l).\gamma_{(y,w)}(a)\\ &= \sum_{y \in R^a(v,l)} \big(\Gamma_{X,u}(a).\gamma_{(u,v)}(a)\big).\Gamma_{v,y}(a, l). \sum_{w \in N_{out}(y)} \gamma_{(y,w)}(a)\\ &= \sum_{y \in R^a(v,l)} \delta_a^{l,y}(\{e\}). \sum_{w \in N_{out}(y)} \gamma_{(y,w)}(a)\\ &= \sum_{w \in R^a(v,l+1)} \delta_a^{l+1,w}(\{e\})\end{aligned}$$

■

We are now able to formalize the change in credit due to a single edge deletion over all the actions in the action set $\mathscr{A}$.

Lemma 5 *Let $G^m = (V, E \setminus \{e\})$, then the change in credit $\Delta(\{e\})$ due to the removal* *of edge e is equal to:*

$$\Delta(\{e\}) = \sum_{a\in\mathscr{A}} \big(\Gamma_{X,u}(a).\gamma_{(u,v)}(a)\big).\big(\sum_{w\in V}\frac{1}{|\mathscr{A}_w|}\Gamma_{v,w}(a)\big)$$

Lemma 5 follows from Lemma 4 and Equation 6.2. Next, we prove the submodularity property of the function Δ. This property helps in designing efficient algorithms for both BIL and ILM.

Theorem 32 *The function Δ is monotone and submodular.*

Proof: The function is monotonic for each action a, as the removal of an edge cannot increase the credit. As a consequence, Δ (sum of credits over all actions) is also monotonic.

To prove submodularity, we consider the deletion of two sets of edges, E_S and E_T where $E_S \subset E_T$, and show that $\Delta(E_S \cup \{e\}) - \Delta(E_S) \geq \Delta(E_T \cup \{e\}) - \Delta(E_T)$ for any edge $e \in C$ such that $e \notin E_S$ and $e \notin E_T$. A non-negative linear combination of submodular functions is also submodular. Thus, it is sufficient to show the property for one action a, as Δ has the following form:

$$\begin{aligned}\Delta(B) &= \sigma_{cd}(G, X) - \sigma_{cd}(G^m, X)\\ &= \frac{1}{|\mathscr{A}_u|}\sum_{a\in\mathscr{A}_u}(\Gamma'_{X,u}(G,a) - \Gamma'_{X,u}(G^m,a))\end{aligned}$$

where $\Gamma'_{X,u}(G,a)$ denotes $\Gamma_{X,u}(a)$ in $G(a)$.

For the same reason, we assume a single node $x \in X$ ($\Gamma_{X,u} = \sum_{s\in X}\Gamma^{V-X+s}_{s,u}$). Edge sets E_S and E_T are removed from the graph and we evaluate $\Delta(\{e\})$ such that $e \notin E_S$ and $e \notin E_T$. Let the credits towards x from node w after removing E_S and E_T edges be

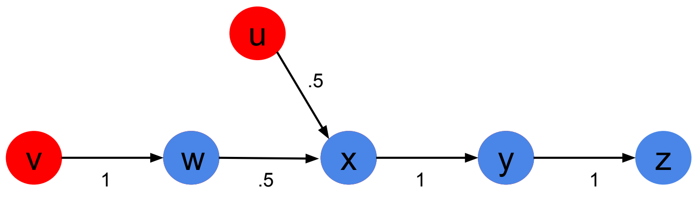


Figure 6.3: This illustrates a counter example in Theorem 34.

$\Gamma'_{x,w}(G^S)$ and $\Gamma'_{x,w}(G^T)$ (omitting a from $\Gamma'(.,a)$ for simplicity) respectively. Moreover, we use the notation $u\overrightarrow{a}v$ if there is a path from u to v in $G(a)$. There are two possible cases:

(1) If $w\overrightarrow{a}v$ does not hold, then removal of $e = (u,v)$ keeps $\Gamma'_{x,w}(G^S)$ and $\Gamma'_{x,w}(G^T)$ unchanged and the marginal gains for both E_S and E_T are 0.

(2) If $w\overrightarrow{a}v$ holds, marginal gains for sets E_S and E_T are equal to $\Gamma'_{x,u}(G^S).\gamma_{(u,v)}.\Gamma'_{v,w}(G^S)$ and $\Gamma'_{x,u}(G^T).\gamma_{(u,v)}.\Gamma'_{v,w}(G^T)$ respectively. Thus,

$$\Gamma'_{x,u}(G^S).\gamma_{(u,v)}.\Gamma'_{v,w}(G^S) \geq \Gamma'_{x,u}(G^T).\gamma_{(u,v)}.\Gamma'_{v,w}(G^T)$$

as $\Gamma'_{x,u}(G^S) \geq \Gamma'_{x,u}(G^T)$ and $\Gamma'_{v,w}(G^S) \geq \Gamma'_{v,w}(G^T)$.

These conclude that Δ is a submodular function. ■

The next section describes the APX-hardness of both versions of the influence limitation problem (ILM and BIL) using the notion of *curvature* [178] and L-reduction [59].

6.5 Curvature and APX-hardness

The ILM problem is NP-hard to approximate within a constant greater than $1 - \frac{1}{e}$. We prove the same about the BIL (budget constrained) problem. To show these results, we first describe a parameter named *curvature* that models the dependencies between elements (edges) in maximizing an objective function.

In ILM, the objective is $max\{\Delta(B), B \subset C\}$ where B is an independent set. Before proving APX-hardness, we define the concept of *total curvature* (c_t) [178].

Definition 13 *The total curvature of a monotone and submodular function* Δ *is defined by:*

$$c_t = 1 - min_{S,e_i} \frac{\Delta(S \cup \{e_i\}) - \Delta(S)}{\Delta(\emptyset \cup \{e_i\}) - \Delta(\emptyset)}$$

The *total curvature* [178] measures how much the marginal gains decrease when an element is added to a set S. Intuitively, it captures the level of dependency between elements in a set S. For instance, if the marginal gains are independent ($c_t = 0$) a simple greedy algorithm will be optimal. Let S^* be the optimal solution set. The *curvature with respect to optimal* (c_o) [178] is defined as follows:

Definition 14 Δ *has curvature with respect to optimal* $c_o \in [0, 1]$ *if* c_o *is the smallest value such that for every* T*:*

$$\Delta(S^* \cup T) - \Delta(S^*) + \textstyle\sum_{j \in S^* \cap T} \big(\Delta(S^* \cup T \setminus \{e_i\}) - \Delta(S^* \cup T)\big) \geq (1 - c_o)\Delta(T)$$

Vondrak [178] proves the following theorem:

Theorem 33 *There is no polynomial time algorithm that generates an approximation within a factor larger than* $\frac{1}{c_o}(1 - e^{-c_o})$ *for maximizing a monotone and submodular function under matroid constraints where* c_o *is the curvature with respect to optimal.*

We use Theorem 33 to prove the APX-hardness of ILM.

Theorem 34 *ILM is APX-hard and cannot be approximated within a factor greater than* $(1 - 1/e)$*.*

Proof: ILM is a monotone and submodular optimization problem under a matroid constraint. We prove the inapproximability result by designing a problem instance where the *curvature with respect to optimal* (c_o) is 1. Consider the example in Figure 6.3, the candidate set $C = \{(w,x),(x,y),(y,z)\}$, $b = 1$ and the target set $X = \{u,v\}$. In this setting, one of the optimal sets $S^* = (w,x),(x,y)$. Assuming $T = (y,z)$ will imply $S^* \cap T = \emptyset$. If $\Delta(S^* \cup T) - \Delta(S^*) = 0$, then c_o has to be 1. Note that, $\Delta(S^* \cup T) = \Delta(S^*) = 2.5$, which leads to $c_o = 1$. Therefore, ILM cannot be approximated within a factor greater than $\frac{1}{1}(1 - e^{-1})$ and our claim is proved. ■

Theorem 35 *BIL is APX-hard and cannot be approximated within a factor greater than $(1 - 1/e)$.*

Proof: We reduce the BIL problem from a problem that is similar to ILM and has matroid constraints with *curvature with respect to optimal* as 1. First, we define this problem as ILMO where maximum b outgoing edges can be deleted form a node (unlike in ILM where the limit was on incoming edges). However, ILMO is NP-hard, follows matroid constraints and has curvature 1 (the proofs are straightforward and similar to the proofs for ILM). Thus ILMO cannot be approximated within a factor greater than $(1 - \frac{1}{e})$ (similarly as Theorem 34).

Now we give an L-reduction [59] from the ILMO problem. The following two equations are satisfied in our reduction:

$$OPT(I_{BIL}) \leq c_1 \cdot OPT(I_{ILMO})$$
$$OPT(I_{ILMO}) - s(T^S) \leq c_2 \cdot (OPT(I_{BIL}) - s(T^B))$$

where I_{ILMO} and I_{BIL} are problem instances, and $OPT(Y)$ is the optimal value for instance Y. $s(T^S)$ and $s(T^B)$ denote any solution of the ILMO and BIL instances, respectively. If the conditions hold and BIL has an α approximation, then ILMO has an $(1 - c_1c_2(1 - \alpha))$ approximation. It is NP-hard to approximate ILMO within a factor greater than $(1 - \frac{1}{e})$. Now, $(1 - c_1c_2(1 - \alpha)) \leq (1 - \frac{1}{e})$, or, $\alpha \leq (1 - \frac{1}{c_1c_2e})$. So, if the conditions are satisfied, it is NP-hard to approximate BIL within a factor greater than $(1 - \frac{1}{c_1 c_2 e})$.

Consider a problem instance I_{ILMO}, where graph $G = (V, E)$, $|V| = n, |E| = m$ and integer b and the target set $X = \{x\}$ are given. This problem becomes a BIL instance when $b = k$ where k is the budget (in BIL). If the solution of I_{ILMO} is $s(T^S)$ then the influence of node x will decrease by $s(T^S)$. Note that $s(T^B) = s(T^S)$ from the construction. Thus, both the conditions are satisfied when $c_1 = 1$ and $c_2 = 1$. So, BIL is NP-hard to approximate within a factor grater than $(1 - \frac{1}{e})$. ■

Both the BIL and ILM problems are APX-hard and cannot be approximated within a constant greater than $1 - \frac{1}{e}$. However, the next two sections describe efficient approximate algorithms for influence limitation based on submodularity. Algorithm 9 (Greedy) provides tight approximation $(1 - \frac{1}{e})$ for BIL and Algorithm 13 (CG) produces the same for ILM with high probability.

6.6 Method: BIL

According to Theorem 12, BIL is a monotone submodular maximization problem under a budget constraint. As a consequence, a simple greedy algorithm produces a constant factor approximation of $(1-1/e)$ [60] for the problem. We introduce an efficient version of this greedy algorithm based on properties of the credit distribution model. The greedy algorithm removes the edge that minimizes the credit of the target set, one at a time. After each removal, the credit $(\Gamma_{u,v})$ of node u for influencing v has to be updated. However, as only one edge e is removed, intuitively, nodes in the entire network should not be affected but only some in the neighborhood of e. We formalize these observations and apply them in an efficient algorithm for BIL. The notation $u \overrightarrow{a} v$ denotes that there is a path from u to v in $G(a)$.

Algorithm 9 Greedy

```
Require: $X$, $C$, $k$
Ensure: A solution set $B$ of $k$ edges
 1: $B \leftarrow \emptyset$
 2: while $|B| \leq k$ do
 3:     for $e \in C \setminus B$ do
 4:         $e.MC \leftarrow$ computeMC($e$)
 5:     end for
 6:     $e^* \leftarrow \arg\max_{e \in C \setminus B}\{e.MC\}$
 7:     $B \leftarrow B \cup \{e^*\}$ and $E \leftarrow E \setminus \{e^*\}$
 8:     updateUC($e, EP, UC, SC$)
 9:     updateSC($e, EP, UC, SC$)
10: end while
11: return $B$
```

Algorithm 10 computeMC

```
Require: $e = (u, v)$, $X$, $UC$, $SC$
Ensure: $mc$
1: $mc = 0$
2: for $a \in \mathscr{A}$ such that $SC[u][a] > 0$ and $EP[u][v][a] > 0$ do
3:     $mc_a = 0$
4:     for each user $w$ such that $UC[v][w][a] > 0$ do
5:         $mc_a = mc_a + UC[v][w][a]/\mathscr{A}_w$
6:     end for
7:     $mc = mc + (SC[u][a] \cdot EP[u][v][a]) \cdot mc_a$
8: end for
```

Observation 3 *For a given action a and DAG $G(a)$, the removal of $e = (u, v)$ changes $\Gamma_{z,w}$ iff $z \overrightarrow{a} u$ and $v \overrightarrow{a} w$.*

Let $G(a)$ and $\{z, w\}$ be an arbitrary DAG and node pair, respectively. Deleting $e = (u, v)$ can only affect the credit $\Gamma_{z,w}$—i.e. credit of node z for influencing w—if e is on a path from z to w in $G(a)$. The edge e is on one of such paths if and only if $z \overrightarrow{a} u$ and $v \overrightarrow{a} w$. The following observations are derived from Lemma 5.

Observation 4 *For given DAG $G(a)$, the removal of $e = (u, v)$ reduces $\Gamma_{z,w}$ by $(\Gamma_{z,u} \cdot \gamma_{(u,v)}) \cdot \Gamma_{v,w}$ iff $z \overrightarrow{a} u$ and $v \overrightarrow{a} w$.*

Algorithm 11 updateUC

```
Require: $e = (u, v)$, $EP$, $UC$, $SC$
1: for $a \in \mathscr{A}$ do
2:     $\gamma \leftarrow EP[u][v][a]$
3:     for each user $z$ such that $UC[z][u][a] > 0$ do
4:         for each user $w$ such that $UC[v][w][a] > 0$ do
5:             $UC[z][w][a] = UC[z][w][a] - (UC[z][u][a] \cdot \gamma) \cdot UC[v][w][a]$
6:         end for
7:     end for
8: end for
```

Algorithm 12 updateSC

```
Require: $e = (u, v)$, $EP$, $UC$, $SC$
1: for $a \in \mathscr{A}$ such that $SC[u][a] > 0$ and $EP[u][v][a] > 0$ do
2:     $\gamma \leftarrow EP[u][v][a]$
3:     for each user $w$ such that $UC[v][w][a] > 0$ do
4:         $SC[w][a] = SC[w][a] - (SC[u][a] \cdot \gamma) \cdot UC[v][w][a]$
5:     end for
6: end for
```

Observation 5 *For given target set X, an action a and DAG $G(a)$, the removal of $e = (u, v)$ reduces $\Gamma_{X,w}$ by $(\Gamma_{X,u} \cdot \gamma_{(u,v)}) \cdot \Gamma_{v,w}$ iff $z \overrightarrow{a} u$ and $v \overrightarrow{a} w$ where $z \in X$.*

Algorithm 9 scans the action log $\mathscr{L}$ to collect information for comparing the effect of each candidate edge. This information is maintained in data structures EP, EC, and SC. More specifically, $EP[u][v][a]$ denotes the edge credit ($\gamma_{(u,v)}(a)$) of u for influencing v when (u, v) exists, $UC[u][v][a]$ is the credit ($\Gamma_{u,v}(a)$) given to u for influencing v, and $SC[u][a]$ is the credit ($\Gamma_{X,u}(a)$) given to X for influencing u, all for an action a.

The contribution of each edge (see Lemma 5), given the current solution B, is computed using *computeMC*. Method *updateUC* (Algorithm 11) identifies the credits that have been changed upon an edge removal and does so by updating the data structure UC following Observation 4. Method *updateSC* does the same for the credits of target set of nodes X by updating the data structure SC following Observation 5.

The expensive steps of Algorithm 9 are steps 4, 7 and 8. The corresponding methods *computeMC*, *updateUC*, and *updateSC*, take $O(\sum_{a\in\mathscr{A}}|V(a)|), O(\sum_{a\in\mathscr{A}}|V(a)|^2)$, and $O(\sum_{a\in\mathscr{A}}|V(a)|)$ time respectively. Thus, the total running time of Greedy is $O(k\cdot|C|\cdot\sum_{a\in\mathscr{A}}|V(a)|+k\cdot\sum_{a\in\mathscr{A}}|V(a)|^2)$. Notice that the total time complexity does not depend on the number of nodes ($|V|$) in the graph, but on the sizes of the action graphs, the budget and the candidate edge set.

Optimization of Greedy in BIL

We propose an intuitive and simple optimization technique to further improve the efficiency of Greedy. The question about optimization is the following: do all the edges in the candidate set (C) of edges need to be evaluated? To answer this, we introduce a concept of *edge dominance*. The idea is very intuitive and simple. If an edge $e' = (w, x)$ is reachable from the target set through only a particular edge $e^* = (u, v)$ in all the DAGs, then we call e' as the dominated and the edge e^* as the dominating edge. In other words, there is no such path from a node in X to w without going through e^* in $G(a)$ for all $a \in \mathscr{A}$. Note that the if the dominating edge e^* is removed from the graph, the marginal contribution towards reducing influence of target set X by removing e' becomes 0. The next lemma depicts the dominance of an edge.

Lemma 6 *If $e' = (w, x)$ and $e* = (u, v)$ are present in $G(a)$, and $\Gamma_{X,w} = \Gamma_{X,u}\cdot\gamma_{(u,v)}\cdot\Gamma v, w$ for all $a \in \mathscr{A}$ then e' is dominated by e^*.*

6.7 Method: ILM

The greedy algorithm (Algorithm 9) for budgeted influence limitation (BIL) might not provide a valid solution for the matroid constrainted problem. Based on Observation 2, we apply continuous relaxation and adopt the continuous greedy technique to design our algorithm. Furthermore, we propose a fast randomized scheme for rounding the relaxed solution that works well in practice.

6.7.1 Continuous Relaxation

Let $\vec{y} = (y_1, y_2, ...y_c)$ be the vector with membership probabilities for each edge in the candidate set C ($|C| = c$). Moreover, let B be a random subset where $e_i \in C$ is included in B with probability y_i. From [173], if f is the continuous extension of Δ, then:

$$f(\vec{y}) = \mathbf{E}_{B\sim\vec{y}}[\Delta(B)] = \sum_{B\subseteq C} \Delta(B) \prod_{e_i\in B} y_i \prod_{e_i\in C\setminus B} (1 - y_i) \tag{6.6}$$

Let $E_{in}(v)$ be the edges incoming to v. Our objective is to find a $\vec{y}$ that maximizes $f(\vec{y})$ with the following constraints:

$$y_i \in [0, 1] \tag{6.7}$$

$$\sum_{e_i\in E_{in}(v)} y_i \leq b \quad \forall v \in V \tag{6.8}$$

While Equation 6.7 (constraint **S1**) maintains the fractional values as probabilities, Equation 6.8 (constraint **S2**) enforces the maximum number of edges incident to each node to be bounded by b. Because the relaxation of Δ as f is continuous, the optimal value for f is an upper bound on Δ (the discrete version). Let B^* and Y^* be the optimal edge sets for Δ and f, respectively. Also, let Z be a vector defined as follows: $z_i = 1$ if $e_i \in B^*$ and $z_i = 0$, otherwise. Then, $\Delta(B^*) = f(Z)$ and Z maintains the constraints. As $f(Y^*)$ is maximum, $\Delta(B^*) = f(Z) \leq f(Y^*)$.

We show that the new function f is smooth (it has a second derivative), monotone

Algorithm 13 Continuous Greedy (CG)

```
Require: $X$, $C$, $b$
Ensure: A vector $\vec{y}$ satisfying constraints (S1) and (S2)
1: Start $\vec{y}$ as a null vector, $t = 0$
2: while $t \leq \tau$ do
3:     Generate $s$ samples $B_1, B_2, ..., B_s$ where $e_i$ belongs to $B_j$ ($\forall j \in [s]$) with probability $y_i$
4:     Set weight of an edge, $e_i$ as $w_i = \frac{\sum_{j=1}^{s} \Delta(B_j \cup \{e_i\}) - \Delta(B_j)}{s}$
5:     Compute an edge set $E^Y$ maintaining the constraint (S2) and maximizes $\sum_{e_i \in E^Y} w_i$
6:     For all $e_i \in E^Y$, set $y_i = y_i + 1/\tau$
7:     $t = t + 1$
8: end while
9: return $\vec{y}$
```

and submodular. Based on these properties, we can design a continuous greedy algorithm that produces a relaxed solution for ILM with a constant-factor approximation [173].

Theorem 36 *The objective function f is smooth, monotone and submodular.*

Proof: Let $\mathcal{Y} = (y_1, y_2, ...y_c)$ be the vector with membership probabilities for each edge in C ($c = |C|$). Let the set B be a random subset of C where the edge $e_i \in C$ is included in set B with probability y_i. If f is the continuous extension of Δ, then,$f(\mathcal{Y}) = \mathbf{E}_{B \sim \mathcal{Y}}[\Delta(B)] = \sum_{B \subseteq C} \Delta(B) \prod_{e_i \in B} y_i \prod_{e_i \in C \setminus B} (1 - y_i)$. To prove the function $f : [0,1]^C \rightarrow \mathbb{R}$ is a smooth monotone submodular function, we need to prove the followings:

i) f has second partial derivatives everywhere.

ii) Monotonicity: For each $e_i \in C$, $\frac{\partial f}{\partial y_i} \geq 0$.

iii) Submodularity: For each $e_i, e_j \in C$, $\frac{\partial^2 f}{\partial y_i \partial y_j} \geq 0$.

We derive a closed form similar in [173] for the second derivative and thus it always exists.

For each $e_i \in C$,

$$\frac{\partial f}{\partial y_i} = \mathbf{E}[\Delta(B)|e_i \in B] - \mathbf{E}[\Delta(B)|e_i \notin B]$$

As Δ is monotone, $\mathbf{E}[\Delta(B)|e_i \in B] - \mathbf{E}[\Delta(B)|e_i \notin B] \geq 0$ and thus, f is also monotone.

For each $e_i, e_j \in C, i \neq j$,

$$\frac{\partial^2 f}{\partial y_i \partial y_j} = \mathbf{E}[\Delta(B)|e_i, e_j \in B] - \mathbf{E}[\Delta(B)|e_i \in B, e_j \notin B] -$$
$$\mathbf{E}[\Delta(B)|e_i \notin B, e_j \in B] - \mathbf{E}[\Delta(B)|e_i, e_j \notin B]$$

As Δ is submodular, $\frac{\partial^2 f}{\partial y_i \partial y_j} \geq 0$ from the above expression. Thus, f is submodular. Note that if $i = j$, $\frac{\partial^2 f}{\partial y_i \partial y_j} = 0$. In other words, the relaxation f is called multi-linear because it is linear in every co-ordinate (y_i). ■

Continuous Greedy (CG): The continuous greedy algorithm (Algorithm 13) provides a solution set $\vec{y}$ such that $f(\vec{y}) \geq (1-\frac{1}{e})f(Y^*) \geq (1-\frac{1}{e})\Delta(B^*)$ with high probability. The approximation guarantee exploits the facts that Δ is submodular (Theorem 36) and ILM follows a matroid constraint (Observation 2). CG is similar to the well-known Frank-Wolfe algorithm [179]. It iteratively increases the coordinates (edge probabilities) towards the direction of the best possible solution with small step-sizes while staying within the feasible region. In [173], the author proves the following:

Theorem 37 *The Continuous Greedy (Algorithm 13) returns a vector $\vec{y}$ that satisfies constraints S1 and S2 and such that $f(\vec{y}) \geq (1-\frac{1}{e})\Delta(B^*)$ when $\tau = c^2$ and $s = c^5$.*

The values τ and s correspond to the number of iterations and samples applied by the CG algorithm. The costliest operations of CG are steps 3, 4 and 5. Step 3 takes $O(c.s)$ time, as it visits each edge in the candidate set C ($|C| = c$). Step 4 computes the contribution of edges, having worst case time complexity $O(s.c\sum_{a\in\mathscr{A}}|V(a)|)$. Step 5 greedily selects the best set of edges, according to the weights. Therefore, the total running time of the algorithm is $O(\tau(s.c\sum_{a\in\mathscr{A}}|V(a)| + c\log c))$. Though Theorem 37 requires high value of τ and s, in practice, the algorithm produces high quality results with low values of them (Section 6.8.2).

6.7.2 Rounding

Algorithm 13 returns a vector $\vec{y}$ satisfying constraints Eq. 6.7 and Eq. 6.8 while producing $f(\vec{y}) \geq (1 - \frac{1}{e})\Delta(B^*)$. However, as $\vec{y}$ contains probability values (in the interval $[0, 1]$), a rounding step is still required for obtaining a deterministic set of edges.

There exists a computationally-intensive lossless rounding procedure for matroids known as *swap rounding* [175]. The computation depends on the number of base matroids which can be very large in the solution set obtained from Algorithm 13. We address the high time complexity issue by proposing a simpler and faster randomized procedure. We show that our independent rounding method produces feasible edges with low error and high probability.

We sort the edges according to their weights (probabilities) and round them while maintaining feasibility. This fast procedure only makes a single pass over the candidate edges in C. In order to analyze this randomized procedure, we assume that it is unaware of the dependency between the edges. Let $\mathcal{B}$ be the edge set produced by rounding, i.e. $f(\vec{y}) = \mathbf{E}[\Delta(\mathcal{B})]$, and let $E_v \subset \mathcal{B}$ be the incoming edges incident on node v. The next theorem shows that the randomized procedure will produce a feasible set within error ε with (high) probability $1 - \frac{1}{n}$, where $n = |V|$ is the number of nodes.

Theorem 38 *The following holds for the number of edges incoming to v in the rounded* *set:*

$$Pr(|E_v| < (1+\varepsilon)b) \geq 1 - \frac{1}{n}$$

where $\varepsilon = \sqrt{\frac{6\log n}{b}}$.

Proof: Let $\mathcal{B}$ be the set of edges produced by the rounding procedure. An edge e_i is included in $\mathcal{B}$ with probability y_i. As $\vec{y}$ is a feasible solution, $\sum_{e_i} y_i \leq b \quad \forall v \in V$ (Equation 6.8) where e_i is incident (incoming) to vertex v. Thus, $\mathbf{E}(|E_v|) \leq b$. Applying the Chernoff's bound:

$$Pr(|E_v| \geq (1+\varepsilon)\mathbf{E}(|E_v|)) < \exp\left(-\frac{\mathbf{E}(|E_v|)\varepsilon^2}{3}\right)$$

Applying the union bound, $\forall v \in V$, we get:

$$Pr(|E_v| \geq (1+\varepsilon)b) < n \cdot \exp\left(-\frac{b\varepsilon^2}{3}\right)$$

Substituting $\varepsilon = \sqrt{\frac{6\log n}{b}}$, we get:

$$Pr(|E_v| < (1+\varepsilon)b) \geq 1 - \frac{n}{n^2} = 1 - \frac{1}{n}$$

■

We emphasize two implications of Theorem 38: (1) The probability that the rounded solution is feasible depends on the error ε which is small whenever b is large; (2) The rounding procedure has a probabilistic bi-criteria approximation, being lossless if the maximum number of edges to be removed per node is $b' = b(1+\varepsilon)$. The proposed randomized rounding scheme is efficient, as it only performs one pass over the candidate edges C in order to generate its output.

Dataset Name	$\|V\|$	$\|E\|$	#Action	#Tuple
ca-AstroPh (CA)	$18K$	$197K$	$1K$	$56K$
email-EuAll (EE)	$265K$	$420K$	–	–
Youtube (CY)	$1.1M$	$2.9M$	–	–
Flixster-small (FXS)	$15K$	$191K$	$1.8K$	$30K$
Flickr-small (FCS)	$15K$	$1.4M$	$1.4K$	$10K$
Flixster (FX)	$1M$	$28M$	$49K$	$8.2M$
Flickr (FC)	$1.3M$	$81M$	$296K$	$36M$

Table 6.2: Statistics of the datasets. We generate synthetic actions via IC model for CA, EE and CY datasets.

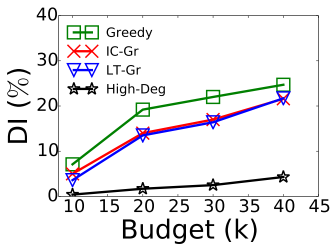
6.7.3 Generalizations

Matroids can capture other influence limitation settings, especially when edges in the solution can be naturally divided into partitions. Examples include the limitation of influence in non-overlapping communities [180, 181], disjoint campaigning [182], and problems where issues of fairness arise [183]. Moreover, influence boosting problems via attribute-level modification [37] and edge addition [4] can also be modelled under matroid constraints.

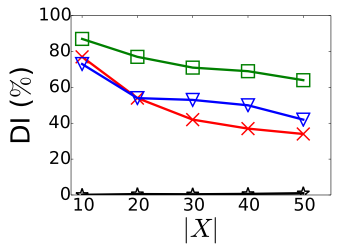
6.8 Experimental Results

Our solutions were implemented in Java and experiments were conducted on 3.30GHz Intel core with 30 GB RAM.

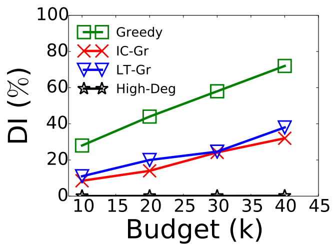
Datasets: The datasets used in the experiments are the following: 1) **Flixster [170]**: Flixster is an unweighted directed social graph, along with the log of performed actions. The log has triples of (u, a, t) where user u has performed action a at time t. Here, an action for a user is rating a movie. 2) **Flickr [184]**: This is a photo sharing platform. Here, an action would be joining an interest group. 3) **Synthetic**: We use the structure of real datasets that come from different genre (e.g., co-authorship, social).



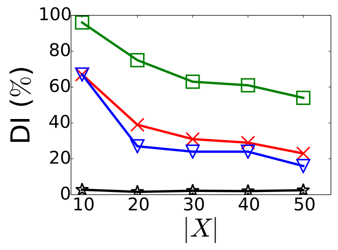
(a) CA (varying k)



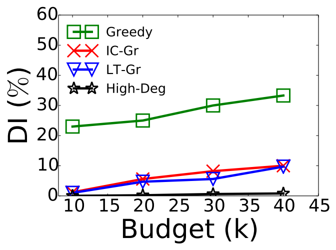
(b) CA (varying $|X|$)



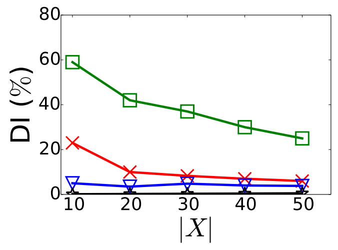
(c) FXS (varying k)



(d) FXS (varying $|X|$)



(e) FCS (varying k)



(f) FCS (varying $|X|$)

Figure 6.4: [BIL] (a, c, e) Decrease in Influence (DI) produced by different algorithms. (b, d, f) DI produced by different algorithms varying the size of the target set, X with $k = 30$.

The networks are available online[2]. We synthetically generate the actions and create associated tuples. Synthetic actions are generated assuming the Independent Cascade (IC) [31] model. The "ca-AstroPh" dataset is a Collaboration network of Arxiv Astro Physics. In the "Youtube" social network, users form friendship with others and can create groups that other users can join. Table 6.2 shows the statistics of the datasets. We use the small extracted networks (from Flixster and Flickr) for the quality-related experiments as our baselines are not scalable. To show the scalability of our methods,

[2]https://snap.stanford.edu

	FXS: # (tuples, actions)$\times 10^3$		
Budget	$(30, 1.7)$	$(50, 4.8)$	$(75, 6.9)$
$k = 50$	58	61	68
$k = 75$	73	83	85
$k = 100$	85	88	91
	FCS: # (tuples, actions)$\times 10^3$		
	$(20, 2.6)$	$(30, 3.8)$	$(50, 5.8)$
$k = 50$	208	383	1187
$k = 75$	269	579	1891
$k = 100$	356	780	2551

Table 6.3: [BIL] Running times (in seconds) of Greedy varying number of tuples. The number of tuples and actions are in thousands.

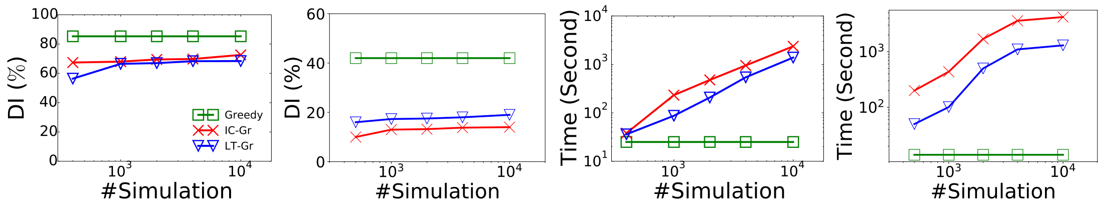
we extract networks of different sizes from the raw large Flixster and Flickr data. For all the networks, we learn the influence probabilities via the widely used method proposed by Goyal et al. [177].

We use the small sub-networks (from Flixster and Flickr) for quality-related experiments as our baselines are not scalable. Influence probabilities are learned using the method proposed in [177].

Performance Metric: The quality of a solution set B (a set of edges) is the percentage of *Decrease in Influence (DI)* of X:

$$DI(B) = \frac{(\sigma_{cd}(G, X) - \sigma_{cd}(G^m, X))}{\sigma_{cd}(G, X)} \times 100 \tag{6.9}$$

The set X is randomly selected from the set of top 150 nodes with highest number of actions. The candidate set C contains edges that appear at least once in any action graph. The number of MC simulations for IC and LT-based baselines is at least 1000.



(a) Quality on CA (b) Quality on FXS (c) Time on CA (d) Time on FXS

Figure 6.5: [BIL] Comparison of our greedy algorithm and simulation based baselines varying number of simulations: (a-b) Quality and (c-d) Running time.

6.8.1 Experiments: BIL

Baselines: 1) **IC-Gr [34]:** Finds the top k edges based on the greedy algorithms that minimize influence via edge deletion under the IC model. 2) **LT-Gr [4]:**Finds the top k edges based on the greedy algorithm proposed in [4]. Here, the authors minimize the influence of a set of nodes according to the LT model via edge deletion. Note that we also apply optimization techniques proposed in [4] for both of these baselines. 3) **High-Degree:** Selects edges between the target nodes X and the top-k high degree nodes. Other heuristics (*Friends of a Friend* and random selection) did not produce better results than High-Deg.

Quality (vs Baselines): We compare our Greedy algorithm (Algorithm 9) against the baselines on three datasets (CA, FXS and FCS) in Figures 6.4a, 6.4c and 6.4e (target size is set as 30). Greedy takes a few seconds to run and significantly outperforms the baselines (by up to 35%) in terms of DI(%). The running time of Greedy is low as it avoids expensive Monte-Carlo simulations. For CA, the action graphs are generated by IC model and hence, IC-Gr produces better results than the other two datasets.

Scalability of Greedy: We show the scalability of our Greedy algorithm (Algorithm 9) for increasing number of tuples (#actions) size of the graph. Table 6.3 shows the results on FXS and FCS. As FCS has higher edge density than FXS, the number of tuples has higher effect on the running time in FCS. Note that we consider all the edges that appear

Dataset	$\lvert V\rvert$	Actions	Tuples	$\lvert C\rvert$	Time (sec)
EE	265K	5K	326K	4.1K	637
CY	1.1M	5K	313K	6.3K	950
FX	200K	2.6K	200K	51K	4020

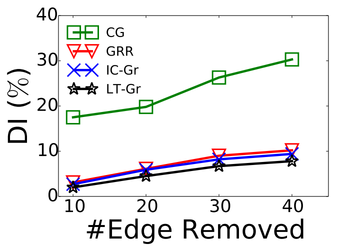
Table 6.4: [BIL] Running Times of Greedy varying graph size for $|X| = 30$ and $k = 30$.

in one of the actions in our candidate set of edges. A larger candidate set results in longer running time. However our algorithm only takes around 2 and 43 minutes to run for $75K$ and $50K$ tuples in FXS and FCS, respectively.

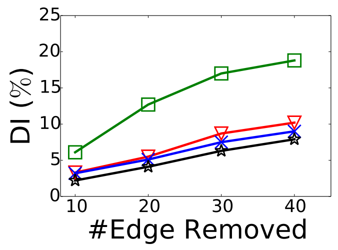
Table 6.4 shows the results varying the graph size. The running times are dominated by the size of both the graphs and the candidate sets. Greedy takes nearly 16 minutes on CY with 1M nodes and 6K candidate edges, whereas, it takes 67 minutes on FX with 200K nodes and 51K candidate edges.

Parameter variations: We also analyze the impact of varying the following parameters: the number of target nodes ($|X|$) and the number of simulations for LT-Gr and IC-Gr. First we vary the size of the target set X. Figures 6.4b, 6.4d, and 6.4f show the results for CA, FXS, and FCS respectively where budget, $k = 30$. Greedy provides better DI (by up to 35%) across all $|X|$ and datasets. With the increase in $|X|$, DI decreases for the top three algorithms. A larger $|X|$ would have a higher influence to reduce. Thus, with the same number of edges removed, the DI would decrease for larger target set. DI is lower for FCS as it is denser than CA and FXS.

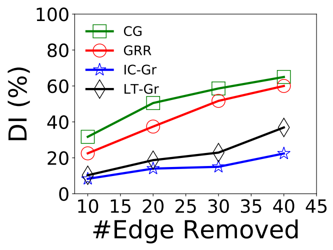
We also evaluate the effect of the number of simulations on LT-Gr and IC-Gr. Figure 6.5 shows the results for $|X| = 30$ and $k = 20$. Our algorithm produces better results even when the baselines perform 10^4 simulations. By comparing Figures 6.5a and 6.5b, it is evident that IC-Gr performs better than LT-Gr in CA as the synthetic actions are generated by IC model. Figure 6.5d also shows that our method is 1-4 orders faster than the simulation based baselines.



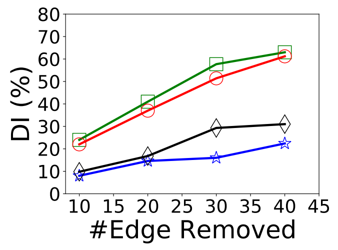
(a) CA (b=1)



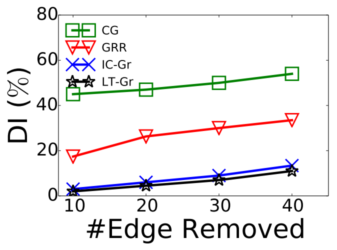
(b) CA (b=2)



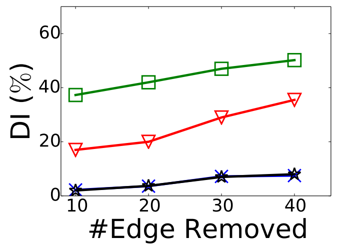
(c) FXS (b=1)



(d) FXS (b=2)



(e) FCS (b=1)



(f) FCS (b=2)

Figure 6.6: [ILM] Decrease in Influence produced by different algorithms on (a-b) CA, (c-d) FXS, and (e-f) FCS. Our algorithm, CG outperforms the baselines by up to 20%.

6.8.2 Experiments: ILM

Baselines and other settings: 1) **Greedy with Restriction (GRR):** Finds the feasible edges (respecting the matroid constraint) using a greedy algorithm (BIL). (2-3) We also apply **IC-Gr** and **LT-Gr** with the edge removal constraint for each node. The number of samples and iterations used in CG are $s = 20$ and $\tau = 100$, respectively. After obtaining the solution vector from CG, we run randomized rounding for 50 times and choose the best solution.

	FCS: # (tuples, actions)$\times 10^3$		
	$(20, 2.6)$	$(30, 3.8)$	$(50, 5.8)$
20	28.1	64.7	180
40	29.1	64.6	181
60	29.2	63.1	167

Table 6.5: [ILM] Running Times (in min.) of CG varying number of tuples for $|X| = 20$ and $b = 2$.

Quality (vs Baselines): We compare the Continuous Greedy (CG) algorithm against the baseline methods on CA, FXS and FCS where $|X| = 30$ varying the number of edge removal. Figure 6.6 shows the results when $b = 1$ and $b = 2$. CG significantly outperforms the baselines by up to 20%. GRR does not produce good results as it has to select the feasible edge that does not violate the maximum edge removal constraint b. While maintaining feasibility, GRR cannot select the current true best edge.

	FCS: # (tuples, actions)$\times 10^3$					
#Edge	$(20, 2.6)$		$(30, 3.8)$		$(50, 5.8)$	
	CG	**GRR**	**CG**	**GRR**	**CG**	**GRR**
20	33	22	50	41	35	20
40	42	32	53	44	45	35
60	44	35	61	54	54	40

Table 6.6: [ILM] Decrease in Influence (%) in FCS by Continuous Greedy (CG) vs GRR varying the number of tuples.

Scalability of Continuous Greedy: CG (Algorithm 13) is generally slower than GRR. However, unlike for GRR, increasing the budget does not affect the running time of CG. We evaluate the running time of CG while increasing the number of tuples (actions) using FCS in Table 6.5. Table 6.6 shows the results on the quality in DI (%) produced by CG and GRR (other baselines are not scalable) on FCS data. CG outperforms GRR by up to 15%. Table 6.8 shows the results on FXS. Because of higher density and thus larger candidate set, CG takes longer in FCS (Table 6.5). These observations validate the running time analysis for CG (Section 6.7.1). Table 6.7 shows the results on FXS data. CG outperforms GRR by up to 8%. CG consistently produces better results than GRR.

	FXS: # (tuples, actions)$\times 10^3$					
#Edge Removed	$(30, 1.7)$		$(50, 4.8)$		$(75, 6.9)$	
	CG	**GRR**	**CG**	**GRR**	**CG**	**GRR**
20	50	44	48	42	51	44
40	51	47	53	45	60	56
60	60	54	61	55	63	57

Table 6.7: [ILM] Decrease in Influence (%) in FXS by Continuous Greedy (CG) vs GRR varying the number of tuples. The number of tuples and actions are in thousands.

Table 6.9 shows results varying the graph size. Running times are dominated by the size of the graphs and the candidate sets (numbers of actions and tuples are same as in Table 6.4). CG takes approximately 31 minutes on CY with 1M nodes and 6K candidate edges, whereas it takes approximately 1.5 hours on FX with 200K nodes and 51K candidate edges. CG also outperforms GRR by up to 9%.

Parameter Variation: The size of the target set X is varied and we observe its effect in Figures 6.7a and 6.7. We set $b = 2$, and remove 20 edges for these experiments. CG provides better DI consistently across different target sizes. As expected, with the increase of target set size, DI generally decreases for all the algorithms. A larger target size would have a higher influence to be reduced. Thus, with the same number of edges removed, the DI would decrease for a larger target set. We compare the Continuous Greedy (CG) algorithm against the baseline methods on CA, FCS and FXS varying b (Figure 6.6). CG significantly outperforms the baselines.

	FXS: # (tuples, actions)$\times 10^3$		
#Edge Removed	$(30, 1.7)$	$(50, 4.8)$	$(75, 6.9)$
20	7.5	20.7	69.4
40	7.1	16.8	69.4
60	7.4	16.7	69.5

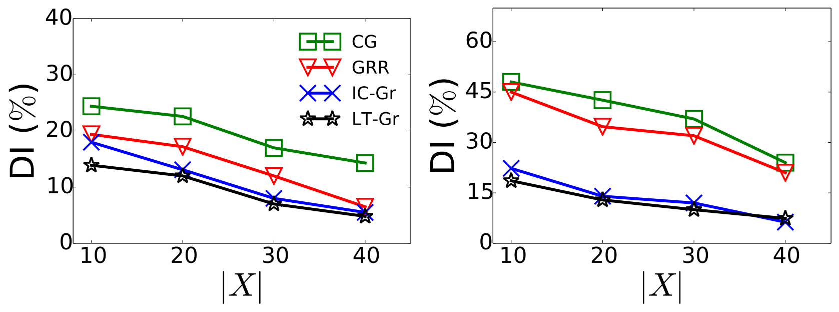
Table 6.8: [ILM] Running times (in minutes) of CG varying number of tuples for $|X| = 20$ and $b = 2$ on FXS.

Dataset	$\|V\|$	$\|C\|$	CG (Time)	CG (DI)	GRR (DI)
CY	1.1M	6.3K	1858	55.1	47.2
FX	200k	51K	5690	46.2	37.3

Table 6.9: [ILM] Running Time (Scalability) in seconds of CG and Decrease in Influence (percentage) by CG and GRR varying graph size for $|X| = 20$, $b = 2$ and the number of edges removed is 20.

6.9 Conclusions

We studied the influence limitation problem via edge deletion. Different from previous work, our formulation is data-driven, taking into account available propagation traces in the selection of edges. Our influence limitation problem was framed under two different types of constraints—budget and matroid. Both versions were shown to be APX-hard and cannot be approximated within a factor greater than $(1 - \frac{1}{e})$. For the budget constrained version, we have developed an efficient greedy algorithm that achieves a good approximation guarantee by exploiting the monotonicity and submodularity of the objective function. The matroid constrained version was solved via continuous relaxation and a continuous greedy technique, achieving a probabilistic approximation guarantee. Experiments showed the effectiveness of our solutions, which outperform the baselines using both real and synthetic datasets.



(a) Quality on CA (b) Quality on FXS

Figure 6.7: [ILM] Decrease in Influence (DI) produced by different algorithms varying the size of the target set, X when $b = 2$.

Chapter 7

Conclusion

This thesis outlines the scope of network design problems and their applications. We have built algorithms to improve different objectives in networks. For instance, in Chapter 2, different from existing techniques, our formulation incorporated the practical considerations that the impact of delay minimization should be noticeable and favor important paths in a network. We have proposed a scalable, importance sampling-based, traffic improvement method to reduce commute delays. The algorithm is evaluated on large traffic data from three major cities (New York, Beijing, and San Fransisco) and shows 70-fold improvement over state-of-the-art techniques.

Chapter 3 have studied several variations of a novel network design problem, *group centrality optimization*. This problem has applications in a variety of domains including social, collaboration, and communication networks. Our randomized algorithms provide theoretical quality guarantees under realistic assumptions and also outperform the baseline methods by up to 5 times on several datasets.

We have discussed the the Hiding Leader problem for the core centrality and degree centrality measures in Chapter 4. The chapter proves several hardness and approximability results for the same problem in different settings as well as design approximation algorithms. Hence, our results prove that, although classical complexity theoretic framework fails to compare relative difficulty of hiding leaders with respect to various centrality measures, hiding leaders may be significantly harder for the core centrality than the degree centrality.

This thesis has also designed an algorithm to profile network resilience via the k-core minimization (KCM) problem in Chapter 5. KCM discovers that the structure of the human gene is more complex and stable than that of the yeast gene in Chapter 5. The proposed algorithm is an efficient sampling based heuristic based on Shapley value to account for the interdependence in the impact of candidate edges.

Finally, the influence limitation problem via edge deletion is described in Chapter 6. Different from previous work, our formulation is data-driven, taking into account available propagation traces in the selection of edges and framed under two different types of constraints—budget and matroid. For the budget constrained version, we have developed an efficient greedy algorithm that achieves a good approximation guarantee by exploiting the monotonicity and submodularity of the objective function. The matroid constrained version was solved via continuous relaxation and a continuous greedy technique, achieving a probabilistic approximation guarantee.

We have also shown that different from search versions (e.g. computing shortest path), the design ones (e.g. optimizing shortest path) are computationally harder[11]. The problems discussed in this thesis cover a large range of the spectrum of intractability. While the k-core minimization problem [12] is NP-hard to approximate within any constant, the shortest path [14] optimization problem cannot be approximated within a constant factor grater than $1 - 1/e$. These algorithmic challenges led us to design novel algorithms. The solutions developed in this thesis involve a combination of randomized algorithms, combinatorial optimization, machine learning and game theory to solve these problems. This thesis also opens up multiple directions of future research.

Dynamic Networks: There are many opportunities for studying network design problems for a variety of structural properties of networks or network processes. For instance, an interesting future direction in optimizing centrality is the dynamic version of the problem [185, 186, 187], where coverage centrality has to be maintained under temporal, and possibly adversarial, edge updates. This problem has interesting connections with existing work on *Game Theory* [188].

Applications of Network Design: We have seen many applications of network design for different objectives. Design would be useful for plenty of other problems such as opinion formation: How to optimize the dynamics of opinion formation via design? A major challenge is to measure the effect of the modifications in well-known models (e.g. DeGroot-Friedkin). Design is also relevant in other applications such as telecommunication failure and disaster management: How to improve the reliability of telecommunication networks? How to minimize the disruption of emergency services by a natural disaster?

Covert Networks: An important future direction is solving security problems in covert networks. For instance, the average case computational complexity of the Hiding Leader problem for popular network centrality measures is a challenging problem. Since the results of Waniek et al. [58] and ours establish that the Hiding Leader problem is intractable only in the worst case, it could very well be possible that there exist heuristics that efficiently solve most randomly generated instances. If this is true, then the apparent complexity shield against manipulating various centrality measures will become substantially weak. Another immediate future work is to resolve the computational complexity of the Hiding Leader problem for the core centrality measure when the core centrality of every leader is at most 2.

Sophisticated Design: Network design problems are not well studied under difficult circumstances: How to design assuming a complex model that might be an approximation (estimation errors, missing data) and might have privacy issues involved? For instance, in network design problems, the common assumption is that a central designer has the authority or option to modify the graph optimally. However, the presence of rational users with strategies as in game theoretic settings will require different objectives and techniques.

Bibliography

[1] W. Ellens and R. E. Kooij, *Graph measures and network robustness*, *arXiv preprint arXiv:1311.5064* (2013).

[2] A. Gupta and J. Könemann, *Approximation algorithms for network design: A survey*, *Surveys in Operations Research and Management Science* (2011) 3–20.

[3] Y. Lin and K. Mouratidis, *Best upgrade plans for single and multiple source-destination pairs*, *GeoInformatica* **19** (2015), no. 2 365–404.

[4] E. B. Khalil, B. Dilkina, and L. Song, *Scalable diffusion-aware optimization of network topology*, in *KDD*, pp. 1226–1235, 2014.

[5] C. J. Kuhlman, G. Tuli, S. Swarup, M. V. Marathe, and S. Ravi, *Blocking simple and complex contagion by edge removal*, in *International Conference on Data Mining (ICDM)*, pp. 399–408, IEEE, 2013.

[6] V. Chaoji, S. Ranu, R. Rastogi, and R. Bhatt, *Recommendations to boost content spread in social networks*, in *WWW*, pp. 529–538, 2012.

[7] B. Dilkina, K. J. Lai, and C. P. Gomes, *Upgrading shortest paths in networks*, in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pp. 76–91, Springer, 2011.

[8] B. Wilder, H. C. Ou, K. de la Haye, and M. Tambe, *Optimizing network structure for preventative health*, in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 841–849, 2018.

[9] H. Tong, B. A. Prakash, T. Eliassi-Rad, M. Faloutsos, and C. Faloutsos, *Gelling, and melting, large graphs by edge manipulation*, in *CIKM*, pp. 245–254, 2012.

[10] A. Meyerson and B. Tagiku, *Minimizing average shortest path distances via shortcut edge addition*, in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX-RANDOM)*, pp. 272–285. Springer, 2009.

[11] S. Medya, P. Bogdanov, and A. Singh, *Towards scalable network delay minimization*, in *ICDM*, pp. 1083–1088, 2016.

[12] S. Medya, T. Ma, A. Silva, and A. Singh, *K-core minimization: A game theoretic approach*, *arXiv preprint arXiv:1901.02166* (2019).

[13] S. Medya, A. Silva, A. Singh, P. Basu, and A. Swami, *Group centrality maximization via network design*, in *Proc. 24th SIAM International Conference on Data Mining*, pp. 126–134, SIAM, 2018.

[14] S. Medya, J. Vachery, S. Ranu, and A. Singh, *Noticeable network delay minimization via node upgrades*, *Proceedings of the VLDB Endowment* **11** (2018), no. 9 988–1001.

[15] S. Medya, A. Silva, and A. Singh, *Influence minimization under budget and matroid constraints: Extended version*, *arXiv preprint arXiv:1901.02156* (2019).

[16] D. Paik and S. Sahni, *Network upgrading problems*, *Networks* (1995) 45–58.

[17] M. Papagelis, F. Bonchi, and A. Gionis, *Suggesting ghost edges for a smaller world*, in *International conference on Information and knowledge management (CIKM)*, pp. 2305–2308, 2011.

[18] N. Parotisidis, E. Pitoura, and P. Tsaparas, *Selecting shortcuts for a smaller world*, in *SIAM International Conference on Data Mining (SDM)*, pp. 28–36, SIAM, 2015.

[19] E. D. Demaine and M. Zadimoghaddam, *Minimizing the diameter of a network using shortcut edges*, *in SWAT, ser.Lecture Notes in Computer Science, H. Kaplan,Ed.* (2010) 420–431.

[20] S. Perumal, P. Basu, and Z. Guan, *Minimizing eccentricity in composite networks via constrained edge additions*, in *MILCOM*, pp. 1894–1899, 2013.

[21] N. Parotsidis, E. Pitoura, and P. Tsaparas, *Centrality-aware link recommendations*, in *Proc. 9th International ACM Conference on Web Search and Data Mining*, pp. 503–512, 2016.

[22] U. Brandes, *A faster algorithm for betweenness centrality*, *Journal of mathematical sociology* (2001) 163–177.

[23] M. Riondato and E. M. Kornaropoulos, *Fast approximation of betweenness centrality through sampling*, in *Proc. 7th International ACM Conference on Web Search and Data Mining*, pp. 413–422, 2014.

[24] Y. Yoshida, *Almost linear-time algorithms for adaptive betweenness centrality using hypergraph sketches*, in *Proc. 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1416–1425, 2014.

[25] A. Mahmoody, E. Charalampos, and E. Upfal, *Scalable betweenness centrality maximization via sampling*, in *22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1765–1773, 2016.

[26] P. Crescenzi, G. D'Angelo, L. Severini, and Y. Velaj, *Greedily improving our own centrality in a network*, in *Proc. 14th Symposium on Experimental Algorithms*, pp. 43–55, 2015.

[27] V. Ishakian, D. Erdos, E. Terzi, and A. Bestavros, *A framework for the evaluation and management of network centrality*, in *Proc. SIAM International Conference on Data Mining*, pp. 427–438, 2012.

[28] G. D'Angelo, L. Severini, and Y. Velaj, *On the maximum betweenness improvement problem*, *Electronic Notes in TCS* **322** (2016) 153 – 168.

[29] V. Amelkin and A. K. Singh, *Fighting opinion control in social networks via link recommendation*, in *Proc. of ACM SIGKDD Conference of Knowledge Discovery and Data Mining (KDD19). ACM, Anchorage, AK, US. https://doi.org/10.1145/3292500.3330960*, 2019.

[30] G. DAngelo, M. Olsen, and L. Severini, *Coverage centrality maximization in undirected networks*, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 501–508, 2019.

[31] D. Kempe, J. Kleinberg, and É. Tardos, *Maximizing the spread of influence through a social network*, in *Proc. 9th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 137–146, ACM, 2003.

[32] C. Gao, J. Liu, and N. Zhong, *Network immunization and virus propagation in email networks: experimental evaluation and analysis*, *Knowledge and Information Systems* **27** (2011), no. 2 253–279.

[33] C. M. Schneider, T. Mihaljev, S. Havlin, and H. J. Herrmann, *Suppressing epidemics with a limited amount of immunization units*, *Physical Review E* **84** (2011), no. 6 061911.

[34] M. Kimura, K. Saito, and H. Motoda, *Minimizing the spread of contamination by blocking links in a network.*, in *AAAI*, vol. 8, pp. 1175–1180, 2008.

[35] I. Bogunovic, *Robust protection of networks against cascading phenomena.* PhD thesis, Master Thesis ETH Zürich, 2012.

[36] D. Sheldon, B. Dilkina, A. N. Elmachtoub, R. Finseth, A. Sabharwal, J. Conrad, C. P. Gomes, D. Shmoys, W. Allen, O. Amundsen, *et. al.*, *Maximizing the spread of cascades using network design*, in *UAI*, 2010.

[37] Y. Lin, W. Chen, and J. C. Lui, *Boosting information spread: An algorithmic approach*, in *International Conference on Data Engineering (ICDE)*, pp. 883–894, IEEE, 2017.

[38] C. Budak, D. Agrawal, and A. El Abbadi, *Limiting the spread of misinformation in social networks*, in *Proceedings of the 20th international conference on World wide web*, pp. 665–674, ACM, 2011.

[39] N. P. Nguyen, G. Yan, M. T. Thai, and S. Eidenbenz, *Containment of misinformation spread in online social networks*, in *Proceedings of the 4th Annual ACM Web Science Conference*, pp. 213–222, ACM, 2012.

[40] X. He, G. Song, W. Chen, and Q. Jiang, *Influence blocking maximization in social networks under the competitive linear threshold model*, in *SIAM International Conference on Data Mining (SDM)*, pp. 463–474, SIAM, 2012.

[41] W. Chen, L. V. Lakshmanan, and C. Castillo, *Information and influence propagation in social networks*, *Synthesis Lectures on Data Management* **5** (2013), no. 4 1–177.

[42] S. Medya, P. Bogdanov, and A. Singh, *Making a small world smaller: Path optimization in networks*, *IEEE Transactions on Knowledge and Data Engineering* **30** (2018), no. 8 1533–1546.

[43] S. Medya, L. Cherkasova, G. Magalhaes, K. Ozonat, C. Padmanabha, J. Sarma, and I. Sheikh, *Towards performance and scalability analysis of distributed memory programs on large-scale clusters*, in *Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering*, pp. 113–116, ACM, 2016.

[44] S. Medya, L. Cherkasova, and A. Singh, *Predictive modeling and scalability analysis for large graph analytics*, in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pp. 63–71, IEEE, 2017.

[45] A. Mittal, A. Dhawan, S. Medya, S. Ranu, and A. Singh, *Learning heuristics over large graphs via deep reinforcement learning*, *arXiv preprint arXiv:1903.03332* (2019).

[46] A. Goyal, W. Lu, and L. V. Lakshmanan, *Simpath: An efficient algorithm for influence maximization under the linear threshold model*, in *2011 IEEE 11th international conference on data mining*, pp. 211–220, IEEE, 2011.

[47] K. Jung, W. Heo, and W. Chen, *Irie: Scalable and robust influence maximization in social networks*, in *2012 IEEE 12th International Conference on Data Mining*, pp. 918–923, IEEE, 2012.

[48] A. Arora, S. Galhotra, and S. Ranu, *Debunking the myths of influence maximization: An in-depth benchmarking study*, in *Proceedings of the 2017 ACM International Conference on Management of Data*, pp. 651–666, ACM, 2017.

[49] P. Banerjee, S. Ranu, and S. Raghavan, *Inferring uncertain trajectories from partial observations*, in *International Conference on Data Mining (ICDM)*, pp. 30–39, IEEE, 2014.

[50] P. Banerjee, P. Yawalkar, and S. Ranu, *Mantra: a scalable approach to mining temporally anomalous sub-trajectories*, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1415–1424, ACM, 2016.

[51] N. Garg and S. Ranu, *Route recommendations for idle taxi drivers: Find me the shortest route to a customer!*, in *Proceedings of the 24th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2018.

[52] A. E. Motter, S. A. Myers, M. Anghel, and T. Nishikawa, *Spontaneous synchrony in power-grid networks*, *Nature Physics* **9** (2013), no. 3 191–197.

[53] Z. Li, R. A. Hassan, M. Shahidehpour, S. Bahramirad, and A. Khodaei, *A hierarchical framework for intelligent traffic management in smart cities*, *IEEE Transactions on Smart Grid* **PP** (2017), no. 99 1–1.

[54] D. Schrank, T. Lomax, and B. Eisele, *2015 urban mobility scorecard and appendices*, *Texas A&M Transportation Institute* **39** (August, 2015).

[55] S. Mitra, S. Ranu, V. Kolar, A. Telang, A. Bhattacharya, R. Kokku, and S. Raghavan, *Trajectory aware macro-cell planning for mobile users*, in *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pp. 792–800, IEEE, 2015.

[56] V. Kolar, S. Ranu, A. P. Subramainan, Y. Shrinivasan, A. Telang, R. Kokku, and S. Raghavan, *People in motion: Spatio-temporal analytics on call detail records*, in *Communication Systems and Networks (COMSNETS), 2014 Sixth International Conference on*, pp. 1–4, IEEE, 2014.

[57] S. Krumke, M. Marathe, H. Noltemeier, R. Ravi, and S. Ravi, *Approximation algorithms for certain network improvement problems*, *Journal of Combinatorial Optimization* **2** (1998) 257–288.

[58] M. Waniek, T. P. Michalak, T. Rahwan, and M. Wooldridge, *On the construction of covert networks*, in *Proc. 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS*, pp. 1341–1349, 2017.

[59] D. P. Williamson and D. B. Shmoys, *The design of approximation algorithms*. Cambridge, 2011.

[60] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, *Best algorithms for approximating the maximum of a submodular set function*, *Math. Oper. Res.* (1978) 177–188.

[61] W. Lu, W. Chen, and L. V. Lakshmanan, *From competition to complementarity: comparative influence diffusion and maximization*, *PVLDB* **9** (2015), no. 2 60–71.

[62] S. Asmussen and P. W. Glynn, *Stochastic simulation: algorithms and analysis*, vol. 57. Springer Science & Business Media, 2007.

[63] A. Silva, P. Bogdanov, and A. K. Singh, *Hierarchical in-network attribute compression via importance sampling*, in *International Conference on Data Engineering (ICDE)*, pp. 951–962, IEEE, 2015.

[64] W. Hoeffding, *Probability inequalities for sums of bounded random variables*, *Journal of the American statistical association* **58** (1963), no. 301 13–30.

[65] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, *T-drive: driving directions based on taxi trajectories*, in *Proceedings of the 18th SIGSPATIAL International conference on advances in geographic information systems*, pp. 99–108, ACM, 2010.

[66] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "CRAWDAD dataset epfl/mobility (v. 2009-02-24)." Downloaded from `http://crawdad.org/epfl/mobility/20090224`, Feb., 2009.

[67] D. Donovan, Brian; Work, *New york city taxi trip data (2010-2013)*, 2016.

[68] S. Medya, A. Silva, A. Singh, P. Basu, and A. Swami, *Maximizing coverage centrality via network design: Extended version*, *arXiv preprint arXiv:1702.04082* (2017).

[69] P. Dey and S. Medya, *Covert networks: How hard is it to hide?*, in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 628–637, International Foundation for Autonomous Agents and Multiagent Systems, 2019.

[70] J. Sabater and C. Sierra, *Reputation and social network analysis in multi-agent systems*, in *Proc. 1st International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, pp. 475–482, 2002.

[71] E. Otte and R. Rousseau, *Social network analysis: a powerful strategy, also for the information sciences*, *J. Inf. Sci.* **28** (2002), no. 6 441–453.

[72] F.-Y. Wang, K. M. Carley, D. Zeng, and W. Mao, *Social computing: From social informatics to social intelligence*, *IEEE Intelligent systems* **22** (2007), no. 2.

[73] P. J. Carrington, J. Scott, and S. Wasserman, *Models and methods in social network analysis*, vol. 28. Cambridge university press, 2005.

[74] H. Chen, H. Atabakhsh, J. J. Xu, A. G. Wang, B. Marshall, S. Kaza, L. C. Tseng, S. Eggers, H. Gowda, T. Petersen, *et. al.*, *Coplink center: social network analysis and identity deception detection for law enforcement and homeland security intelligence and security informatics: a crime data mining approach to developing border safe research*, in *Proc. 2005 National Conference on Digital government research*, pp. 112–113, Digital Government Society of North America, 2005.

[75] S. Ressler, *Social network analysis as an approach to combat terrorism: Past, present, and future research*, *Homeland Security Affairs* **2** (2006), no. 2.

[76] J. Xu and H. Chen, *Criminal network analysis and visualization*, *Commun. ACM* **48** (2005), no. 6 100–107.

[77] Y. Lu, X. Luo, M. Polgar, and Y. Cao, *Social network analysis of a criminal hacker community*, *J. Comp. Inf. Sys* **51** (2010), no. 2 31–41.

[78] H. Eiselt, *Destabilization of terrorist networks*, *Chaos, Solitons & Fractals* **108** (2018) 111–118.

[79] E. Farooq, S. A. Khan, and W. H. Butt, *Covert network analysis to detect key players using correlation and social network analysis*, in *Proc. 2nd International Conference on Internet of things and Cloud Computing*, pp. 94:1–94:6, ACM, 2017.

[80] D. Knoke, *Emerging trends in social network analysis of terrorism and counterterrorism*, *Emerging Trends in the Social and Behavioral Sciences: An Interdisciplinary, Searchable, and Linkable Resource* (2015) 1–15.

[81] A. Bavelas, *A mathematical model for group structures*, *Applied anthropology* **7** (1948), no. 3 16–30.

[82] M. E. Shaw, *Group structure and the behavior of individuals in small groups*, *The Journal of psychology* **38** (1954), no. 1 139–149.

[83] M. A. Beauchamp, *An improved index of centrality*, *Behavioral science* **10** (1965), no. 2 161–163.

[84] S. B. Seidman, *Network structure and minimum degree*, *Soc. networks* **5** (1983), no. 3 269–287.

[85] C. Morselli, C. Giguère, and K. Petit, *The efficiency/security trade-off in criminal networks*, *Soc. Networks* **29** (2007), no. 1 143–153.

[86] M. A. Shaikh and W. Jiaxin, *Network structure mining: locating and isolating core members in covert terrorist networks*, *WSEAS Transactions on Information Science and Applications* **5** (2008), no. 6 1011–1020.

[87] B. R. Memon, *Identifying important nodes in weighted covert networks using generalized centrality measures*, in *Proc. European Intelligence and Security Informatics Conference*, pp. 131–140, IEEE, 2012.

[88] M. Kitsak, L. K. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H. E. Stanley, and H. A. Makse, *Identification of influential spreaders in complex networks*, *Nature physics* **6** (2010), no. 11 888.

[89] K. Bhawalkar, J. Kleinberg, K. Lewi, T. Roughgarden, and A. Sharma, *Preventing unraveling in social networks: the anchored k-core problem*, *SIAM J. Discrete Math.* **29** (2015), no. 3 1452–1475.

[90] V. E. Krebs, *Mapping networks of terrorist cells*, *Connections* **24** (2002), no. 3 43–52.

[91] M. K. Sparrow, *The application of network analysis to criminal intelligence: An assessment of the prospects*, *Soc. Networks* **13** (1991), no. 3 251–274.

[92] M. Sageman, *Understanding terror networks.* University of Pennsylvania Press, 2004.

[93] E. Y. Alimi, L. Bosi, and C. Demetriou, *The dynamics of radicalization: a relational and comparative perspective.* Oxford University Press, 2015.

[94] N. Roberts and S. Everton, *Monitoring and disrupting dark networks: A bias toward the center and what it costs us*, in *Eradicating Terrorism from the Middle East*, pp. 29–42. Springer, 2016.

[95] W. E. Baker and R. R. Faulkner, *The social organization of conspiracy: Illegal networks in the heavy electrical equipment industry*, *Am. Sociol. Rev* (1993) 837–860.

[96] N. F. Johnson, M. Zheng, Y. Vorobyeva, A. Gabriel, H. Qi, N. Velásquez, P. Manrique, D. Johnson, E. Restrepo, C. Song, *et. al.*, *New online ecology of adversarial aggregates: Isis and beyond*, *Science* **352** (2016), no. 6292 1459–1463.

[97] R. Stevenson and N. Crossley, *Change in covert social movement networks: The inner circleof the provisional irish republican army*, *Soc. Mov. Stud.* **13** (2014), no. 1 70–91.

[98] N. Crossley, G. Edwards, E. Harries, and R. Stevenson, *Covert social movement networks and the secrecy-efficiency trade off: The case of the uk suffragettes (1906–1914)*, *Soc. Networks* **34** (2012), no. 4 634–644.

[99] D. Calvey, *Covert research: The art, politics and ethics of undercover fieldwork.* Sage, 2017.

[100] S. Atran, R. Axelrod, R. Davis, and B. Fischhoff, *Challenges in researching terrorism from the field*, *Science* **355** (2017), no. 6323 352–354.

[101] P. Csermely, A. London, L.-Y. Wu, and B. Uzzi, *Structure and dynamics of core/periphery networks*, *Journal of Complex Networks* **1** (2013), no. 2 93–123.

[102] K. Von Lampe, *Organized crime: analyzing illegal activities, criminal structures, and extra-legal governance.* Sage Publications, 2015.

[103] A. Bhaskara, M. Charikar, A. Vijayaraghavan, V. Guruswami, and Y. Zhou, *Polynomial integrality gaps for strong SDP relaxations of densest* k*-subgraph*, in *Proc. 23-rd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pp. 388–405, 2012.

[104] M. Waniek, T. P. Michalak, M. J. Wooldridge, and T. Rahwan, *Hiding individuals and communities in a social network*, *Nature Human Behaviour* **2** (2018), no. 2 139.

[105] K. Liu and E. Terzi, *Towards identity anonymization on graphs*, in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 93–106, ACM, 2008.

[106] E. Zheleva and L. Getoor, *To join or not to join: The illusion of privacy in social networks with mixed public and private user profiles*, in *Proceedings of the 18th International Conference on World Wide Web*, pp. 531–540, ACM, 2009.

[107] Y. Altshuler, N. Aharony, Y. Elovici, A. Pentland, and M. Cebrian, *Stealing reality: when criminals become data scientists (or vice versa)*, in *Security and Privacy in Social Networks*, pp. 133–151. Springer, 2013.

[108] J. Kilberg, *A basic model explaining terrorist group organizational structure*, *Studies in Conflict & Terrorism* **35** (2012), no. 11 810–830.

[109] F. Demiroz and N. Kapucu, *Anatomy of a dark network: the case of the turkish ergenekon terrorist organization*, *Trends in organized crime* **15** (2012), no. 4 271–295.

[110] R. Belli, J. D. Freilich, S. M. Chermak, and K. A. Boyd, *Exploring the crime–terror nexus in the united states: a social network analysis of a hezbollah network involved in trade diversion*, *Dynamics of Asymmetric Conflict* **8** (2015), no. 3 263–281.

[111] W. Enders and X. Su, *Rational terrorists and optimal network structure*, *J. Confl. Resolut.* **51** (2007), no. 1 33–57.

[112] R. Janssen and H. Monsuur, *Stable network topologies using the notion of covering*, *Eur J Oper Res.* **218** (2012), no. 3 755–763.

[113] R. Lindelauf, P. Borm, and H. Hamers, *The influence of secrecy on the communication structure of covert networks*, *Soc. Networks* **31** (2009), no. 2 126–137.

[114] P. A. Duijn, V. Kashirin, and P. M. Sloot, *The relative ineffectiveness of criminal network disruption*, *Scientific reports* **4** (2014) 4238.

[115] W. Enders and P. Jindapon, *Network externalities and the structure of terror networks*, *J. Confl. Resolut.* **54** (2010), no. 2 262–280.

[116] J. Goldenberg, B. Libai, and E. Muller, *Using complex systems analysis to advance marketing theory development: Modeling heterogeneity effects on new product growth through stochastic cellular automata*, *J. Acad. Mark. Sci.* **9** (2001), no. 3 1–18.

[117] F. M. Bass, *A new product growth for model consumer durables*, *Manag. Sci.* **15** (1969), no. 5 215–227.

[118] N. Meade and T. Islam, *Modelling and forecasting the diffusion of innovation–a 25-year review*, *Int. J. Forecast* **22** (2006), no. 3 519–545.

[119] J. M. Anthonisse, *The rush in a graph*, *Amsterdam: Mathematische Centrum* (1971).

[120] L. C. Freeman, *A set of measures of centrality based on betweenness*, *Sociometry* (1977) 35–41.

[121] M. Braverman, Y. Kun-Ko, A. Rubinstein, and O. Weinstein, *ETH hardness for densest*-k-*subgraph with perfect completeness*, in *Proc. 28-th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1326–1341, 2017.

[122] M. R. Garey and D. S. Johnson, *Computers and Intractability*, vol. 174. freeman New York, 1979.

[123] D. Moshkovitz, *The projection games conjecture and the np-hardness of ln n-approximating set-cover*, *Theory Comput.* **11** (2015) 221–235.

[124] A.-L. Barabási and R. Albert, *Emergence of scaling in random networks*, *Science* **286** (1999), no. 5439 509–512.

[125] D. J. Watts and S. H. Strogatz, *Collective dynamics of small-worldnetworks*, *Nature* **393** (1998), no. 6684 440.

[126] K. Shin, T. Eliassi-Rad, and C. Faloutsos, *Corescope: Graph mining using k-core analysis—patterns, anomalies and algorithms*, in *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pp. 469–478, IEEE, 2016.

[127] C. Peng, T. G. Kolda, and A. Pinar, *Accelerating community detection by using k-core subgraphs*, *arXiv preprint arXiv:1403.2226* (2014).

[128] Z.-H. You, Y.-K. Lei, L. Zhu, J. Xia, and B. Wang, *Prediction of protein-protein interactions from amino acid sequences with ensemble extreme learning machines and principal component analysis*, in *BMC bioinformatics*, vol. 14, p. S10, BioMed Central, 2013.

[129] J. I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, and A. Vespignani, *Large scale networks fingerprinting and visualization using the k-core decomposition*, in *Advances in neural information processing systems*, pp. 41–50, 2006.

[130] S. Carmi, S. Havlin, S. Kirkpatrick, Y. Shavitt, and E. Shir, *A model of internet topology using k-shell decomposition*, *Proceedings of the National Academy of Sciences* **104** (2007), no. 27 11150–11154.

[131] W. Zhu, C. Chen, X. Wang, and X. Lin, *K-core minimization: An edge manipulation approach*, in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 1667–1670, ACM, 2018.

[132] Z. Xiangyu, L. Feng, Y. Rui, Z. Xuemin, M. Shengwei, Z. Zhen'an, and L. Xiaomeng, *Identification of key transmission lines in power grid using modified k-core decomposition*, in *Electric Power and Energy Conversion Systems (EPECS)*, pp. 1–6, IEEE, 2013.

[133] R. Laishram, A. E. Sariyüce, T. Eliassi-Rad, A. Pinar, and S. Soundarajan, *Measuring and improving the core resilience of networks*, in *Proceedings of the 2018 World Wide Web Conference*, pp. 609–618, 2018.

[134] A. Pedahzur and A. Perliger, *The changing nature of suicide attacks: a social network perspective*, *Social forces* **84** (2006), no. 4 1987–2008.

[135] A. Perliger and A. Pedahzur, *Social network analysis in the study of terrorism and political violence*, *PS: Political Science & Politics* **44** (2011), no. 1.

[136] F. Morone, G. Del Ferraro, and H. A. Makse, *The k-core as a predictor of structural collapse in mutualistic ecosystems*, *Nature Physics* (2018).

[137] M. Burke, C. Marlow, and T. Lento, *Feed me: motivating newcomer contribution in social network sites*, in *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 945–954, ACM, 2009.

[138] R. Farzan, L. A. Dabbish, R. E. Kraut, and T. Postmes, *Increasing commitment to online communities by designing for social presence*, in *Proceedings of the ACM conference on Computer supported cooperative work*, pp. 321–330, ACM, 2011.

[139] D. Garcia, P. Mavrodiev, and F. Schweitzer, *Social resilience in online communities: The autopsy of friendster*, in *Proceedings of the first ACM conference on Online social networks*, pp. 39–50, ACM, 2013.

[140] V. Batagelj and M. Zaveršnik, *Fast algorithms for determining (generalized) core groups in social networks*, *Advances in Data Analysis and Classification* **5** (2011), no. 2 129–145.

[141] L. S. Shapley, *A value for n-person games*, *Contributions to the Theory of Games* **2** (1953), no. 28 307–317.

[142] F. Zhang, Y. Zhang, L. Qin, W. Zhang, and X. Lin, *Finding critical users for social network engagement: The collapsed k-core problem*, in *Thirty-First AAAI Conference on Artificial Intelligence*, pp. 245–251, 2017.

[143] A. Montresor, F. De Pellegrini, and D. Miorandi, *Distributed k-core decomposition*, *IEEE Transactions on parallel and distributed systems* **24** (2013), no. 2 288–300.

[144] A. E. Saríyüce, B. Gedik, G. Jacques-Silva, K.-L. Wu, and Ü. V. Çatalyürek, *Streaming algorithms for k-core decomposition*, *Proceedings of the VLDB Endowment* **6** (2013), no. 6 433–444.

[145] É. Bonnet, V. T. Paschos, and F. Sikora, *Parameterized exact and approximation algorithms for maximum k-set cover and related satisfiability problems*, *RAIRO-Theoretical Informatics and Applications* **50** (2016), no. 3 227–240.

[146] C. Giatsidis, F. Malliaros, D. M. Thilikos, and M. Vazirgiannis, *Corecluster: A degeneracy based graph clustering framework*, in *IAAA: Innovative Applications of Artificial Intelligence*, 2014.

[147] H. Zhang, H. Zhao, W. Cai, J. Liu, and W. Zhou, *Using the k-core decomposition to analyze the static structure of large-scale software systems*, *The Journal of Supercomputing* **53** (2010), no. 2 352–369.

[148] R. H. Chitnis, F. V. Fomin, and P. A. Golovach, *Preventing unraveling in social networks gets harder*, in *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[149] F. D. Malliaros and M. Vazirgiannis, *To stay or not to stay: modeling engagement dynamics in social graphs*, in *ACM international conference on Information & Knowledge Management*, pp. 469–478, 2013.

[150] D. S. Callaway, M. E. Newman, S. H. Strogatz, and D. J. Watts, *Network robustness and fragility: Percolation on random graphs*, *Physical review letters* **85** (2000), no. 25 5468.

[151] R. Albert, I. Albert, and G. L. Nakarado, *Structural vulnerability of the north american power grid*, *Physical review E* **69** (2004), no. 2 025103.

[152] R. Cohen, K. Erez, D. Ben-Avraham, and S. Havlin, *Resilience of the internet to random breakdowns*, *Physical review letters* **85** (2000), no. 21 4626.

[153] W. Najjar and J.-L. Gaudiot, *Network resilience: A measure of network fault tolerance*, *IEEE Transactions on Computers* (1990), no. 2 174–181.

[154] P. Smith, D. Hutchison, J. P. Sterbenz, M. Schöller, A. Fessi, M. Karaliopoulos, C. Lac, and B. Plattner, *Network resilience: a systematic approach*, *IEEE Communications Magazine* **49** (2011), no. 7.

[155] F. D. Malliaros, V. Megalooikonomou, and C. Faloutsos, *Fast robustness estimation in large social graphs: Communities and anomaly detection*, in *SIAM International Conference on Data Mining*, pp. 942–953, 2012.

[156] A. Adiga and A. K. S. Vullikanti, *How robust is the core of a network?*, in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 541–556, Springer, 2013.

[157] R. Narayanam and Y. Narahari, *A shapley value-based approach to discover influential nodes in social networks*, *IEEE Transactions on Automation Science and Engineering* **8** (2011), no. 1 130–147.

[158] R. Kohli, R. Krishnamurti, and P. Mirchandani, *The minimum satisfiability problem*, *SIAM Journal on Discrete Mathematics* (1994) 275–283.

[159] O. Goldschmidt, D. Nehme, and G. Yu, *Note: On the set-union knapsack problem*, *Naval Research Logistics (NRL)* **41** (1994), no. 6 833–842.

[160] A. Arulselvan, *A note on the set union knapsack problem*, *Discrete Applied Mathematics* **169** (2014) 214–218.

[161] R. Balasubramanian, M. R. Fellows, and V. Raman, *An improved fixed-parameter algorithm for vertex cover*, *Information Processing Letters* **65** (1998), no. 3 163–168.

[162] J. Flum and M. Grohe, *Parameterized complexity theory*. Springer Science & Business Media, 2006.

[163] A. Agarwal, M. Dahleh, and T. Sarkar, *A marketplace for data: An algorithmic solution*, *arXiv preprint arXiv:1805.08125* (2018).

[164] J. Castro, D. Gómez, and J. Tejada, *Polynomial calculation of the shapley value based on sampling*, *Computers & Operations Research* **36** (2009), no. 5 1726–1730.

[165] S. Maleki, L. Tran-Thanh, G. Hines, T. Rahwan, and A. Rogers, *Bounding the estimation error of sampling-based shapley value approximation*, *arXiv preprint arXiv:1306.4265* (2013).

[166] F. Moser, R. Colak, A. Rafiey, and M. Ester, *Mining cohesive patterns from graphs with feature vectors*, in *SIAM International Conference on Data Mining*, pp. 593–604, SIAM, 2009.

[167] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, *On the evolution of user interaction in facebook*, in *Proceedings of the 2nd ACM workshop on Online social networks*, pp. 37–42, ACM, 2009.

[168] S. Wuchty and E. Almaas, *Peeling the yeast protein network*, *Proteomics* **5** (2005), no. 2 444–449.

[169] M. Zitnik, R. Sosic, M. W. Feldman, and J. Leskovec, *Evolution of resilience in protein interactomes across the tree of life*, *bioRxiv* (2019).

[170] A. Goyal, F. Bonchi, and L. V. Lakshmanan, *A data-based approach to social influence maximization*, *Proceedings of the VLDB Endowment* **5** (2011), no. 1 73–84.

[171] H. Aziz, S. Bouveret, I. Caragiannis, I. Giagkousi, and J. Lang, *Knowledge, fairness, and social constraints.*, in *AAAI*, 2018.

[172] C. Chekuri and A. Kumar, *Maximum coverage problem with group budget constraints and applications*, in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pp. 72–83. Springer, 2004.

[173] J. Vondrák, *Optimal approximation for the submodular welfare problem in the value oracle model*, in *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 67–74, 2008.

[174] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, *Maximizing a monotone submodular function subject to a matroid constraint*, *SIAM Journal on Computing* **40** (2011), no. 6 1740–1766.

[175] C. Chekuri, J. Vondrak, and R. Zenklusen, *Dependent randomized rounding via exchange properties of combinatorial structures*, in *Foundations of Computer Science (FOCS)*, pp. 575–584, IEEE, 2010.

[176] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, *Cost-effective outbreak detection in networks*, in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 420–429, ACM, 2007.

[177] A. Goyal, F. Bonchi, and L. V. Lakshmanan, *Learning influence probabilities in social networks*, in *International conference on Web search and data mining (WSDM)*, pp. 241–250, ACM, 2010.

[178] J. Vondrák, *Submodularity and curvature: The optimal algorithm*, .

[179] J. Nocedal and S. J. Wright, *Numerical optimization (second edition)*, 2006.

[180] A. Bozorgi, S. Samet, J. Kwisthout, and T. Wareham, *Community-based influence maximization in social networks under a competitive linear threshold model*, *Knowledge-Based Systems* **134** (2017) 149–158.

[181] A. Tsang, B. Wilder, E. Rice, M. Tambe, and Y. Zick, *Group-fairness in influence maximization*, in *IJCAI*, 2019.

[182] M. Lake, *A new campaign resource allocation model*, in *Applied Game Theory*, pp. 118–132. Springer, 1979.

[183] A. Yadav, B. Wilder, E. Rice, R. Petering, J. Craddock, A. Yoshioka-Maxwell, M. Hemler, L. Onasch-Vera, M. Tambe, and D. Woo, *Bridging the gap between theory and practice in influence maximization: Raising awareness about hiv among homeless youth.*, in *IJCAI*, pp. 5399–5403, 2018.

[184] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, *Measurement and Analysis of Online Social Networks*, in *Proceedings of the 5th ACM/Usenix Internet Measurement Conference (IMC'07)*, 2007.

[185] T. Hayashi, T. Akiba, and Y. Yoshida, *Fully dynamic betweenness centrality maintenance on massive networks*, *Proceedings of the VLDB Endowment* **9** (2015), no. 2 48–59.

[186] K. Lerman, R. Ghosh, and J. H. Kang, *Centrality metric for dynamic networks*, in *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*, pp. 70–77, ACM, 2010.

[187] T. Takaguchi, Y. Yano, and Y. Yoshida, *Coverage centralities for temporal networks*, *The European Physical Journal B* **89** (2016), no. 2 1–11.

[188] E. N. Ciftcioglu, S. Pal, K. S. Chan, D. H. Cansever, A. Swami, A. Singh, and P. Basu, *Topology design under adversarial dynamics*, in *WiOpt*, pp. 1–8, 2016.