



University of  
Massachusetts  
Amherst

## COMBINATORIAL ALGORITHMS FOR GRAPH DISCOVERY AND EXPERIMENTAL DESIGN

|               |   |
|---------------|---|
| Item Type     | Dissertation (Open Access)  |
| Authors       | Addanki, Raghavendra K  |
| DOI           | <a href="https://doi.org/10.7275/31057392">10.7275/31057392</a>                                       |
| Rights        | Attribution 4.0 International   |
| Download date | 2026-03-07 03:33:40   |
| Item License  | <a href="http://creativecommons.org/licenses/by/4.0/">http://creativecommons.org/licenses/by/4.0/</a> |
| Link to Item  | <a href="https://hdl.handle.net/20.500.14394/18932">https://hdl.handle.net/20.500.14394/18932</a>     |

# COMBINATORIAL ALGORITHMS FOR GRAPH DISCOVERY AND EXPERIMENTAL DESIGN

A Dissertation Presented

by

RAGHAVENDRA KIRAN ADDANKI

Submitted to the Graduate School of the  
University of Massachusetts Amherst in partial fulfillment  
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2022

Manning College of Information  
and Computer Sciences

© Copyright by Raghavendra Kiran Addanki 2022

All Rights Reserved

# COMBINATORIAL ALGORITHMS FOR GRAPH DISCOVERY AND EXPERIMENTAL DESIGN

A Dissertation Presented

by

RAGHAVENDRA KIRAN ADDANKI

Approved as to style and content by:

---

Andrew McGregor, Co-chair

---

Cameron Musco, Co-chair

---

Hung Le, Member

---

Daniel Sheldon, Member

---

Shiva Prasad Kasiviswanathan, Member

---

James Allan, Chair of the Faculty  
Manning College of Information  
and Computer Sciences

*This thesis is dedicated to my parents:*

Nadimpalli Padmaja & Addanki Venkata Nageswara Rao

## ACKNOWLEDGMENTS

As I reflect on my stay in grad school, I am humbled by the immense support network of people that helped me get through it. I would like to take the opportunity to thank them, as it is needless to say, without them, this thesis would not have been possible.

I had a mixed experience during my grad school. Until my third year, my progress was slow, and my then advisor showed little interest in working with me, and I was low on confidence. Things changed dramatically in my third year when I left for California for a summer internship at Amazon, where I was introduced to the elegant field of causality by Shiva Kasiviswanathan. After identifying an exciting problem and making good progress, I returned to Amherst to find myself without an advisor. In a moment of sheer courage (and desperation), I decided to jump-start my Ph.D. by asking Cameron Musco, who joined a month ago, and Andrew McGregor to co-advise me. To my pleasant surprise, both of them agreed to it. I cannot overstate the importance of their influence on me – through patience and a lot of kindness – they taught me the art of identifying research problems, writing a research paper, communicating the ideas effectively, and, most importantly, encouraging me to have fun with research. Cameron, coming fresh out of grad school, has been an ideal mentor to look up to: through the daily grind, he showed that being brave, curious, and kind are the only things needed to succeed; Andrew, with his sage wisdom, has a remarkable attitude towards collaboration and open communication, which immensely helped me with my confidence. I will always cherish our memories during various zoom meetings or my occasional drop-in meetings. Through my graduate study, I have benefited immensely from the mentorship of two extraordinary human

beings who expected nothing in return for their hard work. I sincerely hope I put these lessons to use in the future.

I am also extremely grateful to Shiva, who provided great timely opportunities and advice, first as an intern mentor and then helping facilitate my stay at the Simons Institute in my last semester. I want to thank my other thesis committee members, Dan Sheldon and Hung Le, for their valuable feedback during the early stages, which helped improve the presentation in the introduction chapter. I would also like to thank all my thesis committee members for making my defense a memorable in-person experience.

It has been a great privilege and experience to learn from many of my co-authors working on various research papers we wrote together. For tolerating me, I would particularly like to thank Cameron, Andrew, Sainyam, Sarwar, Iro, Anup, Tung, David, Sutanay, and Khushbu. The last semester was particularly stressful because of my job search, and I am very thankful to GV, Sainyam, Sandhya, Raj, Rik, and Sabby for going over my research statements and job talks and for always looking out for me. It goes without saying that the next time we meet, the coffee will be on me!

From my early years of grad school, through the pandemic, and to the end of my stay in Amherst, I have had a great support network of friends who helped me stay grounded and sane. More importantly, they provided much-needed fun, warmth, and meaning, particularly when the world was thrown into chaos because of the pandemic. In particular, I would like to thank Sabby, Satya, Sneha, Niketh, Sarwar, Raj, and Rajvi, for all the hikes, games, and camaraderie during the trying times. I would like to thank Sabby for being a great friend and a sounding board when I needed it the most; Satya, Sainyam, and Raj, who were there from the beginning to the end through all the peaks and pitfalls; Michael Scott (Steve Carrell from *The Office*) for helping me get through many paper rejections.

I would like to thank Sainyam, Suhas, Desai, Sabby, and Satya, for the road trips, for teaching me how to cook, and for being great roommates; Utkarsh, Rishi, Satya, Neha, and Nidhi, for providing fun-filled blissful initial years; Manish, Nishant, Sandhya, Virat, Mohit, Karine, Tomas, Divyendra, and Bhanu for great food and conversations over the years. Every so often, I could peek into an alternate world, different from grad school, providing me with reality checks whenever I met with my undergraduate friends, Sasi, Kalyan, Abhinav, and Chandan. Hopefully, our paths will cross again sometime in the future!

I am also very grateful for our tightly knit theory group and Rik, whose enthusiasm to convert any social activity into a board game night made it even more enjoyable. The recent additions have made our group hopeful, inspiring, and cheerful. I had a wonderful experience spending my last semester at the Simons Institute at Berkeley. I met some of the most inspiring individuals, including Siddharth, Chris, Kiran, and Poonam, and I had great fun filling all the white and green boards with a lot of math.

I owe a lot to the theory faculty at my undergraduate alma mater, IIT Madras, for their excellent courses. A reluctant but fateful decision to enroll in a course titled *Algorithmic Algebra* in my fourth year, taught brilliantly by Prof B.V. Raghavendra Rao, changed the course of my study – I found it challenging and a lot of fun. After finishing the course, there was no looking back, and I decided to go to grad school. I want to thank Prof. N.S. Narayanaswamy, who was my thesis advisor, for patiently listening to my ideas and always having an infectious enthusiasm encouraging me to pursue my dreams.

Although I am reaping the fruits of my labor now, the seeds were sown long back by my parents. Their relentless passion and extraordinary sacrifices to ensure that my brother and I have access to good education have been instrumental in any and all the success that we had. Any words are insufficient to express my gratitude towards my mom, who taught me to be proactive and kind; my dad, who taught me to be

patient and persistent; my younger brother, who inspires me with his achievements and advice.

## ABSTRACT

# COMBINATORIAL ALGORITHMS FOR GRAPH DISCOVERY AND EXPERIMENTAL DESIGN

SEPTEMBER 2022

RAGHAVENDRA KIRAN ADDANKI

B.Tech., INDIAN INSTITUTE OF TECHNOLOGY MADRAS

M.Tech., INDIAN INSTITUTE OF TECHNOLOGY MADRAS

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Andrew McGregor and Professor Cameron Musco

In this thesis, we study the design and analysis of algorithms for discovering the structure and properties of an unknown graph, with applications in two different domains: causal inference and sublinear graph algorithms. In both these domains, graph discovery is possible using restricted forms of *experiments*, and our objective is to design low-cost experiments.

First, we describe efficient experimental approaches to the *causal discovery* problem, which in its simplest form, asks us to identify the causal relations (edges of the unknown graph) between variables (vertices of the unknown graph) of a given system. For causal discovery, we study algorithms for the problem of learning the causal relationships between a set of observed variables in the presence of hidden or unobserved variables while minimizing a suitable cost of interventions on the observed

variables. An intervention on a set of variables helps learn the presence of causal relations adjacent to them. Under various cost models for interventions, we design combinatorial algorithms for causal discovery by identifying new connections between discrete optimization, graph property testing, and efficient intervention design.

Next, we investigate query-efficient experimental approaches for estimating various graph properties, such as the number of edges and graph connectivity. The access to the graph, or equivalently performing an experiment, is via a Bipartite Independent Set (BIS) oracle. The BIS oracle is related to the interventional access model used in our work for causal graph discovery, with other applications in group testing and fine-grained complexity. In this setting, we develop non-adaptive algorithms that lead to efficient implementations in highly parallelized and low-memory streaming settings.

# CONTENTS

|   | Page      |
|---|-----------|
| <b>ACKNOWLEDGMENTS</b> .....  | <b>v</b>  |
| <b>ABSTRACT</b> .....   | <b>ix</b> |
| <b>LIST OF FIGURES</b> .....  | <b>xv</b> |
| <br>  |           |
| <b>CHAPTER</b>  |           |
| <b>1. INTRODUCTION</b> .....  | <b>1</b>  |
| 1.0.1 Causal Discovery .....  | 1         |
| 1.0.2 Sublinear Query Algorithms .....                              | 6         |
| 1.1 Summary of Contributions .....                                  | 8         |
| 1.2 Overview of Results .....                                       | 10        |
| 1.2.1 Identity Cost Model .....                                     | 10        |
| 1.2.2 Linear Cost Model .....                                       | 11        |
| 1.2.2.1 Ancestral Graph Recovery .....                              | 12        |
| 1.2.2.2 Observable Graph Recovery .....                             | 12        |
| 1.2.3 Collaborative Causal Discovery .....                          | 15        |
| 1.2.4 Non-adaptive algorithms for counting and sampling edges ..... | 17        |
| 1.3 Bibliographic Notes .....                                       | 18        |
| <br>  |           |
| <b>2. CAUSAL DISCOVERY UNDER IDENTITY COST MODEL</b> .....          | <b>20</b> |
| 2.1 Preliminaries .....   | 20        |
| 2.2 Causal Discovery .....  | 25        |
| 2.2.1 Recovering the Ancestral Graph .....                          | 26        |
| 2.2.2 Recovering the Observable Graph .....                         | 27        |
| 2.2.3 Detecting the Latents .....                                   | 29        |

|           |  |           |
|-----------|--|-----------|
| 2.3       | Experimental Evaluation  | 32        |
| 2.4       | Additional Proof Details from Section 2.2  | 35        |
| 2.4.1     | Recovering the entire causal graph $\mathcal{G}$   | 35        |
| 2.4.1.1   | Recovering the observable graph $G$  | 35        |
| 2.4.1.2   | Latents Affecting Non-adjacent Nodes in $G$ .  | 40        |
| 2.4.1.3   | Latent Affecting Adjacent Nodes in $G$ .   | 42        |
| 2.4.1.4   | Removing the dependence on the exact value of $\tau$ .   | 46        |
| 2.5       | Conclusion   | 47        |
| <b>3.</b> | <b>CAUSAL DISCOVERY UNDER LINEAR COST MODEL</b>  | <b>48</b> |
| 3.1       | Preliminaries  | 49        |
| 3.2       | Ancestral Graph Recovery   | 53        |
| 3.2.1     | 2-approximation Algorithm  | 54        |
| 3.2.2     | $(1 + \epsilon)$ -approximation Algorithm  | 57        |
| 3.3       | Separating Set Systems for Graphs  | 59        |
| 3.4       | Observable Graph Recovery  | 64        |
| 3.4.1     | $(\rho, \gamma, \epsilon)$ – INDEPENDENT-SET   | 67        |
| 3.4.2     | Greedy Algorithm: Guarantees   | 70        |
| 3.5       | Recovering ancestral graph from a supergraph   | 71        |
| 3.6       | Hyperfinite Graphs : Improved Guarantees   | 73        |
| 3.7       | Additional Proof Details   | 78        |
| 3.7.1     | Additional Proof Details from Section 3.2  | 78        |
| 3.7.1.1   | Proof Details from Section 3.2.1   | 78        |
| 3.7.1.2   | Proof Details from Section 3.2.2   | 86        |
| 3.7.2     | Proof Details from Section 3.3   | 94        |
| 3.7.2.1   | $2 \log n$ Approximation Algorithm for Separating Set System   | 94        |
| 3.7.2.2   | Algorithms for $\epsilon$ -(Strongly) Separating Set System when $m \geq 1/\epsilon$                 | 96        |
| 3.7.3     | Proof Details from Section 3.4   | 96        |
| 3.7.4     | Proof Details from Section 3.5   | 105       |
| 3.7.5     | Additional Details for the analysis of 2-approximation result for $\epsilon$ -Separating Set Systems | 109       |

|           |   |            |
|-----------|---|------------|
| 3.8       | Conclusion  | 115        |
| <b>4.</b> | <b>COLLABORATIVE CAUSAL DISCOVERY</b>                         | <b>117</b> |
| 4.1       | Motivation  | 117        |
| 4.1.1     | Our Contributions   | 119        |
| 4.2       | Our Model and Problem Statement                               | 122        |
| 4.3       | Causal Discovery under $(\alpha, \beta)$ -Clustering Property | 127        |
| 4.3.1     | Recovering the Clusters                                       | 128        |
| 4.3.2     | Learning Causal Graphs from $(\alpha, \beta)$ -Clustering     | 131        |
| 4.4       | Causal Discovery under $\alpha$ -Clustering Property          | 133        |
| 4.4.1     | Recovering the Clusters                                       | 133        |
| 4.4.2     | Learning Causal Graphs from $\alpha$ -Clustering              | 135        |
| 4.4.3     | Lower Bound on the Number of Interventions                    | 135        |
| 4.5       | Experimental Evaluation                                       | 136        |
| 4.6       | Additional Details  | 139        |
| 4.6.1     | Maximal Ancestral Graphs                                      | 139        |
| 4.6.2     | Helper Routines   | 142        |
| 4.6.2.1   | Handling Uncertainty in PAG Estimation                        | 145        |
| 4.6.3     | Causal Discovery under $(\alpha, \beta)$ -Clustering          | 147        |
| 4.6.3.1   | Learning Causal Graphs from Clusters                          | 149        |
| 4.6.4     | Discovery under $\alpha$ -Clustering Property                 | 153        |
| 4.6.4.1   | From $\alpha$ -Clustering to Learning Causal Graphs           | 154        |
| 4.6.4.2   | Causal Discovery without Latents                              | 155        |
| 4.6.4.3   | Causal Discovery with Latents: Bounded Degree<br>MAGs         | 159        |
| 4.6.4.4   | Improved Algorithm for Recovering the Clusters                | 161        |
| 4.6.5     | Lower Bound on the Number of Interventions                    | 165        |
| 4.6.6     | Missing details from the Experimental Evaluation              | 171        |
| 4.6.6.1   | Learning MAGs under $(\alpha, \beta)$ -clustering property    | 171        |
| 4.6.6.2   | Learning MAGs under $\alpha$ -clustering property             | 174        |
| 4.7       | Conclusion  | 177        |

|   |            |
|---|------------|
| <b>5. NON-ADAPTIVE EDGE COUNTING AND SAMPLING VIA BIPARTITE INDEPENDENT SET QUERIES</b> ..... | <b>178</b> |
| 5.1 Query Models for Graph Access .....   | 178        |
| 5.2 The Role of Adaptivity .....  | 180        |
| 5.3 Results .....   | 181        |
| 5.3.1 Our Contributions .....   | 181        |
| 5.4 Technical Overview .....  | 183        |
| 5.4.1 Edge Estimation .....   | 183        |
| 5.4.2 Uniform Edge Sampling and Connectivity .....  | 186        |
| 5.5 Preliminaries .....   | 188        |
| 5.6 Non-adaptive algorithm for edge estimation .....  | 189        |
| 5.6.1 Estimating the size of neighborhood .....   | 189        |
| 5.6.1.1 Approximation Guarantees of<br>Algorithm NEIGHBORHOOD-SIZE .....                      | 191        |
| 5.6.2 Finding good approximation for degrees of vertices .....                                | 194        |
| 5.6.2.1 Proof of Lemma 5.6.4 .....  | 195        |
| 5.6.3 Edge Estimation .....   | 198        |
| 5.6.3.1 Overview of Algorithm EDGE-ESTIMATOR .....  | 200        |
| 5.6.3.2 Proof of Theorem 5.6.5 .....  | 203        |
| 5.7 Non-adaptive algorithms for uniform sampling .....  | 216        |
| 5.7.1 Identifying uniform neighbor of a subset of vertices .....                              | 217        |
| 5.7.2 Identifying uniform neighbour for each vertex .....                                     | 218        |
| 5.7.3 Identifying a uniform edge in the graph .....   | 220        |
| 5.7.3.1 Proof of Theorem 5.3.2 .....  | 222        |
| 5.8 Graph Connectivity .....  | 227        |
| 5.9 Conclusion .....  | 229        |
| <b>BIBLIOGRAPHY</b> .....   | <b>231</b> |

## LIST OF FIGURES

| Figure   | Page |
|--|------|
| 1.1 In the causal graph $\mathcal{G}$ shown, $\{v_1, v_2, v_3, v_4\}$ are observable variables and $\ell_1$ is a latent variable. The causal sufficiency assumption does not hold for $\mathcal{G}$ . . . . .  | 3    |
| 1.2 On the left, we have the causal graph $\mathcal{G}$ . An intervention on the variable $v_3$ eliminates the effect of all the incoming edges into $v_3$ , i.e., $\{v_1, v_2, \ell_1\}$ in $\mathcal{G}$ , resulting in a new graph shown on the right. . . . .  | 4    |
| 2.1 An example of a causal graph $\mathcal{G}$ and the corresponding observable graph $G$ . . . . .  | 21   |
| 2.2 $v_k$ is a $p$ -collider for $v_i, v_j$ as it has a path to $v_p$ , a parent of $v_j$ . . . . .  | 24   |
| 2.3 Comparison of $\tau$ vs. maximum degree in various sparse random graph models. On the x-axis is the number of nodes in the graph. Note that our bound on the number of interventions needed to recover $\mathcal{G}$ is better than those provided by Kocaoglu et al. [88] roughly when $\tau < d^2/n$ . . . . . | 34   |
| 4.1 Examples of $M$ causal graphs constructed from Lung Cancer dataset [90]. Here, the causal graphs differ only in the presence of latents (nodes with dotted square box), but they could differ elsewhere too. . . . .   | 118  |

|     |  |     |
|-----|--|-----|
| 4.2 | Two possible diabetes incidence graphs for an individual from [76] differing in the causal edge between <i>Physical Activity</i> and <i>Incidence of Diabetes</i> . The observed variables include: <i>Diet</i> , <i>Body Mass Index (BMI)</i> , <i>Physical Activity</i> , <i>Alcohol (consumption)</i> , <i>Incidence of Diabetes</i> , and the unobserved variable (latent) is <i>Economic Status</i> . The variable <i>Incidence of Diabetes</i> is observable but can't be intervened on, this is not an issue as it has no outgoing edges in the graphs. In this chapter, we assume we do not know the underlying causal graphs or which individuals share the same graph. As intervening on variables such as <i>Diet</i> , <i>BMI</i> might need expensive and careful experimental organization, we ask the following question – given a collection of independent entities (in this diabetes example, they can refer to a collection of people), can we collaboratively learn each entity's causal graphs while minimizing the number of interventions per entity? . . . . . | 120 |
| 4.3 | MAGs with $(\alpha = 0.75, \beta = 0.5)$ -clustering. Every pair of graphs in $C_1^*$ and $C_2^*$ differ in at least $3(= 0.75 \times 4)$ nodes, while pairs of graphs within clusters differ by at most $2(= 0.5 \times 4)$ nodes. . . . .  | 125 |
| 4.4 | Different DAGs with same MAG. It is easy to observe that, no single vertex interventions can differentiate $\mathcal{D}_1$ from $\mathcal{D}_2$ . . . . .  | 141 |
| 4.5 | An example of MAGs $\mathcal{M}_1$ and $\mathcal{M}_2$ with large distance $d(\mathcal{M}_1, \mathcal{M}_2)$ but generating the same PAG. . . . .  | 141 |
| 4.6 | The MAGs used in the proof of Proposition 4.6.7. . . . .   | 144 |
| 4.7 | Dominant MAGs of the causal network <i>Earthquake</i> constructed using the described procedure. . . . .   | 172 |
| 4.8 | Sample size vs. maximum number of interventions per entity used by Algorithm $(\alpha, \beta)$ -BOUNDEDDEGREE. . . . .   | 173 |
| 4.9 | Sample size vs. maximum number of interventions per entity used by Algorithm $\alpha$ -BOUNDEDDEGREE. . . . .  | 173 |

# CHAPTER 1

## INTRODUCTION

In this thesis, we study the design and analysis of algorithms for discovering the structure and properties of an unknown graph, with applications in two different domains: causal inference and sublinear graph algorithms. In both these domains, graph discovery is possible using restricted forms of *experiments* and our objective is to design low cost experiments. First, we describe efficient experimental approaches to the *causal discovery* problem, which in its simplest form, asks us to identify the causal relations (edges of the unknown graph) between variables (vertices of the unknown graph) of a given system. Next, we investigate query efficient experimental approaches for estimating various graph properties, such as, the number of edges and graph connectivity. The access to the graph, or equivalently performing an experiment, is via certain types of queries that are inspired by the group testing and sublinear time algorithms literature.

### 1.0.1 Causal Discovery

Causality has long been a key tool in studying and analyzing various scientific disciplines, such as genetics, psychology, and economics [104]. With data proliferation, there is an increasing demand to build automated systems that rely on machine learning based approaches. Many, if not most, of these approaches are largely based on learning complicated functional representations in high dimensional spaces of the observed data, which are then used for predictions. Such approaches do not necessarily help us answer the *counterfactual questions*. Answers to counterfactual questions

help us reason about the behaviour of a particular outcome under various changes to different sets of features, some of which might not necessarily be observed in the dataset. Such questions are important when a decision making process relies on historical data, e.g., an algorithm deciding whether a loan should be granted by a bank, or implementing an education policy by a government body.

To provide a firm footing for studying such problems, we employ the Structural Causal Model (SCM) framework of causality, pioneered by Pearl [104]. Under this framework, any system is represented by a graphical diagram, encoding the directed causal relations between variables of the system, often represented using a directed acyclic graph, and a system of functional equations that define the interactions between the variables. The SCM framework allows for a control over data generating process, i.e., it allows for fixing the values of the variables, or changing the functional equations associated with a variable. Using this setup, we can systematically derive answers for *what-if* or counterfactual causal questions, capturing settings not observed directly.

In this thesis, we study one of the fundamental problems in the theory of causality: *causal discovery*. In causal discovery, the goal is to identify all the causal relations existing between variables in a given system.

**Causal Sufficiency** Broadly speaking, the problem of causal discovery has been studied based on an assumption on the types of variables, called the *causal sufficiency* assumption. The variables in any system can be classified into two types: observable or endogenous variables whose values can be observed and recorded, and latent or exogenous variables whose values cannot be observed or measured. The causal sufficiency assumption stipulates that there are no latent variables in the system. Figure 1.1 illustrates it with an example.

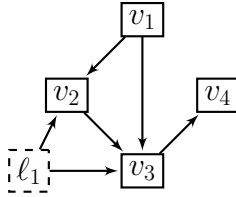


Figure 1.1: In the causal graph  $\mathcal{G}$  shown,  $\{v_1, v_2, v_3, v_4\}$  are observable variables and  $\ell_1$  is a latent variable. The causal sufficiency assumption does not hold for  $\mathcal{G}$ .

Under the causal sufficiency assumption, a culmination of a long series of works resulted in the the well-known *IC* [104] and *PC* [119] algorithms, which identify a causal relation using the *observational data*, i.e., samples drawn from joint distribution on observable variables. These approaches are called *constraint-based* approaches, as they identify causal relations by making use of the constraints imposed by conditional independences in the causal graph. When a pair of variables are independent conditioned on a subset of variables, it results in a conditional independence, which also imposes a restriction on the structure of the underlying causal graph, described using the notion of *d*-separation [105]. Such conditional independences can be identified via conditional independence (CI) tests on the observational data. Much work has focused on understanding the limitations and assumptions underlying these algorithms [66, 68, 70, 92, 116]. There are other well-known *score-based* Bayesian approaches, e.g., Greedy Equivalence Search (GES) [41] that rank the set of all the possible causal graphs based on a score function and return the graph with the optimum score, consistent with the given data. Both score-based and constraint-based approaches recover the same graph, asymptotically. Apart from these, approaches based on functional assumptions in the SCM, such as, LiNGAM [116], are also widely studied. We refer the reader to the survey by Glymour et al. [61] for a comprehensive survey of many causal discovery methods studied under the causal sufficiency assumption.

Under the causal sufficiency assumption, PC or any other constraint or score-based algorithms do not recover all causal relations, but recover only a partial graph, containing both undirected and directed edges. The undirected subgraph of the partial graph is called the *Essential graph*. The Essential graph captures all causal relations that cannot be recovered using CI tests, even with unbounded computational and statistical resources. The set of all directed causal graphs that have the same Essential graph are said to form a *Markov equivalence class*. It is well-known, that to disambiguate a causal graph from its Markov equivalence class, i.e., recover the directions in the Essential graph, interventional data, rather than just observational data is required [50, 51, 65].

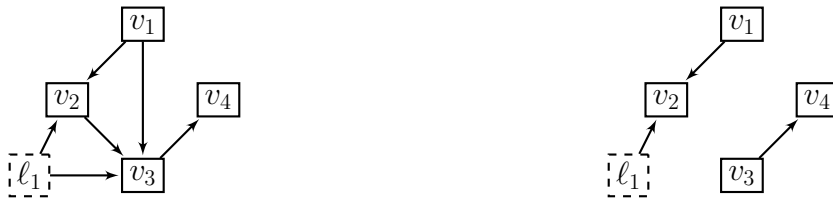


Figure 1.2: On the left, we have the causal graph  $\mathcal{G}$ . An intervention on the variable  $v_3$  eliminates the effect of all the incoming edges into  $v_3$ , i.e.,  $\{v_1, v_2, \ell_1\}$  in  $\mathcal{G}$ , resulting in a new graph shown on the right.

**Intervention** As defined in the SCM framework, an *intervention* (or experiment) requires us to fix a subset of variables to a set of values, inducing a *new distribution* on the free variables. We can observe that any intervention on a set of variables involves a system manipulation and eventually results in a new distribution. Equivalently, when such an intervention is performed, the effect of all the incoming edges in the causal graph (also called parents) is eliminated completely. Figure 1.2 illustrates the definition with an example graph. Such interventions are also called hard interventions, and other types of interventions studied widely include soft and parametric interventions [50]. In a soft (or a parametric) intervention, the effect of incoming edges on intervened set of nodes is not fully eliminated and is specified by a new

distribution [124]. Causal discovery under soft interventions has received attention recently [75, 87, 130]. In this thesis, we focus only on hard interventions. As it is generally expensive to generate new samples from interventional distributions, one of the major goals of causal discovery is to minimize the cost associated with interventions.

In many cases, causal sufficiency is too strong an assumption: it is often contested if the behavior of systems we observe can truly be attributed to measured variables [21, 104]. In light of this, many algorithms avoiding the causal sufficiency assumption, such as IC\* [127] and FCI [119], that only use observational data, have been developed. However, these algorithms only recover a partial directed graph, called the *Partial Ancestral Graph* (PAG), that encodes path relations in the underlying causal graph. Under causal sufficiency, an undirected edge in the Essential graph represents a causal relation. However, when there are latents, the edges of PAG are less informative, as they encode weaker path relations, and do not always represent causal relations. This makes causal discovery highly challenging, as we have an additional step of identifying the causal relations adjacent to every variable. There is a growing interest in constructing efficient intervention designs in this setting [72, 102, 118].

**Cost Models** Interventions are expensive to perform, and several cost models have been proposed in the literature addressing various common scenarios. In a general cost model, intervening on any subset of variables has a cost associated with it, and the goal is to identify all causal relationships and their directions while minimizing the total cost of interventions. This captures the fact that some interventions are more expensive than others. For example, in a medical study, intervening on certain variables might be impractical or unethical. In this thesis, we focusing on recovering the causal graph when there are latents, i.e., without the causal sufficiency assumption, by minimizing the intervention cost using two simplifications of the general cost model.

As an alternative to recovering the entire causal graph, there has been recent interest in recovering the maximum possible number of edges, with a constraint on the total number of interventions (or an upper limit on the cost) [60, 123]. Most of the constraint-based causal discovery approaches, including those developed in this thesis, assume that the conditional independence tests are accurate, which is only possible in the infinite sample limit. Recently, there has been a renewed interest in studying various fundamental problems in causality in the finite sample regime, starting with the work of Acharya et al. [2] and several others [27, 28].

### 1.0.2 Sublinear Query Algorithms

In this thesis, we study the problem of estimating graph properties using a query model (or a test) defined with respect to a graph. The query model is related to the intervention model described in our work on causal discovery.

We study sub-linear query algorithms for estimating the number of edges in a simple, unweighted graph  $G = (V, E)$ , and for sampling uniformly random edges. Access to  $G$  is via a *Bipartite Independent Set (BIS)* oracle [22]. A query to this oracle takes as input two disjoint subsets  $L, R \subseteq V$  and returns

$$BIS(L, R) = \begin{cases} '1' & \text{if there is no edge between } L \text{ and } R \\ '0' & \text{otherwise.} \end{cases}$$

**Local Query Models** Prior work on sub-linear query graph algorithms has largely focused on *local* queries, in particular, (i) vertex degree queries (ii) neighbor queries (output the  $i$ th neighbor of a vertex) and (iii) edge existence queries [54, 63, 114]. In the literature, the first two types of queries form the *adjacency list* query model, while all three types of queries form the *adjacency matrix* query model. Under these models, a variety of graph estimation problems have been well studied, including

edge counting and sampling [53, 63, 114, 126], subgraph counting [10, 31, 52], vertex cover [23, 101], and beyond [110].

**Global Query Models** Motivated by the desire to obtain more query efficient algorithms, Beame et al. [22] studied edge estimation using *global* queries that can make use of information across the graph, including the BIS queries that we will focus on, and the related Independent Set (IS) queries. IS queries were introduced in the literature on query efficient graph recovery [1, 12]. They answer whether or not there exist any edges in the induced subgraph on a subset of nodes  $S \subseteq V$ . We refer the reader to the exposition in [22], which discusses applications of these global query models in group testing [39, 48], computational geometry [13, 33, 56], fine-grained complexity [46, 47], and decision versus counting complexity [47, 111, 121, 122]. Many other variants of global queries have been studied including LINEAR, OR and CUT queries [16, 36, 112]. These queries have been applied to solving maximum matching [84, 100], minimum cut [112], triangle estimation [24, 25, 47], connectivity [16], hitting sets [29], weighted edge estimation [30], hyperedge estimation [47, 26], problems related to linear algebra [107], quantum algorithms [96], and full graph recovery [1, 12].

**The Role of Adaptivity** Notably, for both local and global queries, most sub-linear time graph algorithms are *adaptive*, i.e., a query may depend on the answers to previous queries. In many cases, it is desirable for queries to be *non-adaptive*. This allows them to be completed independently, and might allow for the resulting algorithm to be easily implemented in massively parallel computation frameworks [81]. Non-adaptive algorithms also lead naturally to single-pass, rather than multi-pass, streaming algorithms. In fact, the BIS query model can be seen as a very restricted subset of the more general LINEAR query model, in which each query outputs the inner product of the edge indicator vector with a query vector. This model has long

been studied in the graph-streaming literature [9, 95], in part due to its usefulness in giving single-pass algorithms. However, it has remained open whether non-adaptive algorithms can be given in more restricted global query models.

For these reasons, Assadi et al. [16] and Chakrabarti and Stoeckl [36] have recently sought to reduce query adaptivity under a variety of global query models, including LINEAR, OR, CUT and BIS queries. These works study the *single element recovery* problem, which is a weaker variant of uniform edge sampling, requiring that the algorithm return a single edge in  $G$ . We note that reducing query adaptivity is also a well-studied direction in the closely related literature on group testing [49, 73]. IS and BIS oracles can be thought of as tests if there is a single element in a group of edges, where that group is required to be all edges incident on one node set (IS) or between two disjoint sets (BIS). Attempts to minimize query adaptivity have also been made for sparse recovery [74, 79, 97], sub-modular function maximization [18, 38], property testing [35] and multi-armed bandit learning [8].

In this thesis, we design non-adaptive algorithms for counting and uniform sampling of edges using BIS queries.

## 1.1 Summary of Contributions

In this section, we present a summary of the main contributions of the thesis. First, we start with some useful notation.

**Notation** Let  $\mathcal{G}$  be the (unknown) directed causal graph on both observable variables  $V$  and latent variables  $L$ . A directed edge  $(u, v)$  in  $\mathcal{G}$  indicates a causal relationship from  $u$  to  $v$ . Let  $G$  be the induced subgraph of  $\mathcal{G}$  on the  $|V| = n$  observable variables, also referred to as the *observable graph*.

**Causal Discovery** In recent years, various surprising connections have been discovered between well-studied combinatorial structures in discrete optimization and

intervention designs for causal discovery [71, 82, 94]. Using these connections, much recent work has been devoted to minimizing intervention cost while imposing constraints such as sparsity or cost models [86, 91, 115]. We contribute to this line of work by providing low cost, computationally efficient intervention designs. Firstly, we study causal discovery using the *identity cost model*, in which, every intervention has the same cost, regardless of what variables it contains and therefore minimizing intervention cost is the same as minimizing the number of interventions [88]. Secondly, in the *linear cost model*, where each variable has an intervention cost, and the cost of an intervention on a subset of variables is the sum of costs for each variable in the set [86, 91], our objective is to minimize the total interventional cost. Finally, we describe a new model for causal discovery, called the *Collaborative Causal Discovery*, where the goal is to learn multiple causal graphs with minimum number of unit sized interventions. Here is a summary of our results:

1. **Causal Discovery:** We consider two cost models:

- (a) **Identity Cost Model.** In Chapter 2, we describe algorithms for learning the causal graph  $\mathcal{G}$ , where we try to minimize the number of interventions, i.e., lowest cost in the Identity Cost Model.
- (b) **Linear Cost Model.** In Chapter 3, we describe algorithms that recover ancestral relations, instead of causal relations, with low total cost. Ancestral relations, encoded in the *ancestral graph*, capture path relations, i.e., an edge  $(u, v)$  is present in the ancestral graph iff there is a directed path from  $u$  to  $v$  in  $\mathcal{G}$ .

Under the assumption that we are provided with an undirected causal graph containing all causal relations in  $G$  (the observable graph), we design algorithms that identify the directions using cost efficient algorithms with small relative error approximation. The undirected input graph may

be obtained, e.g., by running algorithms that identify conditional dependencies or consulting a domain expert to identify the presence of causal (undirected) edges.

(c) **Collaborative Causal Discovery.** In Chapter 4, we study a new model model for causal discovery where the goal is to learn multiple causal graphs simultaneously by minimizing the number of unit sized interventions.

2. **Sublinear Query Algorithms.** In Chapter 5, we describe non-adaptive algorithms for counting and sampling edges using bipartite independent set queries. Building upon these ideas, we present improved algorithms for testing whether a graph is connected or not.

## 1.2 Overview of Results

In this section, we present a summary of our results. First, we discuss our contributions for causal discovery under identity and linear cost models. Next, we discuss techniques that overcome limitations of several prior works, under a new causal discovery model, called the Collaborative Causal Discovery. Finally, we discuss our results for estimating various properties of an unknown graph using a query model that is related to interventions in our work on causal discovery.

### 1.2.1 Identity Cost Model

In Chapter 2, for the identity cost model, we present algorithms that recover the causal graph  $\mathcal{G}$  with fewer interventions. First, we present causal graph instances where every non-adaptive algorithm requires  $\Omega(n)$  interventions to recover the observable graph  $G$ . As these graphs are worst case instances, our idea is to parameterize the causal graph in terms of a specific type of variables, that we refer to as *p-colliders*. These variables are a special type of *colliders*, variables that have two incoming edges incident on them in a given directed path of  $\mathcal{G}$ . A node  $v_k$  is *p-collider* for a pair of

nodes  $(v_i, v_j)$  if (i) it is a collider on a path between  $v_i$  and  $v_j$  and (ii) at least one of the parents  $v_i, v_j$  is a descendent of  $v_k$ .

If  $\mathcal{G}$  has at most  $\tau$   $p$ -colliders between every pair of variables (or nodes), then, our algorithm uses at most  $O(\tau \log n)$  interventions for recovering the observable graph  $G$ , improving upon the worst case lower bound of  $\Omega(n)$ . Moreover, our algorithm recovers the entire causal graph  $\mathcal{G}$  using  $O(n\tau \log n + n \log n)$  interventions. The only previous bound on recovering  $\mathcal{G}$  in this setting utilized  $O(\min\{d \log^2 n, \ell\} + d^2 \log n)$  interventions where  $d$  is the maximum (undirected) node degree and  $\ell$  is the length of the longest directed path of the causal graph  $\mathcal{G}$  [88]. Since we use a different parameterization of the causal graph, a direct comparison with this result is not always possible. We argue that a parameterization in terms of  $p$ -colliders is inherently more “natural” as it takes the directions of edges in  $\mathcal{G}$  into account whereas the maximum degree does not. The presence of a single high-degree node can make the number of interventions required extremely high, even if the overall causal graph is sparse. In this case, the notion of  $p$ -colliders is a more global characterization of a causal graph. In Section 2.3 of Chapter 2, we provide a more detailed discussion of different parameter regimes under which our scheme provides a better bound. We also experimentally show that our scheme achieves a better bound over the result due to Kocaoglu et al. [88], in some popular random graph models.

### 1.2.2 Linear Cost Model

Under the linear cost model, we study two settings: (i) we study the ancestral graph recovery, where the goal is to identify all the path relations in  $\mathcal{G}$  (ii) we study the observable graph recovery, under the assumption that a domain expert provides us with an undirected observable graph, and our goal is to recover only their directions.

### 1.2.2.1 Ancestral Graph Recovery

In Chapter 3, under the linear cost model, we present algorithms where the cost of interventions generated by our algorithm is at most twice the cost of the optimum set of interventions, for ancestral graph recovery. Our result is based on a characterization which shows that generating a set of interventions sufficient to recover ancestral relations is equivalent to designing a *strongly separating set system*. We show how to design such a set system with at most twice the optimum cost based on a greedy algorithm that constructs intervention sets that include a variable with high cost in the least number of sets possible.

In the special case when each variable has unit intervention cost, Hyttinen et al. [71] give an exact algorithm to recover ancestral relations in  $G$  with minimal total cost. Their algorithm is based on the Kruskal-Katona theorem in combinatorics [82, 89]. We show that a modification of this approach yields a  $(1 + \epsilon)$ -approximation algorithm in the general linear cost model for any  $0 < \epsilon \leq 1$ , under natural assumptions. To the best of our knowledge, our result is the first to minimize intervention cost under the popular linear cost model in the presence of latents, and without the assumption of unit intervention cost on each variable.

### 1.2.2.2 Observable Graph Recovery

Under the causal sufficiency assumption, for the linear cost model, recovering the causal graph was first studied by Kocaoglu et al. [86]. They showed that causal graph recovery, which translates to recovering the directions of the Essential graph, with optimal minimum cost is NP-hard. To prove the hardness, they showed that causal graph recovery is equivalent to designing a separating set system for the Essential graph. Building on this observation, they presented an approximation algorithm with a multiplicative factor of  $\sim 2$ . As there are no latents, their algorithm makes use of chordality property of the the Essential graph to obtain an efficient approximation al-

gorithm. In the presence of latents, although we presented approximation algorithms for recovering ancestral relations in  $\mathcal{G}$ , there seems to be no known characterization of the optimal intervention sets needed to recover even the observable graph  $G$ , making it hard to design efficient algorithms for recovering  $\mathcal{G}$ .

**Recovering the directions** In Chapter 3, we address these shortcomings in two settings. First, we assume that we are given an undirected graph that contains all causal relations between observable variables, but must identify their directions. There is significant precedent for assumptions on such background knowledge graphs in the literature. For example, Hyttinen et al. [71] and references therein, study intervention design in the same model: a skeleton of possible edges in the causal graph is given via background knowledge, which may come e.g., from domain experts or previous experimental results. In the second setting, we study a relaxation where we are given a supergraph  $H$  of  $G$  containing all causal edges and other additional edges which need not be causal. The second setting is less restrictive, modeling the case where we can ask a domain expert or use observational data to identify a *superset* of possible causal relations. From  $H$  we seek to recover edges of the ancestral graph. Depending on the method by which  $H$  is obtained, it may have special properties that can be leveraged for efficient intervention design. For example, if we use FCI/IC\* [119] to recover a partial ancestral graph from observational data, the remaining undirected edges form a chordal graph [132]. This extends our previous work on ancestral graph recovery where we considered the worst case, when  $H$  is the complete graph. Here, we do not assume anything about how  $H$  is obtained and thus give results holding for general graphs.

**Separating Set Systems** In both settings, we show a connection to separating set systems. Specifically, to solve the recovery problems it is necessary and sufficient to use a set of interventions corresponding to a separating set system when we are given

the undirected observable causal graph  $G$  and a *strongly* separating set system when we are given the supergraph  $H$ . A separating set system is one in which each pair of nodes connected by an edge is separated by at least one intervention – one variable is intervened on and the other is free. A strongly separating set system requires that every connected edge  $(u, v)$  is separated by two interventions – there exists a intervention including  $u$  but not  $v$  and an intervention that includes  $v$  but not  $u$ .

Unfortunately, finding a minimum cost (strongly) separating set system for an arbitrary observable graph  $G$  is NP-hard [91, 71]. We give simple algorithms that achieve  $O(\log n)$  approximation and further argue that, conditioned on the hardness of approximate graph coloring, no polynomial time algorithm can achieve  $o(\log n)$  approximation, where  $n$  is the number of observed variables.

**Bi-criteria approximation** To overcome this limitation, we introduce a *bi-criteria approximation* goal that lets us recover all but  $\epsilon n^2$  edges in the causal or ancestral graph, where  $\epsilon > 0$  is a specified error parameter. For this goal, it suffices to use a relaxed notion of a set system, which we show can be found efficiently using ideas from the graph property testing literature [62]. In the setting where we are given the causal edges in  $G$  and must recover their directions, we give a polynomial time algorithm that finds a set of interventions from which we can recover all but  $\epsilon n^2$  edges with cost at most  $\sim 2$  times the optimal cost for learning the full graph. Similarly, in the setting of ancestral graph recovery, where we are given a super set of edges, we show how to recover all but  $\epsilon n^2$  edges with intervention cost at most  $\sim 4$  times the optimal cost for recovering all edges. Our result significantly extends the applicability of our previous result that gave a 2-relative error approximation to the minimum cost strongly separating set system assuming the worst case when the supergraph  $H$  is a complete graph. The previous approach does not translate to an approximation guarantee better than  $\Omega(\log n)$  for general graphs. Finally, for the special case when

$G$  is a hyperfinite graph [64] with maximum degree  $\Delta$ , we give algorithms that obtain approximation guarantees as above, and recover all but  $\epsilon n \Delta$  edges of  $G$ .

### 1.2.3 Collaborative Causal Discovery

In Chapter 4, we introduce the collaborative causal discovery problem. We assume that we have a collection of  $M$  entities each with their own causal graph. Due to the presence of latent variables, we use a family of mixed graphs known as *maximal ancestral graphs* (MAGs) to model the causal graphs on observed variables.

**Motivation** In a variety of applications, there is no one true causal structure, different entities participating in the application might have different causal structures [58, 106, 76]. In these scenarios, generating a single causal graph by pooling data from these different entities might lead to flawed conclusions [113]. Allowing for interventions, we propose a new model for tackling this problem, referred here as *Collaborative Causal Discovery*, which in its simplest form states that: given a collection of entities, each associated with an individual unknown causal graph and generating their own independent data samples, learn all the causal graphs while minimizing the number of single variable interventions for every entity.

We focus on designing algorithms that have *worst-case* guarantees on the number of unit sized or *atomic* interventions needed to recover (or approximately recover) the MAG of each entity. We assume that there are  $M$  MAGs one for each entity over the same set of  $n$  nodes. From Chapter 2, we know that learning a MAG with atomic interventions, in the worst case requires  $\Omega(n)$  interventions. To facilitate efficient algorithms, we impose a structural assumption on the entities: the set of all the causal graphs (or the entities) can be partitioned into  $k$  clusters such that the causal graphs of any pair of entities belonging to two different clusters are separated by a distance of at least  $\alpha n$ ; entities belonging to same cluster are separated by a distance of at most  $\beta n$  ( $\beta < \alpha$ ), where  $\alpha, \beta$  are constants. The distance between two

causal graphs is defined to be the number of nodes whose direct causal relationships are different. We refer to this clustering of entities as  $(\alpha, \beta)$ -clustering, where  $\alpha$ . A special but important case is when  $\beta = 0$ , in which case all the entities belonging to the same cluster have the same causal MAG, referred to as  $\alpha$ -clustering. Similar assumptions are common for recovering the underlying clusters, in many areas, e.g., crowd-sourcing applications [15, 17]. We show that the worst case  $\Omega(n)$  bound can be substantially reduced if the  $M$  MAGs satisfy the  $(\alpha, \beta)$ -clustering property.

**Guarantees** We first start with the observation that under an  $(\alpha, \beta)$ -clustering, even entities belonging to the same cluster could have a different MAG, which makes exact recovery hard without making a significant number of interventions per entity. If the maximum degree of a MAG is upper bounded by  $\Delta$ , we present an algorithm that uses at most  $O(\Delta \log(M/\delta)/(\alpha - \beta)^2)$  many interventions per entity, with probability at least  $1 - \delta$  (over only the randomness of the algorithm), and recovers an *approximate* MAG for each entity. The approximation is such that for each entity we generate a MAG that is at most  $\beta n$  node-distance from the true MAG of that entity. Here,  $\Delta$  is the maximum undirected degree of the causal MAGs. Our idea is to first recover the underlying clustering of entities by using a randomized set of interventions. Then, we distribute the interventions across the entities in each cluster, thereby, ensuring that the number of interventions per entity is small. By carefully combining the results learnt from these interventions we construct the approximate MAGs. Under the slightly more restrictive  $\alpha$ -clustering assumption, we present algorithms that can *exactly* recover all the MAGs using at most  $\min \{O(\Delta \log(M/\delta)/\alpha), O(\log(M/\delta)/\alpha + k^2)\}$  many interventions per entity. Complementing these upper bounds, we give a lower bound using Yao’s minimax principle [131] that shows for any (randomized or deterministic) algorithm  $\Omega(1/\alpha)$  interventions per entity is required for this causal discovery problem. This implies the  $1/\alpha$  dependence in our upper bound in the  $\alpha$ -clustering case is optimal.

Finally, we provide empirical results on data generated from both real and synthetic networks with added latents and demonstrate the efficacy of our algorithms for learning the underlying clustering and the MAGs.

#### 1.2.4 Non-adaptive algorithms for counting and sampling edges

In Chapter 5, we present algorithms for estimating the number of edges, i.e., *edge estimation* problem and returning a uniform sample from the set of edges of any graph, i.e., *edge sampling* problem, using BIS queries. In particular, we present the first *non-adaptive* algorithm for edge estimation in a graph on  $n$ -nodes, up to  $(1 \pm \epsilon)$  relative error approximation, using  $\text{poly}(1/\epsilon, \log n)$  BIS queries.

**Edge Estimation** We present a non-adaptive algorithm that returns a  $(1 \pm \epsilon)$  relative error approximation to the number of edges, with query complexity  $\tilde{O}(\epsilon^{-5} \log^5 n)$ , where  $\tilde{O}(\cdot)$  hides  $\text{poly}(\log \log n)$  dependencies. Prior methods for  $(1 \pm \epsilon)$  error edge estimation using BIS queries are based on a binary search style approach [22, 47, 26], which is inherently adaptive, and this leads to algorithms requiring  $\Omega(\log^2 n)$  rounds of adaptivity. Beame et al. [22] present a non-adaptive algorithm giving a  $O(\log^2 n)$  approximation factor for bipartite graphs, using  $O(\log^3 n)$  queries. However, no non-adaptive results for general graphs or achieving  $1 \pm \epsilon$  relative error for arbitrary  $\epsilon > 0$  were previously known. We avoid the pitfalls of the previous approaches, by taking a fundamentally different approach, inspired by work on single-pass streaming algorithms. Even with adaptivity, the best known algorithm due to [26] has a query complexity of  $O(\epsilon^{-2} \log^{11} n)$  and succeeds with probability  $1 - 1/n^2$ . Therefore, our non-adaptive result improves upon the current best known algorithms, for constant  $\epsilon$ .

**Edge Sampling** Building on our edge estimation result, we give the first non-adaptive algorithm for outputting a nearly uniformly sampled edge with query com-

plexity  $\tilde{O}(\epsilon^{-6} \log^6 n)$ , improving on the works of Dell et al. [47] and Bhattacharya et al. [26], which require  $\Omega(\log^3 n)$  rounds of adaptivity. Additionally, even ignoring adaptivity, our result improves on the best known algorithm due to Bhattacharya et al. [26] which uses  $O(\epsilon^{-2} \log^{14} n)$  BIS queries, for constant  $\epsilon$ .

**Graph Connectivity** As a consequence of our edge sampling algorithm, we obtain a  $\tilde{O}(n \log^8 n)$  query algorithm for connectivity, using two rounds of adaptivity. This is tight: even in the stronger OR query model (which allows checking the presence of an edge within an arbitrary subset of edges) no non-adaptive algorithm can make  $o(n^2)$  queries. Assadi et al. gave a two-round algorithm in this stronger OR query model. Thus, we close the gap between BIS queries and OR queries for this problem.

### 1.3 Bibliographic Notes

Most of the work presented in the thesis has been previously published or is currently under submission.

1. Chapter 2 and the first half of Chapter 3 have been published as “Efficient intervention design for causal discovery with latents” at the International Conference on Machine Learning, 2020 [5].
2. The latter half of Chapter 3 has been published as “Intervention Efficient Algorithms for Approximate Learning of Causal Graph” at the International Conference on Algorithmic Learning Theory, 2021 [6].
3. Chapter 4 has been published as “Collaborative Causal Discovery with Atomic Interventions” at the Conference on Neural Information Processing Systems, 2021 [4].

4. Chapter 5 has been published as “Non-Adaptive Edge Counting and Sampling via Bipartite Independent Set Queries” at the European Symposium on Algorithms, 2022 [7].

## CHAPTER 2

### CAUSAL DISCOVERY UNDER IDENTITY COST MODEL

In this chapter, we describe algorithms for causal discovery under the identity cost model. In Section 2.1, we present the necessary preliminaries and definitions; in Section 2.2, we describe our algorithms for recovering the causal graph; in Section 2.3, we present an empirical evaluation of our approach; in Section 2.4, we present the necessary proofs of our claims, and in Section 2.5, we present the conclusion.

**Overview** Suppose  $\mathcal{G}$  is the causal graph that we wish to recover. Our approach can be broken down into three steps: first, we construct the ancestral graph  $\text{Anc}(G)$  of  $\mathcal{G}$ , using a set of interventions that satisfy the strongly separating set system property; next, we recover the observable graph  $G$  from  $\text{Anc}(G)$ ; and finally, we identify all the latents  $L$  between the variables in  $V$  to reconstruct  $\mathcal{G}$ . In the last two steps, our main underlying idea is to construct intervention sets such that a special collection of variables, called the  $p$ -colliders, defined for every pair of observable variables, are included in at least one of the intervention sets. As we do not know the graph  $\mathcal{G}$ , we devise randomized strategies that include all the  $p$ -colliders, for every pair of variables, whilst ensuring that we do not create a lot of interventions.

#### 2.1 Preliminaries

**Notation** Following the SCM framework introduced by [105], we represent the set of random variables of interest by  $V \cup L$  where  $V$  represents the set of endogenous (observed) variables that can be measured and  $L$  represents the set of exogenous



Figure 2.1: An example of a causal graph  $\mathcal{G}$  and the corresponding observable graph  $G$ .

(latent) variables that cannot be measured. We define a directed graph on these variables where an edge corresponds to a causal relation between the corresponding variables. The edges are directed with an edge  $(v_i, v_j)$  meaning that  $v_i \rightarrow v_j$ . As is common, we assume that all causal relations that exist between random variables in  $V \cup L$  belong to one of the two categories : (i)  $E \subseteq V \times V$  containing causal relations between the observed variables and (ii)  $E_L \subseteq L \times V$  containing relations of the form  $l \rightarrow v$  where  $l \in L, v \in V$ . Thus, the full edge set of our causal graph is denoted by  $\mathcal{E} = E \cup E_L$ . We also assume that every latent  $l \in L$  influences exactly two observed variables i.e.,  $(l, u), (l, v) \in E_L$  and no other edges are incident on  $l$  following [88]. We let  $\mathcal{G} = \mathcal{G}(V \cup L, \mathcal{E})$  denote the entire causal graph and refer to  $G = G(V, E)$  as the *observable graph*, illustrated in Figure 2.1.

Unless otherwise specified a path between two nodes is a undirected path. For every observable  $v \in V$ , let the parents of  $v$  be defined as  $\text{Pa}(v) = \{w \mid w \in V \text{ and } (w, v) \in E\}$ . For a set of nodes  $S \subseteq V$ ,  $\text{Pa}(S) = \cup_{v \in S} \text{Pa}(v)$ . If  $v_i, v_j \in V$ , we say  $v_j$  is a descendant of  $v_i$  (and  $v_i$  is an ancestor of  $v_j$ ) if there is a directed path from  $v_i$  to  $v_j$ .  $\text{Anc}(v) = \{w \mid w \in V \text{ and } v \text{ is a descendant of } w\}$ . We let  $\text{Anc}(G)$

denote the *ancestral graph*<sup>1</sup> of  $G$  that captures the path relations in  $G$ . Similar to the work of Kocaoglu et al. [88], we define  $\text{Anc}(G)$  as below.

**Definition 2.1.1** (Ancestral Graph  $\text{Anc}(G)$ ). *An edge  $(v_i, v_j) \in \text{Anc}(G)$  iff there is a directed path from  $v_i$  to  $v_j$  in  $G$  (equivalently in  $\mathcal{G}$  due to the semi-Markovian assumption).*

Using Pearl’s do-notation, we represent an intervention on a set of variables  $S \subseteq V$  as  $\text{do}(S = s)$  for a value  $s$  in the domain of  $S$  and the joint probability distribution on  $V \cup L$  conditioned on this intervention by  $\Pr[\cdot \mid \text{do}(S)]$ .

We assume that there exists an oracle that answers queries such as “*Is  $v_i$  independent of  $v_j$  given  $Z$  in the interventional distribution  $\Pr[\cdot \mid \text{do}(S = s)]$ ?*”

**Assumption 2.1.2** (Conditional Independence (CI)-Oracle). *Given any  $v_i, v_j \in V$  and  $Z, S \subseteq V$  we have an oracle that tests whether  $v_i \perp\!\!\!\perp v_j \mid Z, \text{do}(S = s)$ .*

Such conditional independence tests have been widely investigated with sublinear (in domain size) bounds on the sample size needed for implementing this oracle [34, 134]. As is standard in the causality literature, we assume that our causal relationship graph satisfies the *causal Markov condition* and *faithfulness* [119]. We assume that faithfulness holds both in the observational and interventional distributions following [66].

**Semi-Markovian Assumption** Our assumption that each latent only affects two observable variables is commonly known as the semi-Markovian condition and is standard in the literature, e.g., see [125, 88]. In fact, using (pairwise) conditional independence tests, it is impossible to discover latent variables that affect more than two observables, even with unlimited interventions. Consider observables  $x, y, z$  and a

---

<sup>1</sup>We note that the term *ancestral graph* has also been previously used in a different context, see e.g., [108].

latent  $l_{xyz}$  that is a parent of all them. If we test whether  $x, y$ , and  $z$  are all pairwise independent and they all turn out to be false, we can't distinguish the cases where a single latent  $l_{xyz}$  or three separate latents  $l_{xy}, l_{yz}$  and  $l_{xz}$  are causing this non-independence. Thus, we cannot remove the assumption without changing our intervention model or making more restrictive assumptions. As an example, Silva et al. [118] consider the case when latents affect more than two observables, however, they make very strong assumptions – that there are no edges between observables, and each observable has only one latent parent.

**Identity Cost Model** As an intervention on a set of variables requires controlling the variables, and generating a new distribution, we want to use as few interventions as possible. In this cost model, an intervention on any set of observed variables has *unit cost* (no matter how many variables are in the set). We assume that for any intervention, querying the CI-oracle comes free of cost. This model is akin to the model studied in [88].

We recall the notion of  $d$ -separation and introduce  $p$ -colliders that we rely on.

**Definition 2.1.3** ( $d$ -separation). *Given a causal graph  $\mathcal{G}(V \cup L, \mathcal{E})$ , let  $v_i, v_j \in V$  and a set of nodes  $Z \subseteq V$ . We say  $v_i$  and  $v_j$  are  $d$ -separated by  $Z$  if and only if every undirected path  $\pi$  between  $v_i$  and  $v_j$  is blocked by  $Z$ . A path  $\pi$  between  $v_i$  and  $v_j$  is blocked by  $Z$  if at least one of the following holds.*

**Rule 1:**  $\pi$  contains a node  $v_k \in Z$  such that the path  $\pi = v_i \cdots \rightarrow v_k \rightarrow \cdots v_j$  or  $v_i \cdots \leftarrow v_k \leftarrow \cdots v_j$ .

**Rule 2:**  $\pi = v_i \cdots \rightarrow v_k \leftarrow \cdots v_j$  contains a node  $v_k$  and both  $v_k \notin Z$  and no descendant of  $v_k$  is in  $Z$ .

An important observation that relates  $d$ -separation and conditional independence is captured by the following lemma:

**Lemma 2.1.4.** [105] *If  $v_i$  and  $v_j$  are  $d$ -separated by  $Z$ , then  $v_i \perp\!\!\!\perp v_j \mid Z$ .*

**Colliders.** For the path  $\pi = v_i \cdots \rightarrow v_k \leftarrow \cdots v_j$  between  $v_i$  and  $v_j$ ,  $v_k$  is called a *collider* as there are two arrows pointing towards it. We say that  $v_k$  is a collider for the pair  $v_i$  and  $v_j$ , if there exists a path between  $v_i$  and  $v_j$  for which  $v_k$  is a collider. As shown by Rule 2, colliders play an important role in  $d$ -separation. We give a more restrictive definition for colliders that we will rely on henceforth.

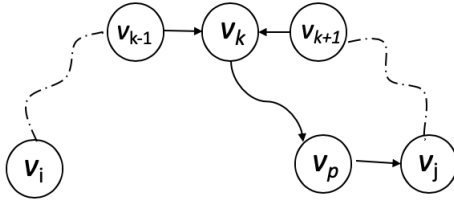


Figure 2.2:  $v_k$  is a  $p$ -collider for  $v_i, v_j$  as it has a path to  $v_p$ , a parent of  $v_j$ .

**Definition 2.1.5 ( $p$ -colliders).** *Given a causal graph  $\mathcal{G}(V \cup L, E \cup E_L)$ . Consider  $v_i, v_j \in V$  and  $v_k \in V$ . We say  $v_k$  is a  $p$ -collider for the pair  $v_i$  and  $v_j$ , if there exists a path  $v_i \cdots \rightarrow v_k \leftarrow \cdots v_j$  in  $\mathcal{G}$  and either  $v_k \in \text{Pa}(v_i) \cup \text{Pa}(v_j)$  or has at least one descendant in  $\text{Pa}(v_i) \cup \text{Pa}(v_j)$ . Let  $P_{ij} \subset V$  denote all the  $p$ -colliders between  $v_i$  and  $v_j$ .*

Intervening on  $p$ -colliders essentially breaks down all the *primitive inducing paths*. Primitive inducing paths are those whose endpoints cannot be separated by any conditioning [108]. Now, between every pair of observable variables, we can define a set of  $p$ -colliders as above. Computing  $P_{ij}$  for the pair of variables  $v_i$  and  $v_j$  explicitly requires the knowledge of  $\mathcal{G}$ , however as we show below we can use randomization to overcome this issue. The following parameterization of a causal graph will be useful in our discussions.

**Definition 2.1.6** ( $\tau$ -causal graph). *A causal graph  $\mathcal{G}(V \cup L, \mathcal{E})$  is a  $\tau$ -causal graph if for every pair of nodes in  $V$  the number of  $p$ -colliders is at most  $\tau$ , i.e.,  $v_i, v_j \in V$  ( $i \neq j$ ), we have  $|P_{ij}| \leq \tau$ .*

Note that every causal graph is at most  $n - 2$ -causal. In practice, we expect  $\tau$  to be significantly smaller. Given a causal graph  $\mathcal{G}$ , it is easy to determine the minimum values of  $\tau$  for which it is  $\tau$ -causal, as checking for  $p$ -colliders is easy. Our algorithm recovers  $\mathcal{G}$  with number of interventions that grow as a function of  $\tau$  and  $n$ .

## 2.2 Causal Discovery

In this section, we consider the identity cost model and present algorithms for recovering the causal graph  $\mathcal{G}$ , with the goal of minimizing the number of interventions. Our algorithm is based on parameterizing the causal graph based on a specific type of collider structure.

We break our approach into multiple steps. First, in section 2.2.1, we discuss the procedure to recover the ancestral graph  $\text{Anc}(G)$  using a strongly separating set system (Definition 2.2.1). Next, the algorithm has two steps: (i) we recover the observable graph  $G$  from  $\text{Anc}(G)$  (section 2.2.2); (ii) after obtaining the observable graph, we identify all the latents  $L$  between the variables in  $V$  to construct  $\mathcal{G}$  (section 2.2.3). In both these steps, an underlying idea is to construct intervention sets with the aim of making sure that all the  $p$ -colliders between every pair of nodes is included in at least one of the intervention sets. As we do not know the graph  $\mathcal{G}$ , we devise randomized strategies to hit all the  $p$ -colliders, whilst ensuring that we do not create a lot of interventions. Missing proofs from the section are collected in Section 2.4.

### 2.2.1 Recovering the Ancestral Graph

For ancestral graph recovery, we will leverage a simple characterization of when a set of interventions  $\mathcal{S} = \{S_1, \dots, S_m\}$  is sufficient to recover  $\text{Anc}(G)$ . In particular,  $\mathcal{S}$  is sufficient if it is a *strongly separating set system* [88].

**Definition 2.2.1** (Strongly Separating Set System). *A collection of subsets  $\mathcal{S} = \{S_1, \dots, S_m\}$  of the ground set  $V$  is a strongly separating set system if for every distinct  $u, v \in V$  there exists  $S_i$  and  $S_j$  such that  $u \in S_i \setminus S_j$  and  $v \in S_j \setminus S_i$ .*

Ancestral graph recovery using a strongly separating set system is simple: we intervene on each of the sets  $S_1, \dots, S_m$ . Using CI-tests we can identify for every pair of  $v_i$  and  $v_j$ , if there is a path from  $v_i$  to  $v_j$  or not in  $G$  using the intervention corresponding to  $S \in \mathcal{S}$  with  $v_i \in S$  and  $v_j \notin S$ . We add an edge to  $\text{Anc}(G)$  if the test returns dependence. Finally, we take the transitive closure and output the resulting graph as  $\text{Anc}(G)$ . In our approach, a strongly separating set system is constructed with  $m = 2 \log n$  interventions by using the binary encoding of the numbers  $1, \dots, n$ , similar to the construction due to Kocaoglu et. al [88].

In Lemma 2.2.2, we show that in fact being strongly separating is *necessary* for any set of interventions to be used to identify  $\text{Anc}(G)$ .

**Lemma 2.2.2.** *Suppose  $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$  is a collection of subsets of  $V$ . For a given causal graph  $G$  if  $\text{Anc}(G)$  is recovered using CI-tests by intervening on the sets  $S_i \in \mathcal{S}$ . Then,  $\mathcal{S}$  is a strongly separating set system.*

*Proof.* Suppose  $\mathcal{S}$  is not a strongly separating set system. If there exists a pair of nodes  $(v_i, v_j)$  such that every set  $S_k \in \mathcal{S}$  contains none of them, then, we cannot recover the edge between these two nodes as we are not intervening on either  $v_i$  or  $v_j$  and the results of an independence test  $v_i \perp\!\!\!\perp v_j$  might not be correct due to the presence of a latent variable  $l_{ij}$  between them. Now, consider the case when only one of them is present in the set system. Let  $(v_i, v_j)$  be such that  $\forall S_k : S_k \cap \{v_i, v_j\} =$

$\{v_i\} \Rightarrow v_i \in S_k, v_j \notin S_k$ . We choose our graph  $G_{ij}$  to have two components  $\{v_i, v_j\}$  and  $V \setminus \{v_i, v_j\}$ ; and include the edge  $v_j \rightarrow v_i$  in it. Our algorithm will conclude from CI-test  $v_i \perp\!\!\!\perp v_j \mid \text{do}(S_k)$  that  $v_i$  and  $v_j$  are independent. However, it is possible that  $v_i \not\perp\!\!\!\perp v_j$  because of a latent  $l_{ij}$  between  $v_i$  and  $v_j$ , but  $v_i \perp\!\!\!\perp v_j \mid \text{do}(S_k)$  as intervening on  $v_i$  disconnects the  $l_{ij} \rightarrow v_i$  edge. Therefore, our algorithm cannot distinguish the two cases  $v_j \rightarrow v_i$  and  $v_i \leftarrow l_{ij} \rightarrow v_j$  without intervening on  $v_j$ . For every  $\mathcal{S}$  that is not a strongly separating set system, we can provide a  $G_{ij}$  such that by intervening on sets in  $\mathcal{S}$ , we cannot recover  $\text{Anc}(G_{ij})$  correctly.  $\square$

### 2.2.2 Recovering the Observable Graph

$\text{Anc}(G)$  encodes all the ancestral relations on observable variables  $V$  of the causal graph  $G$ . To recover  $G$  from  $\text{Anc}(G)$ , we want to differentiate whether  $v_i \rightarrow v_j$  represents an edge in  $G$  or a directed path going through other nodes in  $G$ . We use the following observation, if  $v_i$  is a parent of  $v_j$ , the path  $v_i \rightarrow v_j$  is never blocked by any conditioning set  $Z \subseteq V \setminus \{v_i\}$ . If  $v_i \notin \text{Pa}(v_j)$ , then we show that we can provide a conditioning set  $Z$  in some interventional distribution  $S$  such that  $v_i \perp\!\!\!\perp v_j \mid Z, \text{do}(S)$ . For every pair of variables that have an edge in  $\text{Anc}(G)$ , we design conditioning sets in Algorithm 1 that blocks all the paths between them.

Let  $v_i \in \text{Anc}(v_j) \setminus \text{Pa}(v_j)$ . We argue that conditioning on  $\text{Anc}(v_j) \setminus \{v_i\}$  in  $\text{do}(v_i \cup P_{ij})$  blocks all the paths from  $v_i$  to  $v_j$ . The first simple observation, from  $d$ -separation is that if we take a path that has no  $p$ -colliders between  $v_i$  to  $v_j$  (a  $p$ -collider free path) then it is blocked by conditioning on  $\text{Anc}(v_j) \setminus \{v_i\}$  i.e.,  $v_i \perp\!\!\!\perp v_j \mid \text{Anc}(v_j) \setminus \{v_i\}$ .

The idea then will be to intervene on colliders  $P_{ij}$  to remove these dependencies between  $v_i$  and  $v_j$  as shown by the following lemma.

**Lemma 2.2.3.** *Let  $v_i \in \text{Anc}(v_j)$ .  $v_i \perp\!\!\!\perp v_j \mid \text{do}(v_i \cup P_{ij}), \text{Anc}(v_j) \setminus \{v_i\}$  iff  $v_i \notin \text{Pa}(v_j)$ .*

From Lemma 2.2.3, we can recover the edges of the observable graph  $G$  provided we know the  $p$ -colliders between every pair of nodes. However, since the set of  $p$ -

colliders is unknown without the knowledge of  $\mathcal{G}$ , we construct multiple intervention sets by independently sampling every variable with some probability. This ensures that there exists an intervention set  $S$  such that  $\{v_i\} \cup P_{ij} \subseteq S$  and  $v_j \notin S$  with high probability.

Formally, let  $A_t \subseteq V$  for  $t \in \{1, 2, \dots, 72\tau' \log n\}$  be constructed by including every variable  $v_i \in V$  with probability  $1 - 1/\tau'$  where  $\tau' = \max\{\tau, 2\}$ . Let  $\mathcal{A}_\tau = \{A_1, \dots, A_{72\tau' \log n}\}$  be the collection of the set  $A_t$ 's. Algorithm 1 uses the interventions in  $\mathcal{A}_\tau$ .

---

**Algorithm 1** RECOVERG ( $\text{Anc}(G), \mathcal{A}_\tau$ )

---

```

1:  $E = \phi$ 
2: for  $v_i \rightarrow v_j$  in  $\text{Anc}(G)$  do
3:   Let  $\mathcal{A}_{ij} = \{A \in \mathcal{A}_\tau \text{ such that } v_i \in A, v_j \notin A\}$ 
4:   if  $\forall A \in \mathcal{A}_{ij}, v_i \not\perp\!\!\!\perp v_j \mid \text{Anc}(v_j) \setminus \{v_i\}, \text{do}(A)$  then
5:      $E = E \cup \{(v_i, v_j)\}$ 
6:   end if
7: end for
8: return  $E$ 

```

---

**Proposition 2.2.4.** *Let  $\mathcal{G}(V \cup L, E \cup E_L)$  be a  $\tau$ -causal graph with observable graph  $G(V, E)$ . There exists a procedure to recover the observable graph using  $O(\tau \log n + \log n)$  many interventions with probability at least  $1 - 1/n^2$ .*

**Lower Bound.** Complementing the above result, the following proposition gives a lower bound on the number of interventions by providing an instance of a  $O(n)$ -causal graph such that any non-adaptive algorithm requires  $\Omega(n)$  interventions for recovering it. The lower bound comes because of the fact that the algorithm cannot rule out the possibility of latent.

**Proposition 2.2.5.** *There exists a graph causal  $\mathcal{G}(V \cup L, E \cup E_L)$  such that every non-adaptive algorithm requires  $\Omega(n)$  many interventions to recover even the observable graph  $G(V, E)$  of  $\mathcal{G}$ .*

### 2.2.3 Detecting the Latents

We now describe algorithms to identify latents that effect the observable variables  $V$  to learn the entire causal graph  $\mathcal{G}(V \cup L, E \cup E_L)$ . We start from the observable graph  $G(V, E)$  constructed in the previous section. Our goal will be to use the fact that  $\mathcal{G}$  is a  $\tau$ -causal graph, which means that  $|P_{ij}| \leq \tau$  for every pair  $v_i, v_j$ . Since we assumed that each latent variable (in  $L$ ) effects at most two observable variables (in  $V$ ), we can split the analysis into two cases: a) pairs of nodes in  $G$  without an edge (non-adjacent nodes) and b) pairs of nodes in  $G$  with a direct edge (adjacent). In Algorithm LATENTSNEEDGES (Section 2.4.1.2), we describe the algorithm for identifying the latents effecting pairs of non-adjacent nodes. The idea is to block the paths by conditioning on parents and intervening on  $p$ -colliders. We use the observation that for any non-adjacent pair  $v_i, v_j$  an intervention on the set  $P_{ij}$  and conditioning on the parents of  $v_i$  and  $v_j$  will make  $v_i$  and  $v_j$  independent, unless there is a latent between them.

**Proposition 2.2.6.** *Let  $\mathcal{G}(V \cup L, E \cup E_L)$  be a  $\tau$ -causal graph with observable graph  $G(V, E)$ . Algorithm LATENTSNEEDGES with  $O(\tau^2 \log n + \log n)$  interventions recovers all latents effecting pairs of non-adjacent nodes in the observable graph  $G$  with probability at least  $1 - 1/n^2$ .*

**Latents Affecting Adjacent Nodes in  $G$ .** Suppose we have an edge  $v_i \rightarrow v_j$  in  $G(V, E)$  and we want to detect whether there exists a latent  $l_{ij}$  that effects both of them. Here, we cannot block the edge path  $v_i \rightarrow v_j$  by conditioning on any  $Z \subseteq V$  in any given interventional distribution  $\text{do}(S)$  where  $S$  does not contain  $v_j$ . However, intervening on  $v_j$  also disconnects  $v_j$  from its latent parent. Therefore, CI-tests are not helpful. Hence, we make use of another test called *do-see* test [88], that compares two probability distributions. We assume there exists an oracle that answers whether two distributions are the same or not. This is a well-studied problem with sublinear (in domain size) bound on the sample size needed for implementing this oracle [37].

---

**Algorithm 2** LatentsWEdges( $\mathcal{G}(V \cup L, E \cup E_L), \mathcal{B}_\tau$ )

---

- 1: Consider the edge  $v_i \rightarrow v_j \in E$ .
  - 2: Let  $\mathcal{B}_{ij} = \{B \setminus \{v_i\} \mid B \in \mathcal{B}_\tau \text{ s.t. } v_i \in B, v_j \notin B\}$
  - 3: **if**  $\forall B \in \mathcal{B}_{ij}, \Pr[v_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \neq \Pr[v_j \mid \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)]$   
**then**
  - 4:      $L \leftarrow L \cup l_{ij}, E_L \leftarrow E_L \cup \{(l_{ij}, v_i), (l_{ij}, v_j)\}$
  - 5: **end if**
  - 6: **return**  $\mathcal{G}(V \cup L, E \cup E_L)$
- 

**Assumption 2.2.7** (Distribution Testing (DT)-Oracle). *Given any  $v_i, v_j \in V$  and  $Z, S \subseteq V$  tests whether two distributions  $\Pr[v_j \mid v_i, Z, \text{do}(S)]$  and  $\Pr[v_j \mid Z, \text{do}(S \cup \{v_i\})]$  are identical or not.*

The intuition of the do-see test is as follows: if  $v_i$  and  $v_j$  are the only two nodes in the graph  $G$  with  $v_i \rightarrow v_j$ , then,  $\Pr[v_j \mid v_i] = \Pr[v_j \mid \text{do}(v_i)]$  iff there exists no latent that effects both of them. This follows from the *conditional invariance* principle [20] (or page 24, property 2 in [105]). Therefore, the presence or absence of latents can be established by invoking a DT-oracle.

As we seek to minimize the number of interventions, our goal is to create intervention sets that contain  $p$ -colliders between every pair of variables that share an edge in  $G$ . However, in Lemmas 2.2.8, 2.2.9 we argue that it is not sufficient to consider interventions with only  $p$ -colliders. We must also intervene on  $\text{Pa}(v_i)$  to detect a latent between  $v_i \rightarrow v_j$ . The main idea behind LATENTSWEDGES is captured by the following two lemmas.

**Lemma 2.2.8** (No Latent Case). *Suppose  $v_i \rightarrow v_j \in G$  and  $v_i, v_j \notin B$ , and  $P_{ij} \subseteq B$  then,  $\Pr[v_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] = \Pr[v_j \mid \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)]$  if there is no latent  $l_{ij}$  with  $v_i \leftarrow l_{ij} \rightarrow v_j$ .*

**Lemma 2.2.9** (Latent Case). *Suppose  $v_i \rightarrow v_j \in G$  and  $v_i, v_j \notin B$ , and  $P_{ij} \subseteq B$ , then,  $\Pr[v_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \neq \Pr[v_j \mid \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)]$  if there is a latent  $l_{ij}$  with  $v_i \leftarrow l_{ij} \rightarrow v_j$ .*

From Lemmas 2.2.8, 2.2.9, we know that to identify a latent  $l_{ij}$  between  $v_i \rightarrow v_j$ , we must intervene on all the  $p$ -colliders between them with  $Pa(v_i) \cup \{v_i\}$ . To do this, we again construct random intervention sets. Let  $B_t \subseteq V$  for  $t \in \{1, 2, \dots, 72\tau' \log n\}$  be constructed by including every variable  $v_i \in V$  with probability  $1 - 1/\tau'$  where  $\tau' = \max\{\tau, 2\}$ . Let  $\mathcal{B}_\tau = \{B_1, \dots, B_{72\tau' \log n}\}$  be the collection of the sets. Consider a pair  $v_i \rightarrow v_j$ . To obtain the interventions given by the above lemmas, we iterate over all sets in  $\mathcal{B}_\tau$  and identify all the sets containing  $v_i$ , but not  $v_j$ . From these sets, we remove  $v_i$  to obtain  $\mathcal{B}_{ij}$ . These new interventions are then used in LATENTSWEDGES to perform the required distribution tests using a DT-oracle on the interventions  $B \cup Pa(v_i)$  and  $B \cup Pa(v_i) \cup \{v_i\}$  for every  $B \in \mathcal{B}_{ij}$ . We can show:

**Proposition 2.2.10.** *Let  $\mathcal{G}(V \cup L, E \cup E_L)$  be a  $\tau$ -causal graph with observable graph  $G(V, E)$ .*

*LATENTSWEDGES with  $O(n\tau \log n + n \log n)$  interventions recovers all latents effecting pairs of adjacent nodes in the observable graph  $G$  with probability at least  $1 - 1/n^2$ .*

**Putting it all Together.** Using Propositions 2.2.4, 2.2.6, and 2.2.10, we get the following result. Note that  $\tau \leq n - 2$ .

**Theorem 2.2.11.** *Given access to a  $\tau$ -causal graph  $\mathcal{G} = \mathcal{G}(V \cup L, E \cup E_L)$  through Conditional Independence (CI) and Distribution Testing (DT) oracles, Algorithms RECOVERG, LATENTSNEDGES, and LATENTSWEDGES put together recovers  $\mathcal{G}$  with  $O(n\tau \log n + n \log n)$  interventions, with probability at least  $1 - O(1/n^2)$  (where  $|V| = n$ ).*

**Remark** Note that our algorithms achieve an overall success probability of  $1 - O(1/n^2)$ , however, the success probability can be boosted to any  $1 - O(1/n^c)$  for any constant  $c$ , by adjusting the constant factors (see for example the proof of Lemma 2.4.2). Also for simplicity of discussion, we assume that we know  $\tau$ . How-

ever as we discuss in Section 2.4.1.4, this assumption can be easily removed with an additional  $O(\log \tau)$  factor.

## 2.3 Experimental Evaluation

In this section, we compare the total number of interventions required to recover causal graph  $\mathcal{G}$  parameterized by  $p$ -colliders (see Section 2.2) vs. maximum degree utilized by Kocaoglu et al. [88].

Since the parameterization of these two results are different, a direct comparison between them is not always possible. If  $\tau = o(d^2/n)$ , we use fewer interventions than those used by Kocaoglu et al. [88] for recovering the causal graph. Roughly, for any  $0 \leq \epsilon \leq 1$ , (a) when  $\tau < n^\epsilon, d > n^{(1+\epsilon)/2}$ , our bound is better, (b) when  $\tau > n^\epsilon, \tau < d < n^{(1+\epsilon)/2}$ , then we can identify latents using the algorithms of Kocaoglu et al. [88], after using our algorithm for observable graph recovery, and (c) when  $\tau > d > n^\epsilon, d < n^{(1+\epsilon)/2}$ , the bound achieved by Kocaoglu et al. [88] is better.

In this section, our main motivation is to show that  $p$ -colliders can be a useful measure of complexity of a graph. As discussed in Chapter 1, even few nodes of high degree could make  $d^2$  quite large.

**Setup** We demonstrate our results by considering sparse random graphs generated from the families of: (i) Erdős-Rényi random graphs  $G(n, c/n)$  for constant  $c$ , (ii) Random Bipartite Graphs generated using  $G(n_1, n_2, c/n)$  model, with partitions  $L, R$  and edges directed from  $L$  to  $R$ , (iii) Directed Trees with degrees of nodes generated from power law distribution. In each of the graphs we generate, we additionally include latent variables by sampling 5% of  $\binom{n}{2}$  pairs and adding a latent between them.

**Finding  $p$ -colliders** Let  $\mathcal{G}$  contain observable variables and the latents. To find  $p$ -colliders between every pair of observable nodes of  $\mathcal{G}$ , we enumerate all paths between

them and check if any of the observable nodes on a path can be a possible  $p$ -collider. As this became practically infeasible for larger values of  $n$ , we devise an algorithm that runs in polynomial time (in the size of the graph) by constructing an appropriate flow network and finding maximum flow in this network. We will first describe a construction that takes three nodes  $(v_i, v_j, v_k)$  as input and checks if  $v_k$  is a  $p$ -collider for the pair of nodes  $v_i$  and  $v_j$ . Iterating over all possible nodes  $v_k$  gives us all the  $p$ -colliders for the pair  $v_i, v_j$ .

**Construction** If  $v_k$  is not an ancestor of either  $v_i$  or  $v_j$ , then, output  $v_k$  is not a  $p$ -collider. Else, we describe a modification of  $\mathcal{G}$  to obtain the flow network  $\tilde{\mathcal{G}}$ . First, initialize  $\tilde{\mathcal{G}}$  with  $\mathcal{G}$ . Remove all outgoing edges of  $v_k$  from  $\tilde{\mathcal{G}}$  and set the capacity of all incoming edges incident on  $v_k$  to 1. Add a node  $T_{ij}$  along with the edges  $T_{ij} \rightarrow v_i$  and  $T_{ij} \rightarrow v_j$  to  $\tilde{\mathcal{G}}$  and set the capacity of these edges to 1. For every node  $w \in V \cup L \setminus \{v_k\}$ , create two nodes  $w_{in}$  and  $w_{out}$ . Add edge  $w_{out} \rightarrow w_{in}$  with a capacity 1. Every incoming edge to  $w$  i.e.,  $z \rightarrow w$  is replaced by  $z \rightarrow w_{in}$  and every outgoing edge  $w \rightarrow z$  is replaced by  $w_{out} \rightarrow z$  with capacity 1. Find maximum  $s, t$  flow in  $\tilde{\mathcal{G}}$  with  $T_{ij}, v_k$  as source and sink respectively. If the maximum flow is 2, then output  $v_k$  is a  $p$ -collider, otherwise no.

Now, we outline the idea for the proof of correctness of the above construction.

**Sketch of the Proof** After ensuring that  $v_k$  has a directed path to either  $v_i$  or  $v_j$ , we want to check whether there is an undirected path from  $v_i$  to  $v_j$  containing  $v_k$  as a collider. In other words, we want to check if there are *two vertex disjoint paths* from  $v_i$  and  $v_j$  to  $v_k$  such that both of these paths have incoming edges to  $v_k$ . By adding a node  $T_{ij}$  connected to  $v_i$  and  $v_j$ , we want to route two units of flow from  $T_{ij}$  to  $v_k$  where each node has a vertex capacity of 1. Converting vertex capacities into edge capacities by splitting every node into two nodes (one for incoming and the other for

outgoing edges) gives us the desired flow network on which we can solve maximum flow.

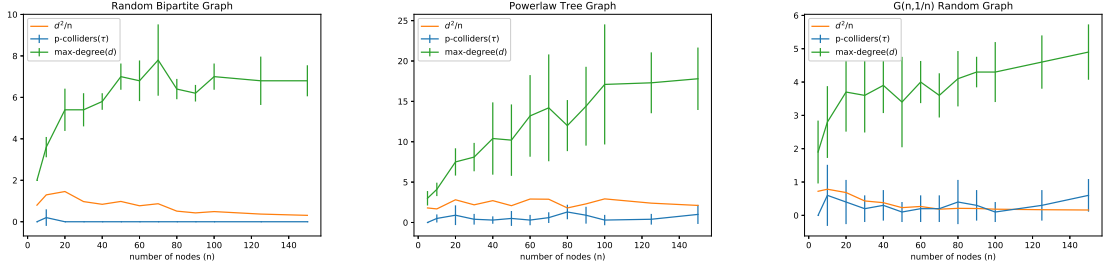


Figure 2.3: Comparison of  $\tau$  vs. maximum degree in various sparse random graph models. On the x-axis is the number of nodes in the graph. Note that our bound on the number of interventions needed to recover  $\mathcal{G}$  is better than those provided by Kocaoglu et al. [88] roughly when  $\tau < d^2/n$ .

**Results** In our plots (Figure 2.3), we compare the maximum undirected degree ( $d$ ) with the maximum number of  $p$ -colliders between any pair of nodes (which defines  $\tau$ ). We ran each experiment 10 times and plot the mean value along with one standard deviation error bars.

Recall that in the worst case, the number of interventions used by our approach (Theorem 2.2.11) is  $O(n\tau \log n + n \log n)$  while the algorithm proposed by Kocaoglu et al. [88] uses  $O(\min\{d \log^2 n, \ell\} + d^2 \log n)$  many interventions where  $\ell$  is the length of the longest directed path in the graph. So roughly when  $\tau < d^2/n$ , our bound is better. For this purpose, we also plot the  $d^2/n$  line using the mean value of  $d$  obtained.

For random bipartite graphs, that can be used to model causal relations over time, we use equal partition sizes  $n_1 = n_2 = n/2$  and plot the results for  $\mathcal{G}(n/2, n/2, c/n)$  for constant  $c = 5$ . We observe that the behaviour is uniform for small constant values of  $c$ . In this case, we observe that the number of  $p$ -colliders is close to zero for all values of  $n$  in the range considered and our bound is better.

For directed random trees, where degrees of nodes follow the powerlaw distribution (observed in real world networks [3]), we again observe that for almost all the values of  $n$ , our bound is better. We run our experiments with small constant values for the exponent  $\gamma$  and show the plots for  $\gamma = 3$  in Figure 2.3. As powerlaw graphs contain only a few nodes concentrated around a very high degree, we expect our algorithm to perform better in such cases. For Erdős-Rényi random graphs  $\mathcal{G}(n, 1/n)$ , we observe that our result is either better or comparable to that of Kocaoglu et al. [88].

It is interesting to see that in the sparse graphs we considered  $\tau$  is considerably smaller compared to  $d$ . Moreover, if we want to identify only the observable graph  $G$  under the presence of latents, our algorithm uses  $O(\tau \log n)$  interventions where as the previous known algorithm, due to Kocaoglu et. al. [88], uses  $O(d \log^2 n)$  interventions. In the random graphs considered above, our algorithms perform significantly better for identifying  $G$ . Therefore, we believe that minimizing the number of interventions based on the notion of  $p$ -colliders is a reasonable direction to consider.

## 2.4 Additional Proof Details from Section 2.2

In this section, we provide all the missing proof details from section 2.2.

### 2.4.1 Recovering the entire causal graph $\mathcal{G}$

In this section, we present the missing proof details from section 2.2. In section 2.4.1.1, we present the missing proof details for recovering the observable graph  $G$ ; in sections 2.4.1.2 and 2.4.1.3, we present the missing proof details for detecting the latents.

#### 2.4.1.1 Recovering the observable graph $G$

In this section, we present the missing proof details from section 2.2.2 for recovering the observable graph  $G$ .

**Lemma 2.4.1** (Lemma 2.2.3 restated). *Let  $v_i \in \text{Anc}(v_j)$ .  $v_i \perp\!\!\!\perp v_j \mid \text{do}(v_i \cup P_{ij}), \text{Anc}(v_j) \setminus \{v_i\}$  iff  $v_i \notin \text{Pa}(v_j)$ .*

*Proof.* Suppose  $v_i \in \text{Anc}(v_j) \setminus \text{Pa}(v_j)$ . Consider the interventional distribution  $\text{do}(v_i \cup P_{ij})$  where  $P_{ij}$  is the set of  $p$ -colliders between  $v_i$  and  $v_j$ . We intervene on  $v_i$  to block the path (if present) given by  $v_i \leftarrow \tilde{l} \rightarrow v_j$  where  $\tilde{l} \in L$ . Consider all the remaining *undirected* paths between  $v_i$  and  $v_j$  denoted by  $\Pi_{ij}$ . We divide  $\Pi_{ij}$  into three cases. Let  $\pi \in \Pi_{ij}$  be a path from  $v_i$  to  $v_j$ .

1.  $\pi$  contains no *colliders*, then,  $\pi$  is blocked by  $\text{Anc}(v_j) \setminus \{v_i\}$ . As  $\pi$  contains no colliders, we can write  $\pi = v_i \cdots v_k \rightarrow v_j$  where  $v_k \in \text{Anc}(v_j)$ . As we are conditioning on  $\text{Anc}(v_j) \setminus \{v_i\} \supseteq \{v_k\}$ ,  $\pi$  is blocked by  $v_k$ .
2.  $\pi$  contains colliders but not a  $p$ -collider. We argue that there are no collider nodes in  $\pi$  that are also in  $\text{Anc}(v_j) \setminus \{v_i\}$ . As there are no  $p$ -colliders, it means that all the colliders have no descendants in the conditioning set  $\text{Anc}(v_j) \setminus \{v_i\}$ . Because if a collider  $v_c$  have a descendant in  $\text{Anc}(v_j) \setminus \{v_i\}$ , then there is a path from  $v_c$  to  $\text{Pa}(v_j)$  through  $\text{Anc}(v_j) \setminus \{v_i\}$ . This means that  $v_c$  is a  $p$ -collider, contradicting our assumption. Therefore, from Rule-2 of  $d$ -separation,  $\pi$  is blocked.
3.  $\pi$  contains at least one  $p$ -collider. We are intervening on  $P_{ij}$  containing all the  $p$ -colliders. In the intervened mutilated graph, all the  $p$ -colliders no longer have an incoming arrow and therefore are not colliders. So  $\pi$  is blocked.

If  $v_i \notin \text{Pa}(v_j)$ , we can conclude that  $v_i \perp\!\!\!\perp v_j \mid \text{do}(\{v_i\} \cup P_{ij}), \text{Anc}(v_j) \setminus \{v_i\}$ . Suppose  $v_i \in \text{Pa}(v_j)$ . In the interventional distribution  $\text{do}(\{v_i\} \cup P_{ij})$ , we still have  $v_i \in \text{Pa}(v_j)$  and any conditioning will not block the path  $\pi = v_i \rightarrow v_j$ . Therefore,  $v_i \not\perp\!\!\!\perp v_j \mid (\text{do}(\{v_i\} \cup P_{ij}), \text{Anc}(v_j) \setminus \{v_i\})$  if  $v_i \in \text{Pa}(v_j)$ .  $\square$

**Lemma 2.4.2.** *Let  $\mathcal{G}(V \cup L, E \cup E_L)$  be a  $\tau$ -causal graph with observable graph  $G(V, E)$ . Given an ancestral graph  $\text{Anc}(G)$ , Algorithm RECOVERG correctly recovers all edges in the observable graph with probability at least  $1 - 1/n^2$ .*

*Proof.* Let  $\tau' = \max\{\tau, 2\}$ . From Lemma 2.2.3, we can recover the edges of  $G$  provided we know the  $p$ -colliders between every pair of nodes. As we do not know the graph  $G$ , we devise a randomized strategy to hit all the  $p$ -colliders, whilst ensuring that we don't create a lot of interventions. Suppose  $\max_{(v_i, v_j) \in V \times V} |P_{ij}| \leq \tau$ . We show that with high probability,  $\forall v_i \in \text{Anc}(v_j), \exists A_t$  such that  $\{v_i\} \cup P_{ij} \subseteq A_t$  and  $v_j \notin A_t$ . We can then use the CI-test described in Lemma 2.2.3 to verify whether  $v_i$  is a parent of  $v_j$ . In Algorithm RECOVERG, we repeat this procedure on every edge of  $\text{Anc}(G)$  and output  $G$ .

Suppose  $v_i \in \text{Anc}(v_j)$ . Let  $\Gamma_t$  denote the event that  $A_t \in \mathcal{A}_\tau$  such that  $\{v_i\} \cup P_{ij} \subseteq A_t$  and  $v_j \notin A_t$  for a fixed  $t \in \{1, \dots, 72\tau' \log n\}$ . Let  $T = 72\tau' \log n$ . As we include a vertex  $v_i \in A_t$  with probability  $1 - 1/\tau'$ , we obtain

$$\Pr[\Gamma_t] = \left(1 - \frac{1}{\tau'}\right)^{|P_{ij}|+1} \frac{1}{\tau'} \geq \left(1 - \frac{1}{\tau'}\right)^{\tau'+1} \frac{1}{\tau'}.$$

Using the inequality  $(1 + \frac{x}{n})^n \geq e^x(1 - \frac{x^2}{n})$  for  $|x| \leq n$ , and since  $\tau' \geq 2$  we have:

$$\begin{aligned} \Pr[\Gamma_t] &\geq \frac{1}{e^{\tau'+1/\tau'}} \left(1 - \frac{(\tau'+1)^2}{\tau'^2(\tau'+1)}\right) \frac{1}{\tau'} \geq \frac{1}{18\tau'} \\ \Rightarrow \Pr[\bar{\Gamma}_t] &\leq 1 - \frac{1}{18\tau'} \text{ and } \Pr[\exists t \in [T] : \Gamma_t] \geq 1 - \left(1 - \frac{1}{18\tau'}\right)^{72\tau' \log n}. \end{aligned}$$

Using the inequality  $(1 + \frac{x}{n})^n \leq e^x$  for  $|x| \leq n$  we have:

$$\Pr[\exists t \in [T] : \Gamma_t] \geq 1 - \frac{1}{n^4}.$$

So the probability that there exists at least one set  $A_t$  for the given pair  $v_i, v_j$  for which  $v_i \cup P_{ij} \subseteq A_t$  and  $v_j \notin A_t$  is at least  $1 - \frac{1}{n^4}$ .<sup>2</sup> To ensure this probability of success for every pair of variables, we use a union bound over the  $n^2$  node pairs.  $\square$

**Proposition 2.4.3** (Proposition 2.2.4 restated). *Let  $\mathcal{G}(V \cup L, E \cup E_L)$  be a  $\tau$ -causal graph with observable graph  $G(V, E)$ . There exists a procedure to recover the observable graph using  $O(\tau \log n + \log n)$  many interventions with probability at least  $1 - 1/n^2$ .*

*Proof.* As is well-known, e.g. [88], a strongly separating set system can be constructed with  $m = 2 \log n$  interventions by using the binary encoding of the numbers  $1, \dots, n$ . Two intervention sets are constructed for every bit location  $k \in [\log n]$ , one with any node  $v_i$  if the number  $i$  has  $k$ th bit set to 1, and other with any node  $v_i$  if the number  $i$  has  $k$ th bit set to 0. Therefore, we require  $2 \log n$  interventions to obtain ancestral graph  $\text{Anc}(G)$  of the observable graph. From Lemma 2.4.2, we require  $O(\tau \log n)$  interventions to recover all the edges of observable graph of  $G$  from  $\text{Anc}(G)$  with probability  $1 - \frac{1}{n^2}$ . Therefore, using  $O(\tau \log n)$  interventions, Algorithm RECOVERG can recover the observable graph  $G(V, E)$  with high probability.  $\square$

It is well established that  $\log(\chi(G))$  interventions are necessary and sufficient in the causally sufficient systems (where there are no latents) where  $\chi(G)$  is the chromatic number of  $G$ . Generalized over all graphs this becomes  $\log(n)$ . Our following lower bound shows that, even if there are no latent variables in the underlying system, if the algorithm cannot rule latents out, and needs to consider latents as a possibility to compute the graph skeleton, then  $\Omega(n)$  interventions are necessary. Shanmugam et al. [115] provide a lower bound in a different setting, when the intervention sets are required to have only limited number of variables.

---

<sup>2</sup>Note by adjusting the constant 72, we could have pushed this probability to any  $1/n^c$  for constant  $c$ .

**Proposition 2.4.4** (Proposition 2.2.5 restated). *There exists a graph causal  $\mathcal{G}(V \cup L, E \cup E_L)$  such that every non-adaptive algorithm requires  $\Omega(n)$  many interventions to recover even the observable graph  $G(V, E)$  of  $\mathcal{G}$ .*

*Proof.* Consider an ordering of observable variables given by  $v_1, v_2, \dots, v_n$ . Let  $G$  be a graph with all directed edges  $(v_a, v_b)$  for all  $b > a$ . Suppose the set of interventions generated by the non-adaptive algorithm is given by  $\mathcal{H}$ . Now consider  $v_i$  for some fixed  $i \geq \frac{n}{4}$ .

We claim that if every intervention  $H \in \mathcal{H}$  is such that for some  $j \in \{3, \dots, i-1\}$ ,  $v_j \notin H$ , then there exists a graph  $G_i$  such that  $G$  and  $G_i$  are both indistinguishable under all the interventions in  $\mathcal{H}$  irrespective of other conditioning. Now consider any set  $H_j \subseteq (\{v_1, v_2, \dots, v_{i-1}\} \setminus \{v_j\}) \cup \{v_{i+1}, \dots, v_n\}$ . Let  $G_i$  be such that it contains all the directed edges  $(v_a, v_b)$  for all  $b > a$  but does not contain the directed edge  $(v_1, v_i)$ . To distinguish between  $G$  and  $G_i$  one needs to determine whether  $v_1 \rightarrow v_i$ . Note that any intervention we use to determine the edge should contain  $v_1$  to rule out the possibility of the influence of latent  $v_1 \leftarrow l_{1i} \rightarrow v_i$  on the CI-tests we perform. Now, under  $\text{do}(H_j)$ , there are only two CI-tests possible to determine whether  $v_1 \rightarrow v_i : v_1 \perp\!\!\!\perp v_i \mid v_j, \text{do}(H_j)$  and  $v_1 \perp\!\!\!\perp v_i \mid \text{do}(H_j)$ . However, for both graphs  $G$  and  $G_i$ , both these independence tests will always turn out negative. In the former case, it is because  $v_j$  will be a collider on the path  $v_i, v_j, v_{j-1}, v_i$ , and in the latter case there is a path  $v_1, v_j, v_i$  that is not blocked. In other words, the CI-tests will provide no information to distinguish between  $G$  and  $G_i$ , unless  $\mathcal{H}$  contains the set  $\{v_1, v_3, \dots, v_{i-1}\}$ .

One can similarly construct these  $G_i$ 's for all  $i \geq \frac{n}{4}$ , thereby  $\mathcal{H}$  needs to contain the intervention sets  $\{v_1, v_3, \dots, v_{i-1}\}$  for all  $n/4 \leq i \leq n$  to separate  $G$  from all the  $G_i$ 's. This proves the claim.  $\square$

---

**Algorithm 3** LATENTSNEDES ( $G(V, E), \mathcal{D}_\tau$ )

---

```
1:  $L \leftarrow \phi, E_L \leftarrow \phi$ 
2: for  $(v_i, v_j) \notin E$  do
3:   Let  $\mathcal{D}_{ij} = \{D \mid D \in \mathcal{D}_\tau \text{ and } v_i, v_j \notin D\}$ 
4:   if  $v_i \perp\!\!\!\perp v_j \mid do(D) \cup Pa(v_i) \cup Pa(v_j)$  for every  $D \in \mathcal{D}_{ij}$  then
5:      $L \leftarrow L \cup l_{ij}, E_L \leftarrow E_L \cup \{(l_{ij}, v_i), (l_{ij}, v_j)\}$ 
6:   end if
7: end for
8: return  $\mathcal{G}(V \cup L, E \cup E_L)$ 
```

---

### 2.4.1.2 Latents Affecting Non-adjacent Nodes in $G$

Let  $\bar{E} = \{(v_i, v_j) \mid (v_i, v_j) \notin E\}$  be the set of non-edges in  $G$ . The entire procedure for finding latents between non-adjacent nodes in  $G$  is described in Algorithm LATENTSNEDES. Similar to Algorithm RECOVERG, we block the paths by conditioning on parents and intervening on  $p$ -colliders. The idea is based on the observation that for any non-adjacent pair  $v_i, v_j$  an intervention on the set  $P_{ij}$  and conditioning on the parents of  $v_i$  and  $v_j$  will make  $v_i$  and  $v_j$  independent, unless there is a latent between them. The following lemma formalizes this idea.

**Lemma 2.4.5.** *Suppose  $(v_i, v_j) \in \bar{E}$ . Then,  $v_i \perp\!\!\!\perp v_j \mid do(P_{ij}), Pa(v_i) \cup Pa(v_j)$  iff  $v_i$  and  $v_j$  has no latent between them.*

*Proof.* Suppose there is no latent between  $v_i$  and  $v_j$ . We follow the proof similar to the Lemma 2.2.3. Consider the pair of variables  $v_i$  and  $v_j$  and all the paths between them  $\Pi_{ij}$ . Let  $\pi \in \Pi_{ij}$ .

1. Let  $\pi$  be a path not containing any colliders. Using Rule-1 of  $d$ -separation, we can block  $\pi$  by conditioning on either  $Pa(v_i)$  or  $Pa(v_j)$ .
2. If  $\pi$  contains colliders and no  $p$ -colliders, then, using Rule-2 of  $d$ -separation,  $\pi$  is blocked as the colliders have no descendants in  $Pa(v_i) \cup Pa(v_j)$ .
3. We block the paths  $\pi$  containing  $p$ -colliders by intervening on  $P_{ij}$

As all the paths in  $\Pi_{ij}$  are blocked, we have  $v_i \perp\!\!\!\perp v_j \mid \text{do}(P_{ij}), \text{Pa}(v_i) \cup \text{Pa}(v_j)$ . If there is a latent  $l_{ij}$  then the path  $v_i \leftarrow l_{ij} \rightarrow v_j$  is not blocked and therefore  $v_i \not\perp\!\!\!\perp v_j \mid (\text{do}(P_{ij}), \text{Pa}(v_i) \cup \text{Pa}(v_j))$ .  $\square$

Formally, let  $D_t \subseteq V$  for  $t \in \{1, 2, \dots, 24\tau'^2 \log n\}$  be constructed by including every variable  $v_i \in V$  with probability  $1 - \frac{1}{\tau'}$  where  $\tau' = \max\{\tau, 2\}$ . Let  $\mathcal{D}_\tau = \{D_1, \dots, D_{24\tau'^2 \log n}\}$  be the collection of the set  $D_t$ 's. Using these interventions  $\mathcal{D}_\tau$ , we argue that we can recover all the latents between non-edges of  $G$  correctly with high probability.

**Proposition 2.4.6** (Proposition 2.2.6 restated). *Let  $\mathcal{G}(V \cup L, E \cup E_L)$  be a  $\tau$ -causal graph with observable graph  $G(V, E)$ . Algorithm LATENTSNEDES with  $O(\tau^2 \log n + \log n)$  many interventions recovers all latents effecting pairs of non-adjacent nodes in the observable graph  $G$  with probability at least  $1 - 1/n^2$ .*

*Proof.* We follow a proof similar to Lemma 2.4.2. Consider a pair of variables  $v_i$  and  $v_j$  such that there is no edge between them in  $G$ . From Lemma 2.4.5, we know that by intervening on all the colliders between  $v_i$  and  $v_j$ , we can identify the presence of a latent. In Algorithm LATENTSNEDES, we iterate over sets in  $\mathcal{D}_{ij}$ . As  $\mathcal{D}_{ij} \subseteq \mathcal{D}_\tau$ , we have  $|\mathcal{D}_{ij}| \leq 24\tau'^2 \log n$ . Let  $\Gamma_t$  denote the event that  $D_t \in \mathcal{D}_{ij}$  is such that  $v_i, v_j \notin D_t$  and  $P_{ij} \subseteq D_t$  for a fixed  $t \in \{1, \dots, 24\tau'^2 \log n\}$ . Let  $T = 24\tau'^2 \log n$ .

$$\Pr[\Gamma_t] = \left(1 - \frac{1}{\tau'}\right)^{|P_{ij}|} \frac{1}{\tau'^2} \geq \left(1 - \frac{1}{\tau'}\right)^{\tau'} \frac{1}{\tau'^2}.$$

Using the inequality  $(1 + \frac{x}{n})^n \geq e^x (1 - \frac{x^2}{n})$  for  $|x| \leq n$ , and since  $\tau' \geq 2$  we have:

$$\begin{aligned} \Pr[\Gamma_t] &\geq \frac{1}{e} \left(1 - \frac{1}{\tau'}\right) \frac{1}{\tau'^2} \geq \frac{1}{2e\tau'^2} \\ \Rightarrow \Pr[\bar{\Gamma}_t] &\leq 1 - \frac{1}{6\tau'^2} \text{ and } \Pr[\exists t \in [T] : \Gamma_t] \geq 1 - \left(1 - \frac{1}{6\tau'^2}\right)^{24\tau'^2 \log n}. \end{aligned}$$

Using the inequality  $(1 + \frac{x}{n})^n \leq e^x$  for  $|x| \leq n$  we have:

$$\Pr[\exists t \in [T] : \Gamma_t] \geq 1 - \frac{1}{n^4}.$$

So the probability that there exists a set  $D_t$  for which  $v_i, v_j \notin D_t$  and  $P_{ij} \subseteq D_t$  is at least  $1 - \frac{1}{n^4}$ . A union bound over at most  $n^2$  pair of variables completes the proof.  $\square$

### 2.4.1.3 Latent Affecting Adjacent Nodes in $G$

We follow an approach similar to the one presented in section 2.4.1.2 for detecting the presence of latent between an edge  $v_i \rightarrow v_j$  in  $G$ . In Algorithm 2, we block all the paths (excluding the edge) between the variables  $v_i$  and  $v_j$  using a conditioning set  $\text{Pa}(v_j)$  in the intervention distribution  $\text{do}(\text{Pa}(v_i) \cup P_{ij})$  in the *do-see* tests we perform. This idea is formalized using the following lemma.

**Lemma 2.4.7.** *Suppose  $v_i \rightarrow v_j \in G$ . Let  $l_{tj}$  be a latent between  $v_t$  and  $v_j$  where  $v_t \neq v_i$  and  $v_i, v_j \notin B$ ,  $P_{ij} \subseteq B$ . Then,  $l_{tj} \perp\!\!\!\perp v_i \mid \text{Pa}(v_j), \text{do}(B \cup \{v_i\} \cup \text{Pa}(v_i))$  and  $l_{tj} \perp\!\!\!\perp v_i \mid \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)$ .*

*Proof.* The proof goes through by analyzing various cases. We give a detailed outline of the proof.

Claim 1:  $l_{tj} \perp\!\!\!\perp v_i \mid \text{Pa}(v_j), \text{do}(B \cup \{v_i\} \cup \text{Pa}(v_i))$ . Suppose  $v_t \in \text{Pa}(v_i) \cup B$ . Consider all the paths between  $v_i$  and  $l_{tj}$  in the interventional distribution  $\text{do}(B \cup \{v_i\} \cup \text{Pa}(v_i))$ . The only paths that are not separated because of the intervention are  $l_{tj} \rightarrow v_j \leftarrow v_i$ ,  $l_{tj} \rightarrow v_j \leftarrow v_k \cdots \leftarrow v_i$  where  $v_k \in \text{Pa}(v_j)$ , and  $l_{tj} \rightarrow v_j \rightarrow \cdots \leftarrow v_i$ . As we are not conditioning on  $v_j$ ,  $l_{tj} \rightarrow v_j \leftarrow v_i$  is blocked (Rule-2 in  $d$ -separation); conditioning on  $\text{Pa}(v_j) \ni v_k$  block the paths  $l_{tj} \rightarrow v_j \leftarrow v_k \cdots \leftarrow v_i$  (Rule-1 in  $d$ -separation); and  $l_{tj} \rightarrow v_j \rightarrow \cdots \leftarrow v_i$  paths have a collider that is not  $\text{Pa}(v_j)$  hence blocked by Rule-2 in  $d$ -separation.

Suppose  $v_t \notin \text{Pa}(v_i) \cup B$ . As before it follows that all paths between  $l_{tj}$  and  $v_i$  going through  $v_j$  are blocked. All other paths between  $l_{tj}$  and  $v_i$  should have a

collider. This is because in any such path  $\pi$  the only edge from  $l_{tj}$  is  $l_{tj} \rightarrow v_t$  and the edges that remain at  $v_i$  are outgoing. It is easy to see that the collider on this path  $\pi$  can't be in  $\text{Pa}(v_j)$  because otherwise it will also be a  $p$ -collider between  $v_i$  and  $v_j$  which are intervened on through  $B$ . When there is a collider on the path that is not in the conditioning set, then the path is blocked (Rule-2 in  $d$ -separation). The same holds for all paths between  $l_{tj}$  and  $v_i$ .

Claim 2:  $l_{tj} \perp\!\!\!\perp v_i \mid \text{Pa}(v_j), \text{do}(B \cup \text{Pa}(v_i))$ . Consider all the paths between  $l_{tj}$  and  $v_i$ . Using the above arguments, we have that all paths containing  $v_j$  are blocked. All other paths between  $l_{tj}$  and  $v_i$  should have a collider. This is because in any such path  $\pi$  the only edge from  $l_{tj}$  is  $l_{tj} \rightarrow v_t$  and  $\pi$  will end at  $v_i$  either as  $l_{tj} \cdots \leftarrow v_i$  or  $l_{tj} \rightarrow \cdots \leftarrow v_k \rightarrow v_i$  where  $v_k \in \text{Pa}(v_i)$ . It is again easy to see that the collider on this path  $\pi$  can't be in  $\text{Pa}(v_j)$  because otherwise it will also be a  $p$ -collider between  $v_i$  and  $v_j$  which are intervened on through  $B$ . As before, when there is a collider on the path that is not in the conditioning set, then the path is blocked (Rule-2 in  $d$ -separation). The same holds for all paths between  $l_{tj}$  and  $v_i$ .  $\square$

**Lemma 2.4.8** (Lemma 2.2.8 restated). *Suppose  $v_i \rightarrow v_j \in G$  and  $v_i, v_j \notin B$ , and  $P_{ij} \subseteq B$  then,  $\Pr[v_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] = \Pr[v_j \mid \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)]$  if there is no latent  $l_{ij}$  with  $v_i \leftarrow l_{ij} \rightarrow v_j$ .*

*Proof.* Suppose  $v_i \rightarrow v_j$  in  $G$  and there is no latent between  $(v_i, v_j)$ . Then, we claim that  $\Pr[v_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] = \Pr[v_j \mid \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)]$ . Let  $L_j$  represents all the latent parents of  $v_j$ . By including  $v_i$  in the intervention,

$$\begin{aligned}
& \Pr[v_j \mid \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)] \\
&= \sum_{L_j} \Pr[v_j \mid L_j, \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)] \Pr[L_j \mid \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)]. \\
&= \sum_{L_j} \Pr[v_j \mid L_j, \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)] \Pr[L_j \mid \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)].
\end{aligned} \tag{2.1}$$

As the value of  $L_j$  is only affected by conditioning on its descendants, and in the interventional distribution  $\text{do}(\text{Pa}(v_i))$ ,  $v_i$  is not a descendant of  $L_j$ , the last statement is true.

Under conditioning on  $v_i$

$$\begin{aligned}
& \Pr[v_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \\
&= \sum_{L_j} \Pr[v_j \mid L_j, v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \Pr[L_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \\
&= \sum_{L_j} \Pr[v_j \mid L_j, v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \Pr[L_j \mid \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)]. \quad (2.2)
\end{aligned}$$

The last statement is true because  $L_j \perp\!\!\!\perp v_i \mid \text{Pa}(v_j)$  in the distribution  $\text{do}(B \cup \text{Pa}(v_i))$  from Lemma 2.4.7. From the invariance principle (page 24 in [105], [88]), for any variable  $v_i$ :

$$\Pr[v_i \mid \text{Pa}(v_i)] = \Pr[v_i \mid Z, \text{do}(\text{Pa}(v_i) \setminus Z)] \text{ for any } Z \subseteq \text{Pa}(v_i)$$

Applying it to our case we get

$$\Pr[v_j \mid L_j, v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] = \Pr[v_j \mid L_j, \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)].$$

Putting this together with (2.1) and (2.2), we get  $\Pr[v_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] = \Pr[v_j \mid \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)]$ , if there is no latent  $l_{ij}$  with  $v_i \leftarrow l_{ij} \rightarrow v_j$ .  $\square$

**Lemma 2.4.9** (Lemma 2.2.9 restated). *Suppose  $v_i \rightarrow v_j \in G$  and  $v_i, v_j \notin B$ , and  $P_{ij} \subseteq B$ , then,  $\Pr[v_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \neq \Pr[v_j \mid \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)]$  if there is a latent  $l_{ij}$  with  $v_i \leftarrow l_{ij} \rightarrow v_j$ .*

*Proof.* Suppose  $v_i \rightarrow v_j$  in  $G$  and there is a latent  $l_{ij}$  between  $(v_i, v_j)$ . Then, we claim that  $\Pr[v_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \neq \Pr[v_j \mid \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)]$ . Let  $L_j$

represents all the latent parents of  $v_j$ , where  $l_{ij} \in L_j$ . Therefore,  $v_i$  is a descendant of  $L_j$ . By including  $v_i$  in the intervention,

$$\begin{aligned}
& \Pr[v_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \\
&= \sum_{L_j} \Pr[v_j \mid L_j, \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)] \Pr[L_j \mid \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)] \\
&= \sum_{L_j} \Pr[v_j \mid L_j, \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)] \Pr[L_j \mid \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)].
\end{aligned}$$

As the value of  $L_j$  is only affected by conditioning on its descendants, and in the interventional distribution  $\text{do}(\text{Pa}(v_i))$ ,  $v_i$  is not a descendant of  $L_j$ , the last statement is true. Under conditioning on  $v_i$ , we have :

$$\begin{aligned}
& \Pr[v_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \\
&= \sum_{L_j} \Pr[v_j \mid L_j, v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \Pr[L_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \\
&= \sum_{L_j} \Pr[v_j \mid L_j, v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \frac{\Pr[v_i \mid L_j, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)]}{\Pr[v_i \mid \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)]} \\
&\qquad \qquad \qquad \cdot \Pr[L_j \mid \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)]
\end{aligned}$$

From the invariance principle (page 24 in [105], [88]), we have for any variable  $v_i$

$$\Pr[v_i \mid \text{Pa}(v_i)] = \Pr[v_i \mid Z, \text{do}(\text{Pa}(v_i) \setminus Z)] \text{ for any } Z \subseteq \text{Pa}(v_i)$$

Applying it to our case we get

$$\Pr[v_j \mid L_j, v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] = \Pr[v_j \mid L_j, \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)].$$

However, since the numerator of

$$\frac{\Pr[v_i \mid L_j, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)]}{\Pr[v_i \mid \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)]}$$

depends on  $L_j$  as  $v_i$  is a descendant of  $l_{ij} \in L_j$ , whereas the denominator is not dependent on  $L_j$ , the ratio is not equal to 1 unless in pathological cases. A similar situation arises in the do-see test analysis for [88]. Hence, we have  $\Pr[v_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \neq \Pr[v_j \mid \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)]$ .  $\square$

**Proposition 2.4.10** (Proposition 2.2.10 restated). *Let  $\mathcal{G}(V \cup L, E \cup E_L)$  be a  $\tau$ -causal graph with observable graph  $G(V, E)$ . Algorithm LATENTSWEDGES with  $O(n\tau \log n + n \log n)$  many interventions recovers all latents effecting pairs of adjacent nodes in the observable graph  $G$  with probability at least  $1 - \frac{1}{n^2}$ .*

*Proof.* From Lemma 2.4.2, we know that with probability  $1 - \frac{1}{n^2}$ , for every pair  $v_i$  and  $v_j$ , there exists, with high probability, an intervention  $B \in \mathcal{B}_\tau$  such that  $v_i \in B, v_j \notin B$  and  $P_{ij} \subseteq B$ . On this  $B$ , using Lemmas 2.2.8 and 2.2.9, we can identify the latent by using a distribution test on  $B \cup \text{Pa}(v_i)$  and  $B \cup \text{Pa}(v_i) \cup \{v_i\}$ .

For every variable  $v_i \in V$ , our algorithm constructs at most  $2|\mathcal{B}_\tau|$  many interventions, given by  $\text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)$  and  $\text{do}(\text{Pa}(v_i) \cup B)$  for every  $B \in \mathcal{B}_\tau$ . Therefore, the total number of interventions used by Algorithm LATENTSWEDGES is  $O(n\tau \log n + n \log n)$ .  $\square$

#### 2.4.1.4 Removing the dependence on the exact value of $\tau$

Let  $\mathcal{G}$  be a  $\tau$ -causal graph. We assume that we know the exact value of  $\tau$  in Algorithms RECOVERG, LATENTSNEDGES, and LATENTSWEDGES. However, this assumption can be easily removed. For a fixed  $\tau$ , let  $\mathcal{G}_\tau$  be graph returned after going through all these above algorithms. Given  $\mathcal{G}_\tau$ , checking whether  $v_k$  is a  $p$ -collider for some pair  $v_i, v_j$  is simple, iterate over all paths between  $v_i$  and  $v_j$  that include  $v_k$ . Let  $\Pi = \{\pi_1, \dots, \pi_r\}$  be these paths. For each  $\pi_w \in \Pi$ , remove the edges in  $\pi_w$  from  $\mathcal{G}_\tau$  see if  $v_k$  has a descendant in  $\text{Pa}(v_i) \cup \text{Pa}(v_j)$  in this modified graph. If this holds for any path  $\pi_w \in \Pi$ , then  $v_k$  is a  $p$ -collider for the pair  $v_i, v_j$ . We describe an efficient algorithm for finding  $p$ -colliders in section 2.3.

The idea is as follows, we invoke Algorithms RECOVERG, LATENTSNEEDGES and LATENTSWEDGES for  $\tau = 1, 2, 4, \dots$ , until we find the first  $\hat{\tau}$  and  $2\hat{\tau}$  such that  $\mathcal{G}_{\hat{\tau}} = \mathcal{G}_{2\hat{\tau}}$ . We now check whether the observable nodes in  $\mathcal{G}_{\hat{\tau}}$  has at most  $\hat{\tau}$   $p$ -colliders, if so we are output  $\mathcal{G}_{\hat{\tau}}$  (and  $\hat{\tau}$ ). Otherwise, we continue by doubling  $\tau$ , i.e., by considering  $2\hat{\tau}$  and  $4\hat{\tau}$ . By increasing  $\tau$  by a constant factor, it is easy to see that process will stop in at most  $\log(2\tau)$  steps and when it stops it produces the correct observable graph  $\mathcal{G}$  and also that  $\hat{\tau} \leq 2\tau$ . Overall, this will increase the number of interventions in Theorem 2.2.11 by a factor of  $O(\log \tau)$  (to  $O(\tau^2 \log n \log \tau + n\tau \log n \log \tau)$  interventions). Through a union bound, the same success probability of  $1 - O(1/n^2)$  can be ensured by adjusting the constants.

## 2.5 Conclusion

We have studied how to recover a causal graph in the presence of latents while minimizing the interventional cost. Under the identity cost model, we gave a randomized algorithm to recover the full causal graph, through a novel characterization based on  $p$ -colliders. In this setting, understanding the optimal interventional cost is open, and an important direction for future research. While we focus on settings that are close to being non-adaptive, where all the interventions are constructed at once in the beginning, an adaptive or sequential setting has received recent attention [67, 115], and is an interesting direction for future work.

## CHAPTER 3

### CAUSAL DISCOVERY UNDER LINEAR COST MODEL

In this chapter, we describe algorithms for recovering the causal graph or the ancestral graph under the linear cost model. In section 3.1, we present the necessary preliminaries and definitions; in section 3.2, we describe our algorithms for recovering the ancestral causal graph; in section 3.3, we discuss the generalizations of separating set systems that are used for causal discovery in Chapter 2 and section 3.2.

For the subsequent portion of the chapter, we assume access to additional information, such as access to the presence of causal edges in the observable graph, but not their directions. Under such additional information, we describe algorithms for recovering the observable graph in section 3.4 and ancestral graph in section 3.5. In section 3.8, we present the conclusion.

**Overview** Suppose  $G$  is the observable graph of the causal graph  $\mathcal{G}$ , that we wish to recover. We present an algorithm that recovers the ancestral graph  $\text{Anc}(G)$  using a set of interventions that satisfy the strongly separating set system property. There might be many candidate set systems that satisfy the strongly separating property, and we argue that a greedy strategy will yield a low-cost collection of sets that is only a multiplicative factor 2 away from the optimal cost. As we do not have a handle on exactly characterizing the set systems that allow for recovering  $G$ , we consider a setting with side information. We assume access to additional knowledge about  $G$ , via access to the presence of exact set (or a superset) of edges in  $G$ , without their directions. The goal, under this setting, is to recover the directions. We argue, even

under such a restriction, that this is computationally hard, in fact, NP-hard. Unfortunately, we observe that identifying good approximation seems also intractable, under certain complexity-theoretic conjectures. To overcome these challenges, we introduce a new goal in causal discovery, where we wish to recover all but  $\epsilon$  fraction of edges of  $G$ . For this approximate recovery goal, we present efficient low-cost algorithms, by generalizing the notions of separating set systems to arbitrary graphs, and drawing connections to graph property testing.

### 3.1 Preliminaries

Following the SCM framework [105], we represent a set of random variables by  $V \cup L$  where  $V$  contains the endogenous (observed) variables that can be measured and  $L$  contains the exogenous (latent) variables that cannot be measured. We define a directed causal graph  $\mathcal{G} = \mathcal{G}(V \cup L, \mathcal{E})$  on these variables where an edge corresponds to a causal relation between the corresponding variables: a directed edge  $(v_i, v_j)$  indicates that  $v_i$  causes  $v_j$ . Throughout we denote  $n = |V|$ .

We assume that all causal relations belong to one of two categories : (i)  $E \subseteq V \times V$  containing direct causal relations between the observed variables and (ii)  $E_L \subseteq L \times V$  containing relations from latents to observable variables. Thus, the full edge set of our causal graph is  $\mathcal{E} = E \cup E_L$ . We also assume that every latent  $l \in L$  influences exactly two observed variables, i.e.,  $(l, u), (l, v) \in E_L$  and no other edges are incident on  $l$ . This *semi-Markovian* assumption is widely used in prior works [88, 117] (see discussion 2.1 for additional details). Let  $G(V, E)$  denote the subgraph of  $\mathcal{G}$  restricted to observable variables, referred to as the observable graph. The Ancestral graph of  $G$  (see Defn. 2.1.1), denoted by  $\text{Anc}(G)$  is defined over the variables  $|V|$  and contains the set of all edges  $(v_i, v_j) \in \text{Anc}(G)$  iff there is a directed path from  $v_i$  to  $v_j$  in  $G$ .

**Intervention Sets** Our primary goal is to recover either  $G$  or  $\text{Anc}(G)$  via interventions on the observable variables. We assume the ability to perform an *intervention*

on a set of variables  $S \subseteq V$  which fixes  $S = s$  for each  $s$  in the domain of  $S$ . We then perform a conditional independence test answering for all  $v_i, v_j$  “Is  $v_i$  independent of  $v_j$  in the interventional distribution  $\text{do}(S = s)$ ?” and denote it using  $v_i \perp\!\!\!\perp v_j \mid \text{do}(S)$ . Here  $\text{do}(S = s)$  uses Pearl’s do-notation to denote the interventional distribution when the variables in  $S$  are fixed to  $s$ .

An *intervention set* is a collection of subsets  $\mathcal{S} = \{S_1, \dots, S_m\}$  that we intervene on in order to recover edges of the observable or ancestral graph. It will also be useful to associate a matrix  $L \in \{0, 1\}^{n \times m}$  with the collection where the  $i$ th column is the characteristic vector of set  $S_i$ , i.e., row entry corresponding to node in  $S_i$  is 1 iff it is present in  $S_i$ . We can also think of  $L$  as a collection of  $n = |V|$  length- $m$  binary vectors that indicate which of the  $m$  intervention sets  $S_1, \dots, S_m$  each variable  $v_i$  belongs to.

As is standard, we assume that  $\mathcal{G}$  satisfies the *causal Markov condition* and assume *faithfulness* [119], both in the observational and interventional distributions following [66]. This ensures that conditional independence tests lead to the discovery of true causal relations rather than spurious associations.

**Linear Cost Model** In the Linear Cost Model, each node  $v \in V$  has a different cost  $C(v) \in [1, W]$  and the cost of intervention on a set  $S \subset V$  is defined as  $\sum_{v \in S} C(v)$ , as also considered by Kocaoglu et al. [91]. That is, interventions that involve a larger number of, or more costly nodes, are more expensive. Our goal is to find an intervention set  $\mathcal{S}$  minimizing  $\sum_{S \in \mathcal{S}} \sum_{v \in S} C(v)$  for recovering a suitable causal structure.

In our setting, we have a constraint on the the number of interventions to be upper bounded by some budget  $m$ . Without such a bound, we can observe that for ancestral graph recovery, the optimal intervention set is  $\mathcal{S} = \{\{v_1\}, \{v_2\}, \dots, \{v_n\}\}$  with cost  $\sum_{v \in V} C(v)$  as intervention on every variable is necessary, as we need to account for the possibility of latent variables (See Lemma 3.2.1 for more details). The optimality

of  $\mathcal{S}$  here follows from a characterization of any feasible set system we establish in Lemma 3.2.1. This *min cost intervention design* problem was first introduced in [86].

Letting  $L \in \{0, 1\}^{n \times m}$  be the matrix associated with an intervention set  $\mathcal{S}$ , the cost  $C(\mathcal{S})$  can be written as  $C(L) = \sum_{j=1}^n C(v_j) \cdot \|L(j)\|_1$ , where  $\|L(j)\|_1$  is the *weight* of  $L$ 's  $j^{\text{th}}$  row, i.e., the number of 1's in that row or the number of interventions in which  $v_j$  is involved.

We study two variants of causal graph recovery, in which we seek to recover the observable graph  $G$  or the ancestral graph  $\text{Anc}(G)$ . We say that an intervention set  $\mathcal{S}$  is  $\alpha$ -optimal for a given recovery task if  $C(\mathcal{S}) \leq \alpha \cdot C(\mathcal{S}^*)$ , where  $\mathcal{S}^*$  is the minimum cost intervention set needed for that task. For both recovery tasks we consider a natural approximate learning guarantee:

**Definition 3.1.1** ( $\epsilon$ -Approximate Learning). *An algorithm  $\epsilon$ -approximately learns  $G(V, E)$  (analogously,  $\text{Anc}(G)$ ) if it identifies the directions of a subset  $\tilde{E} \subseteq E$  of edges with  $|E \setminus \tilde{E}| \leq \epsilon n^2$ .*

Generally, we will seek an intervention set  $\mathcal{S}$  that lets us  $\epsilon$ -approximately learn  $G$  or  $\text{Anc}(G)$ , and which has cost bounded in terms of  $\mathcal{S}^*$ , the minimum cost intervention set needed to *fully* learn the graph. In this sense, our algorithms are bicriteria approximations.

**Independent Sets** Our intervention set algorithms will be based on finding large independent sets of variables, that can be included in the same intervention sets, following the approach of [91]. Given  $G(V, E)$ , a subset of vertices  $Z \subseteq V$  forms an independent set if there are no edges between any vertices in  $Z$ , i.e.,  $E[Z] = \emptyset$  where  $E[Z]$  is set of edges in the sub-graph induced by  $Z$ . Given a cost function  $C : V \rightarrow \mathbb{R}^+$ , an independent set  $Z$  is a maximum cost independent set (MIS) if  $C(Z) = \sum_{u \in Z} C(u)$  is maximized over all independent sets in  $G$ . Since finding MIS is

hard [43], we will use the following two notions of a MIS, with the first often referred to as simply NEAR-MIS, in our approximate learning algorithms :

**Definition 3.1.2** ( $(\gamma, \epsilon)$ -NEAR-MIS). *A set of nodes  $S \subseteq V$  is a  $(\gamma, \epsilon)$ -near-MIS in  $G = (V, E)$  if  $C(S) \geq (1 - \gamma)C(T)$  and  $|E[S]| \leq \epsilon n^2$  where  $T$  is a maximum cost independent set (MIS) in  $G$ .*

**Definition 3.1.3** ( $(\rho, \gamma, \epsilon)$ -Independent-Set). *A set of nodes  $S \subseteq V$  is a  $(\rho, \gamma, \epsilon)$ -independent-set in  $G = (V, E)$  if  $C(S) \geq \rho(1 - \gamma) \cdot C(V)$  and  $|E[S]| \leq \epsilon n^2$ .*

**Hardness of Independent Set** For the linear cost model, the problem of learning a causal graph was introduced in [86]. It was shown recently that the problem of obtaining an optimum cost set of interventions is NP-hard [91]. Under causal sufficiency (no latents), it is well known that the undirected graph (also called Essential Graph [133, 91]) recovered after running the  $IC^*$  algorithm is chordal. Further, an intervention set which is a separating set system (Def. 3.3.1) for the Essential Graph of  $G$  is both necessary and sufficient [50, 115] for learning the causal graph.

The authors of [91] give a greedy algorithm to construct a 2-approximation to the optimal cost separating set system of the essential graph. Their algorithm requires at each step finding a maximum independent set in  $G$  and peeling it off the graph, and is the basis for our approach in Section 3.4. Since  $G$  is *chordal*, there is an algorithm for finding an exact maximum independent set in polynomial time [57]. However, without the assumption of causal sufficiency, we cannot directly extend their algorithm, since finding a maximum independent set in a general graph  $G$  is NP-hard [43]. Moreover, finding an approximate independent set within a factor of  $n^\epsilon$  for any  $\epsilon > 0$  in polynomial time is also not possible unless  $NP \subseteq BPP$  [55].

## 3.2 Ancestral Graph Recovery

In this section, we describe algorithms for recovering the ancestral graph  $\text{Anc}(G)$  in the linear cost model. In section 3.2.1, we present an algorithm that recovers  $\text{Anc}(G)$  using a cost that is within a factor of 2 of the cost obtained by the optimal intervention design. In section 3.2.2 under some mild restrictions, we improve this result to  $1 + \epsilon$ , for any constant  $\epsilon > 0$ .

Recall that, given a budget of  $m$  interventions, our objective is to find a set of interventions  $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$  that can be used to identify  $\text{Anc}(G)$  while minimizing  $\sum_{S \in \mathcal{S}} \sum_{v \in S} C(v)$ . As detailed in Chapter 2, a strongly separating set system (see Defn. 2.2.1) is necessary and sufficient to recover the ancestral graph,  $\text{Anc}(G)$ . Formally, we have:

**Lemma 3.2.1** (Lemma 2.2.2 restated). *Suppose  $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$  is a collection of subsets of  $V$ . For an unknown causal graph, with observable graph  $G$ , if  $\text{Anc}(G)$  is recovered using CI-tests by intervening on the sets  $S_i \in \mathcal{S}$ . Then,  $\mathcal{S}$  is a strongly separating set system.*

Given this characterization, the problem of constructing the ancestral graph  $\text{Anc}(G)$  with minimum linear cost *reduces* to that of constructing a strongly separating set system with minimum cost. In developing our algorithm for finding such a set system, it will be useful to represent a set system by a binary matrix, with rows corresponding to observable variables  $V$  and columns corresponding to interventions (sets  $S_1, \dots, S_m$ ).

**Definition 3.2.2** (Strongly Separating Matrix). *Matrix  $U \in \{0, 1\}^{n \times m}$  is a strongly separating matrix if  $\forall i, j \in [n]$  there exists  $k, k' \in [m]$  such that  $U(i, k) = 1, U(j, k) = 0$  and  $U(i, k') = 0, U(j, k') = 1$ .*

Note that given a strongly separating set system  $\mathcal{S}$ , if we let  $U$  be the matrix where  $U(i, k) = 1$  if  $v_i \in S_k$  and 0 otherwise,  $U$  will be a strongly separating matrix. The other direction is also true. Let  $U(j)$  denote the  $j$ th row of  $U$ . Using Definition 3.2.2

and above connection between recovering  $\text{Anc}(G)$  and strongly separating set system, we can reformulate the problem at hand as:

$$\begin{aligned} \min_U \sum_{j=1}^n C(v_j) \cdot \|U(j)\|_1 & \quad (3.1) \\ \text{s.t. } U \in \{0, 1\}^{n \times m} \text{ is a strongly separating matrix.} \end{aligned}$$

We can thus view our problem as finding an assignment of vectors in  $\{0, 1\}^m$  (i.e., rows of  $U$ ) to nodes in  $V$  that minimizes (3.1). Throughout, we will call  $\|U(j)\|_1$  the *weight* of row  $U(j)$ , i.e., the number of 1s in that row. It is easy to see that  $m \geq \log n$  is necessary for a feasible solution to exist as each row must be distinct.

We start by giving a 2-approximation algorithm for (3.1). In Section 3.2.2, we show how to obtain an improved approximation under certain assumptions.

### 3.2.1 2-approximation Algorithm

In this section, we present an algorithm (Algorithm SSMATRIX) that constructs a strongly separating matrix (and a corresponding intervention set) which minimizes (3.1) to within a 2-factor of the optimum. Missing details from section are collected in Section 3.7.1.1.

**Outline** Let  $U_{\text{OPT}}$  denote a strongly separating matrix minimizing (3.1). Let  $c_{\text{OPT}} = \sum_{j=1}^n C(v_j) \|U_{\text{OPT}}(j)\|_1$  denote the objective value achieved by this optimum  $U_{\text{OPT}}$ . We start by relaxing the constraint on  $U$  so that it does *not need to be strongly separating*, but just must have unique rows, where none of the rows is all zero. In this case, we can optimize (3.1) very easily. We simply take the rows of  $U$  to be the  $n$  unique binary vectors in  $\{0, 1\}^m \setminus \{0^m\}$  with lowest weights. That is,  $m$  rows will have weight 1,  $\binom{m}{2}$  will have weight 2, etc. We then assign the rows to the nodes in  $V$  in descending order of their costs. So the  $m$  nodes with the highest costs will be

assigned the weight 1 rows, the next  $\binom{m}{2}$  assigned weight 2 rows, etc. The cost of this assignment is only lower than  $c_{\text{OPT}}$ , as we have only relaxed the constraint in (3.1).

We next convert this relaxed solution into a valid strongly separating matrix. Given  $m + \log n$  columns, we can do this easily. Since there are  $n$  nodes, in the above assignment, all rows will have weight at most  $\log n$ . Let  $\bar{U} \in \{0, 1\}^{m + \log n}$  have its first  $m$  columns equal to those of  $U$ . Additionally, use the last  $\log n$  columns as ‘row weight indicators’: if  $\|U(j)\|_1 = k$  then set  $\bar{U}(j, m + k) = 1$ . We can see that  $\bar{U}$  is a strongly separating matrix. If two rows have different weights  $k, k'$  in  $\bar{U}$ , then the last  $\log n$  columns ensure that they satisfy the strongly separating condition. If they have the same weight in  $\bar{U}$ , then they already satisfy the condition, as to be unique in  $U$  they must have at least 2 entries on which they differ.

To turn the above idea into a valid approximation algorithm that outputs  $\bar{U}$  with just  $m$  (not  $m + \log n$ ) columns, we argue that we can ‘reserve’ the last  $\log n$  columns of  $\bar{U}$  to serve as weight indicator columns. We are then left with just  $m - \log n$  columns to work with. Thus we can only assign  $m - \log n$  weight 1 rows,  $\binom{m - \log n}{2}$  weight 2 rows, etc. Nevertheless, if  $m \geq \gamma \log n$  (for a constant  $\gamma > 1$ ), this does not affect the assignment much: for any  $i$  we can still ‘cover’ the  $\binom{m}{i}$  weight  $i$  rows in  $U$  with rows of weight  $\leq 2i$ . Thus, after accounting for the weight indicator columns, each weight  $k$  row in  $U$  has weight  $\leq 2k + 1$  in  $\bar{U}$ . Overall, this gives us a 3-approximation algorithm: when  $k$  is 1 the weight of a row may become as large as 3.

To improve the approximation to a 2-approximation we *guess* the number of weight 1 vectors  $a_1$  in the optimum solution  $U_{\text{OPT}}$  and assign the  $a_1$  highest cost variables to weight 1 vectors, achieving optimal cost for these variables. There are  $O(m)$  possible values for  $a_1$  and so trying all guesses is still efficient. We then apply our approximation algorithm to the remaining  $m - a_1$  available columns of  $U$  and  $n - a_1$  variables. Since no variables are assigned weight 1 in this set, we achieve a tighter 2-approximation using our approach. The resulting matrix has the form:

---

**Algorithm 4** SSMATRIX ( $V, m$ )

---

```

1:  $c_{U_{min}} \leftarrow \infty$ 
2: for  $a_1 \in \{0, 1, \dots, 2m/3\}$  do
3:    $U \in \{0, 1\}^{n \times m}$  be initialized with all zeros
4:   Assign the highest cost  $a_1$  nodes with unit weight vectors such that  $U(i, i) = 1$  for
      $i \leq a_1$ 
5:   Set  $m' \leftarrow m - a_1$ 
6:   Mark all vectors of weight at least 1 in  $\{0, 1\}^{m' - \log n}$  as available
7:   for unassigned  $v_i \in V$  (in decreasing order of cost) do
8:     Set  $U(i, (a_1 + 1) : m - \log n)$  to smallest available weight vector in  $\{0, 1\}^{m' - \log n}$ 
     and make this vector unavailable. Let the weight of the assigned vector be  $k$ 
9:     Set ‘row weight indicator’  $U(i, m' - \log n + k) = 1$ 
10:  end for
11:  Compute cost of objective for  $U$  be  $c_U$ 
12:  if  $c_U < c_{U_{min}}$  then
13:     $c_{U_{min}} \leftarrow c_U, U_{min} \leftarrow U$ 
14:  end if
15: end for
16: Return  $U_{min}$ 

```

---

$$U = \begin{pmatrix} \mathbb{I}_{a_1} & 0 & 0 \\ 0 & C_1 & M_1 \\ 0 & C_2 & M_2 \\ \vdots & \vdots & \vdots \end{pmatrix}$$

where  $\mathbb{I}_{a_1}$  is the  $a_1 \times a_1$  identity matrix, the rows of  $C_w$  are all weight  $w$  binary vectors of length  $m - \log n - a_1$ , and the rows of  $M_w$  are length  $\log n$  binary vectors with 1’s in the  $w$ th column. The entire approach is presented in Algorithm SSMATRIX and a proof of the approximation bound in Theorem 3.2.3 is present in Section 3.7.1.1.

**Theorem 3.2.3.** *Let  $m \geq \gamma \log n$  for constant  $\gamma > 1$  and  $U$  be the strongly separating matrix returned by SSMATRIX.<sup>1</sup> Let  $c_U = \sum_{j=1}^n C(v_j) \|U(j)\|_1$ . Then,  $c_U \leq 2 \cdot c_{\text{OPT}}$ , where  $c_{\text{OPT}}$  is the objective value associated with optimum set of interventions corresponding to  $U_{\text{OPT}}$ .*

---

<sup>1</sup>In our proof,  $\gamma = 66$  but this can likely be decreased.

Using the interventions from the matrix  $U$  returned by Algorithm SSMATRIX, we obtain a cost within twice the optimum for recovering  $\text{Anc}(G)$ .

### 3.2.2 $(1 + \epsilon)$ -approximation Algorithm

Hyttinen et al. [71] presented a collection  $\mathcal{A}$  of  $m$  strongly separating intervention sets with minimum average set size, i.e.,  $\sum_{A \in \mathcal{A}} |A|/m$ . This is equivalent to minimizing the objective (3.1) in the linear cost model when the cost of intervening on any node equals 1. In this section, we analyze an adaptation of their algorithm to the general linear cost model, and obtain a  $(1 + \epsilon)$ -approximation for any given  $0 < \epsilon \leq 1$ , an improvement over the 2-approximation of Section 3.2.1. Our analysis requires mild restrictions on the number of interventions and an upper bound on the maximum cost. The algorithm will not depend on  $\epsilon$  but these bounds will. Missing details from this section are collected in Section 3.7.1.2.

**Algorithm  $\epsilon$ -SSMATRIX Outline** The famous Kruskal-Katona theorem in combinatorics forms the basis of the scheme presented in [71] for minimizing the average size of the intervention sets. To deal with with varying costs of node interventions, we augment this approach with a greedy strategy. Let  $\mathcal{A}$  denote a set of  $m$  interventions sets over the nodes  $\{v_1, v_2 \dots, v_n\}$  obtained using the scheme from [71]. Construct a strongly separating matrix  $\tilde{U}$  from  $\mathcal{A}$  with  $\tilde{U}(i, j) = 1$  iff  $v_i \in A_j$  for  $A_j \in \mathcal{A}$ . Let  $\zeta$  denote the ordering of rows of  $\tilde{U}$  in the increasing order of weight. Our Algorithm  $\epsilon$ -SSMATRIX outputs the strongly separating matrix  $U$  where, for every  $i \in [n]$ ,  $U(i) = \tilde{U}(\zeta(i))$  and the  $i$ th row of  $U$  corresponds to the node with  $i$ th largest cost.

Let  $c_{max} = \max_{v_i \in V} C(v_i)/\min_{v_i \in V} C(v_i)$  be the ratio of maximum cost to minimum cost of nodes in  $V$ . For ease of analysis, we assume that the cost of any node is least 1.

**Theorem 3.2.4.** *Let  $U$  be the strongly separating matrix returned by  $\epsilon$ -SSMATRIX.*

*If  $c_{max} \leq \frac{\epsilon n}{3 \binom{m}{t}}$  for  $0 < \epsilon \leq 1$  where  $\binom{m}{k-1} < n \leq \binom{m}{k}$  and  $t = \lfloor k - \epsilon k/3 \rfloor$ , then,*

$$c_U := \sum_{j=1}^n C(v_j) \|U(j)\|_1 \leq (1 + \epsilon) \cdot c_{OPT} ,$$

*where  $c_{OPT}$  is the objective value associated with optimum set of interventions corresponding to  $U_{OPT}$ .*

*Proof.* Suppose the optimal solution  $U_{OPT}$  includes  $a_q^*$  vectors of weight  $q$ . Let  $S$  be the  $a_1^* + a_2^* + \dots + a_t^*$  nodes with highest cost in  $U_{OPT}$ . Since  $a_q^* \leq \binom{m}{q}$ , it immediately follows that  $|S| \leq \sum_{i=q}^t \binom{m}{i}$ . However, a slightly tighter analysis (see Lemma 3.7.9) implies  $|S| \leq \binom{m}{t}$ . Let  $c_{OPT}(S)$  be the total contribution of the nodes in  $S$  to  $c_{OPT}$ . Let  $c_U(S)$  denote the sum of contribution of the nodes in  $S$  to  $c_U$  for the matrix  $U$  returned by  $\epsilon$ -SSMATRIX. Let  $\bar{k}_{|S|}$  and  $\bar{k}_n$  be the average of the smallest  $|S|$  and  $n$  respectively of the vector weights assigned by the algorithm. It is easy to observe that  $\bar{k}_{|S|} \leq \bar{k}_n$ .

$$\begin{aligned} c_U(S) &= \sum_{v_i \in S} C(v_i) \|U(i)\|_1 \leq c_{max} \sum_{v_i \in S} \|U(i)\|_1 \\ &= c_{max} \bar{k}_{|S|} |S| \leq c_{max} \bar{k}_{|S|} \binom{m}{t} \leq \epsilon \bar{k}_{|S|} n/3. \end{aligned}$$

As every node in  $V \setminus S$  receives weight at least  $t = k - \epsilon k/3$  in  $U_{OPT}$  and at most  $k$  in  $U$  returned by  $\epsilon$ -SSMATRIX, we have  $c_U(V \setminus S) \leq \frac{c_{OPT}(V \setminus S)}{1 - \epsilon/3}$ . Now, we give a lower bound on the cost of the optimum solution  $c_{OPT}(V)$ . We know that when costs of all the nodes are 1, then  $\epsilon$ -SSMATRIX achieves optimum cost denoted by  $c'_{OPT}(V)$  (see Section 3.7.1.2 for more details). As all the nodes of  $V$  have costs more than 1, we have:

$$c_{\text{OPT}}(V) \geq c'_{\text{OPT}}(V) = \bar{k}_n \cdot n \geq \bar{k}_{|S|} \cdot n.$$

Hence,

$$\frac{c_U(V)}{c_{\text{OPT}}(V)} \leq \frac{c_U(S)}{\bar{k}_{|S|}n} + \frac{c_U(V \setminus S)}{c_{\text{OPT}}(V \setminus S)} \leq \frac{\epsilon}{3} + \frac{1}{1 - \epsilon/3} \leq 1 + \epsilon.$$

This completes the proof. □

By bounding the binomial coefficients in Thm. 3.2.4, we obtain the following somewhat easier to interpret corollary:

**Corollary 3.2.5.** *If  $c_{max} \leq (\epsilon/6)n^{\Omega(\epsilon)}$  and either a)  $n^{\epsilon/6} \geq m \geq (2 \log_2 n)^{c_1}$  for some constant  $c_1 > 1$  or b)  $4 \log_2 n \leq m \leq c_2 \log_2 n$  for some constant  $c_2$  then the Algorithm  $\epsilon$ -SSMATRIX returns an  $(1 + \epsilon)$ -approximation.*

### 3.3 Separating Set Systems for Graphs

In the previous section, we discussed separating set systems as a collection of sets that satisfy a particular property for every pair of elements (or equivalently vertices). In this section, we generalize this notion by restricting the property to an input graph  $G$ . Using these generalizations, we provide constructions of intervention sets which we show in Sections 3.4 and 3.5 are necessary and sufficient for recovering observable and ancestral graphs. Missing details from this section are collected in section 3.7.2.

**Definition 3.3.1** (Separating Set System). *For any undirected graph  $G(V, E)$ , a collection of subsets  $\mathcal{S} = \{S_1, \dots, S_m\}$  of  $V$  is a separating set system if every edge  $(u, v) \in E$  is separated, i.e., there exists a subset  $S_i$  with  $u \in S_i$  and  $v \notin S_i$  or with  $v \in S_i$  and  $u \notin S_i$ .*

**Definition 3.3.2** (Strongly Separating Set System). *For any undirected graph  $G(V, E)$ , a collection of subsets  $\mathcal{S} = \{S_1, \dots, S_m\}$  of  $V$  is a strongly separating set system if every edge  $(u, v) \in E$  is strongly separated, i.e., there exist two subsets  $S_i$  and  $S_j$  such that  $u \in S_i \setminus S_j$  and  $v \in S_j \setminus S_i$ .*

We can observe that if  $G$  is the complete graph, then, the above definitions are equivalent to those discussed in Chapter 2.

For a separating set system, each pair of nodes connected in  $G$  must simply have different assigned row vectors in the matrix  $L \in \{0, 1\}^{n \times m}$  corresponding to  $\mathcal{S}$  (i.e., the rows of  $L$  form a valid coloring of  $G$ ). For a strongly separating set system, the rows must not only be distinct, but one cannot have support which is a subset of the other's. We say that such rows are *non-dominating*: there are distinct  $i, j \in [m]$  such that  $L(u, i) = L(v, j) = 0$  and  $L(u, j) = L(v, i) = 1$ . We observe that every *strongly separating set system* must satisfy the non-dominating property. When  $\mathcal{S}$  is a (strongly) separating set system for  $G$  we call its associated matrix  $L$  a (*strongly*) *separating matrix* for  $G$ .

Finding an exact minimum cost (strongly) separating set system is NP-Hard [91, 71] and thus we focus on approximation algorithms. We say the  $\mathcal{S}$  is an  $\alpha$ -optimal (strongly) separating set system if  $C(\mathcal{S}) \leq \alpha \cdot C(\mathcal{S}^*)$ , where  $\mathcal{S}^*$  is the minimum cost (strongly) separating set system. Equivalently, for matrices  $C(L) \leq \alpha \cdot C(L^*)$  where  $L, L^*$  correspond to  $\mathcal{S}, \mathcal{S}^*$  respectively.

Unfortunately, even when approximation is allowed, finding a low-cost set system for an arbitrary graph  $G$  is still hard. In particular, we prove a conditional lower bound based on the hardness of approximation for 3-coloring. Achieving a coloring for 3-colorable graphs that uses sub-polynomial colors in polynomial time is a longstanding open problem [128, 32, 80], with the current best known algorithm [14] achieving an approximation factor  $O(n^{0.2111})$ . Thus Theorem 3.3.3 shows the hardness of finding

near optimal separating set systems, barring a major breakthrough on this classical problem.

**Theorem 3.3.3.** *Assuming 3-colorable graphs cannot be colored with sub-polynomial colors in polynomial time, there is no polynomial time algorithm for finding an  $o(\log n)$ -optimal (strongly) separating set system for an arbitrary graph  $G$  with  $n$  nodes when  $m = \beta \log n$  for some constant  $\beta > 2$ .*

*Proof.* We give a proof by contradiction for the case of separating set system  $\mathcal{A}$ . A similar proof can be extended to strongly separating set systems. Suppose  $G$  is a 3-colorable graph containing  $n$  nodes with *unit* costs for every node. We argue that if there is an  $o(\log n)$ -optimal algorithm for separating set system then, we can use it to obtain an algorithm for 3-coloring of  $G$  using  $n^{o(1)}$  colors, thereby giving a contradiction.

First, we observe that the cost of an optimal separating system on  $G$  when  $m = \beta \log n$  is at most  $n$ , as each color class forms an independent set in  $G$  and every node in the color class can be assigned a vector of weight at most 1. Let  $\mathcal{A}(G)$  denote the separating set system output by an  $\alpha$ -optimal algorithm where  $\alpha = o(\log n)$ . We outline an algorithm that takes as input  $\mathcal{A}(G)$  and returns a  $n^{o(1)}$ -coloring of  $G$ .

We have  $C(\mathcal{A}(G)) \leq \alpha C(\mathcal{S}^*)$  where  $\mathcal{S}^*$  is an optimal separating set system for  $G$ . Letting  $L$  be the separating matrix associated with  $\mathcal{A}(G)$ , we thus have

$$C(\mathcal{A}(G)) = \sum_{j=1}^n \|L(j)\|_1 \leq \alpha C(\mathcal{S}^*) \leq \alpha n.$$

Using an averaging argument, we have that in  $\mathcal{A}(G)$ , there are at most  $\frac{n}{4}$  nodes (denoted by  $V \setminus D^{(1)}$ ) with weight  $\|L(j)\|_1$  more than  $4\alpha$ . Consider the remaining  $\frac{3n}{4}$  nodes given by  $D^{(1)}$ . Let  $D_j^{(1)}$  denote the nodes that have been assigned weight  $j$  by  $\mathcal{A}(G)$ . For each of the at most  $\binom{m}{j}$  vectors with weight  $j$  that are feasible, we create a new color and color each node in  $D_j^{(1)}$  using these new colors based on the weight  $j$  vectors assigned to the node in  $\mathcal{A}(G)$ . We repeat this procedure for every weight  $j$

in  $D^{(1)}$ . As the maximum weight of a node in  $D^{(1)}$  is  $4\alpha$ , the total number of colors that we use to color all the nodes of  $D^{(1)}$  is

$$\begin{aligned} \sum_{j=0}^{4\alpha} \binom{m}{j} &\leq \sum_{j=0}^{4\alpha} \frac{m^j}{j!} = \sum_{j=0}^{4\alpha} \frac{(4\alpha)^j}{j!} \left(\frac{m}{4\alpha}\right)^j \leq e^{4\alpha} \left(\frac{m}{4\alpha}\right)^{4\alpha} \leq 2^{4\alpha \log e + 4\alpha \log \frac{m}{4\alpha}} \\ &< 2^{4\alpha \log e + \sqrt{4m\alpha}} \\ &< 2^{o(\log n) + \sqrt{\log n \cdot o(\log n)}} \\ &< n^{o(1)}, \end{aligned}$$

where the first strict inequality used the fact that  $\log \frac{m}{4\alpha} \leq \sqrt{\frac{m}{4\alpha}}$  for  $\frac{m}{4\alpha} > \frac{\beta \log n}{o(\log n)} > 32$ .

After coloring the nodes of  $D^{(1)}$ , we remove these nodes from  $G$  and run  $\alpha$ -optimal algorithm  $\mathcal{A}$  on the remaining nodes  $V \setminus D^{(1)}$ . Observing that a sub-graph of a 3-colorable graph is also 3-colorable, we have that the set of nodes obtained by running  $\mathcal{A}$  on  $V \setminus D^{(1)}$  that have weight at most  $4\alpha$  (denoted by  $D^{(2)}$ ) also require at most  $n^{o(1)}$  colors. As  $|D^{(i)}| \geq \frac{3|V \setminus D^{(i-1)}|}{4}$  for all  $i \in \{1, 2, \dots, \log n\}$ , in at most  $\log n$  recursive calls to  $\mathcal{A}$ , we will fully color  $G$  using at most  $n^{o(1)} \log n = n^{o(1)}$  colors. Hence, we have obtained a  $n^{o(1)}$ -coloring of  $G$  using an  $\alpha$ -optimal algorithm when  $\alpha = o(\log n)$ .  $\square$

**Remark** The results of Theorem 3.3.3 can be extended to any  $m$ . When  $m = o(\log n)$ , in our hardness example that uses 3 colors, any valid separating set system using  $m$  interventions would lead to a coloring of the graph using at most  $2^m = n^{o(1)}$  colors, i.e., a sub-polynomial number of colors. Thus, even finding a valid separating matrix in this scenario is hard, under our assumed hardness of 3-coloring.

We shall now proceed to discuss a  $O(\log n)$  approximation algorithm for finding (strongly) separating set systems on any graph  $G$ .

**A  $2 \log n$ -Approximation Algorithm** It is easy to check that for a strongly separating set system, every node must appear in at least one intervention (because of

non-dominating property), and so the set system has cost as least  $\sum_{v \in V} C(v)$ . At the same time, with  $m \geq 2 \log n$ , we can always find a strongly separating set system where each node appears in  $\log n$  interventions. In particular, we assign each node to a unique vector with weight  $\log n$ . Such an assignment is non-dominating and since  $\binom{2 \log n}{\log n} \geq n$ , is feasible. It achieves cost  $C(\mathcal{S}) = \log n \cdot \sum_{v \in V} C(v)$ , giving a simple  $\log n$ -approximation for the minimum cost strongly separating set system problem. For a separating set system, a simple  $O(\log n)$ -approximation is also achievable by first computing an approximate minimum weight vertex cover and assigning all nodes in its complementary independent set the weight 0 vector i.e., assigning them to no interventions.

**Description of the Algorithm:**

1. Find a 2-approximate weighted vertex cover  $X$  in  $G$  using the classic algorithm from [129].
2. In  $L$ , assign zero vector to all nodes of  $V \setminus X$ ; assign every node in  $X$  with a unique vector of weight  $\log n$  and return  $L$ .

We give a sketch of the arguments involved in proving the approximation ratio of the above algorithm and defer the full details to section 3.7.2. We observe that all the nodes that are part of maximum cost independent set (complement of minimum weighted vertex cover) are assigned a weight 0 vector by optimal separating system for  $G$ . Therefore, the cost of optimal separating set system is at least the cost of minimum cost vertex cover in  $G$ . As every node is assigned a vector of weight  $\log n$  and the cost of vertex cover is at most twice the cost of the minimum weighted vertex cover, we have  $C(L) \leq 2 \log n \cdot C(L^*)$ .

By Theorem 3.3.3, it is hard to improve on the above  $O(\log n)$  approximation factor (up to constants). Therefore, we focus on finding relaxed separating set systems

in which some variables are not separated. We will see that these set systems still suffice for approximately learning  $G$  and  $\text{Anc}(G)$  under the notion of Definition 3.1.1.

**Definition 3.3.4** ( $\epsilon$ -(Strongly) Separating Set System). *For any undirected graph  $G(V, E)$ , a collection of subsets  $\mathcal{S} = \{S_1, \dots, S_m\}$  of  $V$  is an  $\epsilon$ -separating set system if, letting  $L \in \{0, 1\}^{n \times m}$  be the matrix corresponding to  $\mathcal{S}$ ,  $|\{(v_i, v_j) \in E : L(i) = L(j)\}| < \epsilon n^2$ . It is strongly separating if:*

$$|\{(v_i, v_j) \in E : L(i), L(j) \text{ are not non-dominating}\}| < \epsilon n^2.$$

For  $\epsilon$ -strongly separating set systems, when the number of interventions is large, specifically  $m \geq 1/\epsilon$ , a simple approach is to partition the nodes into  $1/\epsilon$  groups of size  $\epsilon \cdot n$ . We then assign the same weight 1 vector to nodes in the same group and different weight 1 vectors to nodes in different groups. For  $\epsilon$ -separating set system, we first find an approximate minimum vertex cover, and then apply the above partitioning. In section 3.7.2, we show that we get within a 2 factor of the optimal (strongly) separating set system. Therefore, for the remainder of this chapter, we assume  $m < 1/\epsilon$ . While  $m$  is an input parameter, smaller  $m$  corresponds to fewer interventions and this is the more interesting regime.

### 3.4 Observable Graph Recovery

In this section, we consider the setting where we are given all edges in the observable graph  $G$  (i.e., all direct causal relations between observable variables) e.g., by a domain expert, and wish to identify the direction of these edges. It is known that, assuming causal sufficiency (no latents), a separating set system is necessary and sufficient to learn  $G$  [50]. In section 3.7.3 we show that this is also the case in the *presence of latents* when we are given the edges in  $G$  but not their directions. We also show that an  $\epsilon$ -separating set system is sufficient to approximately learn  $G$  in this setting:

**Claim 3.4.1.** *Under the assumptions of section 2.1, if  $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$  is an  $\epsilon$ -separating set system for  $G$ ,  $\mathcal{S}$  suffices to  $\epsilon$ -approximately learn  $G$ .*

In particular, if  $\mathcal{S}$  is an  $\epsilon$ -separating set system, we can learn all edges in  $G$  that are separated by  $\mathcal{S}$  up to  $\epsilon n^2$  edges which are not separated. Given Claim 3.4.1, our goal becomes to find an  $\epsilon$ -separating matrix  $L_\epsilon$  for  $G$  satisfying for some small approximation factor  $\alpha$ ,  $C(L_\epsilon) \leq \alpha \cdot C(L^*)$  where  $L^*$  is the minimum cost separating matrix for  $G$ . Missing technical details of this section are collected in section 3.7.3.

We follow a similar approach as that of Kocaoglu et al. [91], observing that every node in an independent set of  $G$  can be assigned the same vector in a valid separating matrix. They show that if we greedily peel off maximum independent sets from  $G$  and assign them the lowest remaining weight vector in  $\{0, 1\}^m$  not already assigned as a row in  $L$ , we will find a 2-approximate separating matrix. Their work focuses on chordal graphs where an MIS can be found efficiently in each step. However for general graphs  $G$ , finding an MIS (even approximately) is hard (see discussion 2.1). Thus, in Algorithm 5, we modify the greedy approach and in each iteration we find a *near* independent set with cost at least as large as the true MIS in  $G$  (Def. 3.1.2). Each such set has few internal edges, this leads to few non-separating assignments between edges of  $G$  in  $L_\epsilon$ . Let  $\epsilon$  be parameter that bounds the number of non-separating edges, and  $\delta$  is the failure probability parameter of our Algorithm 5. All the error parameters are scaled appropriately (see section 3.7.3 for more details) when we pass them along in a procedure call to NEAR-MIS (line 5 in Algorithm 5).

Observe that any subset of fewer than  $\epsilon n$  nodes has at most  $\epsilon^2 n^2$  internal edges and so the NEAR-MIS  $(G, \epsilon^2, \epsilon\delta)$  routine employed in Algorithm 5 always returns at least  $\epsilon n$  nodes. Thus the algorithm terminates in  $1/\epsilon$  iterations. Across all  $1/\epsilon$  NEAR-MIS's there are at most  $\epsilon^2 n^2 \cdot 1/\epsilon = \epsilon n^2$  edges with endpoints assigned the same vector in  $L_\epsilon$ , ensuring that  $L_\epsilon$  is indeed  $\epsilon$ -separating for  $G$ .

---

**Algorithm 5**  $\epsilon$ -SEPARATING MATRIX( $G, m, \epsilon, \delta$ )

---

- 1: **Input** : Graph  $G = (V, E)$ , cost function  $C : V \rightarrow \mathbb{R}^+$ ,  $m$ , error  $\epsilon$ , and failure probability  $\delta$ .
  - 2: **Output** :  $\epsilon$ -Separating Matrix  $L_\epsilon \in \{0, 1\}^{n \times m}$ .
  - 3: Mark all vectors in  $\{0, 1\}^m$  as available.
  - 4: **while**  $|V| > 0$  **do**
  - 5:      $S \leftarrow \text{NEAR-MIS}(G, \epsilon^2, \epsilon\delta)$
  - 6:      $\forall v_j \in S$ , Set  $L_\epsilon(j)$  to smallest weight vector available from  $\{0, 1\}^m$  and mark it unavailable.
  - 7:     Update  $G$  by  $E \leftarrow E \setminus E[S]$  and  $V \leftarrow V \setminus S$ .
  - 8: **end while**
  - 9: **return**  $L_\epsilon$
- 

In Algorithm 6, we implement the NEAR-MIS routine by using the notion of a  $(\rho, \gamma, \epsilon)$ -Independent-Set (Definition 3.1.3). We find a value of  $\rho$  that achieves close to the MIS cost via a search over decreasing powers of  $(1 + \gamma)$ . In Algorithm 7 we show how to obtain a  $(\rho, \gamma, \epsilon)$ -Independent-Set (denoted by  $S$ ) whenever the cost of MIS in  $G$  is at least  $\rho \cdot C(V)$ . So,  $C(S) \geq \rho C(V) - \rho\gamma C(V)$  and we might lose a cost of at most  $\gamma\rho C(V)$  compared to the MIS cost. Therefore, we add  $\epsilon \cdot n$  nodes of highest cost (denoted by  $S_{\epsilon/2}$ ) to  $S$  and argue that by setting  $\gamma = O(\epsilon/W)$ ,  $S \cup S_{\epsilon/2}$  has a cost at least the cost of MIS, i.e.,  $S \cup S_{\epsilon/2}$  is a  $(0, \epsilon)$ -NEAR-MIS.

---

**Algorithm 6** NEAR-MIS

---

- 1: **Input** : Graph  $G(V, E)$ , cost function  $C : V \rightarrow \mathbb{R}^+$ , error  $\epsilon$ , and failure probability  $\delta$ .
  - 2: **Output** : Set of nodes that is a  $(0, \epsilon)$ -NEAR-MIS in  $G$ .
  - 3: Initialize  $\rho = 1$ , and let  $T$  be the set of  $\sqrt{\epsilon n}$  nodes in  $G$  with the highest cost.
  - 4: **while**  $\rho \geq \sqrt{\epsilon}$  **do**
  - 5:      $S \leftarrow \text{INDEPENDENT-SET}(G, \rho, \epsilon/8W, \epsilon, \delta')$  where  $\delta' = \epsilon\delta/4W \log(1/\epsilon)$
  - 6:     Let  $S_{\epsilon/2}$  denote the highest cost  $\epsilon \cdot n/2$  nodes in  $V \setminus S$ .
  - 7:     **if**  $C(S \cup S_{\epsilon/2}) \geq C(T)$  **and**  $|E[S \cup S_{\epsilon/2}]| \leq \epsilon n^2$  **then**
  - 8:         **return**  $S \cup S_{\epsilon/2}$
  - 9:     **end if**
  - 10:      $\rho = \rho/(1 + \gamma)$
  - 11: **end while**
  - 12: **return**  $T$
-

### 3.4.1 $(\rho, \gamma, \epsilon)$ – INDEPENDENT-SET

In this section, we introduce several new ideas and build upon the results for finding a  $(\rho, 0, \epsilon)$ -Independent-Set which has been used to obtain independent set property testers for graphs with unit vertex costs [62]. First, we describe an overview of the general approach.

**Unit Cost Setting** Suppose  $S$  is a fixed MIS in  $G$  with  $|S| \geq \rho \cdot n$  and  $U \subset S$ . Let  $\Gamma(u)$  represent the set of nodes that are neighbors of node  $u$  in  $G$ . Let

$$\Gamma(U) = \bigcup_{u \in U} \Gamma(u) \text{ and } \bar{\Gamma}(U) = V \setminus \Gamma(U).$$

Here,  $\bar{\Gamma}(U)$  denotes the set of nodes with no edges to any node of  $U$ . We claim that  $S \subseteq \bar{\Gamma}(U)$ . First, we observe that  $S \subseteq \bar{\Gamma}(S)$  as  $S$  is an independent set so no node in  $S$  is a neighbor of another node in  $S$  (i.e., all nodes in  $S$  are in  $\bar{\Gamma}(S)$ ). Then, we use the fact  $\bar{\Gamma}(S) \subseteq \bar{\Gamma}(U)$  since  $U \subseteq S$  to conclude  $S \subseteq \bar{\Gamma}(U)$ . Further, [62] proves that, if  $U$  is sampled randomly from  $S$ , taking the lowest degree  $\rho \cdot n$  nodes in the induced subgraph on  $\bar{\Gamma}(U)$  will with high probability yield a  $(0, \epsilon)$ -NEAR-MIS for  $G$ . Intuitively, the nodes in  $\bar{\Gamma}(U)$  have no connections to  $U$  and thus are unlikely to have many connections to  $S$ .

To find a  $U$  that is fully contained in  $S$ , we can sample a small set of nodes in  $G$ ; since we have  $|S| \geq \rho \cdot n$  the sample will contain with good probability a representative proportion of nodes in  $S$ . We can then brute force search over all subsets of this sampled set until we hit  $U$  which is entirely contained in  $S$  and for which our procedure on  $\bar{\Gamma}(U)$  returns a  $(\rho, 0, \epsilon)$ -Independent-Set, i.e., a NEAR-MIS.

**Cost setting for MIS** In the general cost setting, when  $S$  is a high cost MIS, may not contain a large number of nodes, making it more difficult to identify via sampling. To handle this, we partition the nodes based on their costs in powers

of  $(1 + \gamma)$  into  $k = O(\gamma^{-1} \log W)$  (where  $W$  is the maximum cost of a node in  $V$ ) partitions  $V_1, \dots, V_k$ .

A *good partition* is one that contains a large fraction of nodes in  $S$ : at least  $\gamma\rho|V_i|$ . Focusing on these partitions suffices to recover an approximation to  $S$ . Intuitively, all bad partitions have few nodes in  $S$  and thus ignoring nodes in them will not significantly affect the MIS cost.

**Definition 3.4.2** ( $(\gamma, \rho)$ -good partition). *Let  $S$  be an independent set in  $G$  with cost  $\geq \rho C(V)$ . Then  $F_{(\gamma, \rho)} = \{i \mid |V_i \cap S| \geq \gamma\rho|V_i|\}$  is the set of good partitions of  $V$  with respect to  $S$ .*

**Claim 3.4.3.** *Suppose  $S$  is an independent set in  $G$  with cost  $C(S) \geq \rho C(V)$ , then, there exists an independent set  $S' \subseteq S$  such that  $C(S') \geq \rho(1 - 2\gamma)C(V)$  and  $S' \cap V_i = S \cap V_i$  for all  $i \in F_{(\gamma, \rho)}$ .*

While we do not a priori know the set of good partitions, if we sample a small number  $t$  of nodes uniformly from each partition, with good probability, for each good partition we will sample  $\gamma\rho t/2$  nodes in  $S$ . We search over all possible subsets of partitions and in one iteration of our search, we have all the good partitions denoted by  $\{V_1, V_2 \dots V_\tau\}$ . Now, for such a collection of good partitions, we search over all possible subsets  $\mathcal{U} = U_1 \cup U_2 \dots \cup U_\tau$  where  $|U_i| = \gamma\rho t/2$  and in at least one instance have all  $U_i$  in good partitions fully contained in  $S$ . Let

$$Z(\mathcal{U}) := \bigcup_{i=1}^{\tau} V_i \setminus \bigcup_{i=1}^{\tau} \Gamma(U_i)$$

be the nodes in every *good* partition  $V_i$  with no connections to any of the nodes in  $U_i$ . Analogous to unit cost case, we sort the nodes in a *good* partition  $V_i$  by their degree in the induced subgraph on  $Z(\mathcal{U})$ . We select low degree nodes from each partition until the sum of the total degrees of the nodes selected is  $\epsilon n^2/k$ . We output union

of all such nodes iff it is a  $(\rho, 3\gamma, \epsilon)$ -independent set. One key difference is that while including nodes from  $Z(\mathcal{U})$ , we do not include the nodes in the sorted order until sum of degrees is  $\epsilon n^2$ . Instead, we process each *good* partition and include the nodes from each partition separately. Later, we will argue that by doing so we have made sure that the cost contribution of a particular partition is accounted for accurately.

---

**Algorithm 7**  $(\rho, \gamma, \epsilon)$  INDEPENDENT-SET

---

- 1: **Input** : Graph  $G = (V, E)$ , cost function  $C : V \rightarrow \mathbb{R}^+$ , parameters  $\rho, \gamma, \epsilon$  and  $\delta$
  - 2: **Output** :  $(\rho, 3\gamma, \epsilon)$  independent set in  $G$  if one exists.
  - 3: For  $i = 1, \dots, k$ , define  $V_i = \{v \in V \mid (1 + \gamma)^{i-1} \leq C(v) < (1 + \gamma)^i\}$  where  $k = \gamma^{-1} \log W$
  - 4: Sample  $t = O(\frac{k \log(k/\epsilon\delta)}{\epsilon\gamma\rho})$  nodes  $\tilde{V}_i$  in each partition  $V_i$ .
  - 5: **for** every collection of partitions  $\{V_1, V_2, \dots, V_\tau\} \subseteq \{V_1, V_2, \dots, V_k\}$  **do**
  - 6:     **for**  $\mathcal{U} = U_1 \cup U_2 \cup \dots \cup U_\tau$  such that  $U_i \subseteq \tilde{V}_i$  with size  $\gamma\rho t/2$  for all  $i \in [\tau]$  **do**
  - 7:         Let  $Z(\mathcal{U}) := \bigcup_{i=1}^\tau V_i \setminus \bigcup_{i=1}^\tau \Gamma(U_i)$ .
  - 8:         **for**  $i = 1 \dots \tau$  **do**
  - 9:             Sort nodes in  $Z(\mathcal{U}) \cap V_i$  in increasing order of degree in the induced graph on  $Z(\mathcal{U})$ .
  - 10:             Let  $\hat{Z}_i(\mathcal{U}) \subseteq Z(\mathcal{U}) \cap V_i$  be set of nodes obtained by considering the nodes in the sorted order until the total degree is  $\epsilon n^2/k$ .
  - 11:         **end for**
  - 12:         Let  $\hat{Z}(\mathcal{U}) = \bigcup_{i=1}^\tau \hat{Z}_i(\mathcal{U})$ .
  - 13:         **return**  $\hat{Z}(\mathcal{U})$  if  $C(\hat{Z}(\mathcal{U})) \geq \rho(1 - 3\gamma)C(V)$ .
  - 14:     **end for**
  - 15: **end for**
- 

By construction, our output, denoted by  $\hat{Z}(\mathcal{U})$  will have at most  $\epsilon n^2$  internal edges. Thus, the challenge lies in analyzing its cost. We argue that in at least one iteration, all chosen  $U_i$  for good partitions will not only lie within the MIS  $S$ , but their union will accurately represent connectivity to  $S$ . Specifically, any vertex  $v \in Z(\mathcal{U})$ , i.e., with no edges to  $U_i$  for all  $i \in F_{(\gamma, \rho)}$ , should have few edges to  $S$ . We formalize this notion using the definition of  $\epsilon_2$ -IS representative subset below.

**Definition 3.4.4** ( $\epsilon_2$ -IS representative subset).  $R \subseteq \bigcup_{i \in F_{(\gamma, \rho)}} (S \cap V_i)$  is an  $\epsilon_2$ -IS representative subset of  $S$  if for all but  $\epsilon_2 n$  nodes of good partitions, i.e.,  $\bigcup_{i \in F_{(\gamma, \rho)}} V_i$ , we have the following property:

Suppose  $v \in \bigcup_{i \in F(\gamma, \rho)} V_i$ : if  $\Gamma(v) \cap R = \emptyset$  then  $|\Gamma(v) \cap S| \leq \epsilon_2 n$ .

We show that there is a  $\epsilon_2$ -IS representative subset containing at least  $\gamma \rho t / 2$  nodes from each good partition among our sampled nodes  $\bigcup_{i=1}^k \tilde{V}_i$ . Setting  $\epsilon_2 = \epsilon / 2k$  we have:

**Lemma 3.4.5.** *If  $t = O(\frac{k \log(k/\epsilon \delta)}{\epsilon \gamma \rho})$  nodes are uniformly sampled from each partition  $V_i$  to give  $\tilde{V}_i$ , with probability  $1 - \delta$ , there exists an  $\epsilon/2k$ -IS representative subset  $R$  such that, for every  $i \in F(\gamma, \rho)$ ,  $|\tilde{V}_i \cap R| = \gamma \rho t / 2$ .*

Lemma 3.4.5 implies that in at least one iteration, our guess  $\mathcal{U}$  restricted to the good partitions is in fact an  $\epsilon/2k$ -IS representative subset. Thus, nearly all nodes in  $Z(\mathcal{U})$  lying in good partitions have at most  $\epsilon n / 2k$  edges to  $S$ .

In the graph induced by nodes of  $Z(\mathcal{U})$ , with edge set  $E[Z(\mathcal{U})]$ , consider the degree incident on nodes of  $S \cap V_i$  for each partition  $V_i$ . As there are at most  $n$  nodes in  $V_i$ , from Defn. 3.4.4, we have the total degree incident on  $S \cap V_i$  is at most  $\epsilon n^2 / k$ . Thus, including the nodes with lowest degrees in  $\hat{Z}_i(\mathcal{U})$  until the total degree is  $\epsilon n^2 / k$  will yield a set of nodes at least as large as  $S \cap V_i$ . Since all nodes in  $V_i$  have cost within a  $1 \pm \gamma$  factor of each other, we will have  $C(\hat{Z}_i(\mathcal{U})) \geq (1 - \gamma) \cdot C(S \cap V_i)$ . As the cost of  $S$  in the bad partitions is small, using Claim 3.4.3, we have  $\hat{Z}(\mathcal{U}) = \bigcup_{i=1}^r \hat{Z}_i(\mathcal{U})$  is a  $(\rho, O(\gamma), \epsilon)$ -independent set.

### 3.4.2 Greedy Algorithm: Guarantees

Overall, Algorithm 7 implements a  $(\rho, \gamma, \epsilon)$ -INDEPENDENT-SET as required by Algorithm 6 to compute a NEAR-MIS in each iteration of Algorithm 5. It just remains to show that, by greedily peeling off NEAR-MIS from  $G$  iteratively, Algorithm 5 achieves a good approximation guarantee for  $\epsilon$ -Approximate Learning  $G$ . To do this, we use a similar analysis as that of Kocaoglu et al. [91]. In their work, an exact

MIS is computed at each step, since their graph is chordal so the MIS problem is polynomial time solvable [91]. However, the analysis extends to the case when the set returned has cost that is at least the cost of MIS (in our case a NEAR-MIS), allowing us to achieve near 2-factor approximation, as achieved in [91]. Our final result is:

**Theorem 3.4.6.** *For any  $m \geq \eta \log 1/\epsilon$  for some constant  $\eta$ , with probability  $\geq 1 - \delta$ , Algorithm 5 returns  $L_\epsilon$  with  $C(L_\epsilon) \leq (2 + \exp(-\Omega(m))) \cdot C(L^*)$ , where  $L^*$  is the min-cost separating matrix for  $G$ . Moreover  $L_\epsilon$   $\epsilon$ -separates  $G$ . Algorithm 5 has a running time  $O(n^2 f(W, \epsilon, \delta))$  where*

$$f(W, \epsilon, \delta) = O\left(\frac{W}{\epsilon^2} \log \frac{1}{\epsilon} \exp\left(O\left(\frac{W^2 \log^2 W}{\epsilon^6} \log \frac{W}{\epsilon} \log \frac{W \log W \log 1/\epsilon}{\epsilon \delta}\right)\right)\right).$$

### 3.5 Recovering ancestral graph from a supergraph

In section 3.4, we assumed knowledge of the edges in the observable graph  $G$  and sought to identify their directions. In this section, we relax the assumption, assuming we are given any undirected supergraph  $H$  of  $G$  i.e., it includes all edges of  $G$  and may also include edges which do not represent causal edges. When given such a graph  $H$ , we cannot recover  $G$  itself and therefore, we seek to recover all directed edges of the ancestral graph  $\text{Anc}(G)$  appearing in  $H$  (i.e., the set of intersecting edges), which we denote by  $\text{Anc}(G) \cap H$ . This problem strictly generalizes that of section 3.4, as when  $H = G$  we have  $\text{Anc}(G) \cap H = G$ . Missing details of this section are collected in section 3.7.4.

First, we show that to recover  $\text{Anc}(G) \cap H$ , a strongly separating system (Def 3.3.1) for  $H$  is both necessary and sufficient. Furthermore, an  $\epsilon$ -strongly separating system suffices for approximate learning. We formalize this using the following lemma:

**Lemma 3.5.1.** *Under the assumptions of section 3.1, if  $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$  is an  $\epsilon$ -strongly separating set system for  $H$ ,  $\mathcal{S}$  suffices to  $\epsilon$ -approximately learn  $\text{Anc}(G) \cap H$ .*

Given Lemma 3.5.1, our goal becomes to find an  $\epsilon$ -strongly separating matrix for  $H$ ,  $L_\epsilon$  with cost within an  $\alpha$  factor of the optimal strongly separating matrix for  $H$ , for some small  $\alpha$ . To do so, our algorithm builds on the separating set system algorithm of section 3.4. We first run Algorithm 5 to obtain an  $\epsilon$ -separating matrix  $L_\epsilon^S$  and construct  $S_1, S_2, \dots, S_{1/\epsilon}$  where each set  $S_i$  contains all nodes assigned the same vector in  $L_\epsilon^S$  – i.e.,  $S_i$  corresponds to the NEAR-MIS computed at step  $i$  of Algorithm 5. We form a new graph by contracting all nodes in each  $S_i$  into a single *super node* and denote the resulting at most  $1/\epsilon$  vertices by  $V_S$ . In [5], the authors give a 2-approximation algorithm for finding a strongly separating matrix on a set of nodes, provided the graph on these nodes is complete. As  $H$  is an arbitrary super graph of  $G$ , the contracted graph on  $V_S$  is also arbitrary. However we simply assume the worst case, and run the Algorithm of [5] on it to produce  $L_\epsilon^{SS}$ , which strongly separates the complete graph on  $V_S$ . It is easy to show that as a consequence,  $L_\epsilon^{SS}$   $\epsilon$ -strongly separates  $H$ .

---

**Algorithm 8** ANCESTRAL GRAPH( $H, m, \epsilon, \delta$ )

---

- 1:  $L_\epsilon^S := \epsilon$ -SEPARATING MATRIX( $H, m, \epsilon, \delta$ ).
  - 2: Construct  $S_1, S_2, \dots, S_{1/\epsilon}$  where each set  $S_i$  contains nodes assigned the same vectors in  $L_\epsilon^S$ .
  - 3: Construct a set of nodes  $V_S$  by representing  $S_i$  as a single node  $w_i$  and  $C(w_i) = \sum_{u \in S_i} C(u)$ .
  - 4:  $L_\epsilon^{SS} :=$ SSMATRIX( $V_S, m$ ) from Chapter 2.
  - 5: **return**  $L_\epsilon^{SS}$
- 

To prove the approximation bound, we extend the result of [5], showing that their algorithm actually achieves a cost at most 2 times the cost of a *separating matrix* for the complete graph on  $V_S$  which satisfies two additional restrictions: (1) it does not assign the all zeros vector to any node and (2) it assigns the same number of weight one vectors as the optimal strongly separating matrix. Further, we show via a similar analysis to Theorem 3.4.6 that this cost on  $V_S$  is bounded by 2 times the cost of

the optimal strongly separating matrix on the contracted graph over  $V_S$ . Combining these bounds yields the final 4 approximation guarantee of Theorem 3.5.2.

**Theorem 3.5.2.** *Let  $m \geq \eta \log 1/\epsilon$  for some constant  $\eta$  and  $L_\epsilon^{SS}$  be matrix returned by Algorithm 8. Then with probability  $\geq 1 - \delta$ ,  $L_\epsilon^{SS}$  is an  $\epsilon$ -strongly separating matrix for  $H$  and  $C(L_\epsilon^{SS}) \leq (4 + \exp(-\Omega(m))) \cdot C(L^*)$  where  $L^*$  is the min-cost strongly separating matrix for  $H$ . Algorithm 8 runs in time  $O(n^2 f(W, \epsilon, \delta))$  where*

$$f(W, \epsilon, \delta) = O\left(\frac{W}{\epsilon^2} \log \frac{1}{\epsilon} \exp\left(O\left(\frac{W^2 \log^2 W}{\epsilon^6} \log \frac{W}{\epsilon} \log \frac{W \log W \log 1/\epsilon}{\epsilon \delta}\right)\right)\right).$$

### 3.6 Hyperfinite Graphs : Improved Guarantees

In this section, we show that when  $G$  has maximum degree  $\Delta$  and satisfies hyperfinite property, we can obtain the same approximation guarantees, but the number of edges that are not (strongly) separated can be improved to  $\epsilon \cdot n \cdot \Delta$ . Informally, a hyperfinite graph can be partitioned into small connected components by removing  $\epsilon \cdot n$  edges for every  $\epsilon > 0$ . Bounded degree hyperfinite graphs include the class of bounded-degree graphs with excluded minor [11], such as planar graphs, constant tree-width graphs, and also non-expanding graphs [45].

**Definition 3.6.1.** *A Graph  $G(V, E)$  is  $(\epsilon, k)$ -hyperfinite if there exists  $E' \subseteq E$  and  $|E' \setminus E| \leq \epsilon n$  such that every connected component in the induced subgraph of  $E'$  is of size at most  $k$ . A Graph  $G$  is said to be  $\tau$ -hyperfinite, if there exists a function  $\tau : \mathbb{R}^+ \rightarrow \mathbb{R}^+$  such that for every  $\epsilon > 0$ ,  $G$  is  $(\epsilon, \tau(\epsilon))$ -hyperfinite.*

If a  $\tau$ -hyperfinite graph  $G$  has maximum degree  $\Delta$ , we give algorithms for (strongly) separating set systems on  $G$  that obtain the same approximation guarantees, but the number of edges that are not (strongly) separated at most  $\epsilon \cdot n \cdot \Delta$ . In order to obtain that, we extend the additive approximation algorithm of [64] for finding the

maximum independent set to the weighted graphs i.e., when the nodes have costs and return a NEAR-MIS instead of MIS. First, we define a very important partitioning of the nodes  $V$  possible in  $\tau$ -hyperfinite graphs and give the lemma that describes the guarantees associated with finding the partitions.

**Definition 3.6.2** ([64])( $\epsilon, k$ ) partitioning oracle  $\mathcal{O}$ ). *For a given graph  $G(V, E)$  and query  $q$  about  $v \in V$ ,  $\mathcal{O}$  returns the partition  $P[v] \subseteq V$  containing  $v$  that satisfies :*

1. *for every node  $v \in V$ ,  $|P[v]| \leq k$  and  $P[v]$  is connected*
2.  *$|\{(u, w) \in E \mid P[u] \neq P[w]\}| \leq \epsilon \cdot n$  with probability  $9/10$ .*

**Lemma 3.6.3** ([64]). *If  $G$  is  $(\epsilon, \tau(\epsilon))$ -hyperfinite graph with maximum degree  $\Delta$ , then, there is a  $(\epsilon \cdot \Delta, \tau(\epsilon^3/54000))$  partition oracle that answers a given query  $q$  with probability  $1 - \delta$ , using a running time  $O(2^{\Delta^{O(\tau(\epsilon^3))}}/\delta \log 1/\delta)$ .*

Using Lemma 3.6.3, we query every node to obtain the partitioning of  $V$  and formalize this in the following corollary.

**Corollary 3.6.4.** *If  $G$  is  $(\epsilon, \tau(\epsilon))$ -hyperfinite graph with maximum degree  $\Delta$ , then, we can obtain a partitioning of the graph  $G$ , given by  $V_1, V_2, \dots$  such that with probability  $1 - \delta$  and a running time of  $O(\frac{n}{\delta} \cdot 2^{\Delta^{O(\tau(\epsilon^3/\Delta^3))}} \log 1/\delta)$ , we have :*

1. *For every  $i$ ,  $|V_i| \leq \tau(\epsilon^3/\Delta^3 54000)$  and  $V_i$  is connected*
2.  *$|\{(u, w) \mid (u, w) \in E, u \in V_i, w \in V_j \text{ and } i \neq j\}| \leq \epsilon \cdot n$*

Given a  $\tau$ -hyperfinite graph with maximum degree  $\Delta$ , we describe an algorithm that returns a set of nodes that have at most  $\epsilon n \Delta$  edges instead of  $\epsilon n^2$  edges that we saw previously for general graphs  $G$ . To do so, we build upon the previous result from [64] that returns a set of nodes  $R$  which is an additive  $\epsilon n$  approximation of MIS  $S^*$ , i.e.,  $|R| \geq |S^*| - \epsilon n$ . To obtain this, the authors first use the partitioning obtained using Lemma 3.6.3 and find MIS in each partition separately. They show

that ignoring the nodes that are incident on edges across the partitions obtained using Lemma 3.6.3 will only result in a loss of  $\epsilon n$  nodes. We observe that in Algorithm 9, by removing the  $\epsilon \cdot n$  nodes (denoted by  $\widehat{V}$ ) that are incident on the edges across partitions, and adding back  $\epsilon n$  nodes with highest cost, we will obtain a set of nodes with cost at least that of MIS while only adding  $\epsilon n \Delta$  edges amongst the combined set of nodes.

---

**Algorithm 9** NEAR-MIS in  $\tau$ -Hyperfinitive Graph  $G$

---

- 1: **Input** : Graph  $G = (V, E)$ , cost function  $C : V \rightarrow \mathbb{R}^+$ ,  $m$ ,  $\Delta$ , function  $\tau(\cdot)$ , error  $\epsilon$ , failure probability  $\delta$ .
  - 2: **Output** :  $T$  that is a NEAR-MIS with at most  $\epsilon \cdot n \cdot \Delta$  edges.
  - 3: Let the set of partitions is  $\{V_1, V_2 \dots\}$  of  $G(V, E)$  returned using Corollary 3.6.4 with parameters  $\tau(\cdot)$ , error  $\epsilon/2$  and failure probability  $\delta$ .
  - 4: **for** each partition  $V_i$  **do**
  - 5: Calculate the maximum *cost* independent set  $T_i$  in  $V_i$ .
  - 6: **end for**
  - 7:  $\widehat{E} \leftarrow \{(u, v) \mid (u, v) \in E \text{ and there exists } i, j \text{ where } i \neq j, u \in V_i, v \in V_j\}$ .
  - 8:  $\widehat{V} \leftarrow \{u \mid \exists v \text{ such that } (u, v) \in \widehat{E}\}$ .
  - 9:  $T \leftarrow (\bigcup_{i=1} T_i) \setminus \widehat{V}$ .
  - 10: Let  $H$  denote  $\epsilon \cdot n$  nodes of highest cost in  $V \setminus T$ .
  - 11: **return**  $T \cup H$ .
- 

**Lemma 3.6.5.** *In Algorithm 9, we have  $|\widehat{V}| \leq \epsilon \cdot n$  and  $C(T \cup H) \geq C(S^*)$ .*

*Proof.* From Corollary 3.6.4, we have  $|\widehat{E}| \leq \epsilon \cdot n/2$ . From the definition of  $\widehat{V}$ , we have

$$|\widehat{V}| \leq 2|\widehat{E}| \leq \epsilon \cdot n.$$

Suppose  $S^*$  is the maximum cost independent set in  $G$ . Now, consider all nodes in  $\widehat{V}$ . Similar to the above case, it is possible that  $(u, w) \in \widehat{E}$  and  $u \in T_i$ ,  $w \in T_j$  for some  $i \neq j$ . Consider a node  $u \in S$  that is isolated in  $E' \subseteq E$ , and included in some partition  $V_i$ . As  $T_i$  is maximum cost independent set in  $V_i$ , we have  $C(T_i) \geq C(S^* \cap V_i)$  where  $S^* \cap V_i$  is an independent set induced by MIS  $S^*$  in the partition  $V_i$ . Combining it for all partitions, we have  $C(\bigcup_i T_i) \geq C(S^*)$ . As nodes in  $\widehat{V}$ , it is possible that

including those that share an edge in  $\bigcup_i T_i$  will result in the set of nodes not forming an independent set. However, the set  $\bigcup_i T_i \setminus \widehat{V}$  formed by removing all the nodes that are incident with edges across the partitions, is an independent set. Since  $S^*$  is MIS, we have

$$C\left(\bigcup_i T_i \setminus \widehat{V}\right) \leq C(S^*).$$

As  $|\widehat{V}|$  is at most  $\epsilon \cdot n$ , replacing them with  $H$  consisting of  $\epsilon \cdot n$  highest cost nodes from  $T$  will only increase the cost. Therefore, we have

$$\Rightarrow C(T \cup H) = C\left(\left(\bigcup_i T_i \setminus \widehat{V}\right) \cup H\right) \geq C\left(\bigcup_i T_i\right) \geq C(S^*).$$

□

**Theorem 3.6.6.** *Algorithm 9 returns a set  $T \subseteq V$  of nodes such that  $C(T) \geq C(S^*)$  where  $S^*$  is the maximum cost independent set;  $|E[T]| \leq \epsilon \cdot n \cdot \Delta$  and uses a running time  $O\left(\frac{n}{\delta} \cdot 2^{\Delta^{O(\tau(\epsilon^3/\Delta^3))}} \log 1/\delta + n\Delta\right)$  with probability  $1 - \delta$*

*Proof.* From Lemma 3.6.5, we have  $C(T) \geq C(S^*)$  and the nodes in  $H$  include  $\epsilon \cdot n$  nodes that are added (line 10 in Algorithm 6) have at most  $\epsilon \cdot n \cdot \Delta$  edges among themselves. Therefore,  $|E[T]| \leq \epsilon \cdot n \cdot \Delta$ . Using Corollary 3.6.4, we have that it takes  $O\left(\frac{n}{\delta} \cdot 2^{\Delta^{O(\tau(\epsilon^3/\Delta^3))}} \log 1/\delta\right)$  time to find the partitions. After finding the partitions, we find maximum cost independent set in each of the at most  $n$  partitions each of size  $O(\tau(\epsilon^3/\Delta^3))$ , which takes a running time of

$$O(\text{finding maximum cost independent set in each partition}) = O(n \cdot 2^{O(\tau(\epsilon^3/\Delta^3))}).$$

Combining the running times for both these steps, along with  $O(n\Delta)$ , the time to find  $\widehat{E}$ , we have the running time as claimed. □

We can use Algorithm 9 to obtain NEAR-MIS in each iteration of Algorithm 5; from Lemma 3.7.32 and Theorem 3.6.6, we have the following proposition about *separating set system* for  $G$ .

**Proposition 3.6.7.** *Let  $G(V, E)$  be a  $\Delta$ -degree bounded  $\tau$ -hyperfinite graph. For any  $m \geq \eta \log 1/\epsilon$  for some constant  $\eta$ , with probability  $\geq 1 - \delta$ , there is an algorithm that returns  $L_\epsilon$  with  $C(L_\epsilon) \leq (2 + \exp(-\Omega(m))) \cdot C(L^*)$ , where  $L^*$  is the min-cost separating matrix for  $G$  and has a running time  $O\left(\frac{n^3}{\delta} \cdot 2^{\Delta^{O(\tau(\epsilon^3/n^3\Delta^3))}} \log \frac{n}{\delta}\right)$ . Moreover using  $L_\epsilon$ , the number of edges that are not separated in  $G$  is at most  $\epsilon \cdot n \cdot \Delta$ .*

*Proof.* In every iteration, we identify a set of nodes that has the cost at least the cost of MIS. Therefore, total number of iterations possible is at most  $n$ . Scaling the error parameter by setting  $\epsilon' = \epsilon/n$  and  $\delta' = \delta/n$  for each iteration, we have that Algorithm 5 returns  $L_\epsilon$  such that the number of edges that are not separated is  $n \cdot (\epsilon' \cdot n \cdot \Delta) = \epsilon \cdot n \cdot \Delta$ . Using Lemma 3.6.5, we have that the total running time of our algorithm is

$$O\left(n \cdot \frac{n}{\delta'} \cdot 2^{\Delta^{O(\tau(\epsilon'^3/\Delta^3))}} \log 1/\delta'\right) = O\left(\frac{n^3}{\delta} \cdot 2^{\Delta^{O(\tau(\epsilon^3/n^3\Delta^3))}} \log \frac{n}{\delta}\right).$$

□

We can obtain a similar result for *strongly separating set system* for  $G$  using Algorithm 8 and give the following proposition.

**Proposition 3.6.8.** *Let  $G(V, E)$  be a  $\Delta$ -degree bounded  $\tau$ -hyperfinite graph. For any  $m \geq \eta \log 1/\epsilon$  for some constant  $\eta$ , with probability  $\geq 1 - \delta$ , there is an algorithm that returns  $L_\epsilon$  with  $C(L_\epsilon) \leq (4 + \exp(-\Omega(m))) \cdot C(L^*)$ , where  $L^*$  is the min-cost strongly separating matrix for  $G$  and has a running time  $O\left(\frac{n^3}{\delta} \cdot 2^{\Delta^{O(\tau(\epsilon^3/n^3\Delta^3))}} \log \frac{n}{\delta}\right)$ . Moreover using  $L_\epsilon$ , the number of edges that are not strongly separated in  $G$  is at most  $\epsilon \cdot n \cdot \Delta$ .*

*Proof.* In Algorithm 8, we first find  $L_\epsilon^S$ , a separating matrix obtained using Proposition 3.6.7 that does not separate  $\epsilon \cdot n \cdot \Delta$  edges of  $G$ . Next, we find *super nodes* using the NEAR-MIS's returned and assign it vectors appropriately to form strongly separating matrix  $L_\epsilon^{SS}$  on super nodes. Using Theorem 3.7.38, we have the claimed approximation guarantee. The running time follows from Proposition 3.6.7.  $\square$

## 3.7 Additional Proof Details

### 3.7.1 Additional Proof Details from Section 3.2

In this section, we include all the missing details of proofs from Section 3.2.

#### 3.7.1.1 Proof Details from Section 3.2.1

We first argue that the matrix returned by Algorithm SSMATRIX is indeed a strongly separating matrix.

**Lemma 3.7.1.** *The matrix  $U$  returned by Algorithm SSMATRIX is a strongly separating matrix.*

*Proof.* Consider any two nodes  $v_i, v_j$  with corresponding row vectors  $U(i)$  and  $U(j)$ . Suppose  $\|U(i)\|_1 = \|U(j)\|_1$ .

By construction,  $U(i) \neq U(j)$  they will differ in at least one coordinate. However, they have equal weights, so, there must exist one more coordinate such that the strongly separating condition holds. If  $U(i)$  and  $U(j)$  have weights  $r^i \neq r^j$ , then,  $U(i, m' - \log n + r^i) = U(j, m' - \log n + r^j) = 1$  and  $U(i, m' - \log n + r^j) = U(j, m' - \log n + r^i) = 0$  by construction outlined in the Algorithm SSMATRIX. This proves that the matrix  $U$  returned is a strongly separating matrix.  $\square$

The following inequalities about Algorithm SSMATRIX will be useful in analyzing its performance.

**Lemma 3.7.2.** . For  $m \geq 66 \log n$  and  $m'$  as defined in Algorithm SSMATRIX, we have the following

- (a)  $\sum_{t=1}^{\log n} \binom{m' - \log n}{t} \geq n$  (i.e., there are enough vectors of weight  $\leq \log n$  only using  $m' - \log n$  columns to assign a unique vector to each variable).
- (b) Let  $i^*$  be the smallest integer s.t.  $\sum_{t=1}^{i^*} \binom{m'}{t} \geq n$ . Then,  $\sum_{t=1}^{2i-1} \binom{m' - \log n}{t} \geq \sum_{t=1}^i \binom{m'}{t}$  for all  $i \in \{2, \dots, i^*\}$ .

*Proof.* Let  $m \geq 66 \log n$ . From Algorithm SSMATRIX, we have  $m' = m - a_1$  for all guesses  $1 \leq a_1 \leq \frac{2m}{3}$ . By reserving the last “ $\log n$ ” columns in Algorithm SSMATRIX, we want to make sure that  $m' - \log n$  can fully cover  $n$  nodes with weight at most  $\log n$ . We have :

$$m' = m - a_1 \geq \frac{m}{3} \geq 22 \log n \text{ and}$$

$$\sum_{t=1}^{\log n} \binom{m' - \log n}{t} \geq \binom{m' - \log n}{\log n} \geq \left( \frac{21 \log n}{\log n} \right)^{\log n} > n.$$

Moving onto Part (b). Let  $i^*$  be the minimum value of  $i$  such that  $\sum_{t=1}^{i^*} \binom{m'}{t} \geq n$ . Consider  $i$  such that  $2 \leq i \leq i^*$ :

$$\sum_{t=1}^{2i-1} \binom{m' - \log n}{t} \geq \binom{m' - \log n}{2i-1}.$$

Consider now the right hand side:

$$\sum_{t=1}^i \binom{m'}{t} \leq \sum_{t=0}^i \binom{m'}{t} \leq \sum_{t=0}^i \frac{m'^t}{t!} \leq \sum_{t=0}^i \frac{i^t}{t!} \left( \frac{m'}{i} \right)^t \leq e^i \left( \frac{m'}{i} \right)^i.$$

We inductively show that for all  $i \geq 2$

$$\frac{\left(\frac{em'}{i}\right)^i}{\binom{m'-\log n}{2i-1}} \leq 1.$$

Let  $i = 2$ . For  $m \geq \frac{m'}{3} \geq 50 \left(\frac{22}{21}\right)^3$  we have

$$\frac{(em'/2)^2}{\binom{m'-\log n}{3}} \leq \frac{(em'/2)^2}{\left(\frac{m'-\log n}{3}\right)^3} \leq 50 \left(\frac{22}{21}\right)^3 \frac{m'^2}{m'^3} \leq 1.$$

Assume the inequality is correct for some  $i > 2$ . Now, we show that it must also hold for  $i + 1$ .

$$\frac{\left(\frac{em'}{i+1}\right)^{i+1}}{\binom{m'-\log n}{2i+1}} = \frac{\left(\frac{em'}{i+1}\right)^i \frac{em'}{i+1} \binom{m'-\log n}{2i-1}}{\binom{m'-\log n}{2i-1} \binom{m'-\log n}{2i+1}} \leq \frac{\left(\frac{em'}{i}\right)^i \frac{em'}{i+1} \binom{m'-\log n}{2i-1}}{\binom{m'-\log n}{2i-1} \binom{m'-\log n}{2i+1}} \leq \frac{\frac{em'}{i+1} \binom{m'-\log n}{2i-1}}{\binom{m'-\log n}{2i+1}}.$$

For ease of notation, denote  $a = m' - \log n \geq m'(1 - \frac{1}{22}) \geq 21 \log n$ .

Consider the binary entropy function  $H(x) = -x \log x - (1-x) \log(1-x)$ . For  $x \in [\frac{2i-1}{a}, \frac{2i+1}{a}]$ ,  $H(x)$  is an increasing function. For some value of  $x$  in the range we have :

$$\begin{aligned} \frac{H\left(\frac{2i+1}{a}\right) - H\left(\frac{2i-1}{a}\right)}{\frac{2i+1}{a} - \frac{2i-1}{a}} &= H'(x) = \log\left(\frac{1}{x} - 1\right) \geq \log\left(\frac{a}{2i-1} - 1\right) \\ \implies H\left(\frac{2i+1}{a}\right) - H\left(\frac{2i-1}{a}\right) &\geq \frac{2}{a} \log\left(\frac{a}{2i-1} - 1\right). \end{aligned}$$

Now, consider the fraction

$$\binom{a}{2i-1} / \binom{a}{2i+1}.$$

Using the bound from ([93], Page 309)

$$\sqrt{\frac{a}{8b(a-b)}} 2^{mH(b/a)} \leq \binom{a}{b} \leq \sqrt{\frac{a}{2\pi b(a-b)}} 2^{mH(b/a)},$$

$$\begin{aligned} \binom{a}{2i-1} / \binom{a}{2i+1} &\leq \sqrt{8(2i+1)(a-2i-1)/2\pi(2i-1)(a-2i+1)} / 2^{aH(\frac{2i+1}{a})-H(\frac{2i-1}{a})} \\ &\leq \sqrt{20/3\pi} / 2^{aH(\frac{2i+1}{a})-H(\frac{2i-1}{a})} \\ &\leq \sqrt{20/3\pi} / 2^{2\log(\frac{a}{2i-1}-1)} \\ &= \frac{\sqrt{20/3\pi}}{(\frac{a}{2i-1}-1)^2}. \end{aligned}$$

Combining the above, we have :

$$\begin{aligned} \frac{\left(\frac{em'}{i+1}\right)^{i+1}}{\binom{m'-\log n}{2i+1}} &\leq \frac{\frac{m'}{i+1} \sqrt{20e^2/3\pi}}{\left(\frac{a}{2i-1}-1\right)^2} \\ &\leq \frac{4m'i \sqrt{20e^2/3\pi}}{(a-2i)^2} \\ &\leq \frac{4m' \log n \sqrt{20e^2/3\pi}}{m'^2 (1-3/22)^2} = \frac{4 \log n \sqrt{20e^2/3\pi}}{m' (1-3/22)^2} \leq \frac{21.2 \log n}{m'} \leq 1. \end{aligned}$$

Therefore, we have for all  $i \geq 2$

$$\begin{aligned} \left(\frac{em'}{i}\right)^i / \binom{m'-\log n}{2i-1} &\leq 1 \\ \implies \sum_{t=1}^i \binom{m'}{t} &\leq \left(\frac{em'}{i}\right)^i \leq \binom{m'-\log n}{2i-1} \leq \sum_{t=1}^{2i-1} \binom{m'-\log n}{t}. \end{aligned}$$

□

Let  $c_U = \sum_{j=1}^n C(v_j) \|U(j)\|_1$  be value of objective for the matrix  $U$  returned by Algorithm SSMATRIX.

Consider  $U_{\text{OPT}}$ , and let  $V_{\text{OPT}}^{(1)}$  represent all nodes that are assigned weight 1 in it (nodes which have only one 1 in their row). Let  $c_{\text{OPT}}^{(1)}$  denote the sum of cost of the nodes in  $V_{\text{OPT}}^{(1)}$ . In our Algorithm SSMATRIX, we maintain a guess for the size of  $V_{\text{OPT}}^{(1)}$  as  $a_1$ . We want to guess the exact value of  $|V_{\text{OPT}}^{(1)}| \leq m$ . However, we only guess  $a_1$  until  $\frac{2m}{3}$ , so that the remaining columns can be used to obtain a valid separating matrix (for each of our guesses) as observed in Lemma 3.7.2. We show that the cost contribution of nodes in  $V_{\text{OPT}}^{(1)}$  (by allowing this slack in our guesses) due to Algorithm SSMATRIX is not far away from  $c_{\text{OPT}}^{(1)}$ .

First, we show that for any weight  $i \geq 2$  node in  $U_{\text{OPT}}$ , the output  $U$  of Algorithm SSMATRIX assigns vectors with weight at most  $2i$  and for a weight 1 node, we show that the weight assigned by  $U$  is at most 3.

**Lemma 3.7.3.** *Algorithm SSMATRIX assigns a weight of*

- (a) *at most 3 for a weight 1 node in  $U_{\text{OPT}}$ .*
- (b) *at most  $2i$  for a node of weight  $i$  in  $U_{\text{OPT}}$  for  $i \geq 2$ .*

*Proof.* (a) Let  $V$  denote sorted (in the decreasing order of cost) order of nodes. Suppose we assign unique length- $m$  vectors starting from weight 1 to the nodes in the order  $V$ . Let the assignment of vectors be denoted by  $\tilde{U}$ . It is easy to observe that this described assignment  $\tilde{U}$  is not a strongly separating matrix. However, any strongly separating matrix  $U$  is such that the vector assigned to any node  $v_i$  in  $U$  has weight at least that in  $\tilde{U}$  i.e.,  $\|U(i)\|_1 \geq \|\tilde{U}(i)\|_1$ . As  $U$  can be any strongly separating matrix, it also holds for  $U_{\text{OPT}}$  giving us  $\|U_{\text{OPT}}(i)\|_1 \geq \|\tilde{U}(i)\|_1$ .

The number of weight 1 nodes possible in the assignment  $\tilde{U}$  is  $\binom{m}{1}$  and therefore,  $|V_{\text{OPT}}^{(1)}| \leq m$ . Consider all the nodes of weight  $\leq 3$  in  $U$  assigned by Algorithm SSMATRIX. After discarding the first  $m' = m - a_1$  columns assuming our guess  $a_1$  in the current iteration,  $U$  starts assigning vectors with weight 1 in the remaining  $m' - \log n$  while setting a ‘row weight indicator bit’ in the last  $\log n$  columns. In order to obtain

nodes of weight  $\leq 3$ , in  $U$ , we include vectors of weight  $\leq 2$  in the  $m' - \log n$  columns.

Therefore, total number of such nodes is  $a_1 + \binom{m' - \log n}{1} + \binom{m' - \log n}{2}$ .

$$\begin{aligned} a_1 + \binom{m' - \log n}{1} + \binom{m' - \log n}{2} &\geq \binom{m' - \log n}{1} + \binom{m' - \log n}{2} \\ &\geq m' - \log n + \left(\frac{m' - \log n}{2}\right)^2, \text{ using } \binom{m'}{k} \geq \left(\frac{m'}{k}\right)^k \\ &\geq m \geq |V_{\text{OPT}}^{(1)}|, \end{aligned}$$

where the last inequality uses  $m' \geq \frac{m}{3}$  and  $m \geq 66 \log n$ .

Therefore, every weight 1 node in  $U_{\text{OPT}}$  is covered by a vector in  $U$  with weight  $\leq 3$ .

(b) First we argue that using an appropriate  $m'$ , we can give a construction of  $\tilde{U} \in \{0, 1\}^{n \times m'}$  (similar to case (a)) such that weight of node  $v_j$  in  $\tilde{U}$  is at most the weight in  $U_{\text{OPT}}$  for all nodes of weight more than 2 in  $U_{\text{OPT}}$ . Let  $m' = m - \frac{2m}{3}$ . In other words, we are considering the guess  $a_1 = \frac{2m}{3}$ . As our algorithm  $U$  considers all the guesses and returns  $U$  with the lowest cost, arguing that our lemma holds for this guess is sufficient. For this value of  $m'$ , let  $\tilde{U}$  be constructed using vectors from  $\{0, 1\}^{m'}$  in the increasing order of weight, starting with weight 1.

When  $m' = \frac{m}{3}$ , it is possible that a node in  $U_{\text{OPT}}$  can be assigned a vector of weight 1 from  $\{0, 1\}^{m'}$  (this can happen when  $|V_{\text{OPT}}^{(1)}| \geq \frac{2m}{3}$ ). As  $\tilde{U}$  assigns weights in the increasing order occupying the entire  $m'$  columns, it will not result in a strongly separating matrix. Therefore, any node  $v_j$  with weight  $i \geq 2$  in  $U_{\text{OPT}}$ , will be assigned a weight of at most  $i$  in  $\tilde{U}$ .

We know that the number of vectors of weight at most  $i$  in  $\tilde{U}$  is equal to  $\sum_{t=1}^i \binom{m'}{t}$  and number of vectors with weight at most  $2i - 1$  using  $m' - \log n$  columns of  $U$  is equal to  $\sum_{t=1}^{2i-1} \binom{m' - \log n}{t}$ . As Lemma 3.7.2 holds for all guesses of  $a_1$ , we have  $\sum_{t=1}^i \binom{m'}{t} \leq \sum_{t=1}^{2i-1} \binom{m' - \log n}{t}$  for all  $i \geq 2$ . Using induction, we can observe that  $v_j$

is assigned a vector in  $\{0, 1\}^{m' - \log n}$  with weight at most  $2i - 1$ . As  $U$  obtained from Algorithm SSMATRIX mimics the construction used in  $\tilde{U}$  over  $m' - \log n$  columns, we have that weight of node  $v_j$  in  $U$  using  $m' - \log n$  columns is at most  $2i - 1$ . Combining it with the ‘row weight indicator’ bit we set to 1 in the last  $\log n$  columns gives us the lemma.  $\square$

In our next lemma shows that the sum of contribution of the nodes in  $V_{\text{OPT}}^{(1)}$  to  $c_U$  is at most twice that of  $c_{\text{OPT}}^{(1)}$ . Combining this with Lemma 3.7.3, we show that  $U$  achieves a 2-approximation.

**Lemma 3.7.4.** *Let  $c_U^{(1)} = \sum_{v_i \in V_{\text{OPT}}^{(1)}} C(v_i) \|U(i)\|_1$  for the matrix  $U$  returned by Algorithm SSMATRIX, then  $c_U^{(1)} \leq 2c_{\text{OPT}}^{(1)}$ .*

*Proof.* Suppose  $a_1$  represents our guess for the number of weight 1 vectors and  $a_1^*$  represent the number of weight 1 vectors in  $U_{\text{OPT}}$  i.e,  $|V_{\text{OPT}}^{(1)}| = a_1^*$ . In Algorithm SSMATRIX, we use the following bounds for our guess  $0 \leq a_1 \leq 2m/3$ . If  $a_1^* \leq \frac{2m}{3}$ , then it would have been one of our guesses. As we take minimum among all the guesses, we have  $c_U^{(1)} = c_{\text{OPT}}^{(1)}$  in such a case.

Consider the case when  $a_1^* > \frac{2m}{3}$ . Let  $V_{\text{OPT}}^{(1)} = \{v_1, v_2, \dots, v_{a_1^*}\}$  represent an ordering of nodes in the decreasing ordering of cost that are assigned weight 1 in  $U_{\text{OPT}}$ . Consider the contribution of only weight 1 nodes to  $c_{\text{OPT}}$ . We have

$$c_{\text{OPT}}^{(1)} = \sum_{k=1}^{a_1^*} C(v_k) \geq \sum_{k=1}^{2m/3} \frac{2m}{3} C(v_k) \geq \frac{2m}{3} C(v_{2m/3}).$$

We will look at the case when our guess  $a_1$  reaches  $a_1 = \frac{2m}{3}$  and argue about the cost for this particular value of  $a_1$ . As we are taking minimum over all the guesses, we are only going to do better and our approximation ratio will only be better. Among the nodes  $\{v_1, v_2, \dots, v_{a_1^*}\}$  first  $\frac{2m}{3}$  nodes would be assigned weight 1 by  $U$ .

From Lemma 3.7.3, we have that for the remaining  $a_1^* - \frac{2m}{3}$  nodes, Algorithm SSMATRIX might assign a weight 2 or weight 3 vector in  $U$ .

$$\begin{aligned}
c_U^{(1)} &\leq \sum_{i=1}^{2m/3} C(v_i) + 3 \sum_{j=2m/3+1}^{a_1^*} C(v_j) = \sum_{i=1}^{a_1^*} C(v_i) + 2 \sum_{j=2m/3+1}^{a_1^*} C(v_j) \\
&\leq \sum_{i=1}^{a_1^*} C(v_i) + 2 \left( a_1^* - \frac{2m}{3} \right) C(v_{2m/3+1}) \\
&\leq c_{OPT}^{(1)} + 2 \left( a_1^* - \frac{2m}{3} \right) C(v_{2m/3}) \quad (\text{since } C(v_{2m/3}) \geq C(v_{2m/3+1})) \\
&\leq c_{OPT}^{(1)} + \frac{2m}{3} C(v_{2m/3}) \quad (\text{since } a_1^* \leq m) \\
&\leq 2c_{OPT}^{(1)}.
\end{aligned}$$

This completes the proof of the lemma.  $\square$

**Theorem 3.7.5** (Theorem 3.2.3 Restated). *Let  $m \geq 66 \log n$  and  $U$  be the strongly separating matrix returned by Algorithm SSMATRIX. Let  $c_U = \sum_{j=1}^n C(v_j) \|U(j)\|_1$ . Then,*

$$c_U \leq 2 \cdot c_{OPT},$$

where  $c_{OPT}$  is the objective value associated with optimum set of interventions corresponding to  $U_{OPT}$ .

*Proof.* From Lemma 3.7.1, we know that matrix returned by Algorithm SSMATRIX given by  $U$  with cost  $c_U$  is a strongly separating matrix. Consider a strongly separating matrix  $U_{OPT}$  that achieves optimum objective value  $c_{OPT}$ . Let  $V_{OPT}^{(1)}$  represent all nodes that are assigned weight 1 in  $U_{OPT}$ . Let  $c_U^{(1)}$  denote the cost of nodes in  $V_{OPT}^{(1)}$  using  $U$  returned by Algorithm SSMATRIX and  $c_{OPT}^{(1)}$  represents that of  $U_{OPT}$ . We have  $c_{OPT} = c_{OPT}^{(1)} + \sum_{j: \|U_{OPT}(j)\|_1 \geq 2} C(v_j) \|U_{OPT}(j)\|_1$ .

$$\begin{aligned}
c_U &= c_U^{(1)} + \sum_{j: \|U_{\text{OPT}}(j)\|_1 \geq 2} C(v_j) \|U(j)\|_1 \\
&\leq c_U^{(1)} + \sum_{j: \|U_{\text{OPT}}(j)\|_1 \geq 2} C(v_j) 2\|U_{\text{OPT}}(j)\|_1 && \text{(from Lemma 3.7.3)} \\
&\leq 2c_{\text{OPT}}^{(1)} + 2 \sum_{j: \|U_{\text{OPT}}(j)\|_1 \geq 2} C(v_j) \|U_{\text{OPT}}(j)\|_1 && \text{(from Lemma 3.7.4)} \\
&\leq 2c_{\text{OPT}}.
\end{aligned}$$

This completes the proof of the theorem. □

### 3.7.1.2 Proof Details from Section 3.2.2

In this section, we present our algorithm that achieves an improved  $1+\epsilon$ -approximation in the linear cost model setting under mild assumptions on the cost of the nodes and the number of interventions. The algorithm is adapted from that proposed by [71] whose work drew connections between causality and known separating system constructions in combinatorics. In particular, [71] considered a setting where given  $n$  variables and  $m$ , the goal is to construct  $k$  sets that are strongly separating with the objective of minimizing the average size of the intervention sets. Stated differently this provides an algorithm for solving 3.1 when  $C(v) = 1$  for all nodes  $v \in V$ .

Next, we adapt the algorithm from [71] to deal with the case where each node could have a different cost value. Our main contribution is to show that this adaptation constructs a set of interventions which achieves an objective value in the linear cost model that is within a factor  $1+\epsilon$  times of the optimum under some mild restrictions. First, we start with some definitions and statements from the combinatorics that will prove useful for stating and analyzing the algorithm.

**Definition 3.7.6.** (*antichain*). Consider a collection  $\mathcal{S}$  of subsets of  $\{v_1, v_2, \dots, v_n\}$  such that for any two sets  $S_i, S_j \in \mathcal{S}$ , we have  $S_i \not\subset S_j$  and  $S_j \not\subset S_i$ . Then, such a collection  $\mathcal{S}$  is called an *antichain*.

We provide a lemma that shows that an antichain can also be represented as a strongly separating matrix.

**Lemma 3.7.7.** *Let  $\mathcal{T} = \{T_1, T_2, \dots, T_n\}$  be an antichain defined on  $\{1, 2, \dots, m\}$ . Construct a matrix  $U \in \{0, 1\}^{n \times m}$  where  $U(i, j) = 1$  iff  $T_i$  contains  $j$ . Then  $U$  is a strongly separating matrix.*

*Proof.* From the definition of antichain, for any two sets  $T_i, T_j \in \mathcal{T}$ , there exists  $k$  and  $k'$  such that  $k \in T_i \setminus T_j$  and  $k' \in T_j \setminus T_i$ . So, we have  $U(i, k) = U(j, k') = 1$  and  $U(i, k') = U(j, k) = 0$ . It follows that  $U$  is a strongly separating matrix from the definition.  $\square$

In the previous lemma, we gave a construction of a strongly separating matrix that corresponds to an antichain. In the next lemma, we show that given a strongly separating system, we can also obtain a corresponding antichain.

**Lemma 3.7.8.** *Let  $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$  be a strongly separating set system defined on  $\{v_1, v_2, \dots, v_n\}$ . Construct a strongly separating matrix  $U \in \{0, 1\}^{n \times m}$  where  $U(i, j) = 1$  iff  $S_j$  contains  $v_i$ . Define a collection of sets  $\mathcal{T} = \{T_1, T_2, \dots, T_n\}$  defined over the column indices of  $U$  i.e.,  $\{1, 2, \dots, m\}$  such that  $j \in T_i$  iff  $U(i, j) = 1$ . Then,  $\mathcal{T}$  is an antichain.*

*Proof.* From the definition of strongly separating system, we have for every two nodes  $v_i, v_j \in \mathcal{S}$ , there exists  $S_k$  and  $S_{k'}$  such that  $v_i \in S_k \setminus S_{k'}$  and  $v_j \in S_{k'} \setminus S_k$ . This implies  $k \in T_i \setminus T_j$  and  $k' \in T_j \setminus T_i$  as  $U(i, k) = U(j, k') = 1$  and  $U(i, k') = U(j, k) = 0$ . Therefore, for every two sets  $T_i$  and  $T_j$  in  $\mathcal{T}$ , we have  $T_i \not\subset T_j$  and  $T_j \not\subset T_i$ . Hence,  $\mathcal{T}$  is an antichain.  $\square$

**Lemma 3.7.9** (LYM inequality [77]). *Suppose  $\mathcal{S}$  represent an antichain defined over the elements  $\{1, 2, \dots, m\}$ . Let  $a_k = |\{T \mid T \in \mathcal{S} \text{ where } |T| = k\}|$  defined for all  $k \in [m]$ , then,*

$$\sum_{k=0}^m \frac{a_k}{\binom{m}{k}} \leq 1.$$

**Definition 3.7.10** ([77]). *A neighbor of a binary vector  $v$  is a vector which can be obtained from  $v$  by flipping one of its 1-entries to 0. A shadow of a set  $A \subseteq \{0, 1\}^m$  of vectors is the set of all its neighbors and denoted by  $\partial(A)$ .*

Suppose  $A \subseteq \{0, 1\}^m$  consists of weight  $k$  vectors i.e., for all  $v \in A$ ,  $\|v\|_1 = k$ . Then, there is an interesting representation for  $|A|$  i.e., size of  $A$  called the  $k$ -cascade form,

$$|A| = \binom{a_k}{k} + \binom{a_{k-1}}{k-1} + \binom{a_{k-2}}{k-2} + \cdots + \binom{a_s}{s} \text{ where } a_k > a_{k-1} > \cdots > a_s \geq s \geq 1.$$

Moreover, this representation is unique and for every  $|A| \geq 1$ , there exists a  $k$ -cascade form. Given a set  $A$  of such vectors, we can make the following observation.

**Observation 3.7.11.** *Let  $B \subseteq \{0, 1\}^m$  be a collection of vectors with weight exactly  $k-1$ . If  $A \cup B$  is an antichain, then,  $B \cap \partial(A) = \phi$ .*

The above observation implies that if we want to maximize the number of weight  $k-1$  vectors to get a collection of weight  $k$  and  $k-1$  vectors that form an antichain, then, we have to choose weight  $k$  vectors that has a small shadow. Now, we describe the statement of the famous Kruskal-Katona theorem that gives a lower bound on the size of shadow of  $A$ .

**Theorem 3.7.12** (Kruskal-Katona Theorem [77]). *Consider a set  $A \subseteq \{0, 1\}^m$  of vectors such that for all  $v \in A$ ,  $\|v\|_1 = k$  and the  $k$ -cascade form is*

$$|A| = \binom{a_k}{k} + \binom{a_{k-1}}{k-1} + \binom{a_{k-2}}{k-2} + \cdots + \binom{a_s}{s}.$$

Then,

$$|\partial(A)| \geq \binom{a_k}{k-1} + \binom{a_{k-1}}{k-2} + \binom{a_{k-2}}{k-2} + \cdots + \binom{a_s}{s-1}.$$

**Definition 3.7.13.** (*Colexicographic Ordering*) Let  $u$  and  $v$  be two distinct vectors from  $\{0, 1\}^m$ . In the colexicographic ordering  $u$  appears before  $v$  if for some  $i$ ,  $u(i) = 0, v(i) = 1$  and  $u(j) = v(j)$  for all  $j > i$ .

We now state a result that the colexicographic ordering (or colex order) of all vectors of  $\{0, 1\}^m$  achieves the Kruskal-Katona theorem lower bound. Therefore, we can generate a sequence of any number of vectors with weight  $k$  that has the smallest possible shadow.

**Lemma 3.7.14.** *Proposition 10.17 from [77]. Using the first  $T$  of weight  $k$  vectors in the colex ordering of  $\{0, 1\}^m$ , we can obtain a collection  $A \subseteq \{0, 1\}^m$  such that  $|\partial(A)| = \binom{a_k}{k-1} + \binom{a_{k-1}}{k-2} + \binom{a_{k-2}}{k-2} + \dots + \binom{a_s}{s-1}$  where the  $k$ -cascade form of  $T = |A| = \binom{a_k}{k} + \binom{a_{k-1}}{k-1} + \binom{a_{k-2}}{k-2} + \dots + \binom{a_s}{s}$ .*

We state the Flat Antichain theorem, that we will use later.

**Theorem 3.7.15** (Flat Antichain Theorem). [85] *If  $\mathcal{A}$  is an antichain, then, there exists another antichain  $\mathcal{B}$  defined over same elements, such that  $|\mathcal{A}| = |\mathcal{B}|$ ,  $\sum_{A \in \mathcal{A}} |A| = \sum_{B \in \mathcal{B}} |B|$  and for every  $B \in \mathcal{B}$ , we have  $|B| \in \{d-1, d\}$  for some positive integer  $d$ .*

Algorithm  $\epsilon$ -SSMATRIX is an adaptation of Algorithm 4 of [71] for the linear cost model setting. From Lemma 3.7.8 and 3.7.7, it is clear that constructing a strongly separating set system is equivalent to constructing an antichain. A consequence of Flat Antichain theorem [85] is that for every antichain  $\mathcal{A}$  there is another antichain  $\mathcal{B}$  of same size such that  $\sum_{A \in \mathcal{A}} |A| = \sum_{B \in \mathcal{B}} |B|$  and  $\mathcal{B}$  has sets of cardinality either  $d$  or  $d-1$  for some positive integer  $d$ . Therefore, the problem of finding a separating set system reduces to finding an appropriate antichain with weights  $d$  and  $d-1$  that minimizes the objective (assuming all nodes have cost equal to 1).

**Corollary 3.7.16.** [71] *Flat Antichain theorem implies Algorithm 4 achieves optimal cost assuming all nodes have unit costs.*

Algorithm 4 of [71] is a consequence of Kruskal-Katona theorem; using colexicographic ordering we can maximize the  $d - 1$  weight vectors in an antichain of size  $n$  consisting of weight  $d$  and  $d - 1$  vectors. Therefore, choosing  $d = k$  where  $\binom{m}{k-1} < n \leq \binom{m}{k}$ , they consider all possible number of vectors of weight  $k$  and find the one with the minimum number of weight  $k$  vectors. However, unlike [71], we have to deal with different costs of intervention for each node. We adopt a greedy strategy, where we assign the vectors (obtained using the previous algorithm) in the increasing order of weight to the nodes in the decreasing order of their costs. Observe that our Algorithm  $\epsilon$ -SSMATRIX assigns vectors of weight  $k - 1$  or  $k$  that are relatively high to the nodes with large costs. Surprisingly, we show that when the costs are bounded by  $\approx \epsilon n^\epsilon$ , and number of interventions  $m \leq n^\epsilon$ , it achieves a  $1 + \epsilon$ -approximation.

---

**Algorithm 10**  $\epsilon$ -SSMATRIX ( $V, m$ )

---

- 1: Let  $\tilde{U} \in \{0, 1\}^{n \times m}$  be initialized with all zeros
  - 2: Find  $k$  satisfying  $\binom{m}{k-1} < n \leq \binom{m}{k}$
  - 3: **for**  $t = 0$  to  $n$  **do**
  - 4: Let  $A_t$  denote the first  $t$  vectors in the colex ordering of  $\{0, 1\}^m$  with weight  $k$ . Calculate  $|\partial(A_t)|$  using Lemma 3.7.14.
  - 5: **if**  $t - |\partial(A_t)| + \binom{m}{k-1} \geq n$  **then**
  - 6: For the rows  $\tilde{U}(j)$  with  $n - t + 1 \leq j \leq n$  assign the vectors of weight  $k$  using  $A_t$
  - 7: For the rows  $\tilde{U}(j)$  with  $j \leq n - t$ , assign vectors of weight  $k - 1$  from  $\{0, 1\}^m$  that are not contained in  $\partial(A_t)$
  - 8: **break**;
  - 9: **end if**
  - 10: **end for**
  - 11: Let  $\zeta$  denote the ordering of rows of  $\tilde{U}$  in the increasing order of weight.
  - 12: For every  $i \in [n]$  assign  $U(i) = \tilde{U}(\zeta(i))$  where  $i^{\text{th}}$  row of  $U$  corresponds to the node with  $i^{\text{th}}$  largest cost.
  - 13: **Return**  $U$
- 

**Lemma 3.7.17.** *Let  $U$  represent the output of Algorithm  $\epsilon$ -SSMATRIX. Then,  $U$  is a strongly separating matrix.*

*Proof.* From Observation 3.7.11, we have that our set of weight  $k$  vectors  $A_t$  and set of weight  $k - 1$  vectors given by  $B_t = A_t \setminus \partial(A_t)$  satisfy  $B_t \cap \partial(A_t) = \phi$ . So, the collection  $A_t \cup B_t$  is an antichain. In Algorithm  $\epsilon$ -SSMATRIX,  $U$  and  $\tilde{U}$  contain the

same collection of vectors, only differing in the ordering  $\zeta$ . From Lemma 3.7.7, we have  $U$  constructed from  $A_t \cup B_t$  is a strongly separating matrix.  $\square$

The following lemma follows from LYM inequality in Lemma 3.7.9.

**Lemma 3.7.18.** *Let  $U_{\text{OPT}}$  represent the optimum solution with  $a_q^*$  representing the number of rows of  $U$  with weight  $q$ . Then, for any  $t \leq n$  :*

$$\sum_{q=1}^t a_q^* \leq \binom{m}{t}.$$

*Proof.* From Lemma 3.7.17 and Corollary 3.7.16, we know that the matrix  $U_{\text{OPT}}$  is a strongly separating matrix. Therefore, using Lemma 3.7.8, we can construct a collection  $\mathcal{T}$  defined over  $\{1, 2, \dots, m\}$  such that  $\mathcal{T}$  is an antichain.  $T_i \in \mathcal{T}$  corresponds to a row of  $U_{\text{OPT}}$  and  $|T_i| = \|U_{\text{OPT}}(i)\|_1$  represents the weight of  $i^{\text{th}}$  row of  $U$ . Applying LYM inequality from Lemma 3.7.9 gives us:

$$\sum_{q=1}^t \frac{a_q^*}{\binom{m}{t}} \leq \sum_{q=1}^t \frac{a_q^*}{\binom{m}{q}} \leq \sum_{q=0}^m \frac{a_q^*}{\binom{m}{q}} \leq 1$$

and so  $\sum_{q=1}^t a_q^* \leq \binom{m}{t}$ .  $\square$

The next lemma gives an upper bound for  $\binom{m}{t}$  that can be used to simplify the statement of the Theorem 3.2.4.

**Lemma 3.7.19.** *If  $6/k \leq \epsilon \leq 1/2$  and  $m \geq 2 \log_2 n$ :*

$$\binom{m}{t} \leq 2n \cdot 2^{-(\epsilon k/6) \log_2(m/(2k))}.$$

*Proof.* By the definition of  $k$ ,

$$\binom{m}{t} = \binom{m}{k-1} \binom{m}{t} / \binom{m}{k-1} < n \binom{m}{t} / \binom{m}{k-1}.$$

Let  $H(x)$  denote the binary entropy function. Note that  $t = \lfloor k - \epsilon k/3 \rfloor$ . Therefore,

$$(k-1) - t \geq k-1 - k + \epsilon k/3 = \epsilon k/3 - 1 \geq \epsilon k/6,$$

and that for all  $x \in [t/m, (k-1)/m]$ ,

$$H'(x) \geq H'\left(\frac{k-1}{m}\right) = \log_2\left(\frac{m}{k-1} - 1\right) \geq \log_2\left(\frac{m}{2k}\right),$$

where we used the assumption  $t/m \leq (k-1)/m \leq 1/2$ . Hence,

$$|H((k-1)/m) - H(t/m)| \geq \frac{\epsilon k/6}{m} \log_2\left(\frac{m}{2k}\right).$$

Using the bound from ([93], Page 309)

$$\sqrt{\frac{a}{8b(a-b)}} 2^{mH(b/a)} \leq \binom{a}{b} \leq \sqrt{\frac{a}{2\pi b(a-b)}} 2^{mH(b/a)},$$

we get that

$$\binom{m}{t} / \binom{m}{k-1} \leq 2^{m(H(t/m) - H((k-1)/m))} \sqrt{\frac{8(k-1)(m-k+1)}{2\pi t(m-t)}} \leq 2 \cdot 2^{-(\epsilon k/6) \log_2(m/(2k))}$$

where the last inequality used  $\epsilon \leq 1$ .

□

**Corollary 3.7.20.** (Corollary 3.2.5 Restated). Algorithm  $\epsilon$ -SSMATRIX is a  $(1 + \epsilon)$ -approximation if the maximum cost satisfies

$$c_{max} \leq \epsilon/6 \cdot 2^{(\epsilon k/6) \log_2(m/(2k))}$$

assuming  $n^{\epsilon/6} \geq m \geq 2 \log_2 n$ . If a)  $m \geq (2 \log_2 n)^{c_1}$  for some constant  $c_1 > 1$  or b)  $4 \log_2 n \leq m \leq c_2 \log_2 n$  for some constant  $c_2$  then the RHS bound is at least  $\epsilon/6 \cdot n^{\Omega(\epsilon)}$ .

*Proof.* First note that  $k \geq \log_m n$  since

$$m^k \geq \binom{m}{k} \geq n.$$

When  $2 \log n \leq m \leq n^\epsilon$ , we have  $k \geq \log_m n \geq \frac{6}{\epsilon}$ . From the previous lemma 3.7.19,

$$c_{max} \leq \epsilon/6 \cdot 2^{(\epsilon k/6) \log_2(m/(2k))} \leq \epsilon n/3 \binom{m}{t}$$

Using Theorem 3.2.4, we have that Algorithm  $\epsilon$ -SSMATRIX is a  $(1 + \epsilon)$ -approximation.

We next consider the simplification in Part (a). If  $m \geq (2 \log_2 n)^{c_1}$  for some  $c_1 > 1$  then  $k \leq \log_2 n$  as  $\binom{m}{\log_2 n} \geq n$ . So  $2k \leq 2 \log_2 n \leq m^{1/c_1}$ . Hence,

$$\log_2(m/(2k)) \geq (1 - 1/c_1) \log_2 m$$

and so

$$2^{(\epsilon k/6) \log_2(m/(2k))} \geq 2^{(\epsilon k(1-1/c_1)(\log_2 m)/6)} \geq n^{\frac{\epsilon(1-1/c_1)}{6}}.$$

where the last inequality follows since  $k \geq \log_m n$ .

We next consider the simplification in Part (b). Now suppose  $m \leq c_2 \log_2 n$  for some constant  $c_2 \geq 2$  then,  $k \geq \log_{ec_2} n$  since

$$(c_2 e)^k \geq (me/k)^k \geq \binom{m}{k} \geq n .$$

Note that for  $m \geq 4 \log n$ ,

$$\log_2(m/(2k)) \geq \log_2(4 \log n / (2 \log_2 n)) \geq 1$$

and so

$$2^{(\epsilon k/6) \log_2(m/(2k))} \geq 2^{(\epsilon k/6)} \geq n^{\frac{\epsilon}{6 \log_{ec_2} 2}} .$$

□

### 3.7.2 Proof Details from Section 3.3

In this section, we present missing details from section 3.3, about various efficient constructions of separating set systems.

#### 3.7.2.1 $2 \log n$ Approximation Algorithm for Separating Set System

In this section, we show that the algorithm presented in section 3.3 obtains a  $2 \log n$ -optimal separating set system for a given graph  $G$ . To do so, we first make the following two simple claims. Let  $\mathcal{S}^* = \{S_1, S_2, \dots, S_m\}$  be the minimum cost separating set system for  $G$  and  $I$  denote the maximum cost independent set in  $G$ .

**Definition 3.7.21.** (*Vertex Cover*). *A set of nodes  $S$  is a vertex cover for the graph  $G(V, E)$ , if for every edge  $(u, v) \in E$ , we have  $\{u, v\} \cap S \neq \phi$ .*

**Claim 3.7.22.** *The set of vertices in  $V \setminus I$  forms a minimum weighted vertex cover for  $G$ .*

*Proof.* Suppose  $X$  denote a minimum weighted vertex cover in  $G$ , then,  $V \setminus X$  is an independent set in  $G$ . We have  $C(X) = C(V) - C(V \setminus X) \geq C(V) - C(I)$  as  $I$  is maximum cost independent set. Observe that the vertex cover given by  $X := V \setminus I$  satisfies the above equation with equality. Hence, the claim.  $\square$

**Claim 3.7.23.**  $C(\mathcal{S}^*) \geq C(V \setminus I)$

*Proof.* Let  $L^*$  denote optimal separating matrix corresponding to  $\mathcal{S}^*$ . We can rewrite  $C(\mathcal{S}^*)$  in terms of  $C(L^*) = \sum_{j=1}^n C(v_j) \|L(j)\|_1$ . It is easy to observe that every node in an independent set of  $G$  can be assigned the same vector in a separating matrix. So, nodes with weight zero in  $L^*$  are from an independent set (say  $I_{L^*}$ ) in  $G$ . As weight of the vectors assigned to remaining nodes in  $L^*$  is at least 1, we have  $C(L^*) \geq C(V) - C(I_{L^*}) \geq C(V) - C(I)$ , using the definition of  $I$ .  $\square$

Combining Claims 3.7.22 and 3.7.23, we can observe that a good approximation for weighted vertex cover will result in a good approximation for separating set system. There is a well known 2-approximation algorithm for weighted vertex cover problem using linear programming that runs in polynomial time (Page 10, Theorem 1.6 [129]).

**Lemma 3.7.24.** *If  $m \geq 2 \log n$ , then, there is an algorithm that returns a separating set system that is  $2 \log n$ -optimal.*

*Proof.* Let  $X$  denote the minimum weighted vertex cover which is a 2-approximation obtained using the well known linear programming relaxation [129]. In our algorithm, we assign every node in  $X$  with a unique vector of weight  $\log n$ . This is feasible because the set of nodes in  $V \setminus X$  form an independent set, and  $\binom{m}{\log n} \geq \binom{2 \log n}{\log n} \geq n$ . Combining Claims 3.7.22 and 3.7.23, we have

$$C(L) = \log n C(X) \leq 2 \log n C(V \setminus I) \leq 2 \log n C(\mathcal{S}^*).$$

$\square$

### 3.7.2.2 Algorithms for $\epsilon$ -(Strongly) Separating Set System when $m \geq 1/\epsilon$

**$\epsilon$ -Separating Set System** For  $\epsilon$ -separating set system on  $G(V, E)$ , we first find a 2-approximate minimum weighted vertex cover  $X$  using the well-known linear programming based algorithm from [129] (Refer Page 10, Theorem 1.6 in [129]). We then partition the nodes of  $X$  randomly into  $1/\epsilon$  groups of expected size  $\epsilon \cdot n$ . We then assign the same weight 1 vector to nodes in the same group and different weight 1 vectors to nodes in different groups. This is possible since  $m \geq 1/\epsilon$ . It is easy to see that the total number of edges that are not separated on expectation is  $\epsilon|E| \leq \epsilon n^2$ . For the remaining nodes in  $V \setminus X$  that form an independent set, we assign the zero vector. Therefore, total cost of  $\epsilon$ -separating set system is given by  $C(X)$ . From Claim 3.7.22, we have  $C(X) \leq 2C(V \setminus I)$  where  $I$  is maximum weighted independent set in  $G$ . Using Claim 3.7.23, we have  $C(X) \leq 2C(\mathcal{S}^*)$  where  $\mathcal{S}^*$  is optimal separating set system for  $G$ . Therefore, we get within a 2 factor of the optimal separating set system.

**$\epsilon$ -Strongly Separating Set System** For  $\epsilon$ -strongly separating set system on  $H(V, E)$ , we partition the nodes randomly into  $1/\epsilon$  groups of expected size  $\epsilon \cdot n$ . We then assign the same weight 1 vector to nodes in the same group and different weight 1 vectors to nodes in different groups. This is possible since  $m \geq 1/\epsilon$ . It is easy to see that the total number of edges that are not strongly separated on expectation is  $\epsilon|E| \leq \epsilon n^2$ . As every vector assigned to a node in a valid strongly separating matrix should have weight at least 1, this results in an  $\epsilon$ -strongly separating matrix, and the corresponding set system with optimal cost.

### 3.7.3 Proof Details from Section 3.4

**Claim 3.7.25.** *Suppose a set on interventions  $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$  is used for learning the edges of an undirected causal graph  $G$ . Then, under the assumptions of section 3.1,  $\mathcal{S}$  is a separating set system for  $G$ .*

*Proof.* First, we show that when  $\mathcal{S}$  is a separating set system for  $G$ , we can recover the directions of  $G$ . Consider an edge  $(v_i, v_j) \in G$  and let  $S_k \in \mathcal{S}$  be such that  $v_i \in S_k$  and  $v_j \notin S_k$ . As  $\mathcal{S}$  is a separating set system, we know that such a set  $S_k$  exists for every edge in  $G$ . Consider the CI-test between  $v_i$  and  $v_j$  in the interventional distribution  $\text{do}(S_k)$ . If the test returns that  $v_i \perp\!\!\!\perp v_j \mid \text{do}(S_k)$ , then, we infer  $v_i \rightarrow v_j$ , otherwise we infer that  $v_i \leftarrow v_j$ . When we intervene on  $v_i$  obtained by  $\text{do}(S_k)$ , the latent edges affecting  $v_i$  and all other incoming edges to  $v_i$  are removed. As we know that there is a causal edge between the two variables, if the independence test returns true, it must mean that there is no incoming edge into  $v_i$  from  $v_j$ .

In [50], it was shown that a separating set system is necessary for learning the directions among the observable variables assuming causal sufficiency. As we are trying to recover  $G$  using interventions, such a condition will also hold for our case that is a generalization when not assuming causal sufficiency. Hence, the claim.  $\square$

**Claim 3.7.26.** (Claim 3.4.1 restated) *Under the assumptions of Section 3.1, if  $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$  is an  $\epsilon$ -separating set system for  $G$ ,  $\mathcal{S}$  suffices to  $\epsilon$ -approximately learn  $G$ .*

*Proof.* Given  $\mathcal{S}$  denotes an  $\epsilon$ -separating set system for  $G(V, E)$ . So, there are at most  $\epsilon n^2$  edges  $(u, v) \in E$  such that for all  $i \in [m]$ , either  $\{u, v\} \cap S_i = \phi$  or  $\{u, v\} \cap S_i = \{u, v\}$ . For every such edge, any intervention on a set in  $\mathcal{S}$ , say  $S_i$  cannot recover the direction from a CI-test  $u \perp\!\!\!\perp v \mid \text{do}(S_i)$ ? because for both the cases  $u \leftarrow v$  or  $u \rightarrow v$ , the CI-test returns that they are dependent. For the remaining edges  $(u, v) \in E$ , in the intervention  $S_j$  where  $\{u, v\} \cap S_j = \{u\}$ , we can recover the direction using the CI-test :  $u \rightarrow v$  if  $u \not\perp\!\!\!\perp v \mid \text{do}(S_j)$  and  $u \leftarrow v$  otherwise. From Def. 3.1.1, we have that  $\mathcal{S}$   $\epsilon$ -approximately learns  $G$ .  $\square$

**Claim 3.7.27.** (Claim 3.4.3 restated) *Suppose  $S$  is an independent set in  $G$  with cost  $C(S) \geq \rho C(V)$ , then, there exists an independent set  $S' \subseteq S$  such that  $C(S') \geq \rho(1 - 2\gamma)C(V)$  and  $S' \cap V_i = S \cap V_i$  for all  $i \in F_{(\gamma, \rho)}$ .*

*Proof.* Construct  $S'$  using  $(\gamma, \rho)$ -good partitions of  $V$ . For every  $i \in F_{(\gamma, \rho)}$ , include  $S \cap V_i$  in  $S'$ . Therefore, we have

$$\begin{aligned}
C(S') &= C(S) - \sum_{i \notin F_{(\gamma, \rho)}} C(S \cap V_i) \\
&\geq \rho C(V) - \gamma \rho \sum_{i \notin F_{(\gamma, \rho)}} |V_i| (1 + \gamma)^i \\
&\geq \rho C(V) - \gamma \rho (1 + \gamma) \sum_{i \notin F_{(\gamma, \rho)}} |V_i| (1 + \gamma)^{i-1} \\
&\geq \rho C(V) - \gamma \rho (1 + \gamma) C(V) \\
&\geq \rho (1 - 2\gamma) C(V).
\end{aligned}$$

□

**Lemma 3.7.28.** (Lemma 3.4.5 restated) *If  $t = O(\frac{k}{\epsilon \gamma \rho} \log \frac{4k}{\epsilon \delta})$  nodes are uniformly sampled from each partition  $V_i$  to give  $\tilde{V}_i$ , with probability  $1 - \delta$ , there exists an  $\epsilon/2k$ -IS representative subset  $R$  such that, for every  $i \in F_{(\gamma, \rho)}$ ,  $|\tilde{V}_i \cap R| = \gamma \rho t/2$ .*

*Proof.* Consider a good partition  $V_i$  for some  $i \in F_{(\gamma, \rho)}$ . So,  $|V_i \cap S| \geq \gamma \rho |V_i|$ . As  $\tilde{V}_i$  consists of  $t$  nodes that are uniformly sampled from  $V_i$ , using Hoeffding's inequality ([19, 69]), we know that  $|\tilde{V}_i \cap S| \geq \gamma \rho t/2$  with probability at least

$$1 - \exp(-\gamma \rho t/8) \geq 1 - \exp\left(-\frac{k}{\epsilon} \log \frac{4k}{\epsilon \delta}\right) \geq 1 - \frac{\delta}{2k}.$$

Applying union bound, we have for every  $i \in F_{(\gamma, \rho)}$ ,  $|\tilde{V}_i \cap S| \geq \gamma \rho t/2$  with probability at least

$$1 - k \frac{\delta}{2k} \geq 1 - \frac{\delta}{2}.$$

Consider the union of all subsets  $U_j \subseteq \tilde{V}_j \cap S$  of good partitions such that  $|U_j| = \gamma \rho t/2$ , i.e.,

$$R = \bigcup_{j=1 \mid j \in F_{(\gamma, \rho)}}^k U_j.$$

We claim that  $R$  is a  $\epsilon/2k$ -IS representative subset of  $V$  by arguing that if  $v$  has no neighbours in  $R$ , then, the degree to  $S$  is more than  $\epsilon n/2k$  with *low* probability.

First, consider the case when  $v \in S$ , then  $|\Gamma(v) \cap S| = 0$  and  $\Gamma(v) \cap R = \phi$ . Suppose  $v \in V_j \setminus S$  for some  $j \in F_{(\gamma, \rho)}$  and  $|\Gamma(v) \cap S| \geq \epsilon n/2k$ . If  $\Gamma(v) \cap R = \phi$ , then  $\Gamma(v) \cap R \cap V_i = \phi$  for all  $i \in F_{(\gamma, \rho)}$ . As  $R$  is formed using the sampled nodes, we have that every node in  $R$  should be from  $V_i \setminus (\Gamma(v) \cap S \cap V_i)$  for the condition  $\Gamma(v) \cap R = \phi$  to be satisfied. As every element in  $R$  is chosen uniformly at random from the respective good partitions independently, we have :

$$\begin{aligned}
& \Pr_{\forall i, U_i \sim V_i} [\forall i : \Gamma(v) \cap R \cap V_i = \phi \text{ and } |\Gamma(v) \cap S| > \epsilon n/2k] \\
& \leq \prod_{i \in F_{(\gamma, \rho)}} \left( \frac{|V_i| - |\Gamma(v) \cap S \cap V_i|}{|V_i|} \right)^{|U_i|} \\
& \leq \exp \left( - \sum_i \frac{|U_i| |\Gamma(v) \cap S \cap V_i|}{|V_i|} \right) \\
& \leq \exp \left( - \frac{\gamma \rho t}{n} \sum_i |\Gamma(v) \cap S \cap V_i| \right) \\
& \leq \exp \left( - \frac{\gamma \rho t \epsilon n}{n \cdot 2k} \right) \\
& \leq \epsilon \delta / 2k.
\end{aligned}$$

Therefore, on expectation, there are at most  $n \cdot \epsilon \delta / 4k$  nodes such that the number of neighbours in  $S$  is more than  $\epsilon n/2k$ . Using Markov's inequality, with probability  $1 - \delta/2$ , we have that at most  $\epsilon n/2k$  nodes have number of neighbours in  $S$  greater than  $\epsilon n/2k$ . Applying union bound, we have with probability  $1 - \delta$  that  $R$  is a  $\epsilon/2k$ -IS representative subset.  $\square$

**Lemma 3.7.29.** *Suppose  $S$  is an independent set in  $G$  with cost  $C(S) \geq \rho C(V)$  for some  $\rho > 0$  and  $\widehat{Z}(\mathcal{U})$  denote the set found by Algorithm 7 such that  $\mathcal{U}$  is a  $\epsilon/2k$ -IS representative subset. Then, with probability  $1 - \delta$ , we have*

$$C(\widehat{Z}(\mathcal{U})) \geq \rho(1 - 3\gamma)C(V).$$

*Proof.* Consider  $\widehat{Z}_i(\mathcal{U})$  for some  $i \in F_{(\gamma, \rho)}$  and let  $F := \bigcup_{i \in F_{(\gamma, \rho)}} V_i$ . In Algorithm 7, we obtained  $\widehat{Z}_i(\mathcal{U})$  by including nodes from  $Z(\mathcal{U}) \cap V_i$  in the sorted order of degree such that the total degree of nodes in the induced graph  $Z(\mathcal{U})$  is bounded by  $\epsilon n^2/k$ . First, when  $\mathcal{U}$  is a  $\epsilon/2k$ -IS representative subset, we observe that

$$\mathcal{U} \subseteq S \cap F \subseteq F \setminus \Gamma(S \cap F) \subseteq Z(\mathcal{U}).$$

So,  $S \cap V_i \subseteq Z(\mathcal{U}) \cap V_i$ . From Lemma 3.4.5, we have, for every node in  $Z(\mathcal{U}) \cap V_i$  except for  $\epsilon/2k$  many, the maximum degree to  $S \cap V_i$  is at most  $\epsilon n/2k$ , and the remaining nodes can have a maximum degree of  $n$ . Combining these statements, we have that the total degree incident on the nodes in  $S \cap V_i$  from the nodes  $Z(\mathcal{U}) \cap V_i$  is at most

$$\frac{\epsilon n}{2k} \cdot |Z(\mathcal{U}) \cap V_i| + n \cdot \frac{\epsilon n}{2k} \leq \frac{\epsilon n^2}{k}.$$

As we include nodes in  $\widehat{Z}_i(\mathcal{U})$  until sum of degrees is  $\epsilon n^2/k$ , we have that the size of  $\widehat{Z}_i(\mathcal{U})$  will only be more than the size of  $S \cap V_i$  and satisfies  $|\widehat{Z}_i(\mathcal{U})| \geq |S \cap V_i|$ . We know that every node in  $V_i$  has cost in the range  $[(1 + \gamma)^{i-1}, (1 + \gamma)^i)$ , therefore, we have

$$\begin{aligned} C(\widehat{Z}_i(\mathcal{U})) &\geq \frac{1}{(1 + \gamma)} C(S \cap V_i) \\ \sum_{i \in F_{(\gamma, \rho)}} C(\widehat{Z}_i(\mathcal{U})) &\geq \frac{1}{(1 + \gamma)} \sum_{i \in F_{(\gamma, \rho)}} C(S \cap V_i) \end{aligned}$$

From Claim 3.7.27, we know

$$\begin{aligned}
\sum_{i \in F(\gamma, \rho)} C(\widehat{Z}_i(\mathcal{U})) &\geq \frac{1}{(1 + \gamma)} C(S') \\
&\geq (1 - \gamma)(1 - 2\gamma)\rho C(V) \\
C(\widehat{Z}(\mathcal{U})) &\geq \rho(1 - 3\gamma)C(V).
\end{aligned}$$

□

**Lemma 3.7.30.** *Let  $G$  contain an independent set of cost  $\rho C(V)$ , then, Algorithm 7 returns a set of nodes  $\widehat{Z}(\mathcal{U})$  such that  $C(\widehat{Z}(\mathcal{U})) \geq \rho(1 - 3\gamma)C(V)$  and  $|E[\widehat{Z}(\mathcal{U})]| \leq \epsilon n^2$  with probability  $1 - \delta$  and runs in time  $O\left(n^2 \exp\left(O\left(\frac{k^2}{\epsilon} \log \frac{1}{\gamma\epsilon} \log \frac{k}{\epsilon\delta}\right)\right)\right)$ .*

*Proof.* As our Algorithm 7 selects nodes from each partition  $V_i$  such that the total degree of nodes in  $\widehat{Z}_i(\mathcal{U})$  in the graph induced by  $E[Z(\mathcal{U})]$  is at most  $\epsilon n^2/k$ . Therefore, total degree of nodes in  $\widehat{Z}(\mathcal{U}) = \bigcup_{i \in F(\gamma, \rho)} \widehat{Z}_i(\mathcal{U})$  is at most  $k \cdot \epsilon n^2/k$ . Hence,  $|E[\widehat{Z}(\mathcal{U})]| \leq \epsilon n^2$ . From Lemma 3.7.29, we have  $C(\widehat{Z}(\mathcal{U})) \geq \rho(1 - 3\gamma)C(V)$ .

In Algorithm 7, we iterate over all subsets of the partitions  $\{V_1, V_2, \dots, V_k\}$ . Consider a subset  $\{V_1, V_2, \dots, V_\tau\}$  and in each partition, we iterate over all subsets  $U_i$  of size  $\gamma\rho t/2$ . Therefore, total number of subsets  $\mathcal{U}$  formed from the union of subsets in each partition  $\bigcup_{i=1}^\tau U_i$  is given by  $\binom{t}{\gamma\rho t/2}^\tau$ . Using  $t = O\left(\frac{k \log k/\epsilon\delta}{\rho\gamma\epsilon}\right)$  and  $\rho \geq \sqrt{\epsilon}$ , we have that the total number of iterations is at most

$$\begin{aligned}
2^k \cdot \binom{t}{\gamma\rho t/2}^k &\leq 2^k \cdot \left(\frac{2te}{\gamma\rho t}\right)^{\gamma\rho tk/2} \\
&\leq 2^k \cdot \left(\frac{6}{\gamma\rho}\right)^{\gamma\rho tk/2} \leq \exp\left(O\left(\frac{k^2}{\epsilon} \log \frac{1}{\gamma\epsilon} \log \frac{k}{\epsilon\delta}\right)\right).
\end{aligned}$$

In each iteration, we can find  $Z(\mathcal{U})$  in  $O(|\mathcal{U}|n)$  time. After that, we calculate the degree of nodes in  $Z(\mathcal{U}) \cap V_i$  in the induced sub-graph  $E[Z(\mathcal{U})]$  which requires  $O(|Z(\mathcal{U})|^2) = O(n^2)$  running time. Hence, the claim. □

**Lemma 3.7.31.** *Suppose  $S^*$  denotes MIS in  $G(V, E)$ . Algorithm 6 returns a set of nodes  $S$  such that  $C(S) \geq C(S^*)$ ,  $|S| \geq \sqrt{\epsilon}n$  and  $|E[S]| \leq \epsilon n^2$  with probability  $1 - \delta$  and runs in time*

$$O\left(\frac{n^2 W}{\epsilon} \log \frac{1}{\epsilon} \exp\left(O\left(\frac{W^2 \log^2 W}{\epsilon^3} \log \frac{W}{\epsilon} \log \frac{W \log W \log 1/\epsilon}{\epsilon \delta}\right)\right)\right).$$

*Proof.* Let  $T$  denote the set of  $\sqrt{\epsilon}n$  nodes from  $V$  with highest cost. It is easy to observe that  $C(T) \geq \sqrt{\epsilon} C(V)$ . If  $C(S^*) < C(T)$ , then, Algorithm 6 outputs the set  $T$ . Therefore,

$$C(T) > C(S^*) \text{ and } |E[T]| \leq (\sqrt{\epsilon}n)^2 = \epsilon n^2.$$

Otherwise, in Algorithm 6, we search for MIS with cost  $\rho C(V)$  using decreasing powers of  $(1 + \gamma)$  with the help of the parameter  $\rho$  when  $\rho \geq \sqrt{\epsilon}$ . If  $C(S^*) \geq C(T)$ , then,  $|S^*| \geq |T| = \sqrt{\epsilon}n$  and for some  $1 \leq j \leq \frac{1}{2\gamma} \log \frac{1}{\epsilon}$  and  $\rho = \frac{1}{(1+\gamma)^j}$  (i.e.,  $\sqrt{\epsilon} \leq \rho \leq 1$ ) we have

$$\rho C(V) \leq C(S^*) \leq \rho(1 + \gamma)C(V).$$

For this value of  $\rho$ , Algorithm 7 returns a set of nodes  $S$  such that  $|E[S]| \leq \frac{\epsilon}{8W}n^2$ .

We observe that

$$\begin{aligned} C(S) &\geq \frac{1}{(1 + \gamma)^j} (1 - 3\gamma)C(V) \\ &\geq \frac{1 - 3\gamma}{1 + \gamma} \frac{C(V)}{(1 + \gamma)^{j-1}} \geq (1 - 4\gamma)C(S^*). \end{aligned}$$

In our call to the Algorithm 7 from Algorithm NEAR-MIS, we set  $\gamma = \frac{\epsilon}{8W}$ .

$$\begin{aligned} C(S_{\epsilon/2}) &\geq \frac{\epsilon n}{2} \quad (\text{since, cost of a node is at least 1}) \\ \Rightarrow C(S \cup S_{\epsilon/2}) &\geq C(S^*) + \frac{\epsilon n}{2} - \frac{\epsilon}{2W}C(S^*) \\ &\geq C(S^*) + \frac{\epsilon n}{2} - \frac{\epsilon}{2W}n \cdot W \geq C(S^*). \end{aligned}$$

As every node in  $S_{\epsilon/2}$  has degree at most  $n$ , we have

$$|E[S \cup S_{\epsilon/2}]| \leq \frac{\epsilon n^2}{8W} + \frac{\epsilon n^2}{2} \leq \epsilon n^2.$$

As  $C(S \cup S_{\epsilon/2}) \geq C(T)$  where  $T$  contains the  $\sqrt{\epsilon}n$  highest cost nodes, we have  $|S \cup S_{\epsilon/2}| \geq \sqrt{\epsilon}n$ . When  $C(S^*) \geq C(T)$ , we search for the correct value of  $\rho$  and for each guess, we call the routine Algorithm 7. In total, the number of calls that are made to Algorithm 7 is at most  $\frac{1}{2\gamma} \log \frac{1}{\epsilon}$ . However, in each call to Algorithm 7, we fail to output with probability  $\delta'$ . As we set the failure probability to  $\delta' = 2\gamma\delta/\log(1/\epsilon)$ , overall the iterations, using union bound, the failure probability is at most  $\delta' \cdot \frac{1}{2\gamma} \log \frac{1}{\epsilon} = \delta$ .

From Lemma 3.7.30, Algorithm 7 runs in time  $O\left(n^2 \exp\left(O\left(\frac{k^2}{\epsilon} \log \frac{1}{\gamma\epsilon} \log \frac{k}{\epsilon\delta'}\right)\right)\right)$ . Substituting  $k = \gamma^{-1} \log W$ ,  $\delta'$ ,  $\gamma = \frac{\epsilon}{8W}$  and for a total of  $\frac{1}{2\gamma} \log \frac{1}{\epsilon}$  calls to Algorithm 7, the running time of Algorithm 6 is

$$O\left(\frac{n^2 W}{\epsilon} \log \frac{1}{\epsilon} \exp\left(O\left(\frac{W^2 \log^2 W}{\epsilon^3} \log \frac{W}{\epsilon} \log \frac{W \log W \log 1/\epsilon}{\epsilon\delta}\right)\right)\right).$$

□

From Lemma 3.7.31, we have that in each iteration, Algorithm 6 returns a set of nodes  $S$  that have a cost  $C(S) \geq C(S^*)$  where  $S^*$  is the maximum independent set in  $G$ . In a previous work [91], it was shown that by using maximum independent set in each iteration, we obtain a  $(2 + \exp(-\Omega(m)))$ -optimal separating set system. Following the exact same analysis, gives us an approximation factor close to 2. We refer the reader to the analysis in Section 3.7.5, and give the main statement of the Lemma below.

**Lemma 3.7.32.** *For any  $m \geq \eta \log 1/\epsilon$  for some constant  $\eta > 2$ , with probability  $\geq 1 - \delta$ , Algorithm 5 returns  $L_\epsilon$  with  $C(L_\epsilon) \leq (2 + \exp(-\Omega(m))) \cdot C(L^*)$ , where  $L^*$  is the min-cost separating matrix for  $G$*

**Scaling Parameters** In Algorithm 5, we pass a scaled value of  $\epsilon$  by setting it to  $\epsilon^2$  when we call Algorithm 6, as this ensures that total number of edges returned over  $\frac{1}{\epsilon}$  calls is at most  $\epsilon n^2$ . We also set the failure probability for each call as  $\epsilon\delta$ , to ensure that over  $\frac{1}{\epsilon}$  calls, total failure probability using union bound is at most  $\delta$ .

**Theorem 3.7.33.** (Theorem 3.4.6 restated) For any  $m \geq \eta \log 1/\epsilon$  for some constant  $\eta$ , with probability  $\geq 1 - \delta$ , Algorithm 5 returns  $L_\epsilon$  with  $C(L_\epsilon) \leq (2 + \exp(-\Omega(m))) \cdot C(L^*)$ , where  $L^*$  is the min-cost separating matrix for  $G$ . Moreover  $L_\epsilon$   $\epsilon$ -separates  $G$ . Algorithm 5 has a running time  $O(n^2 f(W, \epsilon, \delta))$  where

$$f(W, \epsilon, \delta) = O\left(\frac{n^2 W}{\epsilon^2} \log \frac{1}{\epsilon} \exp\left(O\left(\frac{W^2 \log^2 W}{\epsilon^6} \log \frac{W}{\epsilon} \log \frac{W \log W \log 1/\epsilon}{\epsilon \delta}\right)\right)\right).$$

*Proof.* Using all the above scaled parameters, from Lemma 3.7.31, the running time of Algorithm 5 that internally calls Algorithm 6 for  $\frac{1}{\epsilon}$  number of times, is given by

$$O\left(\frac{n^2 W}{\epsilon^2} \log \frac{1}{\epsilon} \exp\left(O\left(\frac{W^2 \log^2 W}{\epsilon^6} \log \frac{W}{\epsilon} \log \frac{W \log W \log 1/\epsilon}{\epsilon \delta}\right)\right)\right).$$

From Lemma 3.7.32, we have the approximation guarantee. □

**Dependence on  $\epsilon$**  Observe that our running time is exponential in  $1/\epsilon$  and therefore setting  $\epsilon < 1/n^2$  to get a separating system with all edges separated requires exponential running time. As we have argued that finding such a set system with near optimal cost is hard conditioned on the hardness of approximate coloring (Theorem 3.3.3), it is thus also conditionally hard to improve our runtime to be polynomial in  $1/\epsilon$ . It is an interesting open question to study the parameterized hardness beyond polynomial factors with respect to  $\epsilon$ .

By Theorem 3.4.6 with  $m = O(\log(1/\epsilon))$  interventions we can  $\epsilon$ -approximately learn any causal graph  $G$ . For learning the entire graph  $G$ ,  $m \geq \log \chi$  interventions

are necessary, where  $\chi$  is the chromatic number of  $G$ , since the rows of  $L \in \{0, 1\}^{n \times m}$  must be a valid coloring of  $G$  [91].

### 3.7.4 Proof Details from Section 3.5

In this section, we say a pair of nodes  $(v_i, v_j)$  share an ancestral relation, if  $v_i$  has a directed path to  $v_j$  ( $v_i$  is an ancestor of  $v_j$ ) or  $v_j$  has a directed path to  $v_i$  ( $v_j$  is an ancestor of  $v_i$ ).

**Lemma 3.7.34.** *Suppose  $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$  is a collection of subsets of  $V$ . If  $\text{Anc}(G) \cap H$  is recovered from  $H$  using conditional independence tests by intervening on the sets  $S_i \in \mathcal{S}$ . Then, under the assumptions of section 3.1,  $\mathcal{S}$  is a strongly separating set system on  $H$ .*

*Proof.* First, we argue that to recover  $\text{Anc}(G) \cap H$  it is sufficient that  $\mathcal{S}$  is a strongly separating set system on  $H$ . Suppose  $(v_i, v_j) \in H$  and  $v_i, v_j$  share an ancestral relation i.e., either  $v_i$  is an ancestor of  $v_j$  or  $v_j$  is an ancestor of  $v_i$ . Therefore,  $v_i \not\perp\!\!\!\perp v_j$  and  $(v_i, v_j) \in \text{Anc}(G)$ . From Lemma 1 in [88], we know that, we can recover the ancestral relation between  $v_i$  and  $v_j$  using conditional independence tests (or CI-tests) on interventional distributions that strongly separate the two variables  $v_i$  and  $v_j$ . As  $\mathcal{S}$  is a strongly separating set system for  $H$ , we can recover all ancestral relations in  $\text{Anc}(G) \cap H$ .

Now, we show that a strongly separating set system on  $H$  is necessary. Here, we give a proof similar to Lemma A.1 from [5]. Suppose  $\mathcal{S}$  is not a strongly separating set system for  $H$ . If there exists a pair of nodes containing an ancestral relation, say  $(v_i, v_j) \in H \cap \text{Anc}(G)$  such that every set  $S_k \in \mathcal{S}$  contains none of them, then, we cannot recover the ancestral relation between these two nodes as we are not intervening on either  $v_i$  or  $v_j$  and the results of an independence test  $v_i \perp\!\!\!\perp v_j$  might result in a wrong inference, possibly due to the presence of a latent variable  $l_{ij}$  between them. Consider the case when only one of them is present in every set of  $\mathcal{S}$ . Let  $\mathcal{S}$  be such

that  $\forall S_k \in \mathcal{S} : S_k \cap \{v_i, v_j\} = \{v_i\} \Rightarrow v_i \in S_k, v_j \notin S_k$ . We choose our graph  $G$  to have two components  $\{v_i, v_j\}$  and  $V \setminus \{v_i, v_j\}$ ; and include the edge  $v_j \rightarrow v_i$  in it. Observe that  $v_i \not\perp\!\!\!\perp v_j$ . Our algorithm will conclude from the CI-test  $v_i \perp\!\!\!\perp v_j \mid \text{do}(S_k)$ ? that  $v_i$  and  $v_j$  are independent. However, it is possible that  $v_i \not\perp\!\!\!\perp v_j$  because of a latent  $l_{ij}$  between  $v_i$  and  $v_j$ , but when we do CI-test, we get  $v_i \perp\!\!\!\perp v_j \mid \text{do}(S_k)$  as intervening on  $v_i$  disconnects the  $l_{ij} \rightarrow v_i$  edge. Therefore, our algorithm cannot distinguish the two cases  $v_j \rightarrow v_i$  and  $v_i \leftarrow l_{ij} \rightarrow v_j$  without intervening on  $v_j$ . For every  $\mathcal{S}$  that is not a strongly separating set system on  $H$ , we can provide a graph  $G$  such that by intervening on sets in  $\mathcal{S}$ , we cannot recover  $\text{Anc}(G) \cap H$  from  $H$  correctly.  $\square$

**Lemma 3.7.35.** (*Lemma 3.5.1 restated*) *Under the assumptions of Section 3.1, if  $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$  is an  $\epsilon$ -strongly separating set system for  $H$ ,  $\mathcal{S}$  suffices to  $\epsilon$ -approximately learn  $\text{Anc}(G) \cap H$ .*

*Proof.* Given  $\mathcal{S}$  denotes an  $\epsilon$ -strongly separating set system for  $H$ . So, there are at most  $\epsilon n^2$  edges  $(u, v) \in H$  such that for all  $i \in [m]$ , either  $\{u, v\} \cap S_i = \emptyset$  or  $\{u, v\} \cap S_i = \{u, v\}$  or  $\{u, v\} \cap S_i = \{u\}$  (without loss of generality). For every such edge, any intervention on a set in  $\mathcal{S}$ , say  $S_i$  cannot recover the direction from a conditional independence test (CI-test)  $u \perp\!\!\!\perp v \mid \text{do}(S_i)$  because for both the cases  $u \leftarrow v$  or  $u \leftarrow l_{uv} \rightarrow v$ , where  $l_{uv}$  is a latent, the CI-test returns that they are independent. Therefore, we cannot recover the ancestral relation (if one exists) between  $u, v$  that are not strongly separated in  $H$ . For edges  $(u, v) \in H$  that are strongly separated using  $S_i$  and  $S_j$ , we can recover the ancestral relation using CI-tests  $u \perp\!\!\!\perp v \mid \text{do}(S_i)$  and  $u \perp\!\!\!\perp v \mid \text{do}(S_j)$ . From Def. 3.1.1, we have that  $\mathcal{S}$   $\epsilon$ -approximately learns  $G$ .  $\square$

Algorithm SSMATRIX from [5] gives a 2-approximation guarantee for the output strongly separating matrix. However, we cannot directly extend the arguments as the guarantee holds when the input graph is complete. We get around this limitation, and show that Algorithm 8 achieves a close to 4-approximation, by relating the cost

of  $\epsilon$ -strongly separating matrix returned by SSMATRIX on the *supernode* set  $V_S$ , to the cost of 2-approximate  $\epsilon$ -separating matrix that we find using Algorithm 5.

Let  $\text{ALG}_S$  denote the cost of the objective  $\sum_{j=1}^n C(v_j) \|L_\epsilon^S(j)\|_1$  obtained by Algorithm 5 where  $L_\epsilon^S$  is an  $\epsilon$ -separating matrix;  $\text{ALG}_{SS}$  denote the cost of the objective  $\sum_{j=1}^n C(v_j) \|L_\epsilon^{SS}(j)\|_1$  obtained by Algorithm 8 where  $L_\epsilon^{SS}$  is an  $\epsilon$ -strongly separating matrix. For the sake of analysis, during assignment of vectors to nodes in  $L_\epsilon^S$ , we assume that Algorithm 5 only allows vectors of weight at least 1. As SSMATRIX algorithm from [5] assigns vectors with weight atleast 1 (otherwise it will not be a valid strongly separating matrix), this assumption for  $\text{ALG}_S$  helps us in showing a relation between the costs of  $L_\epsilon^{SS}$  and  $L_\epsilon^S$ . As that is not sufficient to obtain the claimed guarantee, instead of assigning  $\binom{m}{1}$  vectors of weight 1, we constraint it to a fixed number  $r \leq \binom{m}{1}$ . In [5], Algorithm SSMATRIX assigns vectors to  $L_\epsilon^{SS}$  by guessing the exact number of weight 1 vectors in  $\text{OPT}_{SS}$ , the parameter  $r$  corresponds to this guess.

Let  $\text{OPT}_{SS}$  and  $\text{OPT}_S$  denote optimum objective values associated with strongly separating and separating matrices for a graph  $H$ . Let  $\text{ALG}_{SS}(r)$  denote the cost  $C(L_\epsilon^{SS})$  assuming first  $r$  columns are used for exactly  $r$  weight 1 vectors during the assignment in  $L_\epsilon^{SS}$ , and the remaining  $m - r$  columns are used for all the remaining vector assignments. Similarly,  $\text{ALG}_S(r)$ ,  $\text{OPT}_S(r)$  and  $\text{OPT}_{SS}(r)$  are defined.

**Lemma 3.7.36.**  $\text{OPT}_S(r) \leq \text{OPT}_{SS}(r)$  for any  $r \geq 0$ .

*Proof.* Observe that any strongly separating matrix for  $H$  is also a separating matrix for  $H$ . Now, consider a strongly separating matrix that achieves cost  $\text{OPT}_{SS}(r)$  using  $r$  weight 1 vectors, then, we have

$$\text{OPT}_S(r) \leq \text{OPT}_{SS}(r).$$

□

**Lemma 3.7.37.**  $C(L_\epsilon^{SS}) \leq (4 + \gamma + \exp(-\Omega(m))) \cdot \text{OPT}_{SS}$ .

*Proof.* First, we note that in *any* strongly separating matrix, for the *non-dominating* property to hold, the support of weight 1 vectors and the support of vectors of weight  $> 1$  are column disjoint. Suppose  $a_1^*$  denote the number of columns of  $m$  that are used by  $\text{OPT}_{SS}$  for weight 1 vectors i.e,  $\text{OPT}_{SS}(a_1^*) = \text{OPT}_{SS}$ .

Following the exact proof of Lemma A.5 in [5] gives us the following guarantee about Algorithm 8

$$C(L_\epsilon^{SS}) \leq 2\text{ALG}_S(a_1^*).$$

From Theorem 3.4.6 and Lemma 3.7.32 in Section 3.7.5 (or the analysis from [91]) :

$$\text{ALG}_S(a_1^*) \leq (2 + \exp(-\Omega(m))) \text{OPT}_S(a_1^*).$$

From Lemma 3.7.36, we know  $\text{OPT}_S(a_1^*) \leq \text{OPT}_{SS}(a_1^*)$ . Therefore, we have:

$$\begin{aligned} \text{ALG}_S(a_1^*) &\leq (2 + \exp(-\Omega(m))) \text{OPT}_{SS}(a_1^*) \\ &= (2 + \exp(-\Omega(m))) \text{OPT}_{SS}. \end{aligned}$$

Hence, the lemma. □

**Theorem 3.7.38.** (*Theorem 3.5.2 restated*) Let  $m \geq \eta \log 1/\epsilon$  for some constant  $\eta$  and  $L_\epsilon^{SS}$  be matrix returned by Algorithm 8. Then with probability  $\geq 1 - \delta$ ,  $L_\epsilon^{SS}$  is an  $\epsilon$ -strongly separating matrix for  $H$  and  $C(L_\epsilon^{SS}) \leq (4 + \exp(-\Omega(m))) \cdot C(L^*)$  where  $L^*$  is the min-cost strongly separating matrix for  $H$ . Algorithm 8 runs in time  $O(n^2 f(W, \epsilon, \delta))$  where

$$f(W, \epsilon, \delta) = O\left(\frac{n^2 W}{\epsilon^2} \log \frac{1}{\epsilon} \exp\left(O\left(\frac{W^2 \log^2 W}{\epsilon^6} \log \frac{W}{\epsilon} \log \frac{W \log W \log 1/\epsilon}{\epsilon \delta}\right)\right)\right).$$

*Proof.* From Lemma 3.7.37, we have  $C(L_\epsilon^{SS}) \leq (4 + \exp(-\Omega(m)))C(L^*)$ . The sets of nodes  $S_1, S_2, \dots, S_{1/\epsilon}$  returned by Algorithm 8 are such that every set  $S_i$  contains at most  $\epsilon^2 n^2$  edges with probability  $1 - \delta$ . Therefore, in total at most  $\frac{1}{\epsilon} \epsilon^2 n^2 \leq \epsilon n^2$  edges do not satisfy strongly separating property. As SSMATRIX has a running time of  $O(|V_S|) = O(\frac{1}{\epsilon})$ , and using the running time of Algorithm 5 from Theorem 3.7.33, our claim follows.  $\square$

### 3.7.5 Additional Details for the analysis of 2-approximation result for $\epsilon$ -Separating Set Systems

In this section, we present already known results from [91] filling in the details in the analysis of our Algorithm 5 for the sake of completion.

Let  $\mathcal{I}$  denote the set of all independent sets in  $G$ . For some  $\mathcal{A} \subseteq \mathcal{I}$ , we have

$$J(\mathcal{A}) = \sum_{v \in \bigcup_{S \in \mathcal{A}} S} C(v)$$

that is, it takes a set of independent sets and returns the sum of the cost of the vertices in their union. We observe that  $J$  is submodular, monotone, and non-negative [91].

Let  $S_0$  denote the set of nodes that are assigned weight 0 vector after the first iteration of Algorithm 5. We set  $V = V \setminus S_0$  for the remainder of this section and handle the cost contribution of nodes in  $S_0$  separately in the analysis of approximation ratio.

**Lemma 3.7.39.** *Given a submodular, monotone, and non-negative function  $J$  over a ground set  $V$  and a cardinality constraint  $k$ . Let Algorithm 5 return  $S_{greedy}$  a collection of at most  $Ck$  (for some constant  $C > 0$ ) sets that are  $(0, \epsilon)$ -NEAR-MIS, then,*

$$J(S_{greedy}) \geq (1 - e^{-C}) \max_{S \subseteq \mathcal{I}, |S| \leq k} J(S).$$

*Proof.* We have that in  $i$ th iteration of Algorithm 5, we pick a set of nodes  $S_i$  with cost at least the cost of MIS  $T_i$  in  $G$ , i.e.,  $S_i$  is a  $(0, \epsilon)$ -NEAR-MIS in  $G$  and satisfies  $C(S_i) \geq C(T_i)$ . Let  $\mathcal{S}^*$  be the collection of independent sets such that  $J(\mathcal{S}^*) = \max_{\mathcal{S} \subseteq \mathcal{I}, |\mathcal{S}| \leq k} J(\mathcal{S})$ . Let  $\bigcup_{j \leq i} S_j$  be denoted by  $S_{1:i}$ . We claim using induction that

$$J(\mathcal{S}^*) - J(S_{1:i}) \leq \left(1 - \frac{1}{k}\right)^i J(\mathcal{S}^*).$$

Consider  $i$ th iteration when Algorithm 5 picks  $S_i$ . Using submodularity, we have

$$J(\mathcal{S}^*) - J(S_{1:i-1}) \leq \sum_{B \in \mathcal{S}^* \setminus S_{1:i-1}} J(S_{1:i-1} \cup B).$$

Therefore, there exists one set  $B \in \mathcal{S}^* \setminus S_{1:i-1}$ , with cost at least  $\frac{\sum_{B \in \mathcal{S}^* \setminus S_{1:i-1}} J(S_{1:i-1} \cup B)}{k}$ .

As argued in Lemma 3.7.31, we are picking a set  $S_i$  with cost  $C(S_i) \geq C(T_i)$  where  $T_i$  is MIS in the  $i$ th iteration, we have :

$$\begin{aligned} C(S_i) \geq C(T_i) &\geq \frac{\sum_{B \in \mathcal{S}^* \setminus S_{1:i-1}} J(S_{1:i-1} \cup B)}{k} \geq \frac{J(\mathcal{S}^*) - J(S_{1:i-1})}{k} \\ J(S_i) = C(S_i) &\geq \frac{J(\mathcal{S}^*) - J(S_{1:i-1})}{k}. \end{aligned}$$

For  $i = 1$ , our claim follows from the above statement, i.e.,  $C(S_1) = J(S_1) \geq \frac{J(\mathcal{S}^*)}{k}$ .

Assuming that our claim holds until iteration  $i - 1$  for some  $i \geq 2$ , we have after the  $i$ th iteration :  $J(\mathcal{S}^*) - J(S_{1:i}) = J(\mathcal{S}^*) - J(S_{1:i-1}) - J(S_i)$ . This is true because  $J(S_{1:i}) = J(S_{1:i-1}) + J(S_i)$  as  $S_i$  is greedily chosen by picking a set containing nodes that are not previously selected. Therefore,

$$\begin{aligned}
J(\mathcal{S}^*) - J(S_{1:i}) &= J(\mathcal{S}^*) - J(S_{1:i-1}) - J(S_i) \\
&\leq J(\mathcal{S}^*) - J(S_{1:i-1}) - \frac{J(\mathcal{S}^*) - J(S_{1:i-1})}{k} \\
&\leq (J(\mathcal{S}^*) - J(S_{1:i-1})) \left(1 - \frac{1}{k}\right) \\
&\leq \left(1 - \frac{1}{k}\right)^i J(\mathcal{S}^*).
\end{aligned}$$

Setting  $i = C \cdot k$ , we have

$$\begin{aligned}
J(\mathcal{S}^*) - J(S_{\text{greedy}}) &\leq \left(1 - \frac{1}{k}\right)^{Ck} J(\mathcal{S}^*) \leq e^{-C} \cdot J(\mathcal{S}^*) \\
\Rightarrow J(S_{\text{greedy}}) &\geq (1 - e^{-C})J(\mathcal{S}^*).
\end{aligned}$$

□

Now, we define two types of submodular optimization problem, called the submodular chain problem and the supermodular chain problem that will be useful later.

**Definition 3.7.40.** *Given integers  $k_1, k_2, \dots, k_m$  and a submodular, monotone, and non-negative function  $J$ , over a ground set  $V$ , the submodular chain problem is to find sets  $A_1, A_2, \dots, A_m \subseteq 2^{[V]}$  such that  $|A_i| \leq k_i$  that maximizes*

$$\sum_{i=1}^m J(A_1 \cup A_2 \cup \dots \cup A_i).$$

**Lemma 3.7.41.** *Let  $A_1^*, A_2^*, \dots, A_m^*$  be the optimal solution to the submodular chain problem. Suppose that for all  $1 \leq p \leq m/2 - 1$  we have that  $\sum_{i=1}^{2p} k_i \geq \tau \sum_{i=1}^p k_i$ . Also assume that  $J(A_1 \cup A_2 \cup \dots \cup A_m) = J(V)$ . Then the greedy algorithm 5 for the submodular chain problem returns set  $A_1, A_2, \dots, A_m$  such that*

$$\sum_{i=1}^m J(A_1 \cup A_2 \cup \dots \cup A_i) \geq J(V) + 2(1 - e^{-\tau}) \sum_{i=1}^{m/2-1} J(A_1^* \cup A_2^* \cup \dots \cup A_i^*).$$

*Proof.* Given  $\sum_{i=1}^{2p} k_i \geq \tau \sum_{i=1}^p k_i$ . From Lemma 3.7.39, we have

$$J(A_1 \cup A_2 \cup \cdots \cup A_{2p}) \geq (1 - e^{-\tau})J(A_1^* \cup A_2^* \cup \cdots \cup A_p^*).$$

$$\sum_{i=1}^{m/2-1} J(A_1 \cup A_2 \cdots \cup A_{2i}) \geq (1 - e^{-\tau}) \sum_{i=1}^{m/2-1} J(A_1^* \cup A_2^* \cup \cdots \cup A_i^*).$$

Now, we use the monotonicity property of the submodular function  $J$  to get

$$\begin{aligned} & \sum_{i=1}^m J(A_1 \cup A_2 \cdots \cup A_i) \\ &= J(A_1 \cup A_2 \cdots \cup A_m) + \sum_{i=1}^{m/2-1} J(A_1 \cup A_2 \cup \cdots \cup A_{2i}) + J(A_1 \cup A_2 \cup \cdots \cup A_{2i+1}) \\ &\geq J(V) + 2 \sum_{i=1}^{m/2-1} J(A_1 \cup A_2 \cup \cdots \cup A_{2i}). \end{aligned}$$

Hence, the lemma. □

**Definition 3.7.42.** Given integers  $k_1, k_2, \dots, k_m$  and a submodular, monotone, and non-negative function  $F$ , over a ground set  $V$ , the supermodular chain problem is to find sets  $A_1, A_2, \dots, A_m \subseteq 2^{[V]}$  such that  $|A_i| \leq k_i$  that minimizes

$$\sum_{i=1}^m J(V) - J(A_1 \cup A_2 \cup \cdots \cup A_i).$$

For the greedy algorithm 5, we give the following claim for the supermodular chain problem.

**Lemma 3.7.43.** Let  $A_1^*, A_2^*, \dots, A_m^*$  be the optimal solution to the supermodular chain problem. Suppose that for all  $1 \leq p \leq m/2 - 1$  we have that  $\sum_{i=1}^{2p} k_i \geq \tau \sum_{i=1}^p k_i$ . Also assume that  $J(A_1 \cup A_2 \cup \cdots \cup A_m) = J(V)$ . Then the greedy algorithm 5 for the supermodular chain problem returns set  $A_1, A_2, \dots, A_m$  such that

$$\sum_{i=1}^m J(V) - J(A_1 \cup A_2 \cup \cdots \cup A_i) \leq e^{-\tau} m \cdot J(V) + 2 \sum_{i=1}^m J(A_1^* \cup A_2^* \cup \cdots \cup A_i^*).$$

*Proof.* From Lemma 3.7.41, we have

$$\begin{aligned}
& (m+1)J(V) - \sum_{i=1}^m J(A_1 \cup A_2 \cup \dots \cup A_i) \\
& \leq mJ(V) - 2(1 - e^{-\tau}) \sum_{i=1}^{m/2-1} J(A_1^* \cup A_2^* \cup \dots \cup A_i^*) \\
& \leq e^{-\tau} mJ(V) + mJ(V) - 2 \sum_{i=1}^{m/2-1} J(A_1^* \cup A_2^* \cup \dots \cup A_i^*) \\
& \leq e^{-\tau} mJ(V) + 2 \sum_{i=1}^{m/2-1} J(V) - J(A_1^* \cup A_2^* \cup \dots \cup A_i^*).
\end{aligned}$$

Now, we use the monotonicity property of  $J$  to get

$$\begin{aligned}
& e^{-\tau} mJ(V) + 2 \sum_{i=1}^{m/2-1} J(V) - J(A_1^* \cup A_2^* \cup \dots \cup A_i^*) \\
& \leq e^{-\tau} mJ(V) + 2 \sum_{i=1}^m J(V) - J(A_1^* \cup A_2^* \cup \dots \cup A_i^*).
\end{aligned}$$

Finally, we have

$$\begin{aligned}
\sum_{i=1}^m J(V) - J(A_1 \cup A_2 \cup \dots \cup A_i) & \leq (m+1)J(V) - \sum_{i=1}^m J(A_1 \cup A_2 \cup \dots \cup A_i) \\
& \leq e^{-\tau} m \cdot J(V) + 2 \sum_{i=1}^m J(V) - J(A_1^* \cup A_2^* \cup \dots \cup A_i^*).
\end{aligned}$$

□

**Lemma 3.7.44.** *Suppose  $S^+$  denote optimal separating set system that uses an additional color of weight 1 and uses weight 0 vector to color  $A_0$ . Let  $S^*$  denote optimal separating system. Then, we have  $C(S^+) - C(S^*) \leq 0$ .*

*Proof.* We give a proof similar to Lemma 22 in [91]. Let the set of nodes  $A_0$  selected in the first iteration of greedy Algorithm 5 and assigned weight 0 vector be denoted

by  $S_0^+$ . Similarly, the set of nodes that are colored with weight 0 vector in  $S^*$  be denoted by  $S_0^*$ . As  $S^+$  denotes optimal solution on  $V \setminus S_0^+$ , we can assume that a solution that uses a weight 1 color for nodes in  $S_0^* \setminus S_0^+$  is only going to be worse. Therefore, we have :

$$\begin{aligned}
C(S^+) - C(S^*) &\leq \sum_{v \in S_0^* \setminus S_0^+} C(v) - \sum_{v \in S_0^+ \setminus S_0^*} C(v) \\
&\leq \sum_{v \in S_0^* \setminus S_0^+} C(v) + \sum_{v \in S_0^* \cap S_0^+} C(v) - \sum_{v \in S_0^+ \setminus S_0^*} C(v) - \sum_{v \in S_0^* \cap S_0^+} C(v) \\
&\leq \sum_{v \in S_0^*} C(v) - \sum_{v \in S_0^+} C(v) \leq 0.
\end{aligned}$$

□

Let  $L_\epsilon$  denote the  $\epsilon$ -separating matrix returned by Algorithm 5, and let  $L_\epsilon = \{A_0, A_1, A_2, \dots, A_m\}$  where we abuse the previous notation and denote  $A_i$  to represent the set of all nodes (instead of a collection of subsets of  $V$ ) that have weight  $i$  assigned by  $L_\epsilon$ .

$$C(L_\epsilon) = \sum_{i=1}^m J(V) - J(A_1 \cup A_2 \cup \dots \cup A_i),$$

where  $|L_i| \leq \binom{m}{i}$ . We observe that this cost representation corresponds to the super-modular chain problem discussed above.

Assuming  $m \geq \eta \log 1/\epsilon$  for  $\eta > 2$ , we have that, the greedy Algorithm 5 only uses vectors of weight at most  $\log 1/\epsilon$  i.e.,  $m/2$ . In Lemma 3.7.41, each of the values  $k_1, k_2 \dots k_i \dots k_p$  represent number of weight  $i$  vectors available and from Lemma 21 in [91], we have that  $\tau = \Omega(m)$ , for every  $p$  in the range.

**Lemma 3.7.45.** *(Lemma 3.7.32 restated) For any  $m \geq \eta \log 1/\epsilon$  for some constant  $\eta > 2$ , with probability  $\geq 1 - \delta$ , Algorithm 5 returns  $L_\epsilon$  with  $C(L_\epsilon) \leq (2 + \exp(-\Omega(m))) \cdot C(L^*)$ , where  $L^*$  is the min-cost separating matrix for  $G$ .*

*Proof.* Using the definition of  $S^+$  from Lemma 3.7.44, we argue that  $C(S^+) \geq J(V)$  as every node in  $V$  is assigned a weight 1 vector. From Lemma 3.7.43, we have:

$$\begin{aligned}
C(L_\epsilon) &= \sum_{i=1}^{m/2} J(V) - J(A_1 \cup A_2 \cup \dots \cup A_i) \\
&\leq e^{-\tau} m \cdot J(V) + 2 \sum_{i=1}^m J(V) - J(A_1^* \cup A_2^* \cup \dots \cup A_i^*) \\
&\leq e^{-\tau} m \cdot C(S^+) + 2 \sum_{i=1}^m J(V) - J(A_1^* \cup A_2^* \cup \dots \cup A_i^*) \\
&\leq e^{-\tau} m \cdot C(S^+) + 2C(S^*).
\end{aligned}$$

From Lemma 3.7.44 and  $\tau = \Omega(m)$ , we have:

$$\Rightarrow C(L_\epsilon) \leq (2 + \exp(-\Omega(m))) \cdot C(S^*) = (2 + \exp(-\Omega(m))) \cdot C(L^*).$$

□

### 3.8 Conclusion

We highlight that in both the settings, although we consider the presence of latents in the system, we provide results for learning causal relations among only the observable variables. Identification of latents is an important goal and has been well-studied [88, 87, 5] when the objective is to minimize the number of interventions. There is no good cost lower bound known, even for recovering the observable (rather than ancestral) graph in the presence of latents. This makes the development of algorithms with approximation guarantees in terms of the optimum cost difficult. We view addressing this difficulty as a major open question.

Our results on bounded degree graphs make an additional assumption that the graph is hyperfinite, which gives more structure but still captures many graph families. It is an interesting open question if we can extend them to general sparse graphs. This

setting is challenging since even finding a Near-MIS with  $\epsilon \cdot |E|$  edges is still open and likely to be hard [109]. We conjecture that if  $|E| = O(n)$ , a constant approximation for our objectives is not possible (assuming standard complexity theoretic conjectures) if we must separate all but  $\epsilon \cdot |E|$  many edges.

It would also be very interesting to extend our work to the setting where we seek to identify a specific subset of edges of the causal graph, or where certain edges are ‘more important’ than others. We hope that our work is a first step in this direction, introducing the idea of partial recovery to overcome hardness results that rule out non-trivial approximation bounds for full graph recovery in the linear cost model.

## CHAPTER 4

### COLLABORATIVE CAUSAL DISCOVERY

In this chapter, we introduce a new model for *causal discovery*. In section 4.1, we present the motivation for the new model; in section 4.2, we describe the model in detail; in sections 4.3 and 4.4, we present causal discovery algorithms under suitable model specific assumptions; in section 4.5, we present an empirical evaluation of our approaches; in section 4.6, we present all the missing proof details, and in section 4.7 we conclude the chapter.

#### 4.1 Motivation

We recall that using observational data, only some ancestral relations as well as certain causal edges can be learned. Moreover, many observationally equivalent structures cannot be distinguished [132] and require interventional data for full causal graph recovery, as discussed in the introduction chapter 1. As interventions are expensive (require carefully controlled experiments) and performing multiple interventions is time-consuming, an important goal in causal discovery is to design algorithms that utilize simple (preferably, single variable) and fewer interventions [115]. However, when there are latents or unobserved variables in the system, in the worst-case, it is not possible to learn the exact causal DAG without intervening on every variable at least once. Furthermore, multivariable interventions are needed in presence of latents, as discussed in Chapter 2.

On the other hand, in a variety of applications, there is no one true causal structure, different entities participating in the application might have different causal

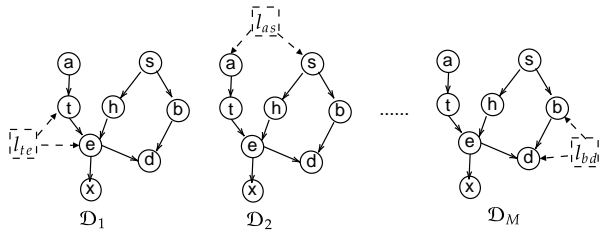


Figure 4.1: Examples of  $M$  causal graphs constructed from Lung Cancer dataset [90]. Here, the causal graphs differ only in the presence of latents (nodes with dotted square box), but they could differ elsewhere too.

structures [58, 106, 76]. For example, see figure 4.1. In these scenarios, generating a single causal graph by pooling data from these different entities might lead to flawed conclusions [113]. Allowing for interventions, we propose a new model for tackling this problem, referred here as *Collaborative Causal Discovery*, which in its simplest form states that: given a collection of entities, each associated with an individual unknown causal graph and generating their own independent data samples, learn all the causal graphs while minimizing the number of *atomic* (single variable) interventions for every entity. An underlying assumption is that each entity on its own generates enough samples in both the observational and interventional settings so that conditional independence tests can be carried out *accurately* on each entity separately. To motivate this model of collaborative causal discovery, let us consider two different scenarios.

- (a) Consider a health organization interested in controlling incidence of a particular disease. The organization has a set of  $M$  individuals (entities) whose data it monitors and can advise interventions on. Each individual is an independent entity that generates its own set of separate data samples<sup>1</sup>. In a realistic scenario, it is highly unlikely that all the  $M$  individuals share the same causal graph (e.g., see

---

<sup>1</sup>As is common in causal discovery, for the underlying conditional independence tests, the data is assumed to be i.i.d. samples from the interventional/observational distributions.

Figures 4.2a and 4.2b from [76]). It would be beneficial for the organization to collaboratively learn all the causal graphs together. The challenge is, *a priori* the organization does not know the set of possible causal graphs or which individual is associated with which graph from this set.

- (b) An alternate setting is where, we have  $M$  companies (entities) wanting to work together to improve their production process. Each company generates their own data (e.g., from their machines) which they can observe and intervene on [99]. Again if we take the  $M$  causal graphs (one associated with each company) it is quite natural to expect some variation in their structure, more so because we do not assume *causal sufficiency* (i.e., we allow for latents). Since interventions might need expensive and careful experimental organization, each company would like to reduce their share of interventions.

The collaborative aspect of learning can be utilized if we assume that there is some underlying (unknown) clustering/grouping of the causal graphs on the entities. Recently, Saeed et al. [113] studied a related model, where the observational data is from a mixture of causal DAGs, and outline ideas that recover a *union graph* (up to Markov equivalence) of these DAGs, without any interventions.

#### 4.1.1 Our Contributions

We formally introduce the collaborative causal discovery problem in section 4.2. We assume that we have a collection of  $M$  entities that can be partitioned into  $k$  clusters such that any pair of entities belonging to two different clusters are separated by large distance (see Definition 4.2.1) in the causal graphs. Due to presence of latents variables, we use a family of mixed graphs known as *maximal ancestral graphs* (MAGs) to model the graphs on observed variables. Each entity is associated with a MAG.

In this chapter, we focus on designing algorithms that have *worst-case* guarantees on the number of atomic interventions needed to recover (or approximately recover)

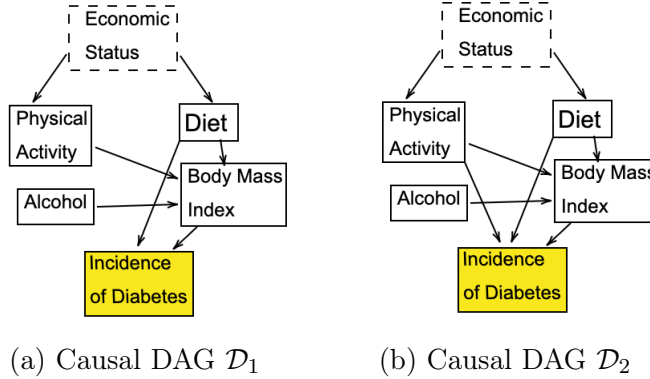


Figure 4.2: Two possible diabetes incidence graphs for an individual from [76] differing in the causal edge between *Physical Activity* and *Incidence of Diabetes*. The observed variables include: *Diet*, *Body Mass Index (BMI)*, *Physical Activity*, *Alcohol (consumption)*, *Incidence of Diabetes*, and the unobserved variable (latent) is *Economic Status*. The variable *Incidence of Diabetes* is observable but can't be intervened on, this is not an issue as it has no outgoing edges in the graphs. In this chapter, we assume we do not know the underlying causal graphs or which individuals share the same graph. As intervening on variables such as *Diet*, *BMI* might need expensive and careful experimental organization, we ask the following question – given a collection of independent entities (in this diabetes example, they can refer to a collection of people), can we collaboratively learn each entity's causal graphs while minimizing the number of interventions per entity?

the MAG of each entity. We assume that there are  $M$  MAGs one for each entity over the same set of  $n$  nodes. Learning a MAG with atomic interventions, in worst case requires  $n$  interventions (see Proposition 4.3.1). We show that this bound can be substantially reduced if the  $M$  MAGs satisfy the property that every pair of MAGs from different clusters have *at least*  $\alpha n$  nodes whose direct causal relationships are different. We further assume that entities belonging to same cluster have similar MAGs in that every pair of them have *at most*  $\beta n$  ( $\beta < \alpha$ ) nodes whose direct causal relationships are different. We refer to this clustering of entities as  $(\alpha, \beta)$ -clustering (Definition 4.2.2). A special but important case is when  $\beta = 0$ , in which case all the entities belonging to the same cluster have the same causal MAG (referred to as  $\alpha$ -clustering, Definition 4.2.4). An important point to notice is that while we assume

there is a underlying clustering on the entities, it is *learnt* by our algorithms. Similar assumptions are common for recovering the underlying clusters, in many areas, for e.g., crowd-sourcing applications [15, 17].

We first start with the observation that under  $(\alpha, \beta)$ -clustering, even entities belonging to the same cluster could have a different MAG, which makes exact recovery hard without making a significant number of interventions per entity. We present an algorithm that using at most  $O(\Delta \log(M/\delta)/(\alpha - \beta)^2)$  many interventions per entity, with probability at least  $1 - \delta$  (over only the randomness of the algorithm), can provably recover an *approximate* MAG for each entity. The approximation is such that for each entity we generate a MAG that is at most  $\beta n$  node-distance from the true MAG of that entity (see section 4.3). Here,  $\Delta$  is the maximum undirected degree of the causal MAGs. Our idea is to first recover the underlying clustering of entities by using a randomized set of interventions. Then, we distribute the interventions across the entities in each cluster, thereby, ensuring that the number of interventions per entity is small. By carefully combining the results learnt from these interventions we construct the approximate MAGs. For the number of interventions, the linear dependence on  $\Delta$  is not uncommon for learning causal graphs [88]. Moreover, most real-world causal bayesian networks are known to have small maximum degrees (see section 4.5).

Under the slightly more restrictive  $\alpha$ -clustering assumption, we present algorithms that can *exactly* recover all the MAGs using at most  $\min\{O(\Delta \log(M/\delta)/\alpha), O(\log(M/\delta)/\alpha + k^2)\}$  many interventions per entity (see section 4.4). Again, randomization plays an important role in our approach.

Complementing these upper bounds, we give a lower bound using Yao’s minimax principle [131] that shows for any (randomized or deterministic) algorithm  $\Omega(1/\alpha)$  interventions per entity is required for this causal discovery problem. This implies the  $1/\alpha$  dependence in our upper bound in the  $\alpha$ -clustering case is optimal.

Finally, a note about parameters. The  $(\alpha, \beta)$ -clustering is universal, in the sense that *any* collection of MAGs will satisfy the  $(\alpha, \beta)$ -clustering property for some value of  $\alpha, \beta$  (with  $\alpha > \beta$ ). Ideally, we would like in our problem instance,  $\alpha$  to be close to 1 and  $\beta$  to be close to 0. In most real-world applications, we would also expect  $k$  to be relatively small and  $M \gg n, k$ .

In section 4.5, we show experiments on data generated from both real and synthetic networks with added latents and demonstrate the efficacy of our algorithms for learning the underlying clustering and the MAGs.

## 4.2 Our Model and Problem Statement

In this section, we introduce the collaborative causal discovery problem. We start with some notations.

**Notation** Following the SCM framework [105], we represent the set of random variables of interest by  $V \cup L$  where  $V$  represents the set of endogenous (observed) variables that can be measured and  $L$  represents the set of exogenous (latent) variables that cannot be measured. Let  $|V| = n$ .

We assume that the causal Markov condition and faithfulness holds for both the observational and interventional distributions [65]. We use conditional independence (CI) tests of the form  $u \perp\!\!\!\perp v \mid Z$  or  $u \perp\!\!\!\perp v \mid \text{do}(w), Z$ , for some  $u, v, w \in V$  and  $Z \subseteq V$  (see section 4.6 for more details).

Throughout this chapter, unless otherwise specified, a path between two nodes is an undirected path. A path of only directed edges is called a directed path.  $u$  is called an ancestor of  $v$  and  $v$  a descendant of  $u$  if  $u = v$  or there is a directed path from  $u$  to  $v$ . A directed cycle occurs in  $G$  when  $u \rightarrow v$  is in  $G$  and  $v$  is an ancestor of  $u$ .

**Our Model** We assume that we have access to  $M$  entities labeled  $1, \dots, M$ , each of which can independently generate their own observational and interventional data. Each entity  $i$  has an associated causal DAG  $\mathcal{D}_i$  over  $V \cup L_i$ , where  $L_i$  represents the latent variables of entity  $i$ . In modeling the problem of causal discovery, complications arise in at least two ways:

- (i) **Latents.** We allow some variables (called latents) in the causal DAG to be unobservable. As regular DAGs are not sufficient to represent the observed distribution when there are latents, we use *ancestral graphical models* that have been proposed as an elegant and useful surrogate for DAG models with latent variables [108].

A mixed graph containing directed ( $\leftarrow$ ) and bidirected ( $\leftrightarrow$ ) edges is said to be *ancestral* if it has no directed cycles, and whenever there is a bidirected edge  $u \leftrightarrow v$ , then there is no directed path from  $u$  to  $v$  or from  $v$  to  $u$ . An ancestral graph on  $V$  (observables) is said to be *maximal*, if, for every pair of nonadjacent vertices  $u, v$ , there exists a set  $Z \subset V$  with  $u, v \notin Z$  such that  $u$  and  $v$  are  $m$ -separated (similar to  $d$ -separation, see Definition 4.6.2) conditioned on  $Z$ . Every DAG with latents (and selection variables) can be transformed into a unique maximal ancestral graph (MAG) over the observed variables [108].

- (ii) **Uniqueness.** Secondly, with just observational data, if the MAGs  $\mathcal{M}_1, \dots, \mathcal{M}_M$  are Markov equivalent, then, without additional strong assumptions they cannot be distinguished, even if they are all structurally different. To overcome the problem of being not identifiable within an equivalence class, we allow for interventions on observed variables. In particular, we focus on atomic interventions, which are the simplest and most commonly used intervention type, modeled through the do-operator [103]. As it turns out, Maximal Ancestral Graphs (MAGs) are uniquely identifiable using atomic interventions.<sup>2</sup>

---

<sup>2</sup>However, in the presence of latents, even with power of atomic interventions, the structure of a causal DAG is not uniquely identifiable. (see, e.g., In Figure 4.4 in section 4.6). Similarly, we can

Our objective will be to minimize these interventions. In particular, since each of these entities independently generate their own data, so we aim to reduce the number of interventions needed per entity. In causal discovery, minimizing the number of interventions while ensuring that they are of small size is an active research area [103, 115, 60, 59].

Given the  $M$  entities, let  $\mathcal{M}_i$  denote the MAG associated with entity  $i$  (the MAG constructed from the DAG  $\mathcal{D}_i$ ). Our goal is to collaboratively learn all these MAGs  $\mathcal{M}_1, \dots, \mathcal{M}_M$  while minimizing the maximum number of interventions per entity.

To facilitate this learning, we make a natural underlying clustering assumption that partitions the entities based on their respective MAGs such that: (i) any two entities belonging to the same cluster have MAGs that are “close” to each other, (ii) any two entities belonging to different clusters have MAGs that are “far” apart. Before stating this assumption formally, we need some definitions.

For MAG  $\mathcal{M}_i = (V, E_i)$ , we denote the children (through outgoing edges), parent (through incoming edges), and spouse (through bidirected edges) of a node  $u \in V$  as

$$\text{ch}_i(u) = \{v \mid u \rightarrow v \in E_i\}, \text{pa}_i(u) = \{v \mid u \leftarrow v \in E_i\}, \text{sp}_i(u) = \{v \mid u \leftrightarrow v \in E_i\}. \quad (4.1)$$

Also, define an incidence set for a vertex  $u \in V$  which contains an entry  $(v, \text{type})$  for every node  $v$  adjacent to  $u$  as

$$N_i(u) = \left\{ \begin{array}{ll} (v, \text{tail}) & \text{if } u \rightarrow v \in E_i \\ (v, \text{head}) & \text{if } u \leftarrow v \in E_i \\ (v, \text{bidirected}) & \text{if } u \leftrightarrow v \in E_i \end{array} \right\}. \quad (4.2)$$

---

show that using single vertex interventions, we also cannot exactly recover a wider class of acyclic graphs like ADMGs (Acyclic Directed Mixed Graphs).

Note that  $|N_i(u)|$  is the undirected degree of  $u$  in  $\mathcal{M}_i$ . We now define a distance measure between MAGs that captures structural similarity between them.

**Definition 4.2.1.** *Given two MAGs  $\mathcal{M}_i = (V, E_i)$  and  $\mathcal{M}_j = (V, E_j)$ , define the node-difference as the set:  $\text{diff}(\mathcal{M}_i, \mathcal{M}_j) = \{u \in V \mid N_i(u) \neq N_j(u)\}$ , and the node-distance as the cardinality of this set:  $d(\mathcal{M}_i, \mathcal{M}_j) = |\text{diff}(\mathcal{M}_i, \mathcal{M}_j)| = |\{u \in V \mid N_i(u) \neq N_j(u)\}|$ .*

Intuitively, the node distance captures the number of nodes whose incidence relationships differ. It is easy to observe that the node distance is a distance metric, and captures a strong structural similarity between the graphs. Two graphs  $\mathcal{M}_i, \mathcal{M}_j$  are identical iff  $d(\mathcal{M}_i, \mathcal{M}_j) = 0$ . For e.g., in Figure 4.3, we have two MAGs that satisfy  $d(\mathcal{M}_{12}, \mathcal{M}_{13}) = 2$  as  $\text{diff}(\mathcal{M}_{12}, \mathcal{M}_{13}) = \{x, z\}$ , where  $d(\mathcal{M}_{12}, \mathcal{M}_{21}) = 3$  as  $\text{diff}(\mathcal{M}_{12}, \mathcal{M}_{21}) = \{x, y, z\}$ . We are now ready to define a simple clustering property on MAGs.

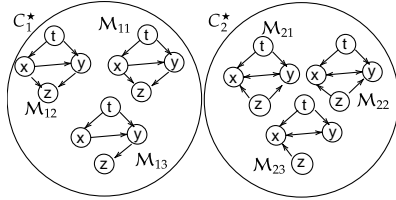


Figure 4.3: MAGs with  $(\alpha = 0.75, \beta = 0.5)$ -clustering. Every pair of graphs in  $C_1^*$  and  $C_2^*$  differ in at least  $3 (= 0.75 \times 4)$  nodes, while pairs of graphs within clusters differ by at most  $2 (= 0.5 \times 4)$  nodes.

**Definition 4.2.2**  $((\alpha, \beta)$ -clustering). *Let  $\mathcal{M}_1, \dots, \mathcal{M}_M$  be a set of  $M$  MAGs. We say that this set of MAGs satisfy the  $(\alpha, \beta)$ -clustering property, with  $\alpha > \beta \geq 0$ , if there exists a partitioning of  $[M]$  into sets (clusters)  $C_1^*, \dots, C_k^* \subset [M]$  (for some  $k \in \mathbb{N}$ ) such that for all  $(i, j) \in [M] \times [M]$ :*

- (i) *if  $i$  and  $j$  belong to same set (cluster), then  $d(\mathcal{M}_i, \mathcal{M}_j) \leq \beta n$ ;*
- (ii) *if  $i$  and  $j$  belong to different sets (clusters), then  $d(\mathcal{M}_i, \mathcal{M}_j) \geq \alpha n$ .*

Under this definition, all the  $M$  MAGs could be different. See, e.g., Figure 4.3. With right setting of  $\alpha > \beta$  we can capture any set of possible  $M$  MAGs. Therefore,

an algorithm such as FCI [119], that constructs PAGs might not be able to recover the clusters, as all the PAGs could be different, and the node-distance between PAGs does not correlate well with the node-distance between corresponding MAGs (e.g., see Figure 4.5 in section 4.6). We use the PAGs generated by FCI as a starting point for all our algorithms and further refine them. We assume that PAGs generated are correct (see section 4.6.2.1 for additional details).

**Definition 4.2.3** (Partial Ancestral Graph (PAG)). *Let  $[\mathcal{M}_i]$  denote the Markov equivalence class of the MAG  $\mathcal{M}_i$  and represented by the Partial Ancestral Graph (or PAG)  $\mathcal{U}_i = (V, \hat{E}_i)$ . Edges  $\hat{E}_i$  have three kinds of endpoints given by arrowheads ( $\leftarrow$ ), circles ( $\circ-$ ) and tails ( $-$ ).*

With this discussion, we introduce our collaborative causal graph learning problem as follows:

**Assumption:** MAGs  $\mathcal{M}_1, \dots, \mathcal{M}_M$  (associated with entities  $1, \dots, M$  respectively) satisfying the  $(\alpha, \beta)$ -clustering property

**Access to each entity:** Through conditional independence (CI) tests on observational and interventional distributions. Each entity generates their own (independent) data samples.

**Goal:** Learn  $\mathcal{M}_1, \dots, \mathcal{M}_M$  while minimizing the max. number of interventions per entity.

An interesting case of the Definition 4.2.2 is when  $\beta = 0$ .

**Definition 4.2.4** ( $\alpha$ -clustering). *We say a set of MAGs  $\mathcal{M}_1, \dots, \mathcal{M}_M$  satisfy the  $\alpha$ -clustering property, if and only if they satisfy  $(\alpha, 0)$ -clustering property.*

Note that  $\alpha$ -clustering is a natural property, wherein each cluster is associated with a single unique MAG, and all entities in the cluster have the same MAGs, and same conditional independences.

### 4.3 Causal Discovery under $(\alpha, \beta)$ -Clustering Property

In this section, we present our main algorithm for learning all the causal MAGs under the  $(\alpha, \beta)$ -clustering property. Missing details from this section are presented in section 4.6.3.

**Outline** All our algorithms are randomized, and succeed with high probability over the randomness introduced by the algorithm. The idea behind all our algorithms is to first learn the true clusters  $C_1^*, \dots, C_k^*$  using very few interventions. Once the true clusters are recovered, the idea is to distribute the interventions across the entities in each cluster and merge the results learned to recover the MAGs (section 4.3.2). For our algorithms, a lower bound for  $\alpha$  and upper bound for  $\beta$  is sufficient. In practice, a clustering of the PAGs (generated from FCI algorithm) can provide guidance about these bounds on  $\alpha, \beta$ , or if we have additional knowledge that  $\alpha \in [1 - \epsilon, 1]$  and  $\beta \in [0, \epsilon]$  for some constant  $\epsilon > 0$ , then, we can use binary search, that increases our intervention bounds by  $\log^2(n\epsilon)/(1 - 2\epsilon)^2$  factor. It is important to note that none of our algorithms require the knowledge of  $k$ .

---

**Algorithm 11** IDENTIFY-OUTNBR  $(\mathcal{U}_i, u)$

---

```

1: Input: node  $u \in V$ , PAG  $\mathcal{U}_i$  of entity  $i$ 
2: Output:  $\text{ch}_i(u)$ 
3:  $\text{ch}_i(u) = \{v \mid u \rightarrow v \in \mathcal{U}_i\}$ 
4: for  $v \in \Gamma_i(u)$  such that  $u \circ - \circ v$  or  $u \circ \rightarrow v \in \mathcal{U}_i$  do
5:   if  $u \not\perp\!\!\!\perp v \mid \text{do}(u)$  then
6:      $\text{ch}_i(u) \leftarrow \text{ch}_i(u) \cup \{v\}$ 
7:   end if
8: end for
9: Return  $\text{ch}_i(u)$ 

```

---

---

**Algorithm 12** IDENTIFY-BIDIRECTED ( $\mathcal{U}_i, u$ )

---

```
1: Input: node  $u \in V$ , PAG  $\mathcal{U}_i$  of entity  $i$ 
2: Output:  $\text{sp}_i(u)$ 
3:  $\text{sp}_i(u) = \{v \mid u \leftrightarrow v \in \mathcal{U}_i\}$ 
4: for  $v \in \Gamma_i(u)$  such that  $u \circ \leftarrow v$  or  $u \leftarrow \circ v$  or  $u \circ \rightarrow v \in \mathcal{U}_i$  do
5:   if  $u \perp\!\!\!\perp v \mid \text{do}(u)$  and  $u \perp\!\!\!\perp v \mid \text{do}(v)$  then
6:      $\text{sp}_i(u) \leftarrow \text{sp}_i(u) \cup \{v\}$ 
7:   end if
8: end for
9: Return  $\text{sp}_i(u)$ 
```

---

**Helper Routines** Let  $\Gamma_i(u)$  denote all nodes that are adjacent to  $u$  in the PAG  $\mathcal{U}_i$ , i.e.,  $\Gamma_i(u) = \{v \mid (u, v) \in \widehat{E}_i\}$ . Given the PAG  $\mathcal{U}_i$ , Algorithm IDENTIFY-OUTNBR identifies all the outgoing neighbors of any node  $u$  in  $\mathcal{M}_i$ . We look at edges of the form  $u \circ \leftarrow v$  or  $u \leftarrow \circ v$  in  $\mathcal{U}_i$  incident on  $u$ , and identify if  $u \rightarrow v$  using the CI-test  $u \perp\!\!\!\perp v \mid \text{do}(u)$ . This is based on the observation that any node  $v$  that is a descendant of  $u$  (including  $\text{ch}_i(u)$ ) satisfies  $u \not\perp\!\!\!\perp v \mid \text{do}(u)$ . Algorithm IDENTIFY-BIDIRECTED identifies all the bidirected edges incident on  $u$ . If there is an edge of the form  $u \circ \leftarrow v$  or  $u \leftarrow \circ v$  or  $u \circ \rightarrow v$  in the PAG, and  $v \notin \text{ch}_i(u)$  and  $u \notin \text{ch}_i(v)$ , then it must be a bidirected edge.

Using these helper routines, we give an Algorithm RECOVERG (in section 4.6.2) that recovers any MAG  $\mathcal{M}_i$  using  $n$  atomic interventions. Complementing this, we show that  $n$  interventions are also required. The missing details are presented in section 4.6.2.

**Proposition 4.3.1.** *There exists a causal MAG  $\mathcal{M}$  such that every adaptive or non-adaptive algorithm requires  $\Omega(n)$  many atomic interventions to recover  $\mathcal{M}$ .*

### 4.3.1 Recovering the Clusters

From the  $(\alpha, \beta)$ -clustering definition, we know that a pair of entities belonging to the same cluster have higher structural similarity between their MAGs than a pair of entities across different clusters. Let us start with a simplifying assumption

that  $\beta = 0$  (i.e.,  $\alpha$ -clustering). So, all the MAGs are separated by a distance of at least  $\alpha n$ . We make the observation that to identify that two MAGs, say  $\mathcal{M}_i$  and  $\mathcal{M}_j$  belong to different clusters, it suffices to find a node  $u$  from the node-difference set  $\text{diff}(\mathcal{M}_i, \mathcal{M}_j)$  and checking their neighbors using Algorithms IDENTIFY-OUTNBR and IDENTIFY-BIDIRECTED. We argue that (see Claim 4.6.14, section 4.6.4.2), with probability at least  $1 - \delta$ , we can identify one such node  $u \in \text{diff}(\mathcal{M}_i, \mathcal{M}_j)$  by sampling  $2 \log(M/\delta)/\alpha$  nodes uniformly from  $V$  as  $|\text{diff}(\mathcal{M}_i, \mathcal{M}_j)| = d(\mathcal{M}_i, \mathcal{M}_j) \geq \alpha n$ .<sup>3</sup> However, this approach will not succeed when  $\beta \neq 0$  because now we have MAGs in the same cluster that are also separated by non-zero distance.

---

**Algorithm 13**  $(\alpha, \beta)$ -BOUNDEDDEGREE

---

- 1: **Input:**  $\alpha > 0, \beta \geq 0 (< \alpha)$ , confidence parameter  $\delta > 0$ , PAGs  $\mathcal{U}_1, \dots, \mathcal{U}_M$  of  $M$  entities
  - 2: **Output:** Partition of  $[M]$  into clusters
  - 3: Let  $S$  denote a uniform sample of  $\frac{4 \log(M/\delta)}{(\alpha-\beta)^2}$  nodes from  $V$  selected with replacement.
  - 4: **for** every entity  $i \in [M]$  and  $u \in S$  **do**
  - 5:    $\text{ch}_i(u) \leftarrow \text{IDENTIFY-OUTNBR}(\mathcal{U}_i, u)$
  - 6:    $\text{sp}_i(u) \leftarrow \text{IDENTIFY-BIDIRECTED}(\mathcal{U}_i, u)$
  - 7:    $\text{pa}_i(u) \leftarrow \Gamma_i(u) \setminus (\text{ch}_i(u) \cup \text{sp}_i(u))$
  - 8:   Construct  $N_i(u)$  (defined in (4.2))
  - 9: **end for**
  - 10: Let  $\mathcal{P}$  denote an empty graph on set of entities  $[M]$
  - 11: **for** every pair of entities  $i, j$  **do**
  - 12:   Let  $\text{COUNT}(i, j) = \sum_{u \in S} \mathbf{1}\{N_i(u) = N_j(u)\}$
  - 13:   **if**  $\text{COUNT}(i, j) \geq (1 - \frac{\alpha+\beta}{2}) |S|$  **then**
  - 14:     Include an edge between  $i$  and  $j$  in  $\mathcal{P}$
  - 15:   **end if**
  - 16: **end for**
  - 17: Return connected components in  $\mathcal{P}$
- 

<sup>3</sup>For theoretical analysis, our intervention targets are randomly chosen, even with the knowledge available from PAGs, because in the worst-case the PAGs might contain no directed edges to help decide which nodes to intervene on. In practice, though if we already know edge orientations from PAG we do not have to relearn them, and a biased sampling based on edges uncertainties in PAGs might be a reasonable approach.

**Overview of Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE** We now build upon the above idea, to recover the true clusters  $C_1^*, \dots, C_k^*$  when  $\beta \neq 0$ . As identifying a node  $u \in \text{diff}(\mathcal{M}_i, \mathcal{M}_j)$  is not sufficient, we maintain a count of the number of nodes among the sampled set of nodes  $S$  that the pair of entities  $i, j$  have the same neighbors, i.e.,  $\text{COUNT}(i, j) = \sum_{u \in S} \mathbf{1}\{N_i(u) = N_j(u)\}$ . Based on a carefully chosen threshold value for the  $\text{COUNT}(i, j)$ , that arises through the analysis of our randomized algorithm, we classify whether a pair of entities belong to the same cluster correctly.

Overall, the idea here is to construct a graph  $\mathcal{P}$  on entities (i.e., the node set of  $\mathcal{P}$  is  $[M]$ ). We include an edge between two entities  $i$  and  $j$  if  $\text{COUNT}(i, j)$  is above the threshold  $(1 - (\alpha + \beta)/2)|S|$ . Using Lemma 4.3.2, we show that this threshold corresponds to the case where if the entities are from same true clusters, then the  $\text{COUNT}$  value corresponding to the pair is higher than the threshold; and if they are from different clusters it will be smaller, with high probability. This ensures that every entity is connected only to the entities belonging to the same true cluster. We return the connected components in  $\mathcal{P}$  as our clusters.

In Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE, we construct a uniform sample  $S$  of size:  $O(\log(M/\delta)/(\alpha - \beta)^2)$ , and identify all the neighbors of  $S$  for every entity  $i \in [M]$ . As we use IDENTIFY-BIDIRECTED to identify all the bi-directed edges, the total number of interventions used by an entity for this step is at most  $\Delta \cdot |S|$ . Combining all the above, using the next lemma, we show that with high probability Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE recovers all the true clusters.

**Lemma 4.3.2.** *If the underlying MAGs  $\mathcal{M}_1, \dots, \mathcal{M}_M$  satisfy  $(\alpha, \beta)$ -clustering property with true clusters  $C_1^*, \dots, C_k^*$  and have maximum undirected degree  $\Delta$ . Then, the Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE recovers the clusters  $C_1^*, \dots, C_k^*$  with probability at least  $1 - \delta$ . Every entity  $i \in [M]$  uses at most  $4(\Delta + 1) \log(M/\delta)/(\alpha - \beta)^2$  many atomic interventions.*

### 4.3.2 Learning Causal Graphs from $(\alpha, \beta)$ -Clustering

In this section, we outline an approach to recover a close approximation of the causal MAGs of all the entities, after correctly recovering the clusters using Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE. First, we note that since the  $(\alpha, \beta)$ -clustering allows the MAGs even in the same cluster to be different, the problem of exactly learning all the MAGs is challenging (with a small set of interventions) as causal edges learnt for an entity may not be relevant for another entity in the same cluster.

In the scenarios mentioned in the introduction, we expect the clusters to be more homogeneous, with many entities in the same cluster sharing the same MAG. We provide an overview of Algorithm  $(\alpha, \beta)$ -RECOVERY that recovers one such MAG called *dominant MAG* for every cluster. Consider a recovered cluster  $C_a^*$ , and a partitioning  $S_a^1, S_a^2, \dots$  of MAGs such that all MAGs in a partition  $S_a^i$  are equal for all  $i$ . We call the MAG  $\mathcal{M}_a^{\text{dom}}$  corresponding to the largest partition  $S_a^{\text{dom}}$  as the *dominant MAG* of  $C_a^*$ . The dominant MAG of a cluster is parameterized by  $\gamma_a = |S_a^{\text{dom}}|/|C_a^*|$  (fraction of the MAGs in the cluster that belong to the largest partition). We defer additional details of Algorithm  $(\alpha, \beta)$ -RECOVERY to section 4.6.3.1.

**Overview of Algorithm  $(\alpha, \beta)$ -RECOVERY.** After recovering the clustering using Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE, our goal is to learn the causal graphs. Using Algorithm  $(\alpha, \beta)$ -RECOVERY, we show that we can learn these graphs approximately up to a distance approximation of  $\beta n$ .

In a cluster  $C_a^*$ , we construct a partitioning of MAGs such that two MAGs belong to a partition if they are equal. The MAG corresponding to the largest partition is called the *dominant MAG*. Using our algorithm, we learn the dominant MAG correctly and return it as an output. As all the MAGs in the cluster satisfy  $(\alpha, \beta)$ -clustering property, the dominant MAG is within a distance of  $\beta n$  from the true MAG and therefore is a good approximation of the true MAG.

For learning the dominant MAG, there are two steps. First, we select a node uniformly at random for every entity and intervene on the node and its neighbors to learn all the edges incident on the node. Next, we construct the dominant MAG by combining the neighborhoods of each individual node. Let  $u$  be any node and  $T_u$  denote the set of all entities which intervened on  $u$  in the first step. Now, among all the neighborhoods identified by the entities in  $T_u$ , we do not know which of them correspond to that of the dominant MAG. In order to identify this, we use a threshold-based approach and assign a score to every entity in  $T_u$ . The score of an entity  $i$  is the number of entities in  $T_u$  that has the same neighborhood of  $u$  as that of entity  $i$ . Finally, we select the entity with the maximum score and assign the neighborhood of the entity as the neighborhood of  $u$  for the dominant MAG (Lines 12-15 in Algorithm  $(\alpha, \beta)$ -RECOVERY). We argue that if the cluster size is large (see Theorem 4.3.3), the neighborhoods of nodes using entities with maximum scores are equal to that of the dominant MAG. This is because the dominant MAG has the largest partition size, and if a sufficiently large number of entities (across all partitions) are assigned node  $u$ , then, many of them will be entities from the dominant MAG partition.

As the entities satisfy  $(\alpha, \beta)$ -clustering property, for all entities the recovered MAGs (dominant MAGs) are close to the true MAGs, and within a distance of at most  $\beta n$ . Note that any MAG from the cluster is within a distance of at most  $\beta n$  due to  $(\alpha, \beta)$ -clustering property, but naively generating a valid MAG from a cluster will require  $n$  interventions on one entity from Proposition 4.3.1. Our actual guarantee is somewhat stronger, as in fact, for the entities whose MAGs are dominant in their cluster, we do recover the exact MAGs. We have the result:

**Theorem 4.3.3.** *Suppose  $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_M$  satisfy  $(\alpha, \beta)$  clustering property. If  $\gamma_a > 1/2$  and  $C_a^* = \Omega(n \log(n/M\delta)(2\gamma_a - 1)^2)$  for all  $a \in [k]$ , then, Algorithm  $(\alpha, \beta)$ -RECOVERY recovers graphs  $\widehat{\mathcal{M}}_1, \dots, \widehat{\mathcal{M}}_M$  such that for every entity  $i \in [M]$ , we have*

$d(\mathcal{M}_i, \widehat{\mathcal{M}}_i) \leq \beta n$  with probability  $1 - \delta$ . Every entity uses at most  $(\Delta + 1) + 4(\Delta + 1) \log(M/\delta)/(\alpha - \beta)^2$  many atomic interventions.

## 4.4 Causal Discovery under $\alpha$ -Clustering Property

In the previous section, we discussed the more general  $(\alpha, \beta)$ -clustering scenario where we manage to construct a good approximation to all the MAGs. Now, we show that we can in fact recover all the MAGs exactly, if we make a stronger assumption. Suppose the MAGs  $\mathcal{M}_1, \dots, \mathcal{M}_M$  of the  $M$  entities satisfy the  $\alpha$ -clustering property (Defn. 4.2.4). Firstly, we can design an algorithm similar to Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE (see Algorithm  $\alpha$ -BOUNDEDDEGREE, Section 4.6.4.3) that recovers the causal MAGs exactly with  $O(\Delta \log(M/\delta)/\alpha)$  many interventions per entity, succeeding with probability  $1 - \delta$ . Note that this has a better  $1/\alpha$  term in the intervention bound, instead of  $1/\alpha^2$  (when  $\beta = 0$ ) term arising in Theorem 4.3.3. In absence of latents, we can further improve it to  $O(\log(M/\delta)/\alpha)$  many interventions per entity (see Algorithm NOLATENTS in Section 4.6.4.2).

### 4.4.1 Recovering the Clusters

In this section, we present another approach (Algorithm  $\alpha$ -GENERAL) with an improved result that requires fewer number of interventions, even when  $\Delta$  is big, provided that each cluster has at least  $\Omega(n \log(M/\delta))$  entities. Missing details of Algorithm  $\alpha$ -GENERAL are in Section 4.6.4.4.

**Overview of Algorithm  $\alpha$ -GENERAL** First, using a similar approach as Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE, we construct a uniform sample  $S \subseteq V$ , and find all the outgoing neighbors of nodes in  $S$ , for every entity  $i \in [M]$ . Then, we construct a graph on entities denoted by  $\mathcal{P}$ , where we include an edge between a pair of entities if the outgoing neighbors of the set of sampled nodes  $S$ , and the set of neighbors in PAGs associated with the entities (obtained from FCI) are the same. However, due

to the presence of bidirected edges, it is possible that the connected components of  $\mathcal{P}$  may not represent the true clusters  $C_1^*, \dots, C_k^*$ .

We make the observation that a pair of entities  $i, j$  that have an edge in this  $\mathcal{P}$  and from different true clusters, can differ only if there is a node  $u$  such that  $u$  has a bidirected edge  $u \leftrightarrow v$  in  $\mathcal{M}_i$ , and a directed edge  $u \leftarrow v$  in  $\mathcal{M}_j$  (or vice-versa). Intervening on both  $u$  and  $v$  will separate these entities, our main idea is to ensure that this happens. First, we show how to *detect* if there are at least two true clusters in any connected component of  $\mathcal{P}$ . Then, we identify all the entities belonging to these two clusters and remove the edges between these entities in  $\mathcal{P}$  and continue.

More formally, let  $T_1, \dots, T_{k'}$  be the partition of  $[M]$  provided by the  $k'$  connected components of  $\mathcal{P}$  and some of these can contain more than one true cluster, hence  $k' \leq k$  and we focus on detecting such events. Let  $\pi : [M] \rightarrow V$  denote a mapping from the set of entities to the nodes in  $V$  such that  $\pi(i)$  is chosen uniformly at random from  $V$  for every entity  $i$ . For every entity  $i$ , we intervene on the node  $\pi(i)$ . To detect that there are at least two clusters in a given subset  $T_a$  of entities, we show that there are two entities  $i, j$  with an edge in  $\mathcal{P}$  and for some node  $u \in S$ , we can identify the neighbor  $v \in \Gamma_i(u) \cap \Gamma_j(u)$  such that  $u \leftrightarrow v$  is an edge in  $\mathcal{M}_i$  and  $u \leftarrow v$  is an edge in  $\mathcal{M}_j$  (or vice-versa). As there are at least  $\Omega(n \log(M/\delta))$  entities in each of these two true clusters in  $T_a$ , for some  $i, j \in T_a$ , we can show that  $\pi(i) = \pi(j) = v$  with probability at least  $1 - \delta$ .

After detecting the event that a component  $T_a$  of  $\mathcal{P}$  contains entities from at least two different true clusters (say,  $C_b^*$  and  $C_c^*$ ) due to an edge  $(u, v)$  as above, we intervene on  $v$  for every entity in  $T_a$ . By intervening on  $v$  (and  $u \in S$ ), we can separate all entities in  $T_a$  that belong to true clusters  $C_b^*$  and  $C_c^*$ , and remove edges between such entity pairs from  $\mathcal{P}$ . We repeat this above procedure of refining  $\mathcal{P}$ . In each iteration, we will have removed all edges between every pair of entities belonging to at least two different true clusters. Since there are at most  $k^2$  different true cluster

pairs, after  $k^2$  iterations the connected components remaining correspond to the true clusters (with high probability). This can be done without knowing the value of  $k$ , by checking whether the connected components in  $\mathcal{P}$  change or not after each iteration of the above idea.

#### 4.4.2 Learning Causal Graphs from $\alpha$ -Clustering

The idea of going from clusters to MAGs is simple and based on distributing the interventions across the entities in the cluster. Since under  $\alpha$ -clustering, entities belonging to a cluster share the same MAG, combining the results is relatively simpler (see Section 4.6.4.1). Combining the guarantees of  $\alpha$ -GENERAL and  $\alpha$ -BOUNDEDDEGREE, we have:

**Theorem 4.4.1.** *If MAGs  $\mathcal{M}_1, \dots, \mathcal{M}_M$  satisfy  $\alpha$ -clustering property with true clusters  $C_1^*, \dots, C_k^*$  such that  $\min_{b \in [k]} |C_b^*| = \Omega(n \log(M/\delta))$ . Then, there is an algorithm that exactly learns all these MAGs with probability at least  $1 - \delta$ . Every entity  $i \in [M]$  uses  $\min \{O(\Delta \log(M/\delta)/\alpha), O(\log(M/\delta)/\alpha + k^2)\}$  many atomic interventions.*

#### 4.4.3 Lower Bound on the Number of Interventions

We now give a lower bound on the number of atomic interventions needed for every algorithm that recovers the true clusters on the MAGs  $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_M$ . Since a lower bound under  $\alpha$ -clustering is also a lower bound under  $(\alpha, \beta)$ -clustering, we work with the  $\alpha$ -clustering property here. First, we show that to identify whether a given pair of entities  $i, j$  belong to the same true cluster or not, every (randomized or deterministic) algorithm must make  $\Omega(1/\alpha)$  interventions for both  $i$  and  $j$ .

Our main idea here is to use the famous Yao’s minimax theorem [131] to get lower bounds on randomized algorithms. Yao’s theorem states that an *average case* lower bound on a deterministic algorithm implies a *worst case* lower bound on randomized algorithms. To show a lower bound using Yao’s minimax theorem, we construct a distribution  $\mu$  on MAG pairs and show that every deterministic algorithm requires

$\Omega(1/\alpha)$  interventions for distinguishing a pair of MAGs drawn from  $\mu$ . The construction of this distribution is presented in Section 4.6.5. We summarize the result:

**Theorem 4.4.2.** *Suppose the underlying MAGs  $\mathcal{M}_1, \dots, \mathcal{M}_M$  satisfy  $\alpha$ -clustering property. In order to recover the clusters with probability  $2/3$ , every (randomized or deterministic) algorithm requires  $\Omega(1/\alpha)$  interventions for every entity in  $[M]$ .*

## 4.5 Experimental Evaluation

In this section, we provide an evaluation of our approaches on data generated from real and synthetic causal networks for learning MAGs satisfying  $(\alpha, \beta)$ -clustering property. We defer additional details, results, and evaluation for  $\alpha$ -clustering to Section 4.6.6.

**Causal Networks** We consider the following real-world Bayesian networks from the *Bayesian Network Repository* which cover a wide variety of domains: *Asia* (Lung cancer) (8 nodes, 8 edges), *Earthquake* (5 nodes, 4 edges), *Sachs* (Protein networks) (11 nodes, 17 edges), and *Survey* (6 nodes, 6 edges). For the synthetic data, we use Erdős-Rényi random graphs (10 nodes). We use the term “causal network” to refer to these ground-truth Bayesian networks.

**Parameters** For each causal network, we start from the corresponding DAG, and generate  $M$  MAGs (one for each entity) split into  $k$  clusters that satisfy the  $(\alpha, \beta)$ -clustering property through random changes to the graph. We also randomly introduce two latents in each graph, and account for them in MAG constructions. For more details, refer Section 4.6.6. We set number of entities  $M = 40$ , number of clusters  $k = 2$ ,  $\alpha = 0.60$ ,  $\beta = 0.20$ , and dominant MAG parameter  $\gamma = 0.90$  for both the clusters. For the synthetic data generated using Erdős-Rényi model, we use  $n = 10$ , probability of edge 0.3.

**Evaluation of Clustering** First, we focus on recovering the clustering using Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE. As a baseline, we employ the well-studied FCI algorithm based on purely observational data [119]. After recovering the PAGs corresponding to the MAGs using FCI, we cluster them by constructing a similarity graph (similar to  $(\alpha, \beta)$ -BOUNDEDDEGREE) defined on the set of entities (refer Section 4.6.6 for more details). For Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE, we first construct a sample  $S$ , and perform various interventions based on the set  $S$  for every entity to obtain the clusters. We also implemented another baseline algorithm (GREEDY) that uses interventions, based on a greedy idea that selects nodes to set  $S$  in Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE by considering nodes in increasing order of their degree in the PAGs returned by FCI. We use this ordering to minimize the no. of interventions as we intervene on every node in  $S$  and their neighbors.

**Metrics** We use the following standard metrics for comparing the clustering performance: *precision* (fraction of pairs of entities correctly placed in a cluster together to the total number of pairs placed in a cluster together), *recall* (fraction of pairs of entities correctly placed in a cluster together to the total number of pairs in the same ground truth clusters), and *accuracy* (fraction of pairs of entities correctly placed or not placed in a cluster to the total number of pairs of entities).

| Causal Network     | FCI             |                  |                 | $(\alpha, \beta)$ -BOUNDEDDEGREE (Alg. 13) |                 |                  | Maximum # Interventions |
|--------------------|-----------------|------------------|-----------------|--|-----------------|------------------|-------------------------|
|                    | Precision       | Recall           | Accuracy        | Precision                                  | Recall          | Accuracy         |                         |
| <i>Earthquake</i>  | $0.57 \pm 0.18$ | $0.94 \pm 0.013$ | $0.58 \pm 0.18$ | $0.78 \pm 0.24$                            | $0.92 \pm 0.03$ | $0.77 \pm 0.23$  | 4                       |
| <i>Survey</i>      | $0.62 \pm 0.21$ | $0.94 \pm 0.013$ | $0.62 \pm 0.2$  | $0.64 \pm 0.23$                            | $0.97 \pm 0.02$ | $0.63 \pm 0.23$  | 5                       |
| <i>Asia</i>        | $0.57 \pm 0.18$ | $0.94 \pm 0.013$ | $0.58 \pm 0.18$ | $0.92 \pm 0.14$                            | $0.95 \pm 0.03$ | $0.91 \pm 0.14$  | 5                       |
| <i>Sachs</i>       | $0.52 \pm 0.12$ | $0.94 \pm 0.01$  | $0.52 \pm 0.12$ | $0.89 \pm 0.20$                            | $0.96 \pm 0.02$ | $0.88 \pm 0.19$  | 6                       |
| <i>Erdős-Rényi</i> | $0.62 \pm 0.21$ | $0.94 \pm 0.02$  | $0.62 \pm 0.21$ | $1.0 \pm 0.00$                             | $0.95 \pm 0.02$ | $0.97 \pm 0.013$ | 6                       |

Table 4.1: In this table, we present the precision, recall and accuracy values obtained by our Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE and using FCI. Each cell includes the mean value along with the standard deviation computed over 10 runs. The last column represents the maximum number of interventions per entity including both Algorithms  $(\alpha, \beta)$ -BOUNDEDDEGREE and  $(\alpha, \beta)$ -RECOVERY.

**Results** In Table 4.1, we compare Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE to FCI on the clustering results. For Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE, we use a sample  $S$  of size 1, and observe in Figure 4.8 (Section 4.6.6), that this corresponds to about 3 interventions per entity. With increase in sample size, we observed that the results were either comparable or better. We observe that our approach leads to considerably better performance in terms of the accuracy metric with an average difference in mean accuracy of about 0.25. This is because FCI recovers partial graphs, and clustering based on the partial information results in poor accuracy. Because of the presence of a dominant MAG within each cluster, we observe that the corresponding entities are always assigned to the same cluster, resulting in high recall for both  $(\alpha, \beta)$ -BOUNDEDDEGREE and FCI. We observe a higher value of precision for our algorithms, because FCI is unable to correctly classify the MAGs that are different from the dominating MAG.

Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE outperforms the GREEDY baseline for the same sample(S) size. For example, on the *Earthquake* and *Survey* causal networks, Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE obtains the mean accuracy values of 0.77 and 0.63 respectively, while GREEDY for the same number of interventions obtained an accuracy of only 0.487 and 0.486 respectively. For the remaining networks, the accuracy values of GREEDY are almost comparable to our Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE.

After clustering, we recover the dominant MAGs using Algorithm  $(\alpha, \beta)$ -RECOVERY, and observe that the additional interventions needed are bounded by the maximum degree of the graphs (see Theorem 4.3.3). This is represented in the last column in Table 4.1. We observe that our *collaborative* algorithms use fewer interventions for dominant MAG recovery compared to the number of nodes in each graph. E.g., in the Erdős-Rényi setup, the number of nodes  $n = 10$ , whereas we use at most 6 interventions per entity. Thus, compared to the worst-case, cutting the number of interventions for each entity by 40%.

## 4.6 Additional Details

In this section, we present missing details from sections 4.1-4.5.

### 4.6.1 Maximal Ancestral Graphs

Ancestral graphical models were introduced motivated by the need to represent data generating processes that involve latent variables. In this chapter, we work with a class of graphical models, the maximal ancestral graph (MAG), which are a generalization of DAGs and are closed under marginalization and conditioning [108]. A maximal ancestral graph (MAG) is a (directed) mixed graph that may contain two kinds of edges: directed edges ( $\rightarrow$ ) and bi-directed edges ( $\leftrightarrow$ ). Before defining a MAG, we need some preliminaries.

Consider a mixed graph  $\mathcal{G}$ . Given an path  $\pi = \langle u, \dots, w, \dots, v \rangle$ ,  $w$  is a collider on  $\pi$  if the two edges incident to  $w$  in  $\pi$  are both into  $w$ , that is, have an arrowhead into  $w$ ; otherwise it is called a non-collider on  $\pi$ . Let  $S$  be any subset of nodes in the graph  $\mathcal{G}$ . An *inducing path* relative to  $S$  is a path on which every node not in  $S$  (except for the endpoints) is a collider on the path and every collider is an ancestor of an endpoint of the path.

**Definition 4.6.1.** *A mixed graph is called a maximal ancestral graph (MAG) if*

1. *The mixed graph is ancestral, i.e., it has no directed cycles, and whenever there is a bidirected edge  $u \leftrightarrow v$ , then there is no directed path from  $u$  to  $v$  or  $v$  to  $u$ .*
2. *There is no inducing path between any two non-adjacent nodes.*

It is straightforward to extend the notion of d-separation in DAGs to mixed graphs using the notion of m-separation [108].

**Definition 4.6.2.** *In a mixed graph, a path  $\pi$  between nodes  $u$  and  $v$  is m-connecting relative to a (possibly empty) set of nodes  $Z$  with  $u, v \notin Z$  if*

1. *every non-collider on  $\pi$  is not a member of  $Z$ ;*

2. every collider on  $\pi$  is an ancestor of some member of  $Z$ .

$u$  and  $v$  are said to be  $m$ -separated by  $Z$  if there is no  $m$ -connected path between  $u$  and  $v$  relative to  $Z$ .

**Conversion of a DAG to a MAG.** The following construction gives us a MAG  $\mathcal{M}$  from a DAG  $\mathcal{D}$ :

1. for each pair of variables  $u, v \in V$ ,  $u$  and  $v$  are adjacent in  $\mathcal{M}$  if and only if there is an inducing path between them relative to  $L$  in  $\mathcal{D}$ . The skeleton or the undirected graph constructed from PAG  $\mathcal{U}$  (obtained using FCI [119]) by ignoring the directions of edges captures all the edges in  $\mathcal{M}$ .
2. for each pair of adjacent variables  $u, v$  in  $\mathcal{M}$ , orient the edge as  $u \rightarrow v$  in  $\mathcal{M}$  if  $u$  is an ancestor of  $v$  in  $\mathcal{D}$ ; orient it as  $u \leftarrow v$  in  $\mathcal{M}$  if  $v$  is an ancestor of  $u$  in  $\mathcal{D}$ ; orient it as  $u \leftrightarrow v$  in  $\mathcal{M}$  otherwise.

Several DAGs can lead to the same MAG (See Figure 4.4c). Essentially a MAG represents a set of DAGs that have the exact same d-separation structures and ancestral relationships among the observed variables. By construction, the MAG is unique for a given DAG.

As a further evidence to the claim that interventions are required, see Figure 4.5, that gives an example of two MAGs separated by a distance of  $\frac{n}{2}$  and have the same Partial Ancestral Graph identified by FCI [133].

**Conditional Independence (CI) Tests** Conditional independence tests are an important building block in causal discovery.

- (i) CI-test in observational distribution: Given  $u, v \in V$ ,  $Z \subset V$  check whether  $u$  is independent of  $v$  given  $Z$ , denoted by  $u \perp\!\!\!\perp v \mid Z$ .

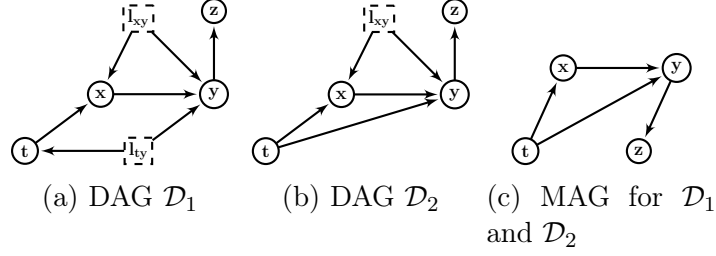


Figure 4.4: Different DAGs with same MAG. It is easy to observe that, no single vertex interventions can differentiate  $\mathcal{D}_1$  from  $\mathcal{D}_2$ .

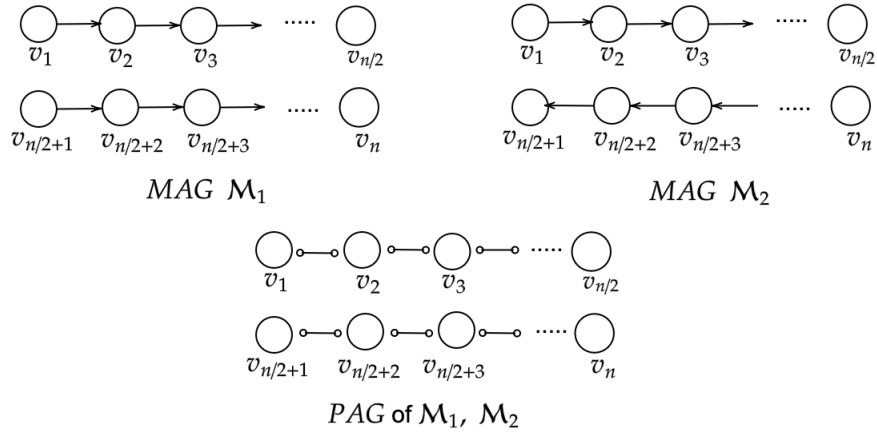


Figure 4.5: An example of MAGs  $\mathcal{M}_1$  and  $\mathcal{M}_2$  with large distance  $d(\mathcal{M}_1, \mathcal{M}_2)$  but generating the same PAG.

- (ii) CI-test in interventional distribution: Given  $u, v \in V$ ,  $Z \subset V$ , and  $w \in V$ , check whether  $u$  is independent of  $v$  given  $Z$  in the interventional distribution of  $w$ , denoted by  $u \perp\!\!\!\perp v \mid Z, \text{do}(w)$  where  $\text{do}(w)$  is the intervention on the variable  $w$ .

The convergence rates of CI tests are well-known [98] which can be used to obtain the required sample size bounds for any of the PAG estimation procedures for the desired Type 1 error bound (omitted here). Note that in our experiments (section 4.5), we do run CI tests on actual data samples generated by our model.

### 4.6.2 Helper Routines

In this section, we present missing details about the helper routines from section 4.3.

**Claim 4.6.3.** *Suppose  $\mathcal{D}_i$  is the DAG and  $\mathcal{M}_i$  is the corresponding MAG for some entity  $i \in [M]$ . Then,  $u \not\perp\!\!\!\perp v \mid \text{do}(u)$  iff  $u$  is an ancestor of  $v$  in the graph  $\mathcal{D}_i$ .*

*Proof.* We follow a proof similar to Lemma 1 in [88]. If  $u$  is an ancestor of  $v$  in the graph  $\mathcal{D}_i$  using the path  $\pi_{uv}$ , then, in the mutilated graph corresponding to  $\text{do}(u)$ , the path  $\pi_{uv}$  remains intact. From d-separation [105],  $\pi_{uv}$  can only be blocked by conditioning on one of the nodes that are not end points. As we do not condition on any variables in the CI-test  $u \perp\!\!\!\perp v \mid \text{do}(u)$  and therefore do not block the path  $\pi_{uv}$ , we have  $u \not\perp\!\!\!\perp v \mid \text{do}(u)$ .

Now, we consider the other direction. If  $u \not\perp\!\!\!\perp v \mid \text{do}(u)$ , then, there is at least a path  $\pi_{uv}$  between  $u$  and  $v$  that is not blocked. In the mutilated graph corresponding to the interventional distribution  $\text{do}(u)$ , the incoming edges into the node  $u$  are removed. In the path  $\pi_{uv}$ , the edge incident on  $u$  is an outgoing edge. If there is a collider on  $\pi_{uv}$ , we have blocked the path by not conditioning on it (from d-separation). As the path is not blocked, it implies that there is no collider on the path. Therefore, the path  $\pi_{uv}$  is a directed path from  $u$  to  $v$ . Hence, the claim.  $\square$

**Claim 4.6.4.** *Given an entity  $i \in [M]$ , and a node  $u \in V$ , Algorithm IDENTIFY-OUTNBR identifies all outgoing edges of  $u$  in  $\mathcal{M}_i$  ( $ch_i(u)$ ) correctly using an intervention on  $u$ .*

*Proof.* We know that  $\mathcal{U}_i = (V, \widehat{E}_i)$  represents the partial ancestral graph of  $\mathcal{M}_i$ . We observe that any outgoing edge  $(u, v)$  incident on a node  $u$  in the PAG  $\mathcal{U}_i$  can be of the form  $u \circ - \circ v$  or  $u \circ \rightarrow v$ . Otherwise, we already know that the edge is not an outgoing edge from  $u$ . We claim that we can identify an outgoing edge  $(u, v)$  from a node  $u$  correctly, if CI-test returns  $u \not\perp\!\!\!\perp v \mid \text{do}(u)$  for every  $v \in \Gamma_i(u)$  satisfying the

condition mentioned above. From Claim 4.6.3, we have that  $u \not\perp\!\!\!\perp v \mid \text{do}(u)$  iff  $u$  is an ancestor of  $v$  in  $\mathcal{D}_i$ , which implies  $u \rightarrow v$  is present in  $\mathcal{M}_i$  and  $v \in \text{ch}_i(u)$ .  $\square$

**Claim 4.6.5.** *Given an entity  $i \in [M]$ , and a node  $u \in V$ , Algorithm IDENTIFY-BIDIRECTED identifies all bidirected edges incident on  $u$  in  $\mathcal{M}_i$  ( $\text{sp}_i(u)$ ) correctly using atomic interventions on all nodes in  $\Gamma_i(u)$ .*

*Proof.* We observe that any bi-directed edge  $(u, v)$  incident on a node  $u \in V$  in the PAG  $\mathcal{U}_i$  can be of the form  $u \circ \leftarrow v$  or  $u \leftarrow \circ v$  or  $u \circ \rightarrow v$ . Otherwise, we already know that the edge is not a bi-directed edge incident at  $u$ . In Algorithm IDENTIFY-BIDIRECTED, for every neighbor  $v$  of  $u$  in the PAG  $\mathcal{U}_i$  satisfying the above condition, we check if  $u \perp\!\!\!\perp v \mid \text{do}(u)$  and  $v \perp\!\!\!\perp u \mid \text{do}(v)$  is satisfied. From Claim 4.6.3, we know that if  $u \not\perp\!\!\!\perp v \mid \text{do}(u)$ , then  $u$  is an ancestor of  $v$  in  $\mathcal{D}_i$  (similarly,  $v$  is an ancestor of  $u$  in  $\mathcal{D}_i$  if  $u \not\perp\!\!\!\perp v \mid \text{do}(v)$ ). So, if  $u \perp\!\!\!\perp v \mid \text{do}(u)$  and  $u \perp\!\!\!\perp v \mid \text{do}(v)$ , then,  $u$  is not an ancestor of  $v$  or vice-versa, which implies  $u \leftrightarrow v$  is present in  $\mathcal{M}_i$ , i.e.,  $v \in \text{sp}_i(u)$ . As we perform an intervention for every neighbor of  $u$  in  $\mathcal{U}_i$ , we have the claim.  $\square$

**Algorithm RECOVERG** For every  $u \in V$ , first identify outgoing neighbors using Algorithm IDENTIFY-OUTNBR and then identify all the bidirected edges incident on  $u$  using Algorithm IDENTIFY-BIDIRECTED.

**Lemma 4.6.6.** *Algorithm RECOVERG recovers all edges of  $\mathcal{M}_i$ , for an entity  $i \in [M]$  using  $n$  atomic interventions.*

*Proof.* Given an entity  $i \in [M]$ , we obtain the partial ancestral graph  $\mathcal{U}_i$  from observational data. Using Algorithm RECOVERG, we create interventions for every node  $u \in V$ . For every node  $u$ , we correctly identify all the outgoing neighbors of  $u$  using Algorithm IDENTIFY-OUTNBR (Claim 4.6.4) and all the bidirected edges using Algorithm IDENTIFY-BIDIRECTED (Claim 4.6.5). Therefore, we have recovered all edges of  $\mathcal{M}_i$  using  $n$  atomic interventions.  $\square$

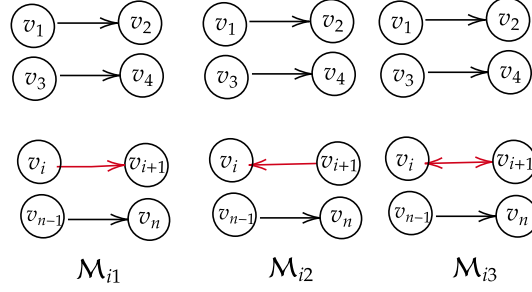


Figure 4.6: The MAGs used in the proof of Proposition 4.6.7.

**Proposition 4.6.7.** [Proposition 4.3.1 restated] *There exists a causal MAG  $\mathcal{M}$  such that every adaptive or non-adaptive algorithm requires  $n$  many atomic interventions to recover  $\mathcal{M}$ .*

*Proof.* Suppose the set of nodes of an unknown MAG  $\mathcal{M}$  is given by  $V = \{v_1, v_2, \dots, v_n\}$ . We denote ALG by any adaptive or non-adaptive *deterministic* algorithm that recovers  $\mathcal{M}$  using the set of interventions  $\mathcal{S} \subseteq V$ . For the sake of contradiction, let ALG recover  $\mathcal{M}$  correctly and  $v_i$  be the vertex that has not been intervened on, i.e.,  $v_i \notin \mathcal{S}$ .

Construct the MAGs  $\mathcal{M}_{i1}, \mathcal{M}_{i2}, \mathcal{M}_{i3}$  with edges  $E_{i1} = \{v_1 \rightarrow v_2, v_3 \rightarrow v_4, \dots, v_i \rightarrow v_{i+1}, \dots, v_{n-1} \rightarrow v_n\}$ ,  $E_{i2} = \{v_1 \rightarrow v_2, v_3 \rightarrow v_4, \dots, v_i \leftarrow v_{i+1}, \dots, v_{n-1} \rightarrow v_n\}$ ,  $E_{i3} = \{v_1 \rightarrow v_2, v_3 \rightarrow v_4, \dots, v_i \leftrightarrow v_{i+1}, \dots, v_{n-1} \rightarrow v_n\}$  respectively (see Figure 4.6).

Upon termination, ALG will have recovered one of the MAGs  $\mathcal{M}_{i1}, \mathcal{M}_{i2}$  or  $\mathcal{M}_{i3}$ . As  $v_i \notin \mathcal{S}$ , we will argue that the true MAG is different from the recovered MAG. We consider two cases:

1. If  $v_{i+1} \in \mathcal{S}$ . First, we observe that for all three MAGs  $\mathcal{M}_{i1}, \mathcal{M}_{i2}$  and  $\mathcal{M}_{i3}$ , the CI-test  $v_i \not\perp\!\!\!\perp v_{i+1}$ . For MAGs  $\mathcal{M}_{i1}$  and  $\mathcal{M}_{i2}$ , we have  $v_i \perp\!\!\!\perp v_{i+1} \mid \text{do}(v_{i+1})$  while  $v_i \not\perp\!\!\!\perp v_{i+1} \mid \text{do}(v_{i+1})$  for MAG  $\mathcal{M}_{i2}$ . As these are the only possible CI-tests for vertices  $v_i$  and  $v_{i+1}$ , the algorithm ALG cannot differentiate between  $\mathcal{M}_{i1}$  and  $\mathcal{M}_{i3}$ . If ALG recovers  $\mathcal{M}_{i1}$ , then, we can set  $\mathcal{M}$  to be  $\mathcal{M}_{i3}$ . This is a contradiction.

2. If  $v_{i+1} \notin \mathcal{S}$ . We observe that for all three MAGs  $\mathcal{M}_{i1}, \mathcal{M}_{i2}$  and  $\mathcal{M}_{i3}$ , the CI-test  $v_i \not\perp\!\!\!\perp v_{i+1}$ , and it is the only possible CI-test involving vertices  $v_i$  and  $v_{i+1}$ . Therefore, the algorithm ALG cannot differentiate between  $\mathcal{M}_{i1}, \mathcal{M}_{i2}$  and  $\mathcal{M}_{i3}$ . If ALG recovers  $\mathcal{M}_{i1}$ , then, we can set  $\mathcal{M}$  to be  $\mathcal{M}_{i2}, \mathcal{M}_{i3}$  and similarly for other cases. This is a contradiction.

Therefore, to recover  $\mathcal{M} \in \{\mathcal{M}_{i1}, \mathcal{M}_{i2}, \mathcal{M}_{i3}\}$  correctly, we must have  $v_i \in \mathcal{S}$ . As  $i$  is chosen arbitrarily, and for every  $i$  we can construct the MAGs  $\mathcal{M}_{i1}, \mathcal{M}_{i2}, \mathcal{M}_{i3}$ , such that any adaptive or non-adaptive deterministic algorithm requires interventions on every node.

We can extend the proof to include *randomized* algorithms, with success probability strictly greater than  $1/2$ , by observing that when  $v_i \notin \mathcal{S}$ , ALG has at least two MAGs among  $\mathcal{M}_{i1}, \mathcal{M}_{i2}, \mathcal{M}_{i3}$  that it cannot differentiate (as argued using two cases above). □

#### 4.6.2.1 Handling Uncertainty in PAG Estimation

We assume throughout this chapter that the initial PAGs (fed to our algorithms) are estimated correctly from observational data. We outline some reasons behind such an assumption.

- (a) PAG estimation is a very well-studied problem in causal discovery from both a theoretical and practical perspective. Well known algorithms for recovering PAGs, such as FCI (Fast Causal Inference), are known to be sound and complete (see [120] and [133]). Also, recent variations of FCI such as Really Fast Causal Inference (RFCI) have sped up the FCI procedure [42]. Today FCI/RFCI procedures are commonly used in practice, with various implementations available [78].
- (b) Note, for all our algorithms and bounds, all that we require from the PAGs is that they have the correct (undirected) skeleton as their corresponding MAGs,

i.e., we could just ignore all the directed edges in the initial PAGs and replace them with edges before using them in our algorithms, and this would not change our results.

- (c) Finally, we could even relax our assumptions and tolerate error even in skeleton estimation. The idea is simple, and we sketch it here. Suppose the MAGs  $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_M$  satisfy the  $\alpha$ -clustering assumption with true clusters  $C_1^*, C_2^*, \dots, C_k^*$ . Now consider the setting where we have errors in the PAG skeleton estimation. Let  $\mathcal{U}_1, \mathcal{U}_2, \dots, \mathcal{U}_M$  be the true skeletons of the MAGs  $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_M$ . Consider for each MAG  $\mathcal{M}_i$ , a corrupted counterpart  $\mathcal{M}_i^{\text{corr}}$ , with the guarantee that  $d(\mathcal{M}_i, \mathcal{M}_i^{\text{corr}}) \leq \beta/2n$ . These corrupted MAGs are only constructed for the sake of proof, and are not actually present. Assume that the skeleton estimation is not precise and instead of  $\mathcal{U}_1, \mathcal{U}_2, \dots, \mathcal{U}_M$ , it produces the skeletons  $\mathcal{U}_1^{\text{corr}}, \mathcal{U}_2^{\text{corr}}, \dots, \mathcal{U}_M^{\text{corr}}$ , associated with these corrupted MAGs  $\mathcal{M}_1^{\text{corr}}, \mathcal{M}_2^{\text{corr}}, \dots, \mathcal{M}_M^{\text{corr}}$ . By triangle inequality, it is easy to observe that the MAGs satisfy  $(\alpha - \beta, \beta)$ -clustering assumption. If  $\beta < \alpha/2$ , then, using Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE on  $\mathcal{U}_1^{\text{corr}}, \mathcal{U}_2^{\text{corr}}, \dots, \mathcal{U}_M^{\text{corr}}$  with parameter  $\alpha$  replaced by  $\alpha - \beta$  will guarantee that we recover the true clusters  $C_1^*, C_2^*, \dots, C_k^*$ . This follows because any pair of entities  $i, j$  that were originally in the same true cluster will still remain together in the same cluster, even under corruption, as their corrupted MAGs will be at most  $\beta n < \alpha/2n$  distance apart. Similarly, if  $i, j$  belonged to different true clusters then they will still remain in different clusters, even under corruption, as their corrupted MAGs will be  $> \alpha/2n$  distance apart. Also, if the corrupted MAGs satisfy the conditions in Theorem 4.3.3, we can recover the dominant MAG. With the right set of parameters, this argument can also be extended starting from an  $(\alpha, \beta)$ -clustering.

### 4.6.3 Causal Discovery under $(\alpha, \beta)$ -Clustering

In this section, we present an algorithm that recovers the underlying clusters  $C_1^*, C_2^*, \dots, C_k^*$  provided they satisfy  $(\alpha, \beta)$ -clustering property. After recovering the clusters, in section 4.6.3.1, we give an algorithm that recovers an approximate MAG for every entity with only few additional interventions.

Firstly, using the next lemma, we show that the threshold used by Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE correctly identifies whether two entities belong to the same true cluster or not. This implies that our algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE recovers the clusters with high probability.

**Lemma 4.6.8** (Lemma 4.3.2 restated). *If the underlying MAGs  $\mathcal{M}_1, \dots, \mathcal{M}_M$  satisfy  $(\alpha, \beta)$ -clustering property with true clusters  $C_1^*, \dots, C_k^*$  and have maximum undirected degree  $\Delta$ . Then, the Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE recovers the clusters  $C_1^*, \dots, C_k^*$  with probability at least  $1 - \delta$ . Every entity  $i \in [M]$  uses at most  $4(\Delta + 1) \log(M/\delta)/(\alpha - \beta)^2$  many atomic interventions.*

*Proof.* Let  $\text{COUNT}(i, j) = \sum_{u \in S} \mathbf{1}\{N_i(u) = N_j(u)\}$  for distinct entities  $i, j$ . If  $i, j$  belong to the same true cluster  $C_t^*$  for some  $t \in [k]$ , we have :

$$\mathbf{E}[\text{COUNT}(i, j)] = \mathbf{E} \left[ \sum_{u \in S} \mathbf{1}\{N_i(u) = N_j(u)\} \right] \geq (1 - \beta)|S|$$

Using Hoeffding's inequality, with probability at least  $1 - \exp(-\frac{\Lambda^2}{2|S|})$

$$\text{COUNT}(i, j) \geq \mathbf{E}[\text{COUNT}(i, j)] - \frac{\Lambda}{2}$$

If  $i, j$  belong to different true clusters, then, we have :

$$\mathbf{E}[\text{COUNT}(i, j)] = \mathbf{E} \left[ \sum_{u \in S} \mathbf{1}\{N_i(u) = N_j(u)\} \right] \leq (1 - \alpha)|S|$$

Using Hoeffding's inequality, with probability at least  $1 - \exp(-\frac{\Lambda^2}{2|S|})$

$$\text{COUNT}(i, j) < \mathbf{E}[\text{COUNT}(i, j)] + \frac{\Lambda}{2}$$

Set  $\Lambda = |S|(\alpha - \beta)$  and  $|S| = \frac{4 \log M/\delta}{(\alpha - \beta)^2}$ .

Using union bound for every pair of entities in  $[M]$ , we have with probability at least  $1 - \delta$ :

if entities  $i, j \in C_t^*$  (belong to the same true cluster) :

$$\text{COUNT}(i, j) \geq \left(1 - \frac{\alpha + \beta}{2}\right) |S|, \text{ and}$$

if entities  $i, j \notin C_b^* \forall b \in [k]$  (do not belong to the same true cluster) :

$$\text{COUNT}(i, j) < \left(1 - \frac{\alpha + \beta}{2}\right) |S|.$$

Therefore, every pair of entities from same true cluster satisfy the condition that COUNT value is larger than  $(1 - \frac{\alpha + \beta}{2})|S|$  and will include an edge in  $\mathcal{P}$ , while we do not include an edge between pair of entities from different clusters. The resulting graph  $\mathcal{P}$ , will have  $k$  connected components and Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE will return the true clusters correctly.

As we intervene on all the neighbors of every node in  $S$ , it will increase the interventions for every entity by a multiplicative  $\Delta + 1$  factor. For an entity  $i$ , the total number of interventional distributions constructed is

$$\sum_{u \in S} (1 + |\Gamma_i(u)|) \leq |S|(\Delta + 1) = 4(\Delta + 1) \log(M/\delta)/(\alpha - \beta)^2 \text{ as } \max_{i \in [M], w \in V} |\Gamma_i(w)| \leq \Delta.$$

This completes the proof. □

### 4.6.3.1 Learning Causal Graphs from Clusters

In this section, we present the full-description of the Algorithm  $(\alpha, \beta)$ -RECOVERY that returns an approximate causal graph for every entity  $i \in [M]$ . We also include the brief overview from section 4.3.2 for clarity.

**Overview of Algorithm  $(\alpha, \beta)$ -RECOVERY** Consider a cluster  $C_a^*$ . We recover the dominant MAG of this cluster,  $\mathcal{M}_a^{\text{dom}}$ , by recovering all the neighbors of every node and carefully merging them. Our idea is to assign a node, selected uniformly at random, to every entity in  $C_a^*$ , and recover the neighborhood of the node using Algorithms IDENTIFY-OUTNBR and IDENTIFY-BIDIRECTED. If the clusters are large such that  $|C_a^*| \gg n$  (see Theorem 4.3.3 for a precise bound), we can show a large number of entities  $T_u$  are assigned node  $u$ , and many of them will share the dominant MAG. We maintain a count  $\text{NCOUNT}(i, u)$  of the number of times the entity  $i$  agrees with other entities in  $T_u$  about neighbors of  $u$ , and guarantee (with high probability) that the entity with the highest count will be that of dominant MAG. After merging the neighbors recovered for every node, we assign the resulting graph to every entity in the cluster.

Consider the cluster  $C_a^*$  for some  $a \in [k]$ . In the next claim, we show that if the size of  $C_a^*$  is sufficiently large, then, each node  $u \in V$  is assigned a large number of entities by  $(\alpha, \beta)$ -RECOVERY using the set  $T_u$ .

**Claim 4.6.9.** *Consider a cluster  $C_a^*$  such that  $\gamma_a > 1/2$  and  $|C_a^*| \geq \frac{8n \log(nM/\delta)}{(2\gamma_a - 1)^2}$ . Let  $T_u$  denote the set of entities assigned to node  $u$  in Algorithm 14. Then, we have with probability  $1 - \delta$ ,  $|T_u| \geq \frac{4 \log(nM/\delta)}{(2\gamma_a - 1)^2}$  for every node  $u \in V$ .*

*Proof.* For a node  $u \in V$ , and cluster  $C_a^*$ , we have:

$$\mathbf{E}[T_u] = \frac{|C_a^*|}{n} \geq \frac{8 \log(nM/\delta)}{(2\gamma_a - 1)^2}.$$

---

**Algorithm 14**  $(\alpha, \beta)$ -RECOVERY
 

---

1: **Input:**  $\alpha > 0, \beta \geq 0 (< \alpha)$ , confidence parameter  $\delta > 0$ , PAGs  $\mathcal{U}_1, \dots, \mathcal{U}_M$  of  $M$  entities  
 2: **Output:**  $\widehat{\mathcal{M}}_1, \widehat{\mathcal{M}}_2, \dots, \widehat{\mathcal{M}}_M$  representing set of  $M$  MAGs.  
 3: Obtain clusters  $C_1^*, C_2^*, \dots, C_k^*$  using Algorithm 13.  
 4: **for** every cluster  $C_a^*$  where  $a \in [k]$  **do**  
 5:   Let  $\widehat{\mathcal{M}}_a^{\text{dom}}$  be an empty graph on the set of nodes  $V$ .  
 6:   For every entity  $i \in C_a^*$ , select a node  $u \in V$  uniformly at random and assign it to  $u$  represented by the set  $T_u$ .  
 7:   **for** every node  $u \in V$  **do**  
 8:     **for** every entity  $i \in T_u$  **do**  
 9:        $\text{ch}_i(u) \leftarrow \text{IDENTIFY-OUTNBR}(\mathcal{U}_i, u)$   
 10:        $\text{sp}_i(u) \leftarrow \text{IDENTIFY-BIDIRECTED}(\mathcal{U}_i, u)$ .  
 11:        $\text{pa}_i(u) \leftarrow \Gamma_i(u) \setminus (\text{ch}_i(u) \cup \text{sp}_i(u))$ .  
 12:       Construct  $N_i(u)$  (defined in (4.2)) and calculate  $\text{NCOUNT}(i, u) = \sum_{j \in T_u: j \neq i} \mathbf{1}\{N_i(u) = N_j(u)\}$   
 13:       **end for**  
 14:       Let  $u_{\max} \leftarrow \arg \max_{i \in T_u} \text{NCOUNT}(i, u)$ .  
 15:       Set neighbors of  $u$  in  $\widehat{\mathcal{M}}_a^{\text{dom}}$  to the set  $N_{u_{\max}}(u)$ .  
 16:     **end for**  
 17:   For every entity  $i \in C_a^*$ , set  $\widehat{\mathcal{M}}_i = \widehat{\mathcal{M}}_a^{\text{dom}}$ .  
 18: **end for**  
 19: Return  $\widehat{\mathcal{M}}_1, \widehat{\mathcal{M}}_2, \dots, \widehat{\mathcal{M}}_M$

---

Using Chernoff bound, with probability at least  $1 - \exp(-\log(nM/\delta)/(2\gamma_a - 1)^2) \geq 1 - \delta/nM$ , we have:

$$T_u \geq \frac{\mathbf{E}[T_u]}{2} \geq \frac{4 \log(nM/\delta)}{(2\gamma_a - 1)^2}.$$

Applying union bound for every node  $u \in V$  and  $a \in [k]$ , gives us the claim.  $\square$

Consider a partitioning of  $C_a^*$  given by  $S_a^1, S_a^2, \dots, S_a^t$  where each  $S_a^i$  for any  $i \in [t]$  represents the maximal collection of MAGs that are equal. Formally, we have:

$$S_a^i = \{\mathcal{M}_p \mid \mathcal{M}_p \in C_a^* \text{ and } \mathcal{M}_p = \mathcal{M}_q \quad \forall \mathcal{M}_q \in S_a^i\}.$$

Let  $|S_a^{\text{dom}}| \geq |S_a^i|$  for every partition  $i \in [t]$  and  $\text{dom}_a$  denote an entity in  $S_a^{\text{dom}}$ . We define:

$$G_a(u) = \{j \mid j \in C_a^* \text{ and } N_j(u) = N_{\text{dom}_a}(u)\} \text{ and } B_a(u) = C_a^* \setminus G_a(u).$$

We can observe that:

$$|G_a(u)| \geq |S_a^{\text{dom}}| \text{ and } |B_a(u)| \leq |C_a^*| - |S_a^{\text{dom}}|.$$

Conditioned on the previous claim that each set  $T_u$  for all  $u \in V$  is large, we argue that for any pair of entities  $i, j \in C_a^*$  where  $\mathcal{M}_i = \mathcal{M}_a^{\text{dom}}$ , and  $\mathcal{M}_j \neq \mathcal{M}_a^{\text{dom}}$ , the NCOUNT value calculated by  $(\alpha, \beta)$ -RECOVERY of entity  $i$  for the node  $u$  is always larger than that of entity  $j$ . Intuitively, after assigning the entities to nodes, we observe that for every node  $u \in V$ , the set  $T_u$  contains a large number of entities with the dominant MAG, i.e.,  $|T_u \cap G_a(u)|$  is large. Because dominant MAGs share the same neighborhood (as they represent the same graph), we can show that the NCOUNT value of dominant MAG is larger than any other MAG in the cluster. We formalize this statement using the following lemma.

**Lemma 4.6.10.** *For every  $a \in [k], u \in V$  and any pair of entities  $i, j \in C_a^*$  that satisfy  $i \in G_a(u)$  and  $j \in B_a(u)$ , we have with probability  $1 - \delta$ ,*

$$\text{NCOUNT}(i, u) > \text{NCOUNT}(j, u).$$

*Proof.* From Algorithm 14, we know that  $\text{NCOUNT}(i, u) = \sum_{j \neq i, j \in T_u} \mathbf{1}\{N_i(u) = N_j(u)\}$  for an entity  $i \in T_u$  and a node  $u \in V$ . Consider the case  $i \in G_a(u)$ . Then, we have:

$$\begin{aligned} \mathbf{E}[\text{NCOUNT}(i, u)] &= \mathbf{E} \left[ \sum_{j \neq i, j \in T_u} \mathbf{1}\{N_i(u) = N_j(u)\} \right] \\ &= \mathbf{E} \left[ \sum_{j \neq i, j \in T_u} \mathbf{1}\{N_{\text{dom}_a}(u) = N_j(u)\} \right] \\ &= \mathbf{E}[|T_u \cap G_a(u)|] \\ &\geq \frac{|S_a^{\text{dom}}|}{n} = \frac{|S_a^{\text{dom}}|}{|C_a^*|} \cdot \frac{|C_a^*|}{n} = \gamma_a \cdot \frac{|C_a^*|}{n} \end{aligned}$$

Using Hoeffding's inequality, with probability at least  $1 - \exp(-\frac{\Lambda^2}{2|T_u|})$

$$\text{NCOUNT}(i, u) \geq \mathbf{E}[\text{NCOUNT}(i, u)] - \frac{\Lambda}{2} \geq \gamma_a \cdot \frac{|C_a^*|}{n} - \frac{\Lambda}{2}$$

If  $i \in B_a(u)$ , then, we have :

$$\begin{aligned} & \mathbf{E}[\text{NCOUNT}(i, u)] \\ &= \mathbf{E} \left[ \sum_{j \neq i, j \in T_u} \mathbf{1}\{N_i(u) = N_j(u)\} \right] \\ &= \mathbf{E} \left[ \sum_{j \neq i, j \in T_u \cap G_a(u)} \mathbf{1}\{N_i(u) = N_{\text{dom}_a}(u)\} + \sum_{j \neq i, j \in T_u \cap B_a(u)} \mathbf{1}\{N_i(u) = N_j(u)\} \right] \\ &= \mathbf{E} \left[ \sum_{j \neq i, j \in T_u \cap B_a(u)} \mathbf{1}\{N_i(u) = N_j(u)\} \right] \\ &= \mathbf{E}[|T_u \cap B_a(u)|] \\ &\leq \frac{|C_a^*| - |S_a^{\text{dom}}|}{n} = \left(1 - \frac{|S_a^{\text{dom}}|}{|C_a^*|}\right) \cdot \frac{|C_a^*|}{n} = (1 - \gamma_a) \cdot \frac{|C_a^*|}{n} \end{aligned}$$

Using Hoeffding's inequality, with probability at least  $1 - \exp(-\frac{\Lambda^2}{2|T_u|})$

$$\text{NCOUNT}(i, u) < \mathbf{E}[\text{COUNT}(i, u)] + \frac{\Lambda}{2} < (1 - \gamma_a) \cdot \frac{|C_a^*|}{n} + \frac{\Lambda}{2}$$

Set  $\Lambda = \frac{|C_a^*|}{n}(2\gamma_a - 1)$  and  $|T_u| \geq \frac{4 \log(nM/\delta)}{(2\gamma_a - 1)^2}$ . Then, for any pair of entities  $i, j \in C_a^*$  such that  $i \in G_a(u)$  and  $j \in B_a(u)$ , we have, with a probability  $1 - \delta/nM^2$ :

$$\text{NCOUNT}(i, u) > \text{NCOUNT}(j, u).$$

Using union bound for every pair of entities in  $[M]$  and  $u \in V$ , with probability at least  $1 - \delta$ , we have the final claim.  $\square$

From the previous Lemma 4.6.10, we know that NCOUNT values are always larger for the dominant MAG partition, and therefore merging the neighborhoods of all the nodes gives us the dominant MAG. As dominant MAG is within a distance of at most  $\beta \cdot n$  from every MAG in the cluster, the dominant MAG returned is a sufficiently good approximation of the true MAG. We formalize this using the following statement.

**Theorem 4.6.11** (Theorem 4.3.3 restated). *Suppose  $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_M$  satisfy  $(\alpha, \beta)$  clustering property. If  $\gamma_a > 1/2$  and  $C_a^* = \Omega(\frac{n \log(n/M\delta)}{(2\gamma_a - 1)^2})$  for all  $a \in [k]$ , then, Algorithm  $(\alpha, \beta)$ -RECOVERY recovers graphs  $\widehat{\mathcal{M}}_1, \dots, \widehat{\mathcal{M}}_M$  such that for every entity  $i \in [M]$ , we have  $d(\mathcal{M}_i, \widehat{\mathcal{M}}_i) \leq \beta n$  with probability  $1 - \delta$ . Moreover, every entity uses at most  $(\Delta + 1) + \frac{4(\Delta+1)\log(M/\delta)}{(\alpha-\beta)^2}$  many atomic interventions.*

*Proof.* From Lemma 4.6.10, we have that  $\text{NCOUNT}(i, u) > \text{NCOUNT}(j, u)$ , which implies  $u_{\max} \in G_a(u)$ . Using Algorithm 14, every entity  $i$  in the cluster  $C_a^*$  is assigned the graph  $\widehat{\mathcal{M}}_i = \mathcal{M}_a^{\text{dom}}$ . From the definition of  $(\alpha, \beta)$ -clustering property, we have that all entities  $i \in C_a^*$  are such that  $d(\mathcal{M}_i, \widehat{\mathcal{M}}_i) = d(\mathcal{M}_i, \mathcal{M}_a^{\text{dom}}) \leq \beta n$ .

Using Algorithm 14 we assign every entity to a single node  $u \in V$ , and perform at most  $\Delta + 1$  interventions to identify all the neighbors of  $u$  for every entity in  $T_u$ . Therefore, we perform at most  $\Delta + 1$  interventions per entity. For obtaining clusters, from Lemma 4.3.2, we know that every entity performs at most  $\frac{4(\Delta+1)\log(M/\delta)}{(\alpha-\beta)^2}$  interventions. Hence, the theorem.  $\square$

#### 4.6.4 Discovery under $\alpha$ -Clustering Property

In this section, we present missing details from section 4.4. First, in section 4.6.4.1, we present a meta algorithm that takes as input clusters of entities and returns the causal MAGs for all the entities. Next, in sections 4.6.4.2-4.6.4.4, we present various algorithms for learning causal MAGs, under  $\alpha$ -clustering property.

#### 4.6.4.1 From $\alpha$ -Clustering to Learning Causal Graphs

Suppose that the underlying MAGs  $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_M$  satisfy the  $\alpha$ -clustering property, our algorithms are based on first accurately recovering these clusters. The idea of going from clusters to MAGs is simple and is based on distributing the interventions across the entities in the cluster. We now discuss a meta-algorithm that returns the associated causal MAG of every entity given the true clustering. Our meta-algorithm takes as input the true clusters  $C_1^*, C_2^*, \dots, C_k^*$  and recovers the MAGs associated with each of them. In any cluster  $C_b^*$  such that  $|C_b^*| < n$ , our meta-algorithm uses an additional  $\lceil n/|C_b^*| \rceil$  many interventions for each entity in  $C_b^*$ . For clusters satisfying  $|C_b^*| \geq n$ , it uses an extra intervention per entity.

**Meta-Algorithm** Consider a true cluster  $C_b^*$  ( $b \in [k]$ ). Construct a mapping  $\phi$  that partitions the  $n$  nodes in  $V$  among all the entities in  $C_b^*$ , such that no entity is assigned to more than  $\lceil n/|C_b^*| \rceil$  many nodes. By definition, all entities in  $C_b^*$  have the same PAG. Let  $\mathcal{U}$  be the common PAG. Construct a MAG  $\mathcal{M}$  from  $\mathcal{U}$  as follows. Consider an edge  $(u, v)$  in  $\mathcal{U}$ . Let  $u = \phi(i)$  and  $v = \phi(j)$  where the entities  $i, j \in C_b^*$  are such that we intervene on node  $u$  in entity  $i$  and node  $v$  in entity  $j$  ( $i$  could be equal to  $j$ ). Now, if  $v \in \text{ch}_i(u)$ , we add  $u \rightarrow v$  into the graph  $\mathcal{M}$ , else if  $u \in \text{ch}_j(v)$ , we add  $u \leftarrow v$ , and  $u \leftrightarrow v$  otherwise. We assign graph  $\mathcal{M}$  for every entity in  $C_b^*$ . Repeating this procedure for every  $C_b^*$  generates the  $M$  MAGs, one for each entity.

**Lemma 4.6.12.** *Suppose there is an Algorithm  $\mathcal{A}$  that recovers the true clusters  $C_1^*, C_2^*, \dots, C_k^*$  of the underlying MAGs  $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_M$  satisfying  $\alpha$ -clustering property such that every entity  $i \in [M]$  uses at most  $f(M)$  interventions. Then, there is an algorithm that can learn all the MAGs  $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_M$  such that every entity  $i \in [M]$  uses at most  $f(M) + \lceil n/\Upsilon \rceil$  many interventions, where  $\Upsilon = \min_{b \in [k]} |C_b^*|$ .*

*Proof.* Consider a cluster  $C_b^*$  for  $b \in [k]$ . As the mapping  $\phi$  assigns every entity at most  $\lceil n/|C_b^*| \rceil$  many nodes to intervene on, we have that every entity in  $C_b^*$  uses at

most  $\lceil n/|C_b^*| \rceil$  additional interventions. Therefore, over all true clusters, every entity uses at most  $f(M) + \lceil n/\Upsilon \rceil$  many interventions.

Consider any cluster  $C_b^*$ . The mapping  $\phi$  in the Meta-Algorithm is well-defined and satisfies the claim that for every node  $u \in V$ , there exists an entity in  $C_b^*$  for which we construct an interventional distribution  $\text{do}(u)$ . Therefore, for cluster  $C_b^*$ , we have  $n$  interventional distributions one for every node in  $V$ , and we use Algorithm RECOVERG to learn the MAG for this cluster (i.e., MAG for all the entities in  $C_b^*$ ). Repeating this for every cluster  $C_1^*, C_2^*, \dots, C_k^*$ , we obtain all the MAGs  $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_M$ .  $\square$

If the clusters are of size at least  $n$ , i.e.,  $\min_{b \in [k]} |C_b^*| \geq n$ , then, we have the following corollary from Lemma 4.6.12.

**Corollary 4.6.13.** *Suppose there is an Algorithm  $\mathcal{A}$  that recovers the true clusters  $C_1^*, C_2^*, \dots, C_k^*$  of the underlying MAGs  $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_M$  satisfying the  $\alpha$ -clustering property such that every entity  $i \in [M]$  uses at most  $f(M)$  interventions. Suppose  $\min_{b \in [k]} |C_b^*| \geq n$ . Then, there is an algorithm that can learn all the MAGs  $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_M$  such that every entity  $i \in [M]$  uses at most  $f(M) + 1$  many interventions.*

#### 4.6.4.2 Causal Discovery without Latents

In this section, we present a randomized algorithm that recovers (with high probability) all the  $M$  MAGs  $\mathcal{M}_1, \dots, \mathcal{M}_M$  when the underlying data generating process for each of these entities do not have any latents (i.e., causal DAGs  $\mathcal{D}_1, \dots, \mathcal{D}_M$  satisfy causal sufficiency). This translates into the fact that the MAGs  $\mathcal{M}_1, \dots, \mathcal{M}_M$  do not have bidirected edges.

We first make the observation that to identify that two graphs, say  $\mathcal{M}_i$  and  $\mathcal{M}_j$  belong to different clusters, it suffices to find a node  $u$  from the node-difference set  $\text{diff}(\mathcal{M}_i, \mathcal{M}_j)$  and checking their outgoing neighbors using Algorithm IDENTIFY-

OUTNBR. We argue that, with probability at least  $1 - \delta$ , we can identify one such node  $u \in \text{diff}(\mathcal{M}_i, \mathcal{M}_j)$  by sampling  $2 \log(M/\delta)/\alpha$  nodes uniformly from  $V$  as  $|\text{diff}(\mathcal{M}_i, \mathcal{M}_j)| = d(\mathcal{M}_i, \mathcal{M}_j) \geq \alpha n$ .

In Algorithm NOLATENTS, we obtain a sample of nodes  $S$  and construct interventional distribution for every entity in  $[M]$ , and for every node in  $S$ . After finding the outgoing neighbors for every entity  $i$  and node in  $S$ , we construct a graph  $\mathcal{P}$  on entities (i.e., the node set of  $\mathcal{P}$  is  $[M]$ ). We include an edge between two entities if they share the same outgoing neighbors for every  $u \in S$ . This ensures that every entity is connected only to the entities belonging to the same true cluster, and we return the connected components in  $\mathcal{P}$  as our clusters.

---

**Algorithm 15** NOLATENTS

---

- 1: **Input:**  $\alpha > 0$ , confidence parameter  $\delta > 0$ , PAGs  $\mathcal{U}_1, \dots, \mathcal{U}_M$  of  $M$  entities.
  - 2: **Output:** Partition of  $[M]$  into clusters
  - 3: Let  $S$  denote a uniform sample of  $\frac{2 \log M/\delta}{\alpha}$  nodes from  $V$  selected with replacement.
  - 4: **for** every entity  $i \in [M]$  and  $u \in S$  **do**
  - 5:      $\text{ch}_i(u) \leftarrow \text{IDENTIFY-OUTNBR}(i, u)$
  - 6: **end for**
  - 7: Let  $\mathcal{P}$  denote an empty graph on set of entities  $[M]$
  - 8: **for** every pair of entities  $i, j$  **do**
  - 9:     **if**  $\text{ch}_i(u) = \text{ch}_j(u)$  and  $\Gamma_i(u) = \Gamma_j(u)$  for every  $u \in S$  **then**
  - 10:         Add an edge between entities  $i$  and  $j$  in  $\mathcal{P}$
  - 11:     **end if**
  - 12: **end for**
  - 13: Return connected components in  $\mathcal{P}$
- 

**Claim 4.6.14.** *Let  $S$  denote a set of  $2 \log(M/\delta)/\alpha$  nodes sampled with replacement uniformly from  $V$ . Then, for every pair of entities  $i, j$  that belong to different true clusters, we have with probability at least  $1 - \delta$ ,  $u \in \text{diff}(\mathcal{M}_i, \mathcal{M}_j)$  for some  $u \in S$ .*

*Proof.* Let  $S$  denote a set of sampled nodes such that  $|S| = 2 \log(M/\delta)/\alpha$ . Therefore, we have

$$\begin{aligned} \Pr_{u \sim V}[u \in \text{diff}(\mathcal{M}_i, \mathcal{M}_j)] &\geq \alpha, \text{ and} \\ \Pr_{S \sim V}[\forall u \in S : u \notin \text{diff}(\mathcal{M}_i, \mathcal{M}_j)] &\leq (1 - \alpha)^{|S|} \\ &\leq e^{-\alpha|S|} \leq \frac{\delta}{M^2}. \end{aligned}$$

Using union bound for every pair of entities in  $[M]$  that belong to two different clusters, we have:

$$\forall i, j \in [M], \Pr_{S \sim V}[\forall u \in S, u \notin \text{diff}(\mathcal{M}_i, \mathcal{M}_j)] \leq \delta.$$

Therefore, for every pair of entities  $i, j \in [M]$  belonging to different true clusters, there exists  $u \in S$  such that :

$$\Pr[u \in \text{diff}(\mathcal{M}_i, \mathcal{M}_j)] \geq 1 - \delta.$$

□

**Lemma 4.6.15.** *Assume causal sufficiency. If MAGs  $\mathcal{M}_1, \dots, \mathcal{M}_M$  satisfy  $\alpha$ -clustering property with true clusters  $C_1^*, \dots, C_k^*$ , then Algorithm NOLATENTS exactly recovers the clusters  $C_1^*, \dots, C_k^*$  with probability at least  $1 - \delta$ . Every entity  $i \in [M]$  uses  $2 \log(M/\delta)/\alpha$  many atomic interventions.*

*Proof.* Consider two entities  $i, j$  and their corresponding MAGs  $\mathcal{M}_i$  and  $\mathcal{M}_j$  respectively. We first observe that if the PAGs of these two entities are different then they belong to different clusters. Now consider the case where the PAGs for both these entities are the same, i.e.,  $\mathcal{U}_i = \mathcal{U}_j$ . Now if  $i$  and  $j$  belong to different true clusters, then we claim that it suffices to find a node  $u$  from the node-difference set  $\text{diff}(\mathcal{M}_i, \mathcal{M}_j) = \{u \mid N_i(u) \neq N_j(u)\}$  to notice this fact. As there are no latents (causal sufficiency), we can identify whether  $u \in \text{diff}(\mathcal{M}_i, \mathcal{M}_j)$ , by checking only the

outgoing neighbors of  $u$  for entities  $i, j$ , i.e.,  $\text{diff}(\mathcal{M}_i, \mathcal{M}_j) = \{u \mid \text{ch}_i(u) \neq \text{ch}_j(u)\}$ . When we identify such a node  $u$ , the set of outgoing neighbors of node  $u$  are different for entities  $i, j$ , and therefore must belong to different true clusters (by  $\alpha$ -clustering property). In order to identify at least one node  $u \in \text{diff}(\mathcal{M}_i, \mathcal{M}_j)$ , we use sampling.

Let  $S$  denote the set of sampled nodes (with replacement) from  $V$  such that  $|S| = 2 \log(M/\delta)/\alpha$ . In Algorithm NOLATENTS, we construct interventional distributions for every node  $u \in S$ , for every entity  $i \in [M]$ . Using these interventional distributions we obtain the outgoing neighbors of nodes in  $S$  using Algorithm IDENTIFY-OUTNBR.

From Claim 4.6.14, we have that for every pair of entities  $i, j$  belonging to different true clusters, there exists  $u \in S$  such that:

$$\Pr[\text{ch}_i(u) \neq \text{ch}_j(u)] = \Pr[u \in \text{diff}(\mathcal{M}_i, \mathcal{M}_j)] \geq 1 - \delta.$$

This implies that, with probability at least  $1 - \delta$ , for every  $i, j$  pair we have the following: in the entity graph  $\mathcal{P}$  there would not be an edge between  $i, j$  if they belong to different true clusters, and there would be an edge if they belong to the same true cluster. The resulting graph  $\mathcal{P}$ , will have  $k$  connected components and Algorithm NOLATENTS will return the true clusters correctly.

Hence, with probability at least  $1 - \delta$ , we can recover all the true clusters  $C_1^*, \dots, C_k^*$  using Algorithm NOLATENTS.  $\square$

**Theorem 4.6.16.** *Assume causal sufficiency. If MAGs  $\mathcal{M}_1, \dots, \mathcal{M}_M$  satisfy  $\alpha$ -clustering property with true clusters  $C_1^*, \dots, C_k^*$  then Algorithm NOLATENTS exactly recovers these clusters with probability at least  $1 - \delta$ . Furthermore, if  $\min_{b \in [k]} |C_b^*| \geq n$ , then there is an algorithm that exactly learns all these MAGs with probability at least  $1 - \delta$ . Every entity  $i \in [M]$  uses  $2 \log(M/\delta)/\alpha + 1$  many atomic interventions.*

*Proof.* From Lemma 4.6.15, we can recover the clusters correctly with probability at least  $1 - \delta$ . Using the Meta-Algorithm discussed in section 4.6.4.1, we can learn the

graphs of every entity with a single additional intervention (see Corollary 4.6.13). This establishes the result.  $\square$

#### 4.6.4.3 Causal Discovery with Latents: Bounded Degree MAGs

Throughout this section, we let :

$$\Delta = \max_{i \in [M], u \in V} |\Gamma_i(u)|.$$

We now discuss an algorithm that recovers clusters  $C_1^*, C_2^* \dots, C_k^*$  using ideas developed in section 4.6.4.2 but now with latents in the system. In the presence of latents, the collection of MAGs  $\mathcal{M}_1, \dots, \mathcal{M}_M$  are mixed graphs that also contain bidirected edges, which introduces issues, as bidirected edges cannot be detected easily. For example, two entities  $i$  and  $j$  might be such that  $u \leftrightarrow v$  could be present in  $\mathcal{M}_i$  and  $u \leftarrow v$  could be present in  $\mathcal{M}_j$ , in which case intervening on just  $u$  alone will not suffice to distinguish  $i$  from  $j$ , we need interventions on both  $u$  and  $v$ . This is the idea behind Algorithm  $\alpha$ -BOUNDEDDEGREE, which identifies all the outgoing and bidirected edges incident on the set of sampled nodes (say  $S$ ), for every entity in  $[M]$ . Since from this we can compute all neighboring relations of  $u$  ( $N_i(u)$ ), Algorithm  $\alpha$ -BOUNDEDDEGREE then checks whether these neighborhoods are the same or not for every node  $u \in S$ . We can now leverage the  $\alpha$ -clustering property to argue that this process succeeds with probability at least  $1 - \delta$ .

As we use Algorithm IDENTIFY-BIDIRECTED, to find all bidirected edges incident on a node  $u \in S$ , we use an additional  $O(\Delta)$  atomic interventions (per entity) where  $\Delta = \max_{i \in [M], u \in V} \Gamma_i(u)$  is the maximum undirected degree in the PAGs  $\mathcal{U}_1, \dots, \mathcal{U}_M$ .

**Lemma 4.6.17.** *If the underlying MAGs  $\mathcal{M}_1, \dots, \mathcal{M}_M$  satisfy  $\alpha$ -clustering property with true clusters  $C_1^*, \dots, C_k^*$ , then Algorithm  $\alpha$ -BOUNDEDDEGREE exactly recovers the clusters  $C_1^*, \dots, C_k^*$  with probability at least  $1 - \delta$ . Every entity  $i \in [M]$  uses at most  $2(\Delta + 1) \log(M/\delta)/\alpha$  many atomic interventions.*

---

**Algorithm 16**  $\alpha$ -BOUNDEDDEGREE

---

- 1: **Input:**  $\alpha > 0$ , confidence parameter  $\delta > 0$ , PAGs  $\mathcal{U}_1, \dots, \mathcal{U}_M$  of  $M$  entities
  - 2: **Output:** Partition of  $[M]$  into clusters
  - 3: Let  $S$  denote a uniform sample of  $\frac{2 \log M / \delta}{\alpha}$  nodes from  $V$  selected with replacement.
  - 4: **for** every entity  $i \in [M]$  and  $u \in S$  **do**
  - 5:      $\text{ch}_i(u) \leftarrow \text{IDENTIFY-OUTNBR}(i, u)$
  - 6:      $\text{sp}_i(u) \leftarrow \text{IDENTIFY-BIDIRECTED}(i, u)$
  - 7:      $\text{pa}_i(u) \leftarrow \Gamma_i(u) \setminus (\text{ch}_i(u) \cup \text{sp}_i(u))$
  - 8:     Construct  $N_i(u)$  (defined in (4.2))
  - 9: **end for**
  - 10: Let  $\mathcal{P}$  denote an empty graph on set of entities  $[M]$
  - 11: **for** every pair of entities  $i, j$  **do**
  - 12:     **if**  $N_i(u) = N_j(u)$  for every  $u \in S$  **then**
  - 13:         Include an edge between  $i$  and  $j$  in  $\mathcal{P}$
  - 14:     **end if**
  - 15: **end for**
  - 16: Return connected components in  $\mathcal{P}$
- 

*Proof.* We follow a proof idea similar to Lemma 4.6.15. Again if two entities  $i, j$  have different PAGs then they belong to different true clusters.

Consider two entities  $i, j$  belonging to different true clusters but having the same PAG. Again it suffices to find a node  $u$  from the node-difference set  $\text{diff}(\mathcal{M}_i, \mathcal{M}_j) = \{u \mid N_i(u) \neq N_j(u)\}$  to conclude that they belong to different clusters.

As there are latents (causal sufficiency), we cannot identify whether  $u \in \text{diff}(\mathcal{M}_i, \mathcal{M}_j)$ , by checking only the outgoing neighbors of  $u$  for entities  $i, j$ , and have to check the set of bidirected edges incident on  $u$  as well. We can identify all the bidirected edges incident on  $u$  for both  $i, j$  using Algorithm IDENTIFY-BIDIRECTED. Identifying such a node  $u$ , whose set of neighbors of node  $u$  are different for entities  $i, j$ , provides a certificate that  $i, j$  belong to different true clusters ( $\alpha$ -clustering property). In order to identify at least one node  $u \in \text{diff}(\mathcal{M}_i, \mathcal{M}_j)$ , we use sampling.

Let  $S$  denote the set of sampled nodes (with replacement) from  $V$  such that  $|S| = 2 \log(M/\delta)/\alpha$ . In Algorithm  $\alpha$ -BOUNDEDDEGREE, we construct interventional distributions for every node  $u \in S$  and all the neighbors in the PAG given by  $\Gamma_i(u)$ , for every entity  $i \in [M]$ . From these interventional distributions, we can compute

$N_i(u)$  and  $N_j(u)$  for all the nodes  $u \in S$  (using Algorithms IDENTIFY-OUTNBR and IDENTIFY-BIDIRECTED).

From Claim 4.6.14, we have that for every pair of entities  $i, j$  belonging to different true clusters, there exists  $u \in S$  such that:

$$\Pr[N_i(u) \neq N_j(u)] = \Pr[u \in \text{diff}(\mathcal{M}_i, \mathcal{M}_j)] \geq 1 - \delta.$$

Hence, with probability at least  $1 - \delta$ , we can recover all the true clusters using Algorithm  $\alpha$ -BOUNDEDDEGREE.

For an entity  $i$ , the total number of interventional distributions constructed is

$$\sum_{u \in S} (1 + |\Gamma_i(u)|) \leq |S|(\Delta + 1) = 2(\Delta + 1) \log(M/\delta)/\alpha \text{ as } \max_{i \in [M], w \in V} |\Gamma_i(w)| \leq \Delta.$$

□

**Theorem 4.6.18.** *If the underlying MAGs  $\mathcal{M}_1, \dots, \mathcal{M}_M$  satisfy  $\alpha$ -clustering property with true clusters  $C_1^*, \dots, C_k^*$ , then Algorithm  $\alpha$ -BOUNDEDDEGREE exactly recovers these clusters with probability at least  $1 - \delta$ . Furthermore, if  $\min_{b \in [k]} |C_b^*| \geq n$ , then there is an algorithm that exactly learns all these MAGs with probability at least  $1 - \delta$ . Every entity  $i \in [M]$  uses at most  $2(\Delta + 1) \log(M/\delta)/\alpha + 1$  many atomic interventions.*

*Proof.* From Lemma 4.6.17, we can recover the clusters correctly with probability at least  $1 - \delta$ . Using the Meta-Algorithm discussed in section 4.6.4.1, and from Corollary 4.6.13, we can obtain an algorithm to learn the graphs of every entity with an additional intervention per entity. This completes the proof. □

#### 4.6.4.4 Improved Algorithm for Recovering the Clusters

In this section, we present the missing details about Algorithm  $\alpha$ -GENERAL, from section 4.4.

In Algorithm  $\alpha$ -GENERAL, first, we obtain all the outgoing neighbors of the sampled set of nodes  $S$ . Then, we construct a graph on set of entities,  $[M]$  such that an edge between a pair of entities  $i, j$  is included if they share same PAGs, i.e.,  $\mathcal{U}_i = \mathcal{U}_j$  and same outgoing neighbors for every node in  $S$ . However, it is possible that the graph  $\mathcal{P}$  can contain more than one true cluster. In the next lemma, we show that we can detect this, and remove all the edges between entities belonging to two different clusters using 2 interventions.

**Lemma 4.6.19.** *Suppose a component  $T_a$  in  $\mathcal{P}_{\text{itr}}$  for some  $\text{itr} \geq 1$  contains all the entities from two true clusters  $C_b^*, C_c^*$ . If  $\min_{r \in [k]} |C_r^*| \geq \Omega(n \log M / \delta)$ , then, we can identify, with a probability  $1 - \delta / 2k^2$ , all the pairs of entities  $i', j' \in T_a$  such that  $i' \in C_b^*$  and  $j' \in C_c^*$  (or vice-versa) using at most 2 interventions for every entity in  $T_a$ .*

*Proof.* We claim that if a component  $T_a$  containing  $C_b^*$  and  $C_c^*$  exists, then, we can identify a pair of entities  $i, j$  that are joined by an edge in  $\mathcal{P}_{\text{itr}}$  such that  $i \in C_b^*$  and  $j \in C_c^*$  or vice-versa.

It suffices to find a node  $u$  from the node-difference set  $\text{diff}(\mathcal{M}_i, \mathcal{M}_j) = \{u \mid N_i(u) \neq N_j(u)\}$  to conclude that they belong to different clusters. From Claim 4.6.14, we know that when  $|S| = 2 \log(2M/\delta)/\alpha$ , we can identify such a  $u \in \text{diff}(\mathcal{M}_i, \mathcal{M}_j)$  with a probability  $1 - \delta/2$ . We make the observation that a pair of entities  $i, j$  that have an edge in this  $\mathcal{P}_{\text{itr}}$  and from different true clusters, can differ only if there is a node  $u \in \text{diff}(\mathcal{M}_i, \mathcal{M}_j)$  such that  $u$  has a bidirected edge  $u \leftrightarrow v$  in  $\mathcal{M}_i$ , and a directed edge  $u \leftarrow v$  in  $\mathcal{M}_j$  (or vice-versa). Intervening on both  $u$  and  $v$  will separate these entities, our main idea is to ensure that this happens.

Consider a mapping  $\pi : [M] \rightarrow V$  where  $\pi(i)$  is assigned a node from  $V$  selected uniformly at random. Using this mapping, we ensure that there are two entities  $i \in T_a \cap C_b^*, j \in T_a \cap C_c^*$  joined by an edge, such that  $\pi(i) = \pi(j) = v$  and  $u \leftrightarrow v$  in  $\mathcal{M}_i$ ,  $u \leftarrow v$  in  $\mathcal{M}_j$  (or vice-versa) for some  $u \in S$ . We have:

$$\Pr[\text{for any } i \in T_a, \pi(i) \neq v] = 1 - \frac{1}{n}, \text{ and}$$

$$\Pr[\forall i \in C_b^* : \pi(i) \neq v] = \left(1 - \frac{1}{n}\right)^{|C_b^*|}.$$

Similarly, we have

$$\Pr[\forall j \in C_c^* : \pi(j) \neq v] = \left(1 - \frac{1}{n}\right)^{|C_c^*|}.$$

$$\begin{aligned} \Pr[\forall i \in C_b^*, j \in C_c^* \text{ such that } \pi(i) \neq v, \pi(j) \neq v] &= \left(1 - \frac{1}{n}\right)^{|C_b^*|+|C_c^*|} \\ &\leq \frac{\delta}{2M^2} \leq \frac{\delta}{2k^2} \\ \Rightarrow \Pr[\exists i \in C_b^*, \exists j \in C_c^* : \pi(i) = \pi(j) = v] &\geq 1 - \frac{\delta}{2k^2}. \end{aligned}$$

As we intervene on  $\pi(i)$  for every entity  $i \in T_a$ , we know that there exists  $i \in C_b^*, j \in C_c^*$ , both in  $T_a$  and that are assigned  $v$  by  $\pi$ . Therefore, we can separate  $i, j$  and remove the edge from  $\mathcal{P}_{\text{itr}}$ . Now, we create an intervention on  $\pi(i) = \pi(j) = v$  for every entity in  $T_a$  and separate all the entity pairs  $(i', j')$  joined by an edge in  $\mathcal{P}_{\text{itr}}$  that satisfy:  $u \leftarrow v$  in  $\mathcal{M}_{i'}$  and  $u \leftrightarrow v$  in  $\mathcal{M}_{j'}$  (or vice-versa). As we use at most two interventions for every entity in  $T_a$ , the lemma follows.  $\square$

**Lemma 4.6.20.** *If the underlying MAGs satisfy  $\alpha$ -clustering property with true clusters  $C_1^*, \dots, C_k^*$  such that  $\min_{b \in [k]} |C_b^*| = \Omega(n \log(M/\delta))$  entities, the Algorithm  $\alpha$ -GENERAL exactly recovers the clusters  $C_1^*, \dots, C_k^*$  with probability at least  $1 - \delta$ . Every entity  $i \in [M]$  uses at most  $O(\log(M/\delta)/\alpha + k^2)$  many atomic interventions.*

*Proof.* From Claim 4.6.14, with probability at least  $1 - \delta/2$ , we have that the set of sampled nodes  $S$  (where  $|S| = 2 \log(2M/\delta)/\alpha$ ) satisfy that for every pair of entities from different clusters there is a node  $u \in S$  that can be used to identify that they

---

**Algorithm 17**  $\alpha$ -GENERAL

---

**Input:**  $\alpha > 0$ , confidence parameter  $\delta > 0$ , PAGs  $\mathcal{U}_1, \dots, \mathcal{U}_M$  of  $M$  entities  $\mathcal{U}_1, \dots, \mathcal{U}_M$ )

**Output:** Partition of  $[M]$  into clusters

Let  $S$  denote a uniform sample of  $\frac{2 \log(2M/\delta)}{\alpha}$  nodes from  $V$  selected with replacement.

**for** every entity  $i \in [M]$  and  $u \in S$  **do**

$\text{ch}_i(u) \leftarrow \text{IDENTIFY-OUTNBR}(i, u)$

**end for**

Let  $\mathcal{P}$  denote an empty graph on the set of entities  $[M]$ .

**for** every pair of entities  $i, j$  **do**

**if**  $\text{ch}_i(u) = \text{ch}_j(u)$  and  $\Gamma_i(u) = \Gamma_j(u) \quad \forall u \in S$  **then**

        Include an edge between  $i$  and  $j$  in  $\mathcal{P}$

**end if**

**end for**

$\text{itr} \leftarrow 1, \mathcal{P}_0 \leftarrow \mathcal{P}$

**while** TRUE **do**

$\mathcal{P}_{\text{itr}} \leftarrow \mathcal{P}_{\text{itr}-1}$

    Let  $T_1, T_2, \dots$  denote the components in  $\mathcal{P}_{\text{itr}}$ .

    For all  $i \in [M]$ , obtain interventional distribution on  $\pi(i)$  picked u.a.r from  $V$ .

**if**  $\exists$  edge  $(i, j) \in \mathcal{P}_{\text{itr}}$  in component  $T_a$  such that  $\pi(i) = \pi(j)$  **then**

        Let  $v = \pi(i) = \pi(j)$

**if**  $v \in \text{sp}_i(u), v \notin \text{sp}_j(u)$  (or vice-versa) for some  $u \in S$  **then**

            Intervene on  $v$  for every entity in  $T_a$ .

            Remove edge  $(i', j')$  from  $\mathcal{P}_{\text{itr}}$  if  $v \in \text{sp}_{i'}(u), v \notin \text{sp}_{j'}(u)$  (or vice-versa)

    for every  $i', j' \in T_a$

**end if**

**end if**

**if** the set of edges in  $\mathcal{P}_{\text{itr}}$  are *same* as the set of edges in  $\mathcal{P}_{\text{itr}-1}$  **then**

        Return connected components in  $\mathcal{P}_{\text{itr}}$

**end if**

$\text{itr} \leftarrow \text{itr} + 1$

**end while**

---

belong to different clusters. Using Lemma 4.6.19, we have that, in every iteration  $\text{itr}$ , we remove all the edges in  $\mathcal{P}_{\text{itr}}$  between entities that are part of the same component but from different true clusters. After  $k^2$  iterations, we would have separated all the pairs of entities between all the true clusters. In Algorithm  $\alpha$ -GENERAL, we return the connected components in  $\mathcal{P}_{\text{itr}}$  when there is no change in the set of edges between entities between  $\mathcal{P}_{\text{itr}-1}$  and  $\mathcal{P}_{\text{itr}}$ .

From Lemma 4.6.19, we have that, every entity performs at most  $|S| + 2k^2$  interventions. As there are at most  $k^2$  iterations, and from Lemma 4.6.19, each iteration fails with probability at most  $\delta/2k^2$ , using union bound, we have that at least one of the iterations fails with probability at most  $\delta/2$ .

Finally, using union bound for failure probability of calculating  $S$  correctly, and failing in at least one of the iterations, we have, with probability at least  $1 - \delta$ , Algorithm  $\alpha$ -GENERAL recovers the true clusters.  $\square$

From Lemma 4.6.20, we know that we can recover the clusters correctly with probability at least  $1 - \delta$ . Using the Meta-Algorithm discussed in Appendix 4.6.4.1, and from Corollary 4.6.13, we can obtain an algorithm to learn the graphs of every entity with an additional intervention per entity. Combining it with guarantees obtained by Algorithm  $\alpha$ -BOUNDEDDEGREE in Theorem 4.6.18, gives us the following result.

**Theorem 4.6.21** (Theorem 4.4.1 restated). *If MAGs  $\mathcal{M}_1, \dots, \mathcal{M}_M$  satisfy  $\alpha$ -clustering property with true clusters  $C_1^*, \dots, C_k^*$  such that  $\min_{b \in [k]} |C_b^*| = \Omega(n \log(M/\delta))$ . Then, there is an algorithm that exactly learns all these MAGs with probability at least  $1 - \delta$ . Every entity  $i \in [M]$  uses  $\min \{O(\Delta \log(M/\delta)/\alpha), O(\log(M/\delta)/\alpha + k^2)\}$  many atomic interventions.*

#### 4.6.5 Lower Bound on the Number of Interventions

In this section, we present a lower bound for the number of interventions required by every entity to recover true clusters. First, we state Yao's minimax theorem, which will be used to prove the lower bound.

**Theorem 4.6.22** (Yao's minimax theorem [131]). *Let  $\mathcal{X}$  be a set of inputs to a problem and  $\mathcal{A}$  the set of all possible deterministic algorithms that solve the problem. For any algorithm  $A \in \mathcal{A}$  and  $x \in \mathcal{X}$ , let  $\text{cost}(A, x)$  denote real-valued measure of cost of an algorithm  $A$  on input  $x$ . Let  $\nu, \mu$  be distributions over  $\mathcal{A}$  and  $\mathcal{X}$  respectively. Then,*

$$\max_{x \in \mathcal{X}} \mathbf{E}_{A \sim \nu} [\text{cost}(A, x)] \geq \min_{a \in \mathcal{A}} \mathbf{E}_{X \sim \mu} [\text{cost}(a, X)]$$

Informally, the theorem states that to prove lower bounds on the cost of *any* randomized algorithm, we have to find some distribution  $\mu$  on inputs, such that *every* deterministic algorithm  $A \in \mathcal{A}$  has high cost.

**Outline of the Lower Bound** Our distribution  $\mu$  places a probability of  $1/2$  for pairs of MAGs that have distance zero and a probability of  $1/2$  equally distributed among all pairs of MAGs with distance equal to  $\alpha n$ . This ensures that both the events considered are equally likely, and we show that to distinguish them, with success probability at least  $2/3$  (over the distribution  $\mu$ ), every deterministic algorithm must make  $\Omega(1/\alpha)$  interventions for both the MAGs. Then, we use Yao’s theorem to translate this into a worst case lower bound for any randomized algorithm. In particular, this means that any algorithm that is based on recovering the clusters to construct the MAGs will require  $\Omega(1/\alpha)$  interventions for every entity in  $[M]$ .

**Construction of the Hard Instances** For the lower bound, consider the case when  $M = 2$ , and assuming causal sufficiency, where we wish to identify the clusters of two MAGs  $\mathcal{M}_1, \mathcal{M}_2$ . We observe that a lower bound on the number of interventions required for every entity in the case of identifying two clusters will also extend for the general case of identifying  $k$  clusters with latents.

Consider two MAGs  $\mathcal{M}_1, \mathcal{M}_2$  on a node set  $V$ , with the promise that either  $d(\mathcal{M}_1, \mathcal{M}_2) = 0$  or  $d(\mathcal{M}_1, \mathcal{M}_2) = \alpha n$ , and the goal is to identify which case holds. Note that in the first case the two entities are in the same cluster ( $k = 1$ ), and in the second case they are in different clusters ( $k = 2$ ).

Let  $V = \{v_1, \dots, v_n\}$  be the set of observable nodes of these MAGs. Consider the node difference set of the MAGs  $\mathcal{M}_1, \mathcal{M}_2$  given by  $\text{diff}(\mathcal{M}_1, \mathcal{M}_2)$  and let  $e \in \{0, 1\}^n$

denote its characteristic vector where  $l$ th coordinate of  $e$  is 1 iff  $v_l \in \text{diff}(\mathcal{M}_1, \mathcal{M}_2)$ . We can observe that, under the above promise,  $e$  is either  $0^n$  or has exactly  $\alpha n$  ones. Therefore, we have reduced our problem to that of finding whether the vector  $e$  contains all zeros or not. Using this reduction, we focus on establishing a lower bound for this modified problem.

We want to check if a given  $n$ -dimensional binary vector is a zero vector, i.e.,  $0^n$  or not, with a promise that if it is not a zero vector, then, it contains  $\alpha n$  coordinates with 1 in them. Using Lemma 4.6.23, we show that  $\Omega\left(\frac{1}{\alpha}\right)$  queries to co-ordinates of  $x$  are required, for any randomized or deterministic algorithm to distinguish between these two cases.

**Lemma 4.6.23.** *Suppose we are given a vector  $x \in \{0, 1\}^n$  with the promise that either  $x = 0^n$  or  $x$  contains  $\alpha n$  ones. In order to distinguish these two cases with probability more than  $2/3$ , every randomized or deterministic algorithm must make at least  $\Omega(1/\alpha)$  queries to the coordinates of the vector  $x$ .*

*Proof.* It is easy to see that every deterministic algorithm for this problem requires  $(1 - \alpha)n + 1$  queries. For obtaining a lower bound on the number of queries of any randomized algorithm, we use Yao's minimax theorem [131]. To do so, we construct an input distribution  $\mu$  on  $\{0, 1\}^n$  and show that every deterministic algorithm on the worst case requires at least  $q$  queries while succeeding with a probability  $2/3$ . From Yao's minimax theorem (Thm 4.6.22), this implies that every randomized algorithm requires at least  $q$  queries to output the correct answer with probability of success  $2/3$ . We construct  $\mu$  by using a probability of  $1/2$  for  $0^n$  vector and a probability of  $1/2$  equally distributed among all vectors in  $\{0, 1\}^n$  containing exactly  $\alpha n$  ones.

Suppose a deterministic algorithm (denoted by ALG) is used to identify whether  $x = 0^n$  or not. Let  $\mathcal{E}(x)$  denote the event that the ALG answers correctly on  $x \in \{0, 1\}^n$ ,  $Q(x)$  denote the set of queries used by ALG such that  $|Q(x)| = q$  and  $L(x)$  denote the coordinates of  $x$  that are non-zero.

Consider the event  $\mathcal{E}(x)$  when ALG answers correctly. We can write it as :

$$\begin{aligned} & \Pr_{x \sim \mu}[\mathcal{E}(x)] \\ &= \Pr[\mathcal{E}(x) \mid Q(x) \cap L(x) \neq \phi] \Pr[Q(x) \cap L(x) \neq \phi] + \Pr[\mathcal{E}(x) \mid Q(x) \cap L(x) = \phi] \Pr[Q(x) \cap L(x) = \phi] \end{aligned}$$

We calculate the probability that the coordinates queried are not part of the non-zero coordinates of  $x$ , given by  $Q(x) \cap L(x) = \phi$  :

$$\begin{aligned} & \Pr_{x \sim \mu}[Q(x) \cap L(x) = \phi] \\ &= \Pr[Q(x) \cap L(x) = \phi \mid x = 0^n] \Pr[x = 0^n] + \Pr[Q(x) \cap L(x) = \phi \mid x \neq 0^n] \Pr[x \neq 0^n] \\ &= \frac{1}{2} (\Pr[Q(x) \cap L(x) = \phi \mid x = 0^n] + \Pr[Q(x) \cap L(x) = \phi \mid x \neq 0^n]) \\ &= \frac{1}{2} \left( 1 + \frac{\binom{n-q}{\alpha n}}{\binom{n}{\alpha n}} \right) = \frac{1 + \tau}{2}, \text{ where } \tau = \frac{\binom{n-q}{\alpha n}}{\binom{n}{\alpha n}}. \end{aligned}$$

Now, we calculate the probability that ALG answers correctly when the queries  $Q(x)$  all return zero. We upper bound this probability by considering the case when ALG answers ‘yes’, and the case when ALG answers ‘no’ separately. It is easy to observe that  $\mathcal{E}(x)$  is correct when ALG = ‘yes’ iff  $x = 0^n$ . Therefore, we have:

$$\begin{aligned} & \Pr[\mathcal{E}(x) \mid Q(x) \cap L(x) = \phi] \\ & \leq \max \left\{ \frac{\overbrace{\Pr[\mathcal{E}(x), Q(x) \cap L(x) = \phi]}^{\text{ALG answers 'yes'}}}{\Pr[Q(x) \cap L(x) = \phi]}, \frac{\overbrace{\Pr[\mathcal{E}(x), Q(x) \cap L(x) = \phi]}^{\text{ALG answers 'no'}}}{\Pr[Q(x) \cap L(x) = \phi]} \right\} \\ & \leq \max \left\{ \frac{1/2}{(1 + \tau)/2}, \frac{\tau/2}{(1 + \tau)/2} \right\} \leq \frac{1}{1 + \tau}. \end{aligned}$$

$$\begin{aligned}
\Pr_{x \sim \mu}[\mathcal{E}(x)] &= \Pr[\mathcal{E}(x) \mid Q(x) \cap L(x) \neq \phi] \Pr[Q(x) \cap L(x) \neq \phi] + \\
&\quad \Pr[\mathcal{E}(x) \mid Q(x) \cap L(x) = \phi] \Pr[Q(x) \cap L(x) = \phi] \\
&\leq \Pr[Q(x) \cap L(x) \neq \phi] + \Pr[\mathcal{E}(x) \mid Q(x) \cap L(x) = \phi] \Pr[Q(x) \cap L(x) = \phi] \\
&\leq \left(1 - \frac{1 + \tau}{2}\right) + \frac{1}{1 + \tau} \frac{1 + \tau}{2} = 1 - \frac{\tau}{2}.
\end{aligned}$$

We know the probability of success for ALG is at least  $\frac{2}{3}$ . Therefore, we have  $\Pr_{x \sim \mu}[\mathcal{E}(x)] \geq \frac{2}{3}$ , which implies  $\tau \leq \frac{2}{3}$ .

Let  $H(x)$  denote the binary entropy function. Using the bound from ([93], Page 309)

$$\sqrt{\frac{a}{8b(a-b)}} 2^{aH(b/a)} \leq \binom{a}{b} \leq \sqrt{\frac{a}{2\pi b(a-b)}} 2^{aH(b/a)}.$$

We have

$$\begin{aligned}
\tau &= \frac{\binom{n-q}{\alpha n}}{\binom{n}{\alpha n}} \leq \sqrt{\frac{8(n-q)(n-\alpha n)}{2\pi n(n-q-\alpha n)}} 2^{(n-q)H(\frac{\alpha n}{n-q}) - nH(\alpha)} \\
&= \sqrt{\frac{4}{\pi} \left(1 + \frac{q\alpha}{n-q-\alpha n}\right)} 2^{(n-q)(H(\frac{\alpha n}{n-q}) - H(\alpha)) - qH(\alpha)}.
\end{aligned}$$

We observe that  $q \leq (1-\alpha)n+1$  for any algorithm, as we can identify whether  $x = 0^n$  or not trivially by querying more than  $(1-\alpha)n+1$  coordinates. Therefore,

$$\begin{aligned}
\tau &\leq \sqrt{\frac{4}{\pi}} (1 - q\alpha) 2^{(n-q)(H(\frac{\alpha n}{n-q}) - H(\alpha)) - qH(\alpha)} \\
&\leq \sqrt{\frac{4}{\pi}} 2^{-q\alpha \log e/2 + (n-q)(H(\frac{\alpha n}{n-q}) - H(\alpha)) - qH(\alpha)}
\end{aligned}$$

Using  $\frac{\alpha n}{n-q} \geq \alpha$  and mean-value theorem, we have:

$$\begin{aligned}
(n-q) \left( H\left(\frac{\alpha n}{n-q}\right) - H(\alpha) \right) &\leq q\alpha H' \left( \frac{\alpha n}{n-q} \right) \\
&= q\alpha \log \left( \frac{n-q}{\alpha n} - 1 \right) \\
&\leq q\alpha \log(1-\alpha) - q\alpha \log \alpha.
\end{aligned}$$

Substituting the above expression and expanding  $H(\alpha)$ , we have :

$$\begin{aligned}
\tau &\leq \sqrt{\frac{4}{\pi}} 2^{-q\alpha \log e/2 + q\alpha \log(1-\alpha) - q\alpha \log \alpha + q\alpha \log \alpha + (1-\alpha)q \log(1-\alpha)} \\
&\leq \sqrt{\frac{4}{\pi}} 2^{-q\alpha \log e/2 + q \log(1-\alpha)} \\
&\leq \sqrt{\frac{4}{\pi}} 2^{-q\alpha \log e/2 - q\alpha} \leq \frac{2}{3}.
\end{aligned}$$

Therefore, for ALG to succeed with probability at least  $2/3$ , we have

$$q \geq \Omega\left(\frac{1}{\alpha}\right).$$

Using this with Yao's minimax theorem (Thm 4.6.22), we get that with every randomized algorithm needs  $\Omega(1/\alpha)$  queries to succeed on this problem with probability at least  $2/3$ .  $\square$

For the above problem of identifying whether a vector is zero or not, we can replace each coordinate query by an intervention on the corresponding node for the two entities (due to the equivalency between the two as explained above). Therefore, from Lemma 4.6.23, we have the following corollary about recovering the clusters.

**Corollary 4.6.24.** *Suppose we are given two MAGs  $\mathcal{M}_1$  and  $\mathcal{M}_2$  corresponding to two entities, with the promise that either  $d(\mathcal{M}_1, \mathcal{M}_2) = 0$  or  $d(\mathcal{M}_1, \mathcal{M}_2) = \alpha n$ . In order to distinguish these two cases with probability at least  $2/3$ , every (randomized or deterministic) algorithm must make at least  $\Omega(1/\alpha)$  interventions on both the entities.*

**Theorem 4.6.25** (Theorem 4.4.2 restated). *Suppose the underlying MAGs  $\mathcal{M}_1, \dots, \mathcal{M}_M$  satisfy  $\alpha$ -clustering property. In order to recover the clusters with probability  $2/3$ , every (randomized or deterministic) algorithm requires  $\Omega(1/\alpha)$  interventions for every entity in  $[M]$ .*

*Proof.* From Corollary 4.6.24, we have that to identify whether two MAGs belong to the same cluster or not, we have to make at least  $\Omega(1/\alpha)$  interventions for every entity. Therefore, to recover all the clusters, we have to make at least  $\Omega(1/\alpha)$  many interventions for every entity  $i \in [M]$ .  $\square$

**Remark** The lower bound for the number of interventions per entity is for the first step of identifying the underlying clustering. Similar to our upper bounds, our lower bound considers the worst-case, where the MAGs satisfy the  $\alpha$ -clustering property are all Markov equivalent. This implies that the PAGs obtained using FCI are identical and will not be helpful in identifying the clusters. In practice, the information available in the PAGs could be useful to reduce the number of interventions.

#### 4.6.6 Missing details from the Experimental Evaluation

In this section, we provide additional details about the experimental evaluation discussed in section 4.5.

##### 4.6.6.1 Learning MAGs under $(\alpha, \beta)$ -clustering property

**(Synthetic) Data Generation** We use following process for each of the five considered causal network (*Asia*, *Earthquake*, *Sachs*, *Survey*, and *Erdős-Renyi*). We construct causal MAGs for  $M$  entities distributed among the clusters  $C_1^*, C_2^*, \dots, C_k^*$  equally, i.e.,  $|C_i^*| = M/k$  for all  $i \in [k]$ . In our experiments, we set  $k = 2$  (i.e., two clusters), and start with  $k = 2$  DAGs that are sufficiently far apart. To do so, we create two copies of the original causal network  $\mathcal{D}$ , and denote the DAG copies by  $\mathcal{D}_1$  and  $\mathcal{D}_2$ . For each of the DAGs  $\mathcal{D}_1$  and  $\mathcal{D}_2$ , we select a certain number of pairs of nodes randomly, and include a latent variable between them, that has a causal edge to both the nodes. In our experiments, we used 2 latents per DAG. This results in two new DAGs  $\mathcal{D}'_1$  and  $\mathcal{D}'_2$ . To ensure  $\alpha n$  node distance between clusters, we modify  $\mathcal{D}'_2$  using random changes until the two MAGs corresponding to the DAGs  $\mathcal{D}'_1$  and  $\mathcal{D}'_2$

are separated by a distance of  $\alpha n$ . These two MAGs, denoted by  $\mathcal{M}_1^{\text{dom}}$  and  $\mathcal{M}_2^{\text{dom}}$  form the dominant MAG for each of the two clusters.

Then, we create  $(1 - \gamma)M/k = (1 - \gamma)M/2$  copies of the dominant MAG and assign it to distinct entities in each cluster. Consider cluster  $C_1^*$  with dominant MAG  $\mathcal{M}_1^{\text{dom}}$ , and corresponding DAG  $\mathcal{D}_1'$ . Note that each cluster has  $M/k = M/2$  entities. For the remaining entities in  $C_1^*$ , we start with  $\mathcal{D}_1$  and include 2 latent variables between randomly selected pairs of nodes. Then, we repeat the previous procedure, of performing a series of random insertions or deletions of edges to the DAG until the distance between the corresponding MAG and  $\mathcal{M}_1^{\text{dom}}$  increases to  $\beta n$ . We follow the same procedure for cluster  $C_2^*$  with dominant MAG  $\mathcal{M}_2^{\text{dom}}$ . Note that in this construction different entities could differ both in latents and their observable graphs. This construction ensures the entities satisfy  $(\alpha, \beta)$ -clustering property. As an example, see Figure 4.7 containing two dominant MAGs of the Causal Network *Earthquake*.

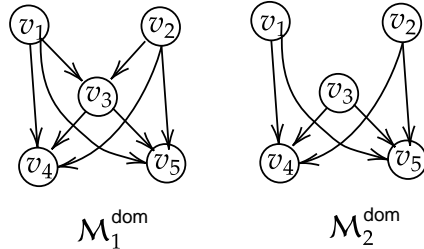


Figure 4.7: Dominant MAGs of the causal network *Earthquake* constructed using the described procedure.

**Sample Set  $S$  Size** For Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE, we use different sample sizes  $S$  ranging from 1 to 3. In Figure 4.8, we plot the mean value of the maximum number of interventions per entity with change in sample set size.

With increase in sample set size, our Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE requires more interventions (see Lemma 4.6.8) and we observe the same in Figure 4.8. We chose the smallest size  $|S| = 1$  in our experiments, as increasing the size will increase the number of interventions but did not lead to much improved clustering results. As

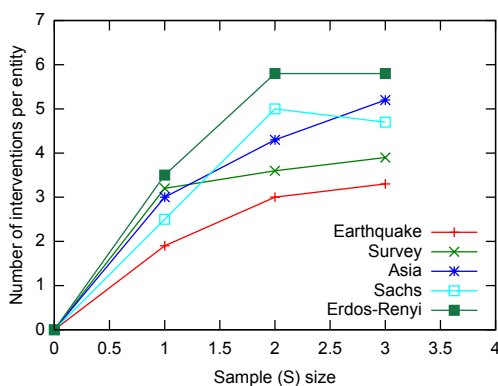


Figure 4.8: Sample size vs. maximum number of interventions per entity used by Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE.

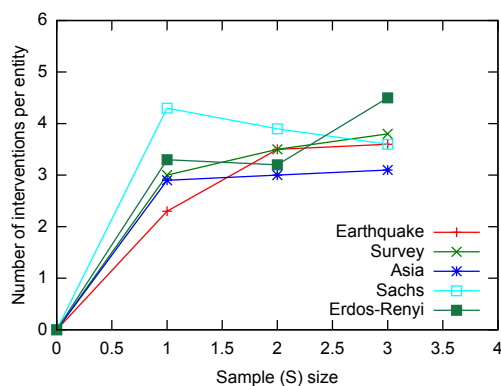


Figure 4.9: Sample size vs. maximum number of interventions per entity used by Algorithm  $\alpha$ -BOUNDEDDEGREE.

a sample set of size 1 roughly corresponds to around 3 interventions (across all causal networks), we use that for results presented in Table 4.1.

**Construction of Clusters from FCI Output** Our first focus is on recovering the true clustering using Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE. As a baseline, we employ the well-studied FCI algorithm [119]. We know that FCI returns the Partial Ancestral Graph(PAG) corresponding to the causal MAG using only the observational data. After recovering the PAGs corresponding to the MAGs using FCI, we cluster them by constructing a weighted graph (similar to Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE) defined on the set of entities. For every pair of entities  $i, j$ , we calculate the number of nodes  $n_{ij}$  that share the same neighborhood using the PAGs associated with them, and assign the weight of the edge as  $n_{ij}$ . This weight captures the similarity between two entities, and whether they belong to the same cluster or not. Now, we use minimum- $k$ -cut algorithm to partition the set of entities into  $k$  components or clusters. In Algorithm  $(\alpha, \beta)$ -BOUNDEDDEGREE, we first construct a sample  $S$ , and perform various interventions based on the set  $S$  for every entity to finally obtain the  $k$  clusters.

| Causal Network     | FCI             |                 |                 | $\alpha$ -BOUNDEDDEGREE (Alg. 16) |                 |                 | Maximum # Interventions |
|--------------------|-----------------|-----------------|-----------------|-----------------------------------|-----------------|-----------------|-------------------------|
|                    | Precision       | Recall          | Accuracy        | Precision                         | Recall          | Accuracy        |                         |
| <i>Earthquake</i>  | $0.79 \pm 0.25$ | $0.98 \pm 0.02$ | $0.79 \pm 0.25$ | $1 \pm 0.00$                      | $1.0 \pm 0.0$   | $1.00 \pm 0.00$ | 3                       |
| <i>Survey</i>      | $0.79 \pm 0.25$ | $0.98 \pm 0.02$ | $0.79 \pm 0.25$ | $0.89 \pm 0.20$                   | $1.0 \pm 0.00$  | $0.89 \pm 0.20$ | 4                       |
| <i>Asia</i>        | $0.84 \pm 0.23$ | $0.98 \pm 0.02$ | $0.84 \pm 0.23$ | $0.89 \pm 0.20$                   | $1.0 \pm 0.00$  | $0.89 \pm 0.20$ | 4                       |
| <i>Sachs</i>       | $1.0 \pm 0.00$  | $1.0 \pm 0.00$  | $1.0 \pm 0.00$  | $0.79 \pm 0.25$                   | $1.0 \pm 0.00$  | $0.79 \pm 0.25$ | 5                       |
| <i>Erdős-Rényi</i> | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$                   | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | 5                       |

Table 4.2: In this table, we present the precision, recall and accuracy values obtained by Algorithm  $\alpha$ -BOUNDEDDEGREE and FCI. Each cell includes the mean value along with the standard deviation computed over 10 runs. The last column contains the maximum number of interventions per entity required (including both Algorithm  $\alpha$ -BOUNDEDDEGREE and the Meta-algorithm) for recovering the DAGs.

**Setup** We used a personal Apple Macbook Pro laptop with 16GB RAM and Intel i5 processor for conducting all our experiments. We use the FCI algorithm implemented in [78]. For every causal network, each experiment took less than 10 minutes to finish all the 10 runs.

#### 4.6.6.2 Learning MAGs under $\alpha$ -clustering property

**(Synthetic) Data Generation** We use following process for each of the five considered causal network (*Asia*, *Earthquake*, *Sachs*, *Survey*, and *Erdős-Rényi*). We construct causal DAGs for  $M$  entities distributed among the clusters  $C_1^*, C_2^*, \dots, C_k^*$  equally, i.e.,  $|C_i^*| = M/k$  for all  $i \in [k]$ . Again we set  $k = 2$ . For each of the DAGs  $\mathcal{D}_1$  and  $\mathcal{D}_2$ , we select a certain number of pairs of nodes randomly, and include a latent variable between them, that has a causal edge to both the nodes. In our experiments, we used 2 latents per DAG. This results in two new DAGs  $\mathcal{D}'_1$  and  $\mathcal{D}'_2$ . To ensure  $\alpha n$  node distance between clusters, we modify  $\mathcal{D}'_2$  using random changes until the two DAGs  $\mathcal{D}'_1$  and  $\mathcal{D}'_2$  are separated by a distance of  $\alpha n$  and are Markov equivalent. Without Markov equivalence, we observe that FCI always recovers the underlying clusters correctly in the  $\alpha$ -clustering case.<sup>4</sup> However, existence of Markov equivalent

---

<sup>4</sup>Again this is not true for  $(\alpha, \beta)$ -clustering, as shown by our experiments results for that case, because now difference in PAGs between two entities does not automatically imply that those two entities must belong to different clusters.

DAGs is a well-known problem in real-world graphs, a popular example to illustrate this comes the “breathing dysfunction” causal graph in Fig. 3 in [133]. We create  $M/2$  copies of each of the two DAGs  $\mathcal{D}'_1$  and  $\mathcal{D}'_2$  and assign it to distinct entities in each of the two clusters.

**Parameters** We present the following settings for the model parameters,  $\alpha$  is at least 0.60,  $M = 40$ . For the synthetic data generated using Erdős-Rényi model, we use  $n = 10$ , probability of edge  $p = 0.30$ . We ran all of our experiments for 10 times with the stated values and report the results.

**Sample Set  $S$  Size** For Algorithm  $\alpha$ -BOUNDEDDEGREE, we again tried different set  $S$  sizes ranging from 1 to 3. In Figure 4.9, we plot the mean value of the maximum number of interventions per entity with increase in sample size. It has a same trend as with  $(\alpha, \beta)$ -clustering (Figure 4.9). A sample size of 1 roughly corresponds to around 3 interventions, and we use that for the results presented in Table 4.2.

**Evaluation of Clustering** We start by results on recovering the clustering using Algorithm  $\alpha$ -BOUNDEDDEGREE. As a baseline, we again employ the well-studied FCI algorithm [119]. After recovering the PAGs corresponding to the DAGs using FCI, we cluster them by constructing a similarity graph (similar to the case of  $(\alpha, \beta)$ -clustering discussed previously) defined on the set of entities. For Algorithm  $\alpha$ -BOUNDEDDEGREE, we first construct a sample  $S$ , and perform various interventions based on the set  $S$  for every entity to finally obtain the  $k$  clusters. We also implemented another baseline algorithm (GREEDY) that uses interventions, based on a greedy idea that selects nodes to set  $S$  in Algorithm  $\alpha$ -BOUNDEDDEGREE by considering nodes in increasing order of their degree in the PAGs returned by FCI. We use this ordering to minimize the number of interventions as we intervene on every node in  $S$  and their neighbors. We use the same metrics as the  $(\alpha, \beta)$ -clustering case.

**Results** In Table 4.2, we compare Algorithm  $\alpha$ -BOUNDEDDEGREE to FCI on the clustering results. For Algorithm  $\alpha$ -BOUNDEDDEGREE, we use a sample  $S$  of size 1, and observe in Figure 4.9, that this corresponds to about 2 interventions per entity. With increase in sample size, we observed that the results were either comparable or better. We observe that our approach leads to considerably better performance in terms of the accuracy metric with an average difference in mean accuracy of about 0.20. We observe that entities belonging to the same true cluster are always assigned to the same cluster, resulting in high recall for both Algorithm  $\alpha$ -BOUNDEDDEGREE and FCI. Further, the higher value of precision for our algorithm is because FCI is unable to correctly detect that there are two clusters, as the DAGs are Markov Equivalent which means that they result in the same PAGs.

Algorithm  $\alpha$ -BOUNDEDDEGREE outperforms the GREEDY baseline for the same sample ( $S$ ) size. For example, on the *Earthquake* and *Survey* causal networks, Algorithm  $\alpha$ -BOUNDEDDEGREE obtains the mean accuracy values of 1.0 and 0.89 respectively, while GREEDY for the same number of interventions obtained an accuracy of only 0.74 and 0.64 respectively. On the remaining causal networks, the accuracy values of GREEDY are almost comparable to our Algorithm  $\alpha$ -BOUNDEDDEGREE.

After clustering, we recover the DAGs using the Meta-algorithm described in section 4.6.4.1, and observe that only one additional intervention is needed. In the last column in Table 4.2, we report the maximum number of interventions for recovering DAGs, which includes both the interventions used by the Algorithm  $\alpha$ -BOUNDEDDEGREE and the Meta-algorithm. We observe that our *collaborative* approach uses fewer interventions for MAG recovery compared to the number of nodes in each causal network. For example, in the Erdős-Rényi setup, the number of nodes  $n = 10$ , whereas we use at most 5 interventions per entity. Thus, compared to the worst-case, cutting the number of interventions for each entity by 50%.

## 4.7 Conclusion

In this chapter, we introduced a new model for causal discovery to capture practical scenarios where there are multiple entities with different causal structures. Under natural clustering assumption(s), we gave efficient provable algorithms for causal learning with atomic interventions and demonstrate its empirical performance. Our model can be extended to the setting where all interventions are non-adaptive, and we plan to study it as part of future work. An interesting future direction would be to use interventional equivalence classes of DAGs as part of the model, instead of the clustering assumption. This might require extending the interventional equivalence between DAGs studied in [65, 83] to the setting without the causal sufficiency assumption and exploit that for learning.

## CHAPTER 5

### NON-ADAPTIVE EDGE COUNTING AND SAMPLING VIA BIPARTITE INDEPENDENT SET QUERIES

In this chapter, we study sub-linear query algorithms for estimating the number of edges and sampling uniformly random edges in a simple, unweighted graph. In section 5.1, we describe various query models for accessing any graph which facilitate sub-linear query algorithms. In section 5.6, we describe our approach for estimating the number of edges; in section 5.7, we extend some of our ideas for counting the edges in a graph, to return a uniform sample among the set of edges; in section 5.8, we give efficient algorithms for testing graph connectivity; in section 5.9, we conclude the chapter with directions for future work.

#### 5.1 Query Models for Graph Access

In this section, we present a brief discussion on various query models studied for graph discovery in the sublinear algorithms literature. First, we present the *Bipartite Independent Set (BIS)* query model, where access to  $G$  is via a *Bipartite Independent Set (BIS)* oracle [22]. A query to this oracle takes as input two disjoint subsets  $L, R \subseteq V$  and returns

$$BIS(L, R) = \begin{cases} '1' & \text{if there is no edge between } L \text{ and } R \\ '0' & \text{otherwise.} \end{cases}$$

**Local Query Models** Prior work on sub-linear query graph algorithms has largely focused on *local* queries, in particular, (i) vertex degree queries (ii) neighbor queries

(output the  $i$ th neighbor of a vertex) and (iii) edge existence queries [54, 63, 114]. In the literature, the first two types of queries form the *adjacency list* query model, while all three types of queries form the *adjacency matrix* query model. Under these models, a variety of graph estimation problems have been well studied, including edge counting and sampling [53, 63, 114, 126], subgraph counting [10, 31, 52], vertex cover [23, 101], and beyond [110].

For a graph with  $n$  nodes and  $m$  edges, given access only to degree queries, Feige [54] presented an algorithm for estimating  $m$  up to  $(2 \pm \epsilon)$  relative error with query complexity  $O(\sqrt{n} \cdot \text{poly}(1/\epsilon, \log n))$ . This work also showed that any  $(2 - o(1))$ -approximation algorithm requires  $\Omega(n)$  queries. In the adjacency list query model, Goldreich and Ron [63] gave a  $(1 \pm \epsilon)$ -approximation algorithm, with query complexity  $O(n/\sqrt{m} \cdot \text{poly}(1/\epsilon, \log n))$ . Recently, Eden and Rosenbaum [53] gave algorithms for near-uniform edge sampling with the same query complexity, and showed that this complexity is nearly tight.

**Global Query Models** Motivated by the desire to obtain more query efficient algorithms, Beame et al. [22] studied edge estimation using *global* queries that can make use of information across the graph, including the BIS queries that we will focus on, and the related Independent Set (IS) queries. IS queries were introduced in the literature on query efficient graph recovery [1, 12]. They answer whether or not there exist any edges in the induced subgraph on a subset of nodes  $S \subseteq V$ . We refer the reader to the exposition in [22], which discusses applications of these global query models in group testing [39, 48], computational geometry [13, 33, 56], fine-grained complexity [46, 47], and decision versus counting complexity [47, 111, 121, 122].

In the IS query model, [22, 40] give a  $O(\min\{\sqrt{m}, n/\sqrt{m}\} \cdot \text{poly}(\log n, 1/\epsilon))$  query algorithm for  $(1 \pm \epsilon)$  approximate edge counting. In the BIS model, numerous authors [22, 47, 26] achieve  $(1 \pm \epsilon)$ -approximation for edge counting and near-uniform

edge sampling using just  $\text{poly}(1/\epsilon, \log n)$  queries. This is exponentially smaller than the query complexities in the IS and local queries models.

Extending the BIS query model to hypergraphs, Dell et al. [47] introduce the *coloured independence oracle* which detects the presence of a size  $k$  hyperedge. They give algorithms for hyperedge estimation and sampling using this generalized oracle. Many other variants of global queries have been studied including LINEAR, OR and CUT queries [16, 36, 112]. These queries have been applied to solving maximum matching [84, 100], minimum cut [112], triangle estimation [24, 25, 47], connectivity [16], hitting sets [29], weighted edge estimation [30], problems related to linear algebra [107], quantum algorithms [96], and full graph recovery [1, 12].

## 5.2 The Role of Adaptivity

For both local and global queries, most sub-linear time graph algorithms are *adaptive*, i.e., a query may depend on the answers to previous queries. In many cases, it is desirable for queries to be *non-adaptive*. This allows them to be completed independently, and might allow for the resulting algorithm to be easily implemented in massively parallel computation frameworks [81]. Non-adaptive algorithms also lead naturally to single-pass, rather than multi-pass, streaming algorithms. In fact, the BIS query model can be seen as a very restricted subset of the more general LINEAR query model, in which each query outputs the inner product of the edge indicator vector with a query vector. This model has long been studied in the graph-streaming literature [9, 95], in part due to its usefulness in giving single-pass algorithms. However, it has remained open whether non-adaptive algorithms can be given in more restricted global query models.

For these reasons, Assadi et al. [16] and Chakrabarti and Stoeckl [36] have recently sought to reduce query adaptivity under a variety of global query models, including LINEAR, OR, CUT and BIS queries. These works study the *single element recovery*

problem, which is a weaker variant of uniform edge sampling, requiring that the algorithm return a single edge in  $G$ . Assadi et al. also study the problem of checking connectivity, presenting a BIS query algorithm making  $\tilde{O}(n)$  queries and using three rounds of adaptivity. They give a two-round algorithm in the stronger OR query model, and show that even in this model, there is no non-adaptive algorithm for connectivity making  $o(n^2)$  queries.

We note that reducing query adaptivity is also a well-studied direction in the closely related literature on group testing [49, 73]. IS and BIS oracles can be thought of as tests if there is a single element in a group of edges, where that group is required to be all edges incident on one node set (IS) or between two disjoint sets (BIS). Attempts to minimize query adaptivity have also been made for sparse recovery [74, 79, 97], sub-modular function maximization [18, 38], property testing [35] and multi-armed bandit learning [8].

## 5.3 Results

In this section, we present our main results. First, in section 5.3.1, we present a discussion of our results, comparing it with prior work. Next, in section 5.4, we present an overview of all the techniques and summarize main ideas behind our approach.

### 5.3.1 Our Contributions

Our main result is the first *non-adaptive* algorithm for edge estimation up to  $(1 \pm \epsilon)$  relative error, using  $\text{poly}(1/\epsilon, \log n)$  BIS queries. Formally, we show:

**Theorem 5.3.1** (Theorem 5.6.5 restated). *Given a graph  $G$  with  $n$  nodes and  $m$  edges, there is an algorithm that makes  $O(\epsilon^{-5} \log^5 n \log^5(\log n) \log(\epsilon^{-1} \log n))$  non-*

adaptive BIS queries to  $G$  and returns an estimate  $\hat{m}$  satisfying:  $m(1 - \epsilon) \leq \hat{m} \leq m(1 + \epsilon)$ , with probability at least  $3/5$ .<sup>1</sup>

Prior methods for  $(1 \pm \epsilon)$  error edge estimation using BIS queries are based on a binary search style approach [22, 47, 26], which is inherently adaptive, and this leads to algorithms requiring  $\Omega(\log^2 n)$  rounds of adaptivity. Beame et al. [22] present a non-adaptive algorithm giving a  $O(\log^2 n)$  approximation factor for bipartite graphs, using  $O(\log^3 n)$  queries. However, no non-adaptive results for general graphs or achieving  $1 \pm \epsilon$  relative error for arbitrary  $\epsilon > 0$  were previously known. Even with adaptivity, the best known algorithm due to [26] has a query complexity of  $O(\epsilon^{-2} \log^{11} n)$  and succeeds with probability  $1 - 1/n^2$ . Therefore, our non-adaptive result improves upon the current best known algorithms, for constant  $\epsilon$ .

Our second result builds on our edge estimation approach, giving the first non-adaptive BIS query algorithm that returns a near-uniformly sampled edge. Formally:

**Theorem 5.3.2** (Theorem 5.7.10 restated). *Given a graph  $G$  with  $n$  nodes,  $m$  edges, and edge set  $E$ , there is an algorithm that makes:*

*$O(\epsilon^{-4} \log^6 n \log(\epsilon^{-1} \log n) + \epsilon^{-6} \log^5 n \log^6(\log n) \log(\epsilon^{-1} \log n))$  non-adaptive BIS queries which, with probability at least  $1 - \epsilon$ , outputs an edge from a probability distribution  $P$  satisfying  $(1 - \epsilon)/m \leq P(e) \leq (1 + \epsilon)/m$  for every  $e \in E$ .*

Prior results for near-uniform edge sampling required  $\Omega(\log^3 n)$  rounds of adaptivity [26, 47]. Additionally, even ignoring adaptivity, our results improves on the best known query complexity of  $O(\epsilon^{-2} \log^{14} n)$ , due to [26], for constant  $\epsilon$ .

By combining Theorem 5.3.2 with prior work on sublinear query graph connectivity, via edge sampling, we obtain a connectivity algorithm using two rounds for adaptivity:

---

<sup>1</sup>Note that the success probability can be boosted in the standard way, by running multiple independent instantiations of the algorithm and taking their median estimate.

**Theorem 5.3.3** (Theorem 5.8.1 restated). *Given a graph  $G$  with  $n$  nodes, there is a 2-round adaptive algorithm that determines if  $G$  is connected with probability at least  $1 - 1/n$  using  $\tilde{O}(n \log^8 n)$  BIS queries, where  $\tilde{O}(\cdot)$  ignores the  $\log^{O(1)} \log n$  dependencies.*

Theorem 5.3.3 improves on a three-round algorithm of Assadi et al. [16] and is tight: even in the stronger OR query model (which allows checking the presence of an edge within an arbitrary subset of edges) no non-adaptive algorithm can make  $o(n^2)$  queries. Assadi et al. gave a two-round algorithm in this stronger OR query model. Thus, Theorem 5.3.3 closes the gap between BIS queries and OR queries for this problem. We note that there is a separation from the even stronger LINEAR query model, where non-adaptive algorithms for connectivity and cut approximation are well-known [9]. Understanding if there remain interesting separations between the BIS and OR query models in terms of adaptivity would be very interesting.

## 5.4 Technical Overview

In this section, we present an overview of our non-adaptive BIS query algorithms for edge estimation (Theorem 5.3.1) and near-uniform edge sampling (Theorem 5.3.2), along with our 2-round algorithm for connectivity (Theorem 5.3.3).

### 5.4.1 Edge Estimation

A simple idea to estimate the number of edges in a graph via BIS queries is to sample small random subsets of nodes and run BIS queries to check the presence of an edge between them. The fraction of these queries that return ‘1’ (i.e., indicating the presence of no edge) can then be used to estimate the number of edges. In particular, for a graph containing  $m$  edges, if the random subgraphs have  $O(n/\sqrt{m})$  nodes in them, then we expect a ‘1’ answer with constant probability. Beame et al. [22] describe a non-adaptive algorithm along these lines, which gives a  $O(\log^2 n)$  approximation

for bipartite graphs using  $O(\log^3 n)$  queries. Unfortunately, going beyond this coarse approximation factor is difficult since many dependencies due to common neighbors arise and this increases the variance of the estimators. Beame et al. handle the issue by using the coarse estimates to subdivide the graph into smaller sub-graphs, until these divided graphs only contain  $O(\log^{O(1)} n)$  edges, at which point all their edges can be discovered with few queries. This strategy yields a  $(1 \pm \epsilon)$  approximation, however, it is inherently adaptive.

Our non-adaptive edge estimation algorithm takes a different approach. Suppose we could sample each node with probability  $p_v \approx \epsilon^{-2}d(v)/m$  and compute the degree of the sampled nodes then it is straightforward to show  $\sum_v \mathbb{I}[v \text{ sampled}] \cdot d(v)/p_v$  equals  $2m$  in expectation. Furthermore, an application of Bernstein bound implies that it is a  $(1 \pm \epsilon)$  with sufficient probability. The challenge is showing that this type of approach can be approximated in the BIS query model.

**Subsampling Nodes.** The first idea, drawn from work on streaming algorithms, is to subsample the nodes of  $G$  at different rates of the form  $1/\gamma^j$  where  $\gamma > 1$  is constant and  $j \in \{0, 1, \dots, O(\log n)\}$ . At each rate, we will “recover” all sampled nodes (along with a corresponding degree estimate) whose degree is roughly  $d(v) \approx \epsilon^2 m/\gamma^j$ . In this way, each node will be recovered with probability roughly  $1/\gamma^j \approx \epsilon^{-2}d(v)/m$ , as desired. We describe this subsampling procedure in Section 5.6.3, as part of our main algorithm EDGE-ESTIMATOR (Algorithm 20).

**Recovering Heavy Nodes.** The next challenge is to show that we can actually recover the appropriate nodes and degree estimates at each sampling rate. If we can approximate the degree of all nodes sampled at rate  $1/\gamma^j$  up to additive error  $O(\epsilon^3 \cdot m/\gamma^j)$ , we will obtain a  $(1 \pm \epsilon)$  relative error approximation to the degree of any node we hope to recover at that sampling rate, i.e., any node with degree roughly  $\epsilon^2 m/\gamma^j$ . Using these approximations, we can determine which nodes should be recovered at that rate, and form our edge estimate.

**Degree Estimation via Neighborhood Size Estimation.** To achieve such an additive error approximation, we also use ideas from the sparse recovery and streaming literature. In particular, we implement an approach reminiscent of the Count-Min sketch algorithm [44]. The approach is described in detail in Section 5.6.2, where we present Algorithm ESTIMATE-DEGREE (Algorithm 19). First observe that when sampling at rate  $1/\gamma^j$ , conditioned on any node  $v$  being included in the sample, the expected total degree of the sampled nodes other than  $v$  is  $O(m/\gamma^j)$ . If we further subdivide these nodes into  $\tilde{O}(1/\epsilon^3)$  random groups, the expected total degree of all nodes other than  $v$  in any group is  $\tilde{O}(\epsilon^3 \cdot m/\gamma^j)$ .

Now, if  $v$  is placed in group  $S$ , we can approximately upper bound its degree by the total *neighborhood size of  $S$* . This upper bound holds approximately as long as  $v$  does not have too many neighbors in  $S$ , which it won't with good probability. The neighborhood size of  $S$  is in turn upper bounded by the degree of  $v$  plus the total degree of other nodes in  $S$ , and thus by  $d(v) + \tilde{O}(\epsilon^3 \cdot m/\gamma^j)$  in expectation. So, in expectation, this approach gives an additive  $\tilde{O}(\epsilon^3 \cdot m/\gamma^j)$  error approximation to the degree of each sampled node  $v$ , with constant probability. Repeating this procedure  $O(\log n)$  times, and, as in the Count-Min sketch, taking the minimum degree estimate for each node sampled at rate  $1/\gamma^j$ , gives us high probability approximation for such nodes.

**Neighborhood Size Estimation.** The final step is to implement an algorithm that can estimate the neighborhood size of the random subset of nodes  $S$ , to be used in our degree estimation procedure. We do this in Section 5.6.1, where we present Algorithm NEIGHBORHOOD-SIZE (Algorithm 18). This algorithm takes as input two disjoint subsets  $L, R$  and returns a  $(1 \pm \epsilon)$ -approximation for the size of the neighborhood of  $L$  in  $R$ . We highlight that this may be very different than the *number of edges connecting  $L$  to  $R$*  – the neighborhood size is the number of nodes in  $R$  with at least one edge to  $L$ . This difference is critical in removing the correlations

discussed previously due to common neighbors. Such correlations lead to the adaptive nature of prior algorithms [22, 47]. To estimate the size of the neighborhood of  $L$  in  $R$ , we sample the nodes in  $R$  at different rates and ask BIS queries on  $L$  and the sampled subset of  $R$ . Intuitively, when the sampling rate is the inverse of the size of the neighborhood, we will observe a ‘1’ response with constant probability. We can detect this and thus estimate the neighborhood size.

**Non-adaptivity.** The approach is inherently non-adaptive as all random sampling of nodes and random subsets can be formed ahead of time, independently of any query responses. The only catch is that to determine which nodes should be recovered at each sampling rate, i.e., those nodes with degree  $d(v) \approx \epsilon^{-2} \cdot m/\gamma^j$ , we need a coarse estimate to the edge count  $m$  in the first place. Fortunately, we can bootstrap such an estimate starting with a very coarse  $O(\log^2 n)$ -relative error approximate estimation, due to Beame et al. [22]. We then refine this estimate iteratively using Algorithm REFINE-ESTIMATE (Algorithm 21). Each refinement improves the approximation factor by  $\epsilon$ , and after  $O(\log_{1/\epsilon} \log n)$ , refinements our estimate will result in a  $(1 \pm \epsilon)$ -approximation factor. The key observation here is that each refine step does not require any additional BIS queries. Thus, our algorithm remains non-adaptive.

#### 5.4.2 Uniform Edge Sampling and Connectivity

In the full version, we prove Theorem 5.3.2 by designing and analyzing a non-adaptive algorithm for returning a near-uniform sample among the edges of the graph. Our approach builds heavily on our edge estimation algorithm. If we knew the degree  $d(v)$  of all vertices, then to sample a uniform edge, we could sample a vertex  $v \in V$  with probability  $d(v)/\sum_{w \in V} d(w)$  and return a uniform neighbor among the neighbors of  $v$ . We can observe that the probability that an edge  $(v, u)$  is sampled is  $d(v)/\sum_{w \in V} d(w) \cdot 1/d(v) + d(u)/\sum_{w \in V} d(w) \cdot 1/d(u) = 1/m$ , i.e., this approach yields a uniformly random edge sample.

**Node Sampling.** We implement the above approach approximately using BIS queries in Algorithm SAMPLING. First note that recovered vertices in our edge estimation algorithm are sampled with probabilities roughly proportional to their degrees. We argue that we can select a random vertex from this set, which overall is equal to any vertex  $v$  with probability approximately  $d(v)/\sum_{w \in V} d(w)$ . To do so, we leverage our degree estimates, and the fact that our edge count estimator, which is the sum of scaled degrees of recovered vertices, is well-concentrated.

**Random Neighbor Sampling.** It remains to show how to return a random neighbor of the sampled vertex. To do so, in the full version, we describe an algorithm that takes as input two disjoint subsets  $L, R$  and returns a uniform neighbor among the neighbors of  $L$  in  $R$ . By showing an equivalence between the substantially more powerful OR queries and BIS queries in this specific setting, we argue that an existing algorithm for OR queries can be extended to return a uniform neighbor using BIS queries. An OR query takes as input a subset of pairs of vertices and returns ‘1’ iff there is an edge in the subset queried. Building on this, in the full version, we present Algorithm UNIFORM-NEIGHBOR that takes as input the subset of nodes sampled at any rate  $1/\gamma^j$  as in our edge estimation algorithm, and approximately returns a uniform neighbor for every vertex  $v$  sampled in this set. As before, we construct  $\tilde{O}(1/\epsilon^4)$  random partitions of the sampled nodes. For every vertex  $v$  in a random subset  $S$ , we return a uniform neighbor (obtained using the idea just described) of  $S$  as the neighbor of  $v$ . If  $v$  has large degree compared to the total degree of nodes in the partition, which it will if it is meant to be recovered at that sampling rate, this output will most likely be a neighbor of  $v$ , and will be close to a uniformly random one.

**A Two-Round Algorithm for Connectivity.** Our non-adaptive edge sampling algorithm (Theorem 5.3.2) yields a two-round algorithm for graph connectivity (Theorem 5.3.3), improving on a prior three-round algorithm of [16]. In particular, the

algorithm of [16] selects  $O(\log^2 n)$  random neighbors per vertex, and contracts the connected components of this random graph into *supernodes*. This random sampling step can be performed using one round of  $\tilde{O}(n)$  BIS queries. They prove that in the contracted graph on the supernodes, there are at most  $O(n \log n)$  edges. Using this fact, they then show how to identify whether all the supernodes are connected using  $\tilde{O}(n)$  BIS queries and two additional rounds of adaptivity.

We follow the same basic approach: using a first round of  $\tilde{O}(n)$  queries to randomly sample  $O(\log^2 n)$  neighbors per vertex and contract the graph into supernodes. Once this is done, we observe that we have BIS query access to the contracted graph simply by always grouping together the set of nodes in each supernode. So, we can directly apply the non-adaptive sampling algorithm of Theorem 5.3.2 to sample edges from the contracted graph. By a coupon collecting argument, drawing  $O(n \log^2 n)$  near-uniform edge samples (with replacement) from the contracted graph suffices to recover all  $O(n \log n)$  edges in the graph, and thus determine connectivity of the contracted graph, and, in turn, the original graph.

## 5.5 Preliminaries

Let  $G(V, E)$  denote the graph on vertex set  $V$  with edges  $E \subseteq V \times V$ . Let  $|V| = n$  be the number of nodes and  $|E| = m$  be the number of edges. For any set of nodes  $S \subseteq V$ , let  $E[S] \subseteq E$  denote the edges in the induced subgraph on  $S$ . For any two disjoint sets of nodes  $L, R \subseteq V$ , let  $E[L, R] = \{(u, v) \in E \mid u \in L, v \in R\}$  denote the edges between them. For any  $v \in V$ , let  $\Gamma(v) = \{u \mid (v, u) \in E \text{ for some } v \in V\}$  be its set of neighbours. Let  $d(v) = |\Gamma(v)|$  be its degree. For  $S \subseteq V$ , let  $\Gamma(S) = \bigcup_{u \in S} \Gamma(u)$  and let  $d(S) = \sum_{u \in S} d(u)$ .

**Definition 5.5.1** (OR query). *An OR query takes as input a collection  $E_q$  of pairs of vertices given by  $E_q = \{(x_1, y_1), (x_2, y_2), \dots, (x_k, y_k) \mid x_i, y_i \in V \forall i \in [k]\}$  and satisfies the following:*

$$\mathcal{OR}(E_q) = \begin{cases} '1' & \text{if } E_q \cap E = \phi \\ '0' & \text{otherwise.} \end{cases}$$

**Lemma 5.5.2** (Bernstein's inequality). *Let  $X_1, X_2, \dots, X_n$  be independent random variables. Suppose  $|X_i| \leq M \forall i \in [n]$ . Then:*

$$\Pr \left[ \left| \sum_i X_i - \mathbf{E}[X_i] \right| \geq t \right] \leq \exp \left( - \frac{t^2}{2 \sum_i \mathbf{E}[(X_i - \mathbf{E}[X_i])^2] + \frac{2}{3}Mt} \right)$$

**Fact 5.5.3.**

$$\left(1 + \frac{x}{n}\right)^n \geq e^x \left(1 - \frac{x^2}{n}\right) \geq e^x \quad \text{for } |x| \leq n, \quad n \geq 1.$$

## 5.6 Non-adaptive algorithm for edge estimation

In this section, we present our non-adaptive algorithm for edge estimation using BIS queries. In Section 5.6.1, we describe an algorithm that takes as input two disjoint subsets  $L, R$  and returns an estimate of the size of the neighborhood  $|\Gamma(L) \cap R|$ . Next, in Section 5.6.2, we use this algorithm to give additive error approximations of degrees of all the vertices in a given subset. Finally, in Section 5.6.3, using the approximate degree estimates, we construct a  $(1 \pm \epsilon)$ -approximate estimator for  $m$  by sampling nodes with probabilities roughly proportional to their degrees.

### 5.6.1 Estimating the size of neighborhood

Algorithm NEIGHBORHOOD-SIZE takes as input two disjoint subsets  $L, R \subseteq V$  and returns a  $(1 \pm \epsilon)$ -approximation of the size of neighborhood of  $L$  in  $R$ , i.e.,  $|\Gamma(L) \cap R|$  using  $\text{poly}(1/\epsilon, \log n)$  BIS queries. We overview the analysis of this algorithm here, before presenting the details in section 5.6.1.1.

The main idea is to sample subsets of vertices in  $R$  (denoted  $\widehat{R}_1, \widehat{R}_2, \dots$ ) with exponentially decreasing probability values  $1/2, 1/4, 1/8, \dots$ . When the sampling rate  $1/2^i$  falls below  $1/|\Gamma(L) \cap R|$ , we expect  $L$  to no longer have any neighbors in  $\widehat{R}_i$  with

good probability. In particular, we can return the inverse of the smallest probability  $1/2^i$  for which  $\mathcal{BIS}(L, \widehat{R}_i) = '1'$ , as a coarse estimate for  $|\Gamma(L) \cap R|$ .

To boost the accuracy of this estimate, we repeat the process  $T = O(\epsilon^{-2} \log(\delta^{-1} \cdot \log n))$  times, and at each sampling rate count the number of times the BIS query  $\mathcal{BIS}(L, \widehat{R}_i)$  returns '1'. This count is denoted  $\text{count}(i)$  in Algorithm 18, and its expectation can be written in closed form as  $\mathbf{E}[\text{count}(i)] = T \cdot (1 - 1/2^i)^{|\Gamma(L) \cap R|}$ . Suppose  $2^{\widehat{i}} \leq |\Gamma(L) \cap R| < 2^{\widehat{i}+1}$ , then,  $\mathbf{E}[\text{count}(\widehat{i})] = \Theta(T)$ . Via a standard Chernoff bound, it will be approximated to  $(1 \pm \epsilon)$  error with high probability by  $\text{count}(\widehat{i})$ . Thus, we can compute an accurate estimate of the neighborhood size by inverting our estimate of  $\mathbf{E}[\text{count}(\widehat{i})]$ , as  $\log_{(1-1/2^{\widehat{i}})}(\text{count}(\widehat{i})/T)$ . We identify the appropriate  $\widehat{i}$  in line 12 of Algorithm 18, and compute the corresponding estimate in lines 13-14. There is one edge case handled in line 13: if  $|\Gamma(L) \cap R| = 1$  we will have  $\widehat{i} = 0$ , and  $\text{count}(\widehat{i}) = 0$ . The final error bound for Algorithm 18 is stated below.

---

**Algorithm 18** NEIGHBORHOOD-SIZE: Estimating the neighborhood size of  $L$  in  $R$

---

**Input:**  $L, R \subseteq V$ , approximation error  $\epsilon$ , failure probability  $\delta$ .

**Output:**  $\eta_{\text{est}}(L)$  as an estimate of  $|\Gamma(L) \cap R|$ .

```

1: Initialize  $\eta_{\text{est}}(L) \leftarrow 0$ .
2: for  $i = 0, 1, \dots, \log_2 n$  do
3:    $\text{count}(i) \leftarrow 0$ .
4:   for  $t = 1, 2, \dots, T = 2e^8 \ln(\log n / \delta) \cdot \epsilon^{-2}$  do
5:      $\widehat{R}_i^t \leftarrow \{u \in R \mid u \text{ is included independently with probability } 1/2^i\}$ .
6:      $\text{count}(i) = \text{count}(i) + \mathcal{BIS}(L, \widehat{R}_i^t)$ 
7:   end for
8: end for
9: if  $\text{count}(0) = T$  then
10:   return  $\eta_{\text{est}}(L) = 0$ .
11: else
12:   Set  $\widehat{i} \leftarrow \max \left\{ i \mid \frac{\text{count}(i)}{T} < \frac{(1-\epsilon)}{2e^2} \right\}$ .
13:   if  $\widehat{i} = 0$  then return  $\eta_{\text{est}}(L) = 1$ .
14:   else return  $\eta_{\text{est}}(L) = \log_{(1-1/2^{\widehat{i}})}(\text{count}(\widehat{i})/T)$ .
15:   end if
16: end if

```

---

### 5.6.1.1 Approximation Guarantees of Algorithm NEIGHBORHOOD-SIZE

For any  $i \in \{0, 1, \dots, \log_2 n\}$  let  $\widehat{R}_i$  denote a set constructed by sampling vertices of  $R$  with probability  $1/2^i$ . In Algorithm 18, we construct  $T = O(\epsilon^{-2} \log(\delta^{-1} \cdot \log n))$  such sets, each denoted by  $\widehat{R}_i^t \forall t \in [T]$ . Let  $\text{count}(i) = \sum_{t=1}^T \mathcal{BIS}(L, \widehat{R}_i^t)$  denotes the number of times the BIS query  $\mathcal{BIS}(L, \widehat{R}_i^t)$  returns ‘1’. For any  $t \in [T]$ , we define:

$$p(i) = \Pr \left[ \mathcal{BIS}(L, \widehat{R}_i^t) = \text{‘1’} \right] = \Pr \left[ \Gamma(L) \cap \widehat{R}_i^t = \phi \right] \text{ and } \widehat{p}(i) = \frac{\text{count}(i)}{T}.$$

Suppose  $L$  satisfies:

$$2^{i^*} \leq |\Gamma(L) \cap R| < 2^{i^*+1} \text{ for some } i^* \in \{0, 1, \dots, \log_2 n\}.$$

**Claim 5.6.1.** *We have the following bounds:*

$$p(i^*) \geq \frac{1}{2e^2}, \quad p(i^* - 2) > \frac{1}{2e^8}, \text{ and } p(i^* - 2) \leq \frac{1}{e^4}.$$

*Proof.*

$$\begin{aligned} p(i) &= \Pr \left[ \Gamma(L) \cap \widehat{R}_i^t = \phi \right] = \Pr[u \notin \widehat{R}_i^t \forall u \in R \cap \Gamma(L)] = \prod_{u \in R \cap \Gamma(L)} \Pr[u \notin \widehat{R}_i^t] \\ &= \left( 1 - \frac{1}{2^i} \right)^{|\Gamma(L) \cap R|}. \end{aligned}$$

We can lower bound  $p(i^*)$  by

$$\begin{aligned} p(i^*) &= \left( 1 - \frac{1}{2^{i^*}} \right)^{|\Gamma(L) \cap R|} \geq \left( 1 - \frac{1}{2^{i^*}} \right)^{2^{i^*+1}} \geq e^{-2} \left( 1 - \frac{2^{i^*+1}}{2^{2i^*}} \right) \text{ (using inequality 5.5.3)} \\ &\geq \frac{1}{2e^2} \quad \text{for } i^* \geq 2. \\ p(i^*) &\geq \frac{1}{8} \quad \text{for } i^* = 1. \end{aligned}$$

If  $i = i^* - 2$ , we have:

$$p(i) = \left(1 - \frac{1}{2^i}\right)^{|\Gamma(L) \cap R|} \leq \left(1 - \frac{1}{2^i}\right)^{2^{i^*}} \leq e^{-2^{i^*-i}} = \frac{1}{e^4}$$

$$\begin{aligned} p(i) &= \left(1 - \frac{1}{2^i}\right)^{|\Gamma(L) \cap R|} \geq \left(1 - \frac{1}{2^{i^*-2}}\right)^{|\Gamma(L) \cap R|} > \left(1 - \frac{1}{2^{i^*-2}}\right)^{2^{i^*+1}} \\ &\geq e^{-8} \left(1 - \frac{2^{i^*+1}}{2^{2^{i^*-4}}}\right) \\ &\geq \frac{1}{2e^8} \text{ for } i^* \geq 5, \end{aligned}$$

where we used the inequality 5.5.3, for the penultimate inequality.

For  $i^* \leq 4$ , the inequality is satisfied. So, we have:  $p(i^* - 2) > 1/2e^8$ .

□

**Claim 5.6.2.** *For sufficiently small  $\epsilon$ , with probability at least  $1 - \delta$ , we have:*

$$\text{count}(i) \geq \frac{1 - \epsilon}{2e^2} \cdot T \quad \forall i \geq i^* \text{ and } \text{count}(i^* - 2) < \frac{1 - \epsilon}{2e^2} \cdot T.$$

*Proof.* As  $\text{count}(i) = \sum_{t=1}^T \mathcal{BLS}(L, \widehat{R}_i^t)$ , we have:  $\mathbf{E}[\text{count}(i)] = T \cdot p(i)$ . Using Claim 5.6.1, we have:

$$T = 2e^8 \ln(\log n / \delta) \cdot \epsilon^{-2} \geq \frac{4 \ln(\log n / \delta) \cdot \epsilon^{-2}}{p(i^* - 2)} \geq \frac{4 \ln(\log n / \delta) \cdot \epsilon^{-2}}{p(i^*)}, \text{ as } p(i^*) \geq p(i^* - 2).$$

Suppose  $i \in \{i^*, i^* - 2\}$ . Then, we have:

$$\begin{aligned} \Pr[|\widehat{p}(i) - p(i)| \geq \epsilon \cdot p(i)] &= \Pr[|T \cdot \widehat{p}(i) - T \cdot p(i)| \geq T \cdot \epsilon \cdot p(i)] \\ &= \Pr[|\text{count}(i) - \mathbf{E}[\text{count}(i)]| \geq \epsilon \mathbf{E}[\text{count}(i)]] \\ &\leq 2 \exp\left(-\frac{\epsilon^2 T p(i)}{2}\right) \leq \frac{\delta}{\log n} \quad (\text{Using Chernoff bound}). \end{aligned}$$

Using Claim 5.6.1, we get:

$$\begin{aligned} \text{count}(i^*) &\geq (1 - \epsilon) \cdot T \cdot p(i^*) \geq \frac{(1 - \epsilon)}{2e^2} \cdot T \\ \text{count}(i^* - 2) &< (1 + \epsilon) \cdot T \cdot p(i) \leq \frac{(1 + \epsilon)}{e^4} \cdot T \\ \implies \text{count}(i^* - 2) &< \frac{(1 + \epsilon)}{e^4} \cdot T \leq \frac{(1 - \epsilon)}{2e^2} \cdot T, \text{ when } \epsilon \leq \frac{e^2/2 - 1}{e^2/2 + 1}. \end{aligned}$$

From the definition, we can observe that  $p(i) \geq p(i^*) \forall i \geq i^*$ . So, the concentration around expected values for  $\text{count}(i)$  obtained using Chernoff bounds will hold for all  $i \geq i^*$ . Using union bound on at most  $\log n$  sampling levels, we have, with probability  $1 - \delta$ :

$$\text{count}(i^* - 2) < \frac{(1 - \epsilon)}{2e^2} \cdot T \text{ and } \text{count}(i) \geq \frac{(1 - \epsilon)}{2e^2} \cdot T \forall i \geq i^*.$$

□

**Lemma 5.6.3.** *Algorithm 18 uses  $O(\epsilon^{-2} \log n \log(\delta^{-1} \cdot \log n))$  BIS queries and returns an estimate  $\eta_{\text{est}}(L)$  of  $|\Gamma(L) \cap R|$  such that with probability at least  $1 - \delta$ ,*

$$(1 - \epsilon) \cdot |\Gamma(L) \cap R| \leq \eta_{\text{est}}(L) \leq (1 + \epsilon) \cdot |\Gamma(L) \cap R|.$$

*Proof.* If  $|\Gamma(L) \cap R| = 0$ , then,  $\text{count}(i) = T$  for every  $i \in \{0, 1, 2, \dots, \log n\}$ . So,  $\hat{i} = 0$ , as none of the values  $\text{count}(i)$ , for any  $i$  will be below the threshold value of  $(1 - \epsilon)T/2e^2$ . So, the estimate  $\eta_{\text{est}}(L) = 0$  returned is exact.

Suppose  $|\Gamma(L) \cap R| = 1$ . When we sample with probability  $1/2^i$  when  $i = 0$ , we obtain  $\hat{R}_i^t = R$ , for every  $t \in [T]$ . As  $\mathcal{BIS}(L, R) = '1'$ , we have  $\text{count}(i) = 0$ , and our estimate  $\eta_{\text{est}}(L) = 1$  is exact. For the remainder of the proof, we assume  $i^* \geq 1$ .

From Algorithm 18, we define  $\hat{i} = \arg \max\{i \mid \text{count}(i) < (1 - \epsilon)T/2e^2\} + 1$ . From Claim 5.6.2, this implies:  $\hat{i} \geq i^* - 2$ . Therefore, with probability at least  $1 - \delta$ , we have:

$$i^* - 2 \leq \hat{i} \leq i^* - 1.$$

Now, we argue that  $\eta_{\text{est}}(L)$  defined by:

$$\eta_{\text{est}}(L) := \log_{(1-1/2^{\hat{i}})} \widehat{p}(\hat{i}) \text{ obtains a } (1 \pm \epsilon) \text{ approximation for } |\Gamma(L) \cap R|.$$

$$\begin{aligned} (1 - \epsilon)p(\hat{i}) &\leq \widehat{p}(\hat{i}) \leq (1 + \epsilon)p(\hat{i}) \\ \log_{(1-1/2^{\hat{i}})}(1 - \epsilon) \cdot p(\hat{i}) &\leq \log_{(1-1/2^{\hat{i}})} \widehat{p}(\hat{i}) \leq \log_{(1-1/2^{\hat{i}})}(1 + \epsilon) \cdot p(\hat{i}) \\ |\Gamma(L) \cap R| + \log_{(1-1/2^{\hat{i}})}(1 - \epsilon) &\leq \log_{(1-1/2^{\hat{i}})} \widehat{p}(\hat{i}) \leq |\Gamma(L) \cap R| + \log_{(1-1/2^{\hat{i}})}(1 + \epsilon) \\ \Rightarrow |\Gamma(L) \cap R| - 2^{\hat{i}} \cdot \epsilon &\leq \log_{(1-1/2^{\hat{i}})} \widehat{p}(\hat{i}) \leq |\Gamma(L) \cap R| + 2^{\hat{i}} \cdot \epsilon \\ \Rightarrow |\Gamma(L) \cap R| - 2^{i^*-2} \cdot \epsilon &\leq \log_{(1-1/2^{\hat{i}})} \widehat{p}(\hat{i}) \leq |\Gamma(L) \cap R| + 2^{i^*-1} \cdot \epsilon \\ \Rightarrow (1 - \epsilon/4) \cdot |\Gamma(L) \cap R| &\leq \log_{(1-1/2^{\hat{i}})} \widehat{p}(\hat{i}) \leq (1 + \epsilon/2) \cdot |\Gamma(L) \cap R|. \end{aligned}$$

Therefore,  $\eta_{\text{est}}(L) := \log_{(1-1/2^{\hat{i}})} \widehat{p}(\hat{i})$  is a  $(1 \pm \epsilon)$ -relative error approximation of  $|\Gamma(L) \cap R|$ .

The total number of BIS queries used by Algorithm 18 is:

$$O(\log n \cdot T) = O(\epsilon^{-2} \log n \log(\log n / \delta)).$$

□

### 5.6.2 Finding good approximation for degrees of vertices

We now describe how to use the NEIGHBORHOOD-SIZE algorithm to estimate the degrees of all vertices in a given subset  $S \subseteq V$  up to additive error depending on the total degree of  $S$ . Our approach is inspired by the count-min sketch algorithm [44]. We randomly partition  $S$  into subsets  $S^1, S^2, \dots, S^\lambda$  where  $\lambda = O(\epsilon^{-3} \log^2 n)$ . The choice of the parameter  $\lambda$  is based on the analysis in Section 5.6.3. For each  $S^i$ , we estimate the size of the neighborhood of  $S^i$  in  $V \setminus S^i$  using NEIGHBORHOOD-SIZE. We then return this neighborhood size estimate as the degree estimate for all vertices in  $S^i$ . For  $v \in S^i$ ,  $|\Gamma(S^i) \cap V \setminus S^i|$  is nearly an overestimate for  $d(v)$ , as long as  $v$  has

few neighbors in  $S^i$ , which it will with high probability. Additionally, it is not too large an overestimate – we can observe that  $|\Gamma(S^i) \cap V \setminus S^i| - d(v) \leq d(S^i \setminus v)$ . I.e., the error in the overestimate is at most the total degree of the other nodes in  $S^i$ . In expectation, this error is at most  $\frac{d(S)}{\lambda} = O\left(d(S) \cdot \frac{\epsilon^3}{\log^2 n}\right)$  due to our random choice of  $S^i$ .

As in the count-min sketch algorithm, to obtain high probability estimates, we repeat the process  $T = O(\log n)$  times and assign the minimum among the neighborhood estimates as the degree estimate of  $d(v)$ . The full approach is given in Algorithm 19 (ESTIMATE-DEGREE) and the error bound in the Lemma 5.6.4 below.

**Lemma 5.6.4.** *Suppose  $S \subseteq V$ . Then, Algorithm 19 uses  $O(\epsilon^{-5} \log^4 n \log(\epsilon^{-1} \log n))$  BIS queries and with probability  $1 - O(1/\log n)$ , returns degree estimates  $\hat{d}(v)$  for every vertex  $v \in S$  satisfying:*

$$d(v)(1 - \epsilon) \leq \hat{d}(v) \leq d(v) + \frac{\epsilon^3}{\log^2 n} \cdot d(S).$$

---

**Algorithm 19** ESTIMATE-DEGREE: Obtain additive approximate degree estimates

---

**Input:**  $S$  is a subset of  $V$ ,  $\epsilon$  is approximation error.

**Output:** Degree estimates of vertices in  $S$ .

- 1: Scale  $\epsilon \leftarrow \epsilon/3$  and initialize  $\hat{d}(v) \leftarrow n$  for every  $v \in S$ .
  - 2: **for**  $t$  in  $\{1, 2, \dots, O(\log n)\}$  **do**
  - 3:     Consider a random partitioning of  $S$  into  $S^{t1}, S^{t2}, \dots, S^{t\lambda}$  where  $\lambda = O(\epsilon^{-3} \log^2 n)$ .
  - 4:     **for** every partition  $S^{ta}$  where  $a \in [\lambda]$  **do**
  - 5:          $\eta_{\text{est}}(S^{ta}) \leftarrow \text{NEIGHBORHOOD-SIZE}(S^{ta}, V \setminus S^{ta}, \epsilon, \delta)$ , where  $\delta = O(1/\log^4 n)$ .
  - 6:          $\hat{d}(v) \leftarrow \min\{\hat{d}(v), \eta_{\text{est}}(S^{ta})\} \forall v \in S^{ta}$ .
  - 7:     **end for**
  - 8: **end for**
  - 9: **return**  $\hat{d}(v)$  for every  $v \in S$ .
- 

### 5.6.2.1 Proof of Lemma 5.6.4

Consider a vertex  $v \in S$ . It is easy to observe that in any partition  $S^{ta}$  containing  $v$ , where  $t \in [T]$  and  $a \in [\lambda]$ , the degree of  $v$  outside the partition (denoted by

$d(v, V \setminus S^{ta})$  is upper bounded by the total size of the neighborhood of  $S^{ta}$  (denoted by  $|\Gamma(S^{ta}) \cap V \setminus S^{ta}|$ ) which is upper bounded by the total degree of vertices present in the partition (denoted by  $d(S^{ta})$ ). Similar to the analysis of count-min sketch, a simple, yet important observation is that the total degree of the partition except for vertex  $v$ , i.e.,  $d(S^{ta} \setminus \{v\})$  is less than  $c \cdot d(S)/\lambda$  for some constant  $c$  and results in the additive approximation factor of  $O(d(S)/\lambda)$ . Now, we present the proof of Lemma 5.6.4:

*Proof.* Given  $S \subseteq V$ . Consider a vertex  $v \in S^{ta}$  for some  $a \in \{1, 2, \dots, \lambda\}$  and  $t \in \{1, 2, \dots, T\}$ . For each call to neighborhood size estimation, we set the failure probability to be  $\delta = O(\epsilon^3/\log^4 n)$ . From Lemma 5.6.3, we have with probability  $1 - \delta$ :

$$(1 - \epsilon)|\Gamma(S^{ta}) \cap (V \setminus S^{ta})| \leq \eta_{\text{est}}(S^{ta}) \leq (1 + \epsilon)|\Gamma(S^{ta}) \cap (V \setminus S^{ta})|$$

We can observe that  $d(v, V \setminus S^{ta}) = |\Gamma(v) \cap (V \setminus S^{ta})| \leq |\Gamma(S^{ta}) \cap (V \setminus S^{ta})|$ .

Therefore:

$$d(v, V \setminus S^{ta}) \leq \frac{\eta_{\text{est}}(S^{ta})}{1 - \epsilon}.$$

Consider the following:

$$\mathbf{E}[d(v, S^{ta})] = \mathbf{E} \left[ \sum_{u \in V} \mathbb{1}\{u \in \Gamma(v) \cap S^{ta}\} \right] = \frac{d(v)}{\lambda} = \frac{d(v)\epsilon^3}{c \log^2 n} \leq \epsilon d(v), \text{ as } c > 1.$$

From Markov's inequality, with probability at least  $1/2$ , we have:  $d(v, S^{ta}) \leq 2\epsilon d(v)$ .

Combining the above, with probability  $1/2 - \delta$ , we have:

$$\frac{\eta_{\text{est}}(S^{ta})}{1 - \epsilon} \geq d(v, V \setminus S^{ta}) = d(v) - d(v, S^{ta}) \geq d(v)(1 - 2\epsilon),$$

$$\implies \eta_{\text{est}}(S^{ta}) \geq (1 - 3\epsilon)d(v).$$

$$\begin{aligned}
\mathbf{E} [d(S^{ta} \setminus \{v\})] &= \mathbf{E} \left[ \sum_{u \in S \setminus \{v\}} d(u) \mathbb{1}\{u \in S^{ta}\} \right] = \sum_{u \in S \setminus \{v\}} d(u) \Pr[u \in S^{ta}] \\
&= \sum_{u \in S \setminus \{v\}} d(u) \cdot \frac{1}{\lambda} \\
&\leq d(S) \cdot \frac{\epsilon^3}{c \log^2 n}, \text{ for some constant } c > 1 \\
&\leq d(S) \cdot \frac{\epsilon^3}{\log^2 n}.
\end{aligned}$$

From Markov's inequality, it follows that:

$$\Pr \left[ d(S^{ta} \setminus \{v\}) \geq d(S) \cdot \frac{2\epsilon^3}{\log^2 n} \right] \leq \frac{\mathbf{E}[d(S^{ta} \setminus \{v\})]}{d(S) \cdot \frac{2\epsilon^3}{\log^2 n}} = \frac{1}{2}.$$

So, with probability at least  $1/2$ , we have:

$$\begin{aligned}
d(S^{ta}) &= d(v) + d(S^{ta} \setminus \{v\}) \\
&\leq d(v) + d(S) \cdot \frac{2\epsilon^3}{\log^2 n} \\
\implies \eta_{\text{est}}(S^{ta}) &\leq |\Gamma(S^{ta}) \cap (V \setminus S^{ta})| \leq d(S^{ta}, V \setminus S^{ta}) \leq d(S^{ta}) \\
\eta_{\text{est}}(S^{ta}) &\leq d(v) + d(S) \cdot \frac{2\epsilon^3}{\log^2 n}.
\end{aligned}$$

Using union bound on all possible sets  $S^{ta}$  for all  $t \in [T]$  and  $a \in [\lambda]$ , with probability at least  $1 - T \cdot \lambda \cdot \delta \geq 1 - 1/2 \log n$ , the neighborhood estimates are  $(1 \pm \epsilon)$ -relative approximations. By taking minimum of all the  $T = O(\log n)$  estimates, we argue that  $\widehat{d}(v)$  is a good approximation of  $d(v)$ . We take minimum of all the  $T$  estimates containing  $v$  and obtain the final degree estimate, given by:

$$\widehat{d}(v) = \min_{t \in T} \eta_{\text{est}}(S^{ta}).$$

We observe that:

$$\begin{aligned}
\Pr \left[ \widehat{d}(v) < (1 - 3\epsilon)d(v) \right] &= \Pr \left[ \left\{ \min_{t \in T} \eta_{\text{est}}(S^{ta}) \right\} < (1 - 3\epsilon)d(v) \right] \\
&= \prod_{t \in T} \Pr \left[ \eta_{\text{est}}(S^{ta}) < (1 - 3\epsilon)d(v) \right] \\
&\leq \left( \frac{1}{2} \right)^T \leq \frac{1}{2n^4}, \text{ and} \\
\Pr \left[ \widehat{d}(v) > d(v) + d(S) \cdot \frac{2\epsilon^3}{\log^2 n} \right] &= \Pr \left[ \left\{ \min_{t \in T} \eta_{\text{est}}(S^{ta}) \right\} > d(v) + d(S) \cdot \frac{2\epsilon^3}{\log^2 n} \right] \\
&= \prod_{t \in T} \Pr \left[ \eta_{\text{est}}(S^{ta}) > d(v) + d(S) \cdot \frac{2\epsilon^3}{\log^2 n} \right] \\
&\leq \left( \frac{1}{2} \right)^T \leq \frac{1}{2n^4}.
\end{aligned}$$

By taking a union bound on all the vertices in  $S$  and the event that neighborhood estimates are accurate, the total failure probability is at most  $1/2 \log n + 1/n^3 \leq 1/\log n$ . Therefore, for every vertex  $v \in S^{ta}$ , we have with probability at least  $1 - 1/\log n$ :

$$(1 - 3\epsilon)d(v) \leq \eta_{\text{est}}(S^{ta}) \leq d(v) + d(S) \cdot \frac{2\epsilon^3}{\log^2 n}.$$

Set  $\epsilon = \epsilon/2^{1/3}$  to appropriately scale the value of  $\epsilon$  for the final guarantees. Algorithm 19 uses  $O(\epsilon^{-3} \log^2 n \cdot T)$  many calls to the sub-routine NEIGHBORHOOD-SIZE, i.e., Algorithm 18. From Lemma 5.6.3, we know that Algorithm 18 uses  $O(\epsilon^{-2} \log n \log(\delta^{-1} \log n))$  BIS queries, where we set  $\delta = O(\epsilon^3/\log^4 n)$ . Therefore, the query complexity of Algorithm 19 is  $O(\epsilon^{-5} \log^4 n \log(\epsilon^{-1} \log n))$ .

□

### 5.6.3 Edge Estimation

In this section, we describe the algorithm EDGE-ESTIMATOR that obtains a  $(1 \pm \epsilon)$ -approximation for the number of edges  $m$ . The constants used  $c_1, c_2$  satisfy  $c_1 \leq c_2/10$

and  $c_2 \geq 50$ , and we do not explicitly mention them for the sake of brevity. Missing details are presented in the full version.

**Our Approach.** A naive strategy to estimate the number of edges (denoted by  $m$ ) is to sample roughly  $\tilde{O}(\epsilon^{-2})$  nodes uniformly, and estimate  $m$  given the degrees of the sampled nodes. However, the variance of such an estimator depends on the maximum degree, which could be as high as  $O(n)$ . To fix this issue, we sample vertices at different rates. Our sampling rates are given by the sequence  $1/\gamma^j$  where  $\gamma > 1$  is a constant,  $j \in \{0, 1, \dots, \log n\}$ . We use the term  $j^{\text{th}}$  level to refer to the sampling rate  $\gamma^{-j}$ . It is easy to observe that when a vertex  $v$  is sampled at rate  $\tilde{O}(\epsilon^{-2}d(v)/m)$ , its contribution is  $\tilde{O}(\epsilon^2m)$ . In other words, if  $d(v) \approx \epsilon^2m/\gamma^j$ , for some sampling level  $j$ , we can use it in our estimator. However, there are three main challenges in implementing this approach which we detail below.

**Approximate degrees.** Algorithm ESTIMATE-DEGREE returns degree estimates that are *approximate* with an additive approximation error of  $\tilde{O}(\epsilon^3m/\gamma^j)$  at sampling level  $j$ . To include a vertex  $v$ , we have to ensure that this error term is small and given by  $O(\epsilon d(v))$ . When  $d(v) = \tilde{\Omega}(\epsilon^2m/\gamma^j)$ , the returned degree estimate  $\hat{d}(v)$  will be a  $(1 \pm \epsilon)$ -approximation to the actual degree  $d(v)$ . Observe that this corresponds to the threshold we mentioned earlier. Therefore, our goal is to identify all vertices at every level  $j$  that pass the threshold of  $\tilde{\Omega}(\epsilon^2m/\gamma^j)$ . When that happens, we say that the vertex  $v$  has been recovered at level  $j$  and can be safely included in our estimator.

**Knowledge of  $m$ .** As we do not know the value of  $m$ , we start with an  $O(\log^2 n)$ -relative error approximate estimate, obtained by the Algorithm COARSEESTIMATOR in Beame et al. [22]. We repeatedly refine the approximate estimate using Algorithm REFINE-ESTIMATE, until we get a  $(1 \pm \epsilon)$ -relative error approximation of  $m$ . Each *refinement* improves the approximation factor from the previous stage by a multiplicative factor of  $\epsilon$ . We note that each refinement does not require any additional BIS queries and uses the approximate degree estimates obtained previously.

**Boundary Vertices.** It is possible that some vertices have degrees close to the threshold values at each sampling level. We denote such vertices  $V_{\text{boundary}}$  and refer to them as *boundary vertices*. For such boundary vertices, as we use approximate degree estimates, they might be recovered at a level different from its true level (defined with respect to exact degrees). Such a scenario could potentially affect the contribution of the recovered vertex in our estimator by an additional multiplicative factor dependent on  $\gamma$  and the difference between recovered level and true level. As a result, our estimator might not be a  $(1 \pm \epsilon)$ -relative error approximation anymore. We get around this limitation by dividing the region between any two consecutive levels into  $B$  buckets and shifting the boundaries of all the levels by a random shift selected uniformly from the first  $B$  buckets. We account for this by changing the sampling rates to  $\gamma^{-\mu(j)}$  where  $\mu(j)$  encodes the random shift.

With the random shift of the level boundaries, we ensure that every vertex will lie close to the boundary with probability at most  $\epsilon$ . Moreover, we argue that every boundary vertex is recovered at its true level or level adjacent to its true level. Therefore, the total contribution of  $V_{\text{boundary}}$  to our edge estimator is  $O(\epsilon m)$ .

### 5.6.3.1 Overview of Algorithm EDGE-ESTIMATOR

**Random Boundary Shift.** Let  $\epsilon$  denote the approximation parameter,  $B = 2/\epsilon$  denote the total number of buckets between two consecutive levels and  $\gamma = 1/(1 - \epsilon)$  the probability of sampling parameter. The region between two consecutive levels is divided into  $B$  buckets with the boundaries of buckets proportional to the values given by  $\{[1/\gamma^B, 1/\gamma^{B-1}), \dots, [1/\gamma^2, 1/\gamma), [1/\gamma, 1)\}$ . We select a random integer offset for shifting our levels, denoted by  $s$ , which is selected uniformly at random from  $[0, B)$ . Now, the level boundaries are located at values proportional to  $\gamma^{-\mu(j)}$  where  $\mu(j) = j \cdot B - s$  and  $0 \leq j \leq L$ . Observe that the number of sampling levels is given by  $L = \frac{1}{B} \cdot \log_{\gamma} n + 1 \leq \frac{1}{2} \log n + 1$ .

---

**Algorithm 20** EDGE-ESTIMATOR: Non-adaptive algorithm for estimating edges
 

---

**Input:**  $V$  set of  $n$  vertices and  $\epsilon > 0$  error parameter.

**Output:** Estimate  $\widehat{m}$  of number of edges in  $G$ .

- 1: Scale  $\epsilon \leftarrow \frac{\epsilon}{600 \log_{1/\epsilon} \log n}$  and initialize  $\gamma \leftarrow 1/(1 - \epsilon)$  and  $B \leftarrow 2/\epsilon$ .
- 2: Let  $s$  be an integer selected uniformly at random from the interval  $[0, B)$ .
- 3: Let  $\mu(j) \leftarrow -s + j \cdot B$  for every integer  $j$  in the interval  $[0, \frac{1}{B} \cdot \log_\gamma n + 1]$ .
- 4: Initialize  $S_0 \leftarrow V$  and construct  $S_1$  by sampling vertices in  $S_0$  with probability  $1/\gamma^{\mu(1)}$ .
- 5: Construct  $S_2 \supseteq \dots \supseteq S_L$  for  $L = \frac{1}{B} \cdot \log_\gamma n$  where each  $S_j$  is obtained by sampling vertices in  $S_{j-1} \forall j \geq 2$ , independently with probability  $1/\gamma^B$ .
- 6: **for**  $j = 0, 1, \dots, L$  **do**
- 7:     Run ESTIMATE-DEGREE ( $S_j$ ) to obtain the estimates  $\widehat{d}_j(v)$  for all  $v \in S_j$  satisfying:

$$(1 - \epsilon)d(v) \leq \widehat{d}_j(v) \leq d(v) + \frac{c_1 \epsilon^3 \cdot m}{\log n \cdot \gamma^{\mu(j)}}.$$

- 8: **end for**
  - 9: Let  $\bar{m}_0$  be the  $O(\log n)$ -approximate estimate from the Algorithm COARSEESTIMATOR in Beame et al. [22] on a random partition of  $V$ .
  - 10: Set  $\bar{m}_0 \leftarrow \max\{2, 16 \log n \cdot \bar{m}_0\}$ , so that we have  $m \leq \bar{m}_0 \leq (64 \log^2 n) \cdot m$ .
  - 11: **for**  $t = 1, 2, \dots, T = 2 \log_{1/\epsilon} \log n$  **do**
  - 12:      $\bar{m}_t$  is assigned the output of REFINE-ESTIMATE that takes as input approximate degree values  $\widehat{d}_j(v) \forall v \in S_j \forall j \in [L]$ , the previous estimate  $\bar{m}_{t-1}$  and the iteration  $t$ .
  - 13: **end for**
  - 14: **return**  $\widehat{m} \leftarrow \bar{m}_T$ .
- 

In Algorithm EDGE-ESTIMATOR, we construct sets  $V = S_0 \supseteq S_1 \supseteq \dots \supseteq S_L$  where a set  $S_j$  (for all  $j \geq 2$ ) is obtained by sampling vertices in  $S_{j-1}$  with probability  $1/\gamma^B$ . The set  $S_1$  is obtained by sampling vertices in  $V$  with probability  $1/\gamma^{-s+B}$ . Our sampling scheme results in each vertex being included in a set  $S_j$  with probability  $1/\gamma^{\mu(j)}$ . We can easily show that with constant probability,  $d(S_j) = O(m \log n / \gamma^{\mu(j)})$ , for all  $j$ . Using Algorithm 19, we obtain approximate degree estimates of vertices in  $S_j$  for every sampling level  $j \leq L$  with an approximation error of  $O(\epsilon^3 / \log^2 n \cdot d(S_j)) = O(m \epsilon^3 / \gamma^{\mu(j)} \log n)$ . By starting with a bad estimate  $\bar{m}_0$  for the total number of edges  $m$  and initialized to a  $O(\log^2 n)$ -approximate estimate, we refine it to obtain an improved estimate  $\bar{m}_1$ . We repeat this process  $T = 2 \log_{1/\epsilon} \log n$  times, such that

the estimate  $\bar{m}_{t-1}$  is used to construct an improved estimate  $\bar{m}_t$ . Finally, we return the estimate  $\bar{m}_T$  as our final estimate for  $m$ .

---

**Algorithm 21** REFINED-ESTIMATE: Refines the current estimate of number of edges

---

**Input:**  $\bar{m}$  satisfying  $m \leq \bar{m} \leq m(1 + \alpha)$ , approximate degree values  $\hat{d}_j(v) \forall v \in S_j \forall j \in [L]$  obtained using Algorithm 20,  $\bar{m}_0$ , and iteration  $t$ .  
**Output:** Estimate  $\hat{m}$  satisfying  $m \leq \hat{m} \leq m(1 + \epsilon \cdot \alpha)$  of number of edges in  $G$ .

- 1: Initialize  $\hat{m} \leftarrow 0$ .
- 2: Initialize  $r(v) \leftarrow 0$  for all  $v$  (indicator if  $v$  has been recovered yet).
- 3: **for**  $j = 0, 1, \dots, L$  **do**
- 4:     **for**  $v \in S_j$  **do**
- 5:         **if**  $r(v) = 0$  and  $\hat{d}_j(v) \geq \frac{\bar{m}}{\gamma^{\mu(j)}} \cdot \frac{c_2 \epsilon^2}{\log n}$  **then**
- 6:              $\hat{m} \leftarrow \hat{m} + \gamma^{\mu(j)} \cdot \hat{d}(v)$
- 7:              $\hat{\ell}(v) \leftarrow j$  and  $r(v) \leftarrow 1$ .
- 8:         **end if**
- 9:     **end for**
- 10: **end for**
- 11: **if**  $t < T = 2 \log_{1/\epsilon} \log n$  **then**
- 12:      $\hat{m} = \hat{m}/2 + (\epsilon \log \log n)^t \bar{m}_0$ .      $\triangleright$  *We normalize  $\hat{m}$  so that we have  $\hat{m} \geq m$ .*
- 13: **else**
- 14:      $\hat{m} = \hat{m}/2$ .
- 15: **end if**
- 16: **return**  $\hat{m}$ .

---

**Overview of Algorithm REFINED-ESTIMATE.** Suppose we are given an initial estimate  $\bar{m}$  satisfying  $m \leq \bar{m} \leq (1 + \alpha)m$  for some unknown approximation factor  $\alpha$  satisfying  $\epsilon \leq \alpha \leq \binom{n}{2}$ . We set the threshold value for recovering a vertex at a level  $j$  as  $\frac{\bar{m}}{\gamma^{\mu(j)}} \cdot \frac{c_2 \epsilon^2}{\log n}$  where  $c_2$  is a constant. So, when a vertex  $v$  with degree estimate  $\hat{d}_j(v)$  (obtained from Algorithm 20) satisfies  $\hat{d}_j(v) \geq \frac{\bar{m}}{\gamma^{\mu(j)}} \cdot \frac{c_2 \epsilon^2}{\log n}$ , we set the level of recovery  $\hat{\ell}(v) = j$  and recovered flag  $r(v) = 1$ . From construction, we can observe that once a vertex is recovered at a particular level it is not available to be recovered at higher level later. Our estimator is the summation of terms  $\gamma^{\mu(\hat{\ell}(v))} \cdot \hat{d}(v)$  for every  $v$  satisfying  $r(v) = 1$ . We normalize  $\hat{m}$  to ensure that the final estimate returned satisfies  $m \leq \hat{m}$  (see the full version for additional details).

Using Bernstein's inequality, we argue that in iteration  $t$ , we can improve the approximation factor of the previous estimate  $\bar{m}_{t-1}$  by a multiplicative factor of  $\epsilon$  in the new estimate  $\bar{m}_t$ . After  $T = O(\log_{1/\epsilon} \log n)$  iterations, the edge estimate will be a  $(1 \pm \epsilon)$ -relative error approximation satisfying:

**Theorem 5.6.5.** *Given a graph  $G$  with  $n$  nodes and  $m$  edges, there is an algorithm that makes  $O(\epsilon^{-5} \log^5 n \log^5(\log n) \log(\epsilon^{-1} \log n))$  non-adaptive BIS queries to  $G$  and returns an estimate  $\hat{m}$  satisfying:  $m(1-\epsilon) \leq \hat{m} \leq m(1+\epsilon)$ , with probability at least  $3/5$ .*

### 5.6.3.2 Proof of Theorem 5.6.5

First, we show that our degree estimates are calculated accurately at every level with constant probability of success.

**Claim 5.6.6.** *With probability  $3/4$ , for all levels  $j \in \{1, 2, \dots, L\}$ , we have:*

$$d(S_j) \leq \frac{8m \cdot L}{\gamma^{\mu(j)}}.$$

*Proof.* As every vertex is included in  $S_j$  with probability  $1/\gamma^{\mu(j)}$ , we get:

$$\mathbf{E}[d(S_j)] = \frac{\sum_{v \in V} d(v)}{\gamma^{\mu(j)}} = \frac{2m}{\gamma^{\mu(j)}}$$

Therefore, by Markov's Inequality,  $\Pr[d(S_j) \geq 8m \cdot L/\gamma^{\mu(j)}] \leq 1/(4L)$ . Taking a union bound over all the levels, with probability at least  $3/4$ ,

$$d(S_j) \leq 8m \cdot L/\gamma^{\mu(j)} \leq 8m \cdot \log n/\gamma^{\mu(j)} \text{ for every level } j \in [L].$$

□

Combining Claim 5.6.6 and Lemma 5.6.4, for sufficiently large  $n$ , we have:

**Corollary 5.6.7.** *The degree estimates returned by Algorithm 19 for each sampling level  $j \in [L]$ , satisfy the following with probability at least 0.70:*

$$(1 - \epsilon)d(v) \leq \widehat{d}_j(v) \leq d(v) + \frac{c_1 \epsilon^3 m}{\gamma^{\mu(j)} \log n} \quad \forall v \in S_j.$$

*Proof.* From Lemma 5.6.4, we have that, for every  $j \in [L]$ , with probability at least  $1 - O(1/\log n)$ , the approximate degree estimates satisfy:

$$(1 - \epsilon)d(v) \leq \widehat{d}_j(v) \leq d(v) + \frac{\epsilon^3}{\log^2 n} \cdot d(S_j) \quad \forall v \in S_j.$$

From Claim 5.6.6, we know that  $d(S_j) \leq 8m \cdot \log n / \gamma^{\mu(j)}$ , for every level  $j \in [L]$ , with probability at least 3/4. Combining both of them, we have the claim about the approximate degree estimates.

Using union bound, we have that the total failure probability is at most  $1/4 + O(L \cdot 1/\log n) \leq 0.30$ , as  $L = O(\log n)$ . Hence, the corollary.  $\square$

For each vertex  $v \in S$  for some subset  $S \subseteq V$ , we associate a level  $\ell(v)$  such that the *actual* degree of  $v$  is a large fraction of the total degree of  $S$ , i.e.,  $\ell(v)$  is the minimum  $j \in \{0, 1, \dots, L\}$  satisfying  $d(v) \geq \frac{\bar{m}}{\gamma^{\mu(j)}} \cdot \frac{c_2 \epsilon^2}{\log n}$ . The value  $\frac{\bar{m}}{\gamma^{\mu(j)}} \cdot \frac{c_2 \epsilon^2}{\log n}$  is called threshold for level  $\ell(v)$ , and it depends on the estimate  $\bar{m}$  for the number of edges  $m$ .

**Definition 5.6.8** (Actual Level). *For every vertex  $v \in V$ , we associate a level  $\ell(v)$  defined as*

$$\ell(v) = \arg \min_{j \in \{0, 1, \dots, L\}} d(v) \geq \frac{\bar{m}}{\gamma^{\mu(j)}} \cdot \frac{c_2 \epsilon^2}{\log n}.$$

The vertices that lie close to the threshold of a level and within a  $\gamma$ -multiplicative factor are called the boundary vertices and are defined as below:

**Definition 5.6.9** (Boundary vertices). *The vertices closer to the boundary are denoted by the set:*

$$V_{\text{boundary}} = \left\{ v \mid d(v) \in \left[ \frac{\bar{m} \cdot c_2 \epsilon^2}{\gamma^{\mu(\ell(v))} \log n}, \frac{\bar{m} \cdot c_2 \epsilon^2}{\gamma^{\mu(\ell(v))-1} \log n} \right) \text{ or } d(v) \in \left( \frac{\bar{m} \cdot c_2 \epsilon^2}{\gamma^{\mu(\ell(v)-1)+1} \log n}, \frac{\bar{m} \cdot c_2 \epsilon^2}{\gamma^{\mu(\ell(v)-1)} \log n} \right) \right\}.$$

**Claim 5.6.10.** *For any vertex  $v \in V$ , with probability  $1 - \epsilon$ , there is some  $j \in \{0, 1, \dots, L\}$  such that:*

$$\frac{\bar{m}}{\gamma^{\mu(j)-1}} \cdot \frac{c_2 \epsilon^2}{\log n} \leq d(v) \leq \frac{\bar{m}}{\gamma^{\mu(j-1)+1}} \cdot \frac{c_2 \epsilon^2}{\log n}.$$

*In other words,  $\Pr[v \in V_{\text{boundary}}] \leq \epsilon$ .*

*Proof.* For notational convenience, let  $\sigma = \frac{\bar{m} \cdot c_2 \epsilon^2}{\log n}$ . Note that for any  $v \in V$  there is some  $j$  such that  $\frac{\sigma}{\gamma^{\mu(j)}} \leq d(v) < \frac{\sigma}{\gamma^{\mu(j-1)}}$ . We claim that every such vertex will not lie close to the edges of the interval  $\left[ \frac{\sigma}{\gamma^{\mu(j)}}, \frac{\sigma}{\gamma^{\mu(j-1)}} \right)$ , i.e.,  $d(v) \notin \left[ \frac{\sigma}{\gamma^{\mu(j)}}, \frac{\sigma}{\gamma^{\mu(j-1)}} \right)$  and  $d(v) \notin \left[ \frac{\sigma}{\gamma^{\mu(j-1)+1}}, \frac{\sigma}{\gamma^{\mu(j-1)}} \right)$ . We will show that both events occur with probability at most  $1/B$ , giving the claim via a union bound.

For any  $v$ , there is a unique  $i$  such that  $d(v) \in \left[ \frac{\sigma}{\gamma^i}, \frac{\sigma}{\gamma^{i-1}} \right)$ . Thus, the claim only fails to hold if  $i = \mu(j)$  for some  $j$  or  $i = \mu(j-1) + 1$  for some  $j$ . For the first case, when  $i = \mu(j) = j \cdot B - s$  for some  $j$  is satisfied only if  $s = j \cdot B - i$ , which occurs with probability  $1/B$  since  $s$  is selected uniformly at random from  $\{0, 1, \dots, B-1\}$ . Similarly,  $i = \mu(j-1) + 1 = (j-1) \cdot B + 1 - s$  only if  $s = (j-1) \cdot B + 1 - i$ , which again occurs with probability  $1/B$ . Using union bound, we have:

$$\Pr \left[ d(v) \in \left[ \frac{\sigma}{\gamma^{\mu(j)}}, \frac{\sigma}{\gamma^{\mu(j-1)}} \right) \text{ or } d(v) \in \left[ \frac{\sigma}{\gamma^{\mu(j-1)+1}}, \frac{\sigma}{\gamma^{\mu(j-1)}} \right) \right] \leq \frac{2}{B} = \epsilon.$$

Hence, the claim. □

**Definition 5.6.11** (Recovered Level). *A vertex  $v$  is recovered at level  $\widehat{\ell}(v)$  iff*

$$\widehat{\ell}(v) = \arg \min_{j \in [L]} \widehat{d}_j(v) \geq \frac{\bar{m}}{\gamma^{\mu(j)}} \cdot \frac{c_2 \epsilon^2}{\log n}.$$

We associate the following sets with the set of recovered vertices:

$$\mathcal{R} = \{v \in V \mid r(v) = 1\}, \mathcal{R}_{\text{bad}} = \{v \in \mathcal{R} \mid \widehat{\ell}(v) \neq \ell(v)\}, \text{ and } \mathcal{R}_{\text{boundary}} = \mathcal{R} \cap V_{\text{boundary}}.$$

Here,  $\mathcal{R}_{\text{bad}}$  represents set of recovered vertices  $v$  at a level  $\widehat{\ell}(v)$  different from  $\ell(v)$ . Recall that  $\ell(v)$  represents the level at which the vertex  $v$  is recovered if we knew the degree  $d(v)$  exactly.

Using the next claim, we argue that if  $v$  is included in the set of sampled vertices at level  $\ell(v)$ , i.e.,  $v \in S_{\ell(v)}$ , then, it will be recovered at that level, provided degree estimates satisfy Corollary 5.6.7.

**Claim 5.6.12.** *Suppose  $v \in S_{\ell(v)}$  and  $v \notin V_{\text{boundary}}$  satisfying:*

$$(1 - \epsilon)d(v) \leq \widehat{d}_{\widehat{\ell}(v)}(v) \leq d(v) + \frac{c_1 \epsilon^3 \cdot m}{\log n \cdot \gamma^{\mu(\widehat{\ell}(v))}}, \text{ then, we have } \widehat{\ell}(v) = \ell(v).$$

*Proof.* As  $v \notin V_{\text{boundary}}$ , and  $v \in S_{\ell(v)}$ , from the definition of boundary vertices, we have:

$$\frac{\bar{m}}{\gamma^{\mu(\ell(v))-1}} \cdot \frac{c_2 \epsilon^2}{\log n} \leq d(v) \leq \frac{\bar{m}}{\gamma^{\mu(\ell(v)-1)+1}} \cdot \frac{c_2 \epsilon^2}{\log n}.$$

This implies:

$$\begin{aligned} \widehat{d}_{\widehat{\ell}(v)}(v) &\geq (1 - \epsilon)d(v) \geq (1 - \epsilon) \cdot \frac{\bar{m}}{\gamma^{\mu(\ell(v))-1}} \cdot \frac{c_2 \epsilon^2}{\log n} \\ &= (1 - \epsilon)\gamma \cdot \frac{\bar{m}}{\gamma^{\mu(\ell(v))}} \cdot \frac{c_2 \epsilon^2}{\log n} \\ &= \frac{\bar{m}}{\gamma^{\mu(\ell(v))}} \cdot \frac{c_2 \epsilon^2}{\log n}, \end{aligned}$$

as  $\gamma = 1/(1 - \epsilon)$ , and so in Algorithm REFINE-ESTIMATE (Alg. 21),  $v$  will be recovered, and  $\widehat{\ell}(v) = \ell(v)$  as long as it hasn't been recovered at a prior level.

At any prior level  $j \leq \ell(v) - 1$ , from Lemma 5.6.4, we have:

$$\begin{aligned}
\widehat{d}_j(v) &\leq d(v) + \frac{c_1 \epsilon^3 \cdot m}{\log n \cdot \gamma^{\mu(j)}} \\
&\leq \frac{\bar{m}}{\gamma^{\mu(\ell(v)-1)+1}} \cdot \frac{c_2 \epsilon^2}{\log n} + \frac{c_1 \epsilon^3 \cdot m}{\log n \cdot \gamma^{\mu(j)}} \\
&\leq \frac{\bar{m}}{\gamma^{\mu(j)}} \cdot \frac{c_2 \epsilon^2}{\log n} \cdot \left( \frac{1}{\gamma \cdot \gamma^{(\ell(v)-1-j) \cdot B}} + \frac{c_1 \epsilon}{c_2} \right) \\
&\leq \frac{\bar{m}}{\gamma^{\mu(j)}} \cdot \frac{c_2 \epsilon^2}{\log n} \left( \frac{1}{\gamma} + \frac{c_1 \epsilon}{c_2} \right) \\
&= \frac{\bar{m}}{\gamma^{\mu(j)}} \cdot \frac{c_2 \epsilon^2}{\log n} \left( 1 - \epsilon + \frac{c_1 \epsilon}{c_2} \right) \\
&< \frac{\bar{m}}{\gamma^{\mu(j)}} \cdot \frac{c_2 \epsilon^2}{\log n},
\end{aligned}$$

as long as we set  $c_1 < c_2$ . Thus,  $v$  will be rejected at any level  $j < \ell(v)$ .  $\square$

The following corollary is immediate from the previous Claim 5.6.12, as every vertex that is not at the boundary is recovered at the actual level.

**Corollary 5.6.13.** *If all the degree estimates of sampled vertices at every level are good approximations, i.e., satisfy the Corollary 5.6.7, then,  $\mathcal{R}_{\text{bad}} \subseteq \mathcal{R}_{\text{boundary}} \subseteq V_{\text{boundary}}$ .*

For a vertex  $v$  that lies in the boundary, i.e.,  $v \in V_{\text{boundary}}$ , it is possible that  $v$  is recovered at a level far away from  $\ell(v)$ . Using the next claim, we argue that it will be recovered in the adjacent levels if it has not been recovered at  $\ell(v)$ .

**Claim 5.6.14.** *Suppose  $v \in S_{\ell(v)+1}$  and  $v \in V_{\text{boundary}}$  satisfying:*

$$(1 - \epsilon) \leq \widehat{d}_{\widehat{\ell}(v)}(v) \leq d(v) + \frac{c_1 \epsilon^3 \cdot m}{\log n \cdot \gamma^{\mu(\widehat{\ell}(v))}}, \text{ then, we have } \widehat{\ell}(v) \in \{\ell(v)+1, \ell(v), \ell(v)-1\}.$$

*Proof.* For notational convenience, let  $\sigma = \frac{\bar{m} \cdot c_2 \epsilon^2}{\log n}$ . As  $v \in S_{\ell(v)+1}$ , we have  $v \in S_{\ell(v)-1}$  and  $v \in S_{\ell(v)}$  from construction.

First, we observe that  $\widehat{\ell}(v) \leq \ell(v) + 1$ , because,

$$\widehat{d}_{\widehat{\ell}(v)}(v) \geq (1 - \epsilon)d(v) \geq (1 - \epsilon) \frac{\sigma}{\gamma^{\mu(\ell(v))}} \geq (1 - \epsilon)\gamma^B \frac{\sigma}{\gamma^{\mu(\ell(v)+1)}} \geq \frac{\sigma}{\gamma^{\mu(\ell(v)+1)}}.$$

The last inequality follows from  $(1 - \epsilon)\gamma^B \geq (1 - \epsilon)(1 + \epsilon)^B \geq 3 - 3\epsilon \geq 1$ , when  $\epsilon \leq \frac{2}{3}$ .

As  $v \in V_{\text{boundary}}$ , we have the following cases:

(a)  $d(v) \in \left( \frac{\sigma}{\gamma^{\mu(\ell(v)-1)+1}}, \frac{\sigma}{\gamma^{\mu(\ell(v)-1)}} \right)$ . We use proof by contradiction. Suppose  $\widehat{\ell}(v) \leq \ell(v) - 2$ . Then:

$$\begin{aligned} \widehat{d}_{\widehat{\ell}(v)}(v) &\leq d(v) + \frac{c_1 \epsilon^3 \cdot m}{\log n \cdot \gamma^{\mu(\widehat{\ell}(v))}} \\ &< \frac{\bar{m}}{\gamma^{\mu(\ell(v)-1)}} \cdot \frac{c_2 \epsilon^2}{\log n} + \frac{c_1 \epsilon^3 \cdot m}{\log n \cdot \gamma^{\mu(\widehat{\ell}(v))}} \\ &= \frac{\bar{m}}{\gamma^{\mu(\widehat{\ell}(v))}} \cdot \frac{c_2 \epsilon^2}{\log n} \cdot \left( \frac{1}{\gamma^{(\ell(v)-1-\widehat{\ell}(v)) \cdot B}} + \frac{c_1 \epsilon}{c_2} \right) \\ &\leq \frac{\bar{m}}{\gamma^{\mu(\widehat{\ell}(v))}} \cdot \frac{c_2 \epsilon^2}{\log n} \left( \frac{1}{\gamma^B} + \frac{c_1 \epsilon}{c_2} \right) \\ &= \frac{\bar{m}}{\gamma^{\mu(\widehat{\ell}(v))}} \cdot \frac{c_2 \epsilon^2}{\log n} \left( (1 - \epsilon)^B + \frac{c_1 \epsilon}{c_2} \right) \\ &\leq \frac{\bar{m}}{\gamma^{\mu(\widehat{\ell}(v))}} \cdot \frac{c_2 \epsilon^2}{\log n} \left( e^{-2} + \frac{c_1 \epsilon}{c_2} \right) \leq \frac{\bar{m}}{\gamma^{\mu(\widehat{\ell}(v))}} \cdot \frac{c_2 \epsilon^2}{\log n}. \end{aligned}$$

The last inequality follows because  $c_1 \leq c_2(1 - e^{-2})$ . Therefore,  $\widehat{\ell}(v) > \ell(v) - 2$ .

(b)  $d(v) \in \left[ \frac{\sigma}{\gamma^{\mu(\ell(v))}}, \frac{\sigma}{\gamma^{\mu(\ell(v)-1)}} \right)$ . Using a similar argument, we obtain that  $\widehat{\ell}(v) \geq \ell(v)$ .

For the sake of contradiction, let  $\widehat{\ell}(v) \leq \ell(v) - 1$ . Then:

$$\begin{aligned}
\widehat{d}_{\widehat{\ell}(v)}(v) &\leq d(v) + \frac{c_1 \epsilon^3 \cdot m}{\log n \cdot \gamma^{\mu(\widehat{\ell}(v))}} \\
&< \frac{\bar{m}}{\gamma^{\mu(\ell(v))-1}} \cdot \frac{c_2 \epsilon^2}{\log n} + \frac{c_1 \epsilon^3 \cdot m}{\log n \cdot \gamma^{\mu(\widehat{\ell}(v))}} \\
&= \frac{\bar{m}}{\gamma^{\mu(\widehat{\ell}(v))}} \cdot \frac{c_2 \epsilon^2}{\log n} \cdot \left( \frac{\gamma}{\gamma^{(\ell(v)-\widehat{\ell}(v)) \cdot B}} + \frac{c_1 \epsilon}{c_2} \right) \\
&\leq \frac{\bar{m}}{\gamma^{\mu(\widehat{\ell}(v))}} \cdot \frac{c_2 \epsilon^2}{\log n} \left( \frac{2}{\gamma^B} + \frac{c_1 \epsilon}{c_2} \right) \\
&= \frac{\bar{m}}{\gamma^{\mu(\widehat{\ell}(v))}} \cdot \frac{c_2 \epsilon^2}{\log n} \left( 2(1-\epsilon)^B + \frac{c_1 \epsilon}{c_2} \right) \\
&\leq \frac{\bar{m}}{\gamma^{\mu(\widehat{\ell}(v))}} \cdot \frac{c_2 \epsilon^2}{\log n} \left( 2e^{-2} + \frac{c_1 \epsilon}{c_2} \right) \leq \frac{\bar{m}}{\gamma^{\mu(\widehat{\ell}(v))}} \cdot \frac{c_2 \epsilon^2}{\log n}.
\end{aligned}$$

The last inequality follows because  $c_1 \leq c_2(1-2e^{-2})$ . Therefore,  $\widehat{\ell}(v) > \ell(v) - 1$ .

Therefore, we have  $\ell(v) - 2 < \widehat{\ell}(v) \leq \ell(v) + 1$ .  $\square$

**Definition 5.6.15** (Random variables). *Let  $\widehat{X}(v)$  be the random variable with  $\widehat{X}(v) = \gamma^{\mu(\widehat{\ell}(v))} \cdot \widehat{d}(v)$  if  $v$  is recovered at level  $\widehat{\ell}(v)$  and  $\widehat{X}(v) = 0$  otherwise. We define  $X(v)$  similarly, assuming we run Algorithm 21 with exact degrees, i.e.,  $X(v) = \gamma^{\mu(\ell(v))} \cdot d(v)$  if  $v$  is recovered at its actual level  $\ell(v)$  and  $X(v) = 0$  otherwise.*

In the analysis that follows next, we will argue that for most of the vertices in  $\mathcal{R}$ , except for those in  $\mathcal{R}_{\text{boundary}}$ , the  $\widehat{X}(v)$  and  $X(v)$  are close to each other, i.e.,  $1 \pm \epsilon$  approximations of each other. Separately, we show that the contribution of  $\sum_{v \in \mathcal{R}_{\text{boundary}}} \widehat{X}(v)$  is small. As  $X(v)$  values do not contain any degree approximations, they are easier to handle and we will show concentration for  $\sum_{v \in \mathcal{R}} X(v)$ . As a result, the concentration will also hold for the actual estimate  $\sum_{v \in \mathcal{R}} \widehat{X}(v)$ .

Throughout the remaining section, unless explicitly stated otherwise, we will assume that the degrees are *good* approximations. Formally stated, we define  $\mathcal{E}_0$  as the event indicating all the degree estimates at every sampling level satisfy Corollary 5.6.7. Note that  $\Pr[\mathcal{E}_0] \geq 0.70$  (from Corollary 5.6.7).

**Claim 5.6.16.** *With probability at least  $1 - 1/11 \log \log n$ , we have:*

$$\sum_{v \in \mathcal{R}_{\text{boundary}}} \widehat{X}(v) \leq 572\epsilon m \log \log n \quad \text{and} \quad \sum_{v \in \mathcal{R}_{\text{boundary}}} X(v) \leq 44\epsilon m \log \log n.$$

*Proof.* For the sake of brevity, we omit that all the expected values include conditioning on the event  $\mathcal{E}_0$ .

Consider a vertex  $v \in \mathcal{R}_{\text{boundary}}$ . We have that the probability a vertex  $v \in \mathcal{R}_{\text{boundary}}$  is recovered at a level  $\widehat{\ell}(v)$  satisfying  $\widehat{\ell}(v) \in \{\ell(v) + 1, \ell(v), \ell(v) - 1\}$  (From Claim 5.6.14):

$$\begin{aligned} \Pr[v \in \mathcal{R}_{\text{boundary}}] &= \Pr[v \in V_{\text{boundary}}] \cdot \Pr[v \in \mathcal{R} \mid v \in V_{\text{boundary}}] \\ &= \sum_{\widehat{\ell}(v) \in \{\ell(v)+1, \ell(v), \ell(v)-1\}} \Pr[v \in V_{\text{boundary}}] \cdot \Pr[v \in S_{\widehat{\ell}(v)}]. \end{aligned}$$

Therefore, we have:

$$\begin{aligned} \mathbf{E}[\widehat{X}(v)] &= \sum_{\widehat{\ell}(v) \in \{\ell(v)+1, \ell(v), \ell(v)-1\}} \gamma^{\mu(\widehat{\ell}(v))} \widehat{d}_{\widehat{\ell}(v)} \Pr[v \in V_{\text{boundary}}] \cdot \Pr[v \in S_{\widehat{\ell}(v)}] \\ &\leq \epsilon \cdot \sum_{\widehat{\ell}(v) \in \{\ell(v)+1, \ell(v), \ell(v)-1\}} \widehat{d}_{\widehat{\ell}(v)} \\ &\leq 3\epsilon \cdot \left( d(v) + \frac{c_1 \epsilon^3 m}{\gamma^{\mu(\ell(v)-1)} \log n} \right) \quad (\text{using Corollary 5.6.7}) \\ &= 3\epsilon \cdot \left( d(v) + \left( \frac{1}{1-\epsilon} \right)^B \cdot \frac{c_1 \epsilon^3 m}{\gamma^{\mu(\ell(v))} \log n} \right) \\ &\leq 3\epsilon \cdot (d(v) + 5\epsilon d(v)), \quad \text{because } d(v) \geq \frac{c_2 \epsilon^2 \bar{m}}{\gamma^{\mu(\ell(v))} \log n} \\ &\quad \geq \frac{c_1 \epsilon^2 m}{\gamma^{\mu(\ell(v))} \log n} \\ &\leq 18\epsilon \cdot d(v) \\ \Rightarrow \mathbf{E} \left[ \sum_{v \in \mathcal{R}_{\text{boundary}}} \widehat{X}(v) \right] &\leq 18\epsilon \cdot \sum_{v \in V} d(v) \leq 36\epsilon m. \end{aligned}$$

Using Markov's inequality, we have  $\sum_{v \in \mathcal{R}_{\text{bad}}} \widehat{X}_v \leq 572\epsilon m \log n$ , with probability  $\geq 1 - 1/22 \log \log n$ .

Similarly, we can bound the sum:

$$\mathbf{E} \left[ \sum_{v \in \mathcal{R}_{\text{boundary}}} X(v) \right] \leq \sum_{v \in V} \epsilon \cdot d(v) \leq 2\epsilon m.$$

Using Markov's inequality, we have, with probability  $1 - 1/22 \log \log n$ ,  $\sum_{v \in \mathcal{R}_{\text{bad}}} X(v) \leq 44\epsilon m \log n$ . Taking a union bound for both the events, gives us the claim.  $\square$

**Claim 5.6.17.**  $\mathbf{E}[X(v)] = d(v) \forall v \in V$ . Also,  $\mathbf{E}[\sum_{v \in V} X(v)] = 2m$ .

*Proof.* By Claim 5.6.12,  $X(v)$  is nonzero which requires that  $v \in S_{\ell(v)}$ . As  $v$  is included in  $S_{\ell(v)}$  with probability  $1/\gamma^{\mu(\ell(v))}$ . Therefore, we have:

$$\begin{aligned} \mathbf{E}[X(v)] &= \gamma^{\mu(\ell(v))} \cdot d(v) \cdot 1/\gamma^{\mu(\ell(v))} = d(v) \\ \mathbf{E} \left[ \sum_{v \in V} X(v) \right] &= \sum_{j \in [L]} \sum_{v \in V \cap S_j} \mathbf{E}[X(v)] = \sum_{v \in V} d(v) = 2m. \end{aligned}$$

$\square$

**Claim 5.6.18.**  $\left| \sum_{v \in \mathcal{R} \setminus \mathcal{R}_{\text{boundary}}} \widehat{X}(v) - \sum_{v \in \mathcal{R} \setminus \mathcal{R}_{\text{boundary}}} X(v) \right| \leq \epsilon \cdot \sum_{v \in \mathcal{R} \setminus \mathcal{R}_{\text{boundary}}} X(v)$ .

*Proof.* Consider a vertex  $v \in \mathcal{R} \setminus \mathcal{R}_{\text{boundary}}$ . From Claim 5.6.12, we have  $\widehat{\ell}(v) = \ell(v)$  provided  $v \in S_{\ell(v)}$ . As we have already conditioned on the event  $\mathcal{E}_0$ , from Corollary 5.6.7, we have:

$$\begin{aligned} (1 - \epsilon)d(v) &\leq \widehat{d}_{\ell(v)}(v) \leq d(v) + \frac{c_1 \epsilon^3 m}{\gamma^{\mu(\ell(v))} \log n} \\ (1 - \epsilon)\gamma^{\mu(\ell(v))}d(v) &\leq \gamma^{\mu(\ell(v))}\widehat{d}_{\ell(v)}(v) \leq \gamma^{\mu(\ell(v))}d(v) + \gamma^{\mu(\ell(v))} \cdot \frac{c_1 \epsilon^3 m}{\gamma^{\mu(\ell(v))} \log n} \\ \Rightarrow (1 - \epsilon)X(v) &\leq \widehat{X}(v) \leq X(v) + \frac{c_1 \epsilon^3 m}{\log n}. \end{aligned}$$

As  $\bar{m} \geq m$  and  $c_2 > c_1$ , we have:

$$\begin{aligned} d(v) &\geq \frac{\bar{m} \cdot c_2 \epsilon^2}{\gamma^{\mu(\ell(v))} \log n} \geq \frac{m \cdot c_2 \epsilon^2}{\gamma^{\mu(\ell(v))} \log n} \\ \Rightarrow \frac{m \cdot c_1 \epsilon^3}{\log n} &\leq \frac{m \cdot c_2 \epsilon^3}{\log n} \leq \gamma^{\mu(\ell(v))} \cdot \epsilon d(v) = \epsilon X(v). \end{aligned}$$

Therefore,

$$(1 - \epsilon)X(v) \leq \widehat{X}(v) \leq (1 + \epsilon)X(v).$$

Thus, if  $\widehat{X}(v)$  is nonzero,  $(1 - \epsilon)\widehat{X}(v) \leq X(v) \leq (1 + \epsilon)\widehat{X}(v)$ , which gives the claim after summing up over all the terms.  $\square$

**Claim 5.6.19.** *The variables  $X(v) \forall v \in V$  are independent and bounded given by*

$$X(v) \leq \frac{2\bar{m} \cdot c_2 \epsilon^2}{\log n}$$

*Proof.* Since the vertices are included independently in the sets  $S_0, \dots, S_L$ , the independence of  $X(v)$  follows immediately. Additionally, since  $\ell(v)$  is the smallest  $j \in \{0, 1, \dots, L\}$  for which  $d(v) \geq \frac{\bar{m}}{\gamma^{\mu(j)}} \cdot \frac{c_2 \epsilon^2}{\log n}$  (See the definition of  $\ell(v)$  in Claim 5.6.12), we obtain  $d(v) \leq \frac{\bar{m}}{\gamma^{\mu(\ell(v)-1)}} \cdot \frac{c_2 \epsilon^2}{\log n}$ . Thus, if  $X(v)$  is nonzero,

$$X(v) = \gamma^{\mu(\ell(v))} \cdot d(v) \leq \frac{\gamma^B \bar{m} \cdot c_2 \epsilon^2}{\log n} \leq \frac{1}{(1 - \epsilon)^B} \cdot \frac{\bar{m} \cdot c_2 \epsilon^2}{\log n} \leq \frac{2\bar{m} \cdot c_2 \epsilon^2}{\log n}.$$

$\square$

Combining Claims 5.6.17 and 5.6.19 we obtain:

**Claim 5.6.20.** *With probability at least  $1 - 1/n$ ,  $|\sum_v X(v) - 2m| \leq \max\{\epsilon, \epsilon \cdot \alpha\} \cdot 2m$ .*

*Proof.* By Claims 5.6.19 and 5.6.17, we have:

$$\begin{aligned}
\sum_{v \in V} \mathbf{E}[(X(v) - d(v))^2] &= \sum_v \mathbf{E}[X(v)^2] - 2d(v) \mathbf{E}[X(v)] + d(v)^2 \\
&\leq \sum_v \mathbf{E}[X(v)^2] \\
&\leq \frac{2\bar{m} \cdot c_2 \epsilon^2}{\log n} \cdot \sum_{v \in V} \mathbf{E}[X(v)] = \frac{2\bar{m} \cdot 2m \cdot c_2 \epsilon^2}{\log n}.
\end{aligned}$$

If  $\alpha > 1$ , then,  $m \leq \bar{m} \leq m(1 + \alpha) \leq 2m\alpha$ . From Bernstein's inequality (Lemma 5.5.2), we have:

$$\begin{aligned}
\Pr \left[ \left| \sum_v X(v) - 2m \right| \geq \epsilon' \alpha \cdot m \right] &\leq \exp \left( -\frac{\epsilon'^2 \alpha^2 \cdot m^2}{\frac{2 \cdot 2\bar{m} \cdot 2m \cdot c_2 \epsilon^2}{\log n} + \frac{2}{3} \cdot \frac{2\bar{m} \cdot c_2 \epsilon^2}{\log n} \cdot m \epsilon' \alpha} \right) \\
&\leq \exp \left( -\frac{\epsilon'^2 \alpha^2 \cdot m^2 \log n}{2 \cdot 4m\alpha \cdot 2m \cdot c_2 \epsilon^2 + \frac{2}{3} \cdot 4m\alpha \cdot c_2 \epsilon^2 \cdot m \epsilon' \alpha} \right) \\
&= \exp \left( -\frac{\epsilon'^2 \alpha^2 \log n}{16c_2 \epsilon^2 \alpha + \frac{8}{3} \cdot c_2 \epsilon^2 \cdot \epsilon' \alpha^2} \right) \leq \frac{1}{n}, \text{ because,} \\
\frac{\epsilon'^2 \alpha^2}{16c_2 \epsilon^2 \alpha + \frac{8}{3} \cdot c_2 \epsilon^2 \cdot \epsilon' \alpha^2} &\geq 1, \text{ as } 1 \geq \epsilon' \geq \epsilon \left( \frac{56c_2}{3} \right)^{1/2}.
\end{aligned}$$

If  $\alpha \in [\epsilon', 1]$ , then,  $\bar{m} \leq 2m$ . From Bernstein's inequality (Lemma 5.5.2), we have:

$$\begin{aligned}
\Pr \left[ \left| \sum_v X(v) - 2m \right| \geq \epsilon' \cdot m \right] &\leq \exp \left( -\frac{\epsilon'^2 \cdot m^2}{\frac{2 \cdot 2\bar{m} \cdot 2m \cdot c_2 \epsilon^2}{\log n} + \frac{2}{3} \cdot \frac{2\bar{m} \cdot c_2 \epsilon^2}{\log n} \cdot m \epsilon'} \right) \\
&\leq \exp \left( -\frac{\epsilon'^2 \cdot m^2 \log n}{2 \cdot 4m \cdot 2m \cdot c_2 \epsilon^2 + \frac{2}{3} \cdot 4m \cdot c_2 \epsilon^2 \cdot m \epsilon'} \right) \\
&\leq \exp \left( -\frac{\epsilon'^2 \log n}{64c_2 \epsilon^2 + \frac{16}{3} \cdot c_2 \epsilon^2} \right) \leq \frac{1}{n}, \text{ because,} \\
\frac{\epsilon'^2}{64c_2 \epsilon^2 + \frac{16}{3} \cdot c_2 \epsilon^2} &\geq 1, \text{ as } 1 \geq \epsilon' \geq \epsilon \left( \frac{208c_2}{3} \right)^{1/2}.
\end{aligned}$$

Combining both the statements, and scaling  $\epsilon$  gives us the lemma.  $\square$

Note that Claim 5.6.20 basically gives us what we want, after adjusting constants on  $\alpha$  and scaling up our estimate appropriately. We formalize this using the below lemma:

**Lemma 5.6.21.** *Suppose the input  $\bar{m}$  to Algorithm REFINE-ESTIMATE satisfies  $m \leq \bar{m} \leq m(1 + \alpha)$  for some approximation factor  $\epsilon \leq \alpha \leq \frac{\binom{n}{2}}{m}$ . Then, with probability  $1 - 1/11 \log \log n - 1/n$ :*

$$2m(1 - \epsilon \log \log n - \epsilon \cdot \alpha) \leq \sum_{v \in V} \hat{X}(v) \leq 2m(1 + \epsilon \cdot \log \log n + \epsilon \cdot \alpha).$$

*Proof.* We note that for the vertices that are not recovered, i.e.,  $r(v) = 0$ , we have  $\hat{X}(v) = 0$ , and therefore need to only consider vertices in  $\mathcal{R}$ . From Claim 5.6.18, we have:

$$\left| \sum_{v \in \mathcal{R} \setminus \mathcal{R}_{\text{boundary}}} \hat{X}(v) - \sum_{v \in \mathcal{R} \setminus \mathcal{R}_{\text{boundary}}} X(v) \right| \leq \epsilon \cdot \sum_{v \in \mathcal{R} \setminus \mathcal{R}_{\text{boundary}}} X(v)$$

Combining it with  $\sum_{v \in \mathcal{R}_{\text{boundary}}} \hat{X}(v) \leq 572\epsilon m \log \log n$  from Claim 5.6.16, we have:

$$\begin{aligned} \sum_{v \in \mathcal{R}} \hat{X}(v) &\leq (1 + \epsilon) \sum_{v \in \mathcal{R} \setminus \mathcal{R}_{\text{boundary}}} X(v) + \sum_{v \in \mathcal{R}_{\text{boundary}}} \hat{X}(v) \\ &\leq (1 + \epsilon) \sum_{v \in V} X(v) + 572\epsilon m \log \log n \\ &\leq 2m(1 + \epsilon \cdot \alpha + \epsilon \cdot \log \log n), \end{aligned}$$

where the last step follows by scaling  $\epsilon$  (with a constant) appropriately. From Claim 5.6.16, we have  $\sum_{v \in \mathcal{R}_{\text{boundary}}} X(v) \leq 44\epsilon m \log \log n$ . Therefore, we get:

$$\sum_{v \in V} X(v) \leq \sum_{v \in \mathcal{R} \setminus \mathcal{R}_{\text{boundary}}} X(v) + 44\epsilon m \log \log n.$$

Similarly, we have:

$$\begin{aligned}
\sum_{v \in \mathcal{R}} \widehat{X}(v) &\geq (1 - \epsilon) \sum_{v \in \mathcal{R} \setminus \mathcal{R}_{\text{boundary}}} X(v) + \sum_{v \in \mathcal{R}_{\text{boundary}}} \widehat{X}(v) \\
&\geq (1 - \epsilon) \sum_{v \in \mathcal{R} \setminus \mathcal{R}_{\text{boundary}}} X(v) \\
&\geq (1 - \epsilon) \cdot \sum_{v \in V} X(v) - (1 - \epsilon) 44\epsilon m \log \log n \\
&\geq 2m(1 - \epsilon \log \log n - \epsilon \cdot \alpha) \geq 2m(1 - \alpha \cdot \epsilon \log \log n).
\end{aligned}$$

The last step follows by scaling  $\epsilon$  appropriately. Using union bound, we get the final probability claim. Hence, the lemma.  $\square$

In Algorithm REFINE-ESTIMATE, we start with an estimate  $\bar{m}_0$  for the number of edges, due to Beame et al. [22]. The estimate for the number of edges  $\bar{m}_0$  satisfies  $m \leq \bar{m}_0 \leq m(1 + \alpha_0)$ , for some  $\alpha_0$ . As  $m \leq \bar{m}_0 \leq O(m \log^2 n)$ , the approximation factor  $\alpha_0$  could be as large as  $O(\log^2 n)$ . In Lemma 5.6.21, we showed that we can improve our estimator by  $\epsilon \cdot (\alpha + \log \log n)$  multiplicative factor. We call this multiplicative improvement as *refinement*. In the next theorem, we argue that our Algorithm 20 which performs repeated refinements results in a  $(1 \pm \epsilon)$ -approximation.

**Theorem 5.6.22** (Theorem 5.6.5 restated). *Given a graph  $G$  with  $n$  nodes and  $m$  edges, there is an algorithm that makes  $O(\epsilon^{-5} \log^5 n \log^5(\log n) \log(\epsilon^{-1} \log n))$  non-adaptive BIS queries to  $G$  and returns an estimate  $\widehat{m}$  satisfying:  $m(1 - \epsilon) \leq \widehat{m} \leq m(1 + \epsilon)$ , with probability at least  $3/5$ .*

*Proof.* We denote  $\epsilon' = O(\epsilon \log \log n)$ . As  $m \leq \bar{m}_0 \leq O(m \log^2 n)$ , the approximation factor  $\alpha_0$  could be as large as  $O(\log^2 n)$ . From Lemma 5.6.21, we have that each refinement improves the approximation factor to  $m(1 + \epsilon \cdot \alpha + \epsilon \cdot \log n) \leq m(1 + \epsilon' \alpha_0) \leq m + \epsilon' m \alpha_0$ . Therefore, after  $O(\log_{1/\epsilon'} \log n)$  refinements, we expect the upper bound in the approximation to reduce from  $\alpha_0 \cdot m$  to  $O(\epsilon' \cdot m)$ . However, each *refinement*

worsens the lower bound from  $m$  to  $m(1 - \epsilon' \cdot \alpha)$ . In order to maintain the invariant that the input to Algorithm 21 always satisfies  $\bar{m}_1 = \hat{m} \geq m$ , we normalize it by adding  $\epsilon' m_0$ . This implies that  $\hat{m} \leq m + 2\epsilon' m \alpha_0$ , and the new approximation factor  $\alpha_1 \leq 2\epsilon' \alpha_0$ . Continuing this, after  $T - 1 = 3 \log_{1/\epsilon'} \log n$  refinements, we will have  $\hat{m}_T \leq m + (2\epsilon')^T m \alpha_0$ . By scaling  $\epsilon = O(\epsilon'/\log \log n)$ , we have,  $\epsilon' \leq 1/c$  for some integer constant  $c > 2$  and  $\hat{m}_T \leq m + \epsilon \cdot m$ . So, the final estimate  $\hat{m}$  returned satisfies:

$$m(1 - \epsilon) \leq \hat{m} \leq m(1 + \epsilon)$$

For each level of sampling, we use Algorithm 19 to return degree estimates which requires  $O(\epsilon^{-5} \log^4 n \log(\epsilon^{-1} \log n))$  BIS queries and  $O(\epsilon^{-5} \log^5 n \log(\epsilon^{-1} \log n))$  in total including all the  $L$  levels (without scaling of  $\epsilon$ ). As we have scaled by setting  $\epsilon = O(\epsilon'/\log_{1/\epsilon} \log n)$ , the total number of BIS queries is  $O(\epsilon^{-5} \log^5 n \log^5(\log n) \log(\epsilon^{-1} \log n))$ .

Recall that we have conditioned on the event  $\mathcal{E}_0$  that our degree estimates are accurate. Using union bound, the total probability of failure across  $O(\log_{1/\epsilon} \log n) = O(\log \log n)$  refinements is:

$$\begin{aligned} & \Pr[\neg \mathcal{E}_0] + (1/10 \log \log n + 1/n) \cdot \log \log n \\ & \leq 0.30 + 1/11 + \log \log n/n \leq 2/5 \text{ for sufficiently large } n. \end{aligned}$$

Hence, the theorem. □

## 5.7 Non-adaptive algorithms for uniform sampling

In this section, we describe algorithms for sampling a *near*-uniform edge in the graph. In Section 5.7.1, we discuss connections between OR queries (Definition 5.5.1) and BIS queries, and outline an algorithm that takes as input two disjoint subsets  $L, R$  and returns a uniform vertex in  $|\Gamma(L) \cap R|$ . Next, in Section 5.7.2, we use this algorithm to return uniform neighbors for every vertex in a given subset sampled from

$V$ . Finally, in Section 5.7.3, we combine these neighbors obtained for each vertex, and return a *near*-uniform sample among the edges of the graph.

### 5.7.1 Identifying uniform neighbor of a subset of vertices

We will describe connections between OR queries (Definition 5.5.1) and BIS queries that give us algorithms for sampling uniformly an edge from the neighborhood of a subset of vertices  $L$  in another disjoint subset  $R$ . This is similar to Section 5.6, where we discussed algorithms for counting the number of edges in the neighborhood of a set  $L$  in another disjoint subset  $R$ .

In [16], the authors discuss various algorithms for the well-studied single element recovery problem using OR-queries. In the single element recovery problem, we are given a boolean vector, and we want to identify a non-zero index (also called element) of the vector.

**Definition 5.7.1** (Single Element Recovery [16]). *Given a boolean vector  $x \in \{0, 1\}^N$ , return a non-zero element from the support of  $x$ , denoted by  $\text{supp}(x)$ .*

**Lemma 5.7.2** (Lemma 4.3 from [16] restated). *Suppose  $x \in \{0, 1\}^N$  is a boolean vector. There is a non-adaptive randomized algorithm that recovers a uniform element  $j \in \text{supp}(x)$  with probability  $1 - \delta$  and uses  $O(\log^2 N \log(1/\delta))$  OR queries.*

**Simulating a BIS query using OR query.** We start with an observation that any BIS query can be simulated using a single OR query. An OR query (Definition 5.5.1) takes as input a subset  $S \subseteq V \times V$  of pairs of vertices and returns if an edge of the graph is present among the subset. Therefore, a BIS query  $\mathcal{BIS}(L, R)$  is equivalent to an OR query of the subset  $S = \{(u, v) \mid u \in L, v \in R\}$ .

Now, we will show a connection in the other direction for the problem of single element recovery, i.e., we show that OR queries used for finding single element recovery can be simulated using appropriate BIS queries.

Suppose we are given two disjoint subsets  $L, R \subseteq V$ , and we want to output a neighbor of  $L$  in  $R$ , i.e., a vertex in the set  $\Gamma(L) \cap R$ . Intuitively, this is equivalent to finding a non-zero element (corresponds to a neighbor vertex) in the vector defined over the subset  $R$  for a given subset  $L$ .

**Simulating an OR query using BIS query for finding a neighbor.** Let  $x_R \in \{0, 1\}^R$  denote a vector such that  $i$ th element of the vector corresponds to the  $i$ th vertex  $v_i$  in  $R$  (according to some fixed ordering of vertices). If  $v_i \in \Gamma(L) \cap R$  for some  $i \in \{1, 2, \dots, |R|\}$ , then,  $x_R[i] = 1$ , otherwise it is 0. Let  $\mathcal{Q}$  denotes the set of OR queries used to recover a uniform element from  $x_R$  using the algorithm from Lemma 5.7.2. Each of the OR queries  $q \in \mathcal{Q}$  is defined over a subset  $R_q \subseteq \{v_1, v_2, \dots, v_{|R|}\}$  and can be replaced with a corresponding BIS query  $\mathcal{BIS}(L, R_q)$  with the same output. Therefore, we can restate Lemma 5.7.2 in terms of BIS queries as follows:

**Lemma 5.7.3.** *Suppose  $L, R \subseteq V$  are disjoint subsets. There is a non-adaptive randomized algorithm that recovers a uniform neighbor  $u \in \Gamma(L) \cap R$  with probability  $1 - \delta$  and uses  $O(\log^2 |R| \log(1/\delta))$  BIS queries.*

## 5.7.2 Identifying uniform neighbour for each vertex

In this section, given a subset  $S \subseteq V$ , we describe an algorithm that returns a uniform neighbor for every vertex in  $S$  based on the ideas from Section 5.7.1.

**Overview of Algorithm UNIFORM-NEIGHBOR.** Our algorithm extends ESTIMATE-DEGREE by also returning a uniform neighbor for every vertex in  $S$ , along with the degree estimates. Consider a vertex  $v$  contained in the partition  $S^{ta}$ . Along with the estimating the neighborhood size of the partition containing a vertex  $v$ , we also return a uniform neighbor from the set  $\Gamma(S^{ta}) \cap V \setminus S^{ta}$  using Lemma 5.7.3 from Section 5.7.1. So, we will have a set of  $T = O(\log n)$  neighbors for every vertex. By selecting the neighbor corresponding to the partition  $S^{t_{\min(v)}a}$  containing  $v$ , we ensure

that the neighbor is a uniform neighbor of  $v$  with high success probability. Here, the partition  $S^{t_{\min}(v)a}$  where  $t_{\min}(v) \in [T]$ , corresponds to the random partition with the minimum neighborhood estimate size and is used for degree estimate of  $v$ , i.e.,  $\widehat{d}(v)$ .

---

**Algorithm 22** UNIFORM-NEIGHBOR: Uniform neighbor for each vertex in a given subset  $S$

---

**Input:** Subset  $S \subseteq V$ .  
**Output:** Degree estimates and a uniform neighbour for each vertex  $v \in S$ .

- 1: Initialize  $\widehat{d}(v) \leftarrow n$  for every  $v \in S$ .
- 2: **for**  $t$  in  $\{1, 2, \dots, T = O(\log n)\}$  **do**
- 3:   Consider a random partitioning of  $S$  into  $S^{t1}, S^{t2}, \dots, S^{t\lambda}$  where  $\lambda = O(\epsilon^{-4} \log^2 n)$ .
- 4:   **for** every partition  $S^{ta}$  where  $a \in [\lambda]$  **do**
- 5:     Let  $z^{ta}$  is sample returned using Lemma 5.7.3 where  $L = S^{ta}, R = V \setminus S^{ta}$  and  $\delta = O(\epsilon / \log n^4)$ .
- 6:      $\eta_{\text{est}}(S^{ta}) \leftarrow \text{NEIGHBORHOOD-SIZE}(S^{ta}, V \setminus S^{ta})$ .
- 7:     **for**  $v \in S^{ta}$  **do**
- 8:       **if**  $\widehat{d}(v) > \eta_{\text{est}}(S^{ta})$  **then**
- 9:          $\widehat{d}(v) \leftarrow \eta_{\text{est}}(S^{ta})$ .
- 10:          $t_{\min}(v) \leftarrow t$ .
- 11:       **end if**
- 12:     **end for**
- 13:   **end for**
- 14: **end for**
- 15:  $\mathcal{U}_j(v) = z^{t_{\min}(v)a}$  for every  $v \in S$ .
- 16: **return**  $\widehat{d}(v), \mathcal{U}_j(v)$  for every  $v \in S$ .

---

We extend Lemma 5.7.3 and obtain the following corollary:

**Corollary 5.7.4.** *If  $v \in S^{ta}$  for some  $t \in [T], a \in [\lambda]$ , then, for every neighbor  $w \in \Gamma(v) \cap V \setminus S^{ta}$ , we have:*

$$\Pr[w = z^{ta}] = \frac{1}{|\Gamma(S^{ta}) \cap V \setminus S_j^{ta}|}$$

*Proof.* From Lemma 5.7.3, we know that any vertex  $w \in \Gamma(S^{ta}) \cap V \setminus S^{ta}$  will satisfy :

$$\Pr[w = z^{ta}] = \frac{1}{|\Gamma(S^{ta}) \cap V \setminus S_j^{ta}|}.$$

As  $\Gamma(v) \cap V \setminus S^{ta} \subseteq \Gamma(S^{ta}) \cap V \setminus S^{ta}$ , we have the corollary.  $\square$

**Lemma 5.7.5.** *Suppose  $S \subseteq V$ . Then, Algorithm 22 uses  $O(\epsilon^{-4} \log^5 n \log(\epsilon^{-1} \log n))$  BIS queries and with probability  $1 - O(\epsilon/\log n)$ , returns degree estimates  $\widehat{d}(v)$  for every vertex  $v \in S$  satisfying:*

$$d(v)(1 - \epsilon) \leq \widehat{d}(v) \leq d(v) + \frac{\epsilon^4}{\log^2 n} \cdot d(S).$$

*Proof.* For every random partition, we use Lemma 5.7.3 to return a uniform neighbor. This step uses  $O(\log^2 n \log(1/\delta))$  BIS queries and succeeds with probability at least  $1 - \delta$ , where  $\delta = O(\epsilon/T\lambda \log n)$ ,  $T = O(\log n)$ , and  $\lambda = O(\epsilon^{-4} \log^2 n)$ . The total number of random partitions considered is  $O(T \cdot \lambda) = O(\epsilon^{-4} \log^3 n)$ . Following the proof of Lemma 5.6.4, we get the lemma.  $\square$

### 5.7.3 Identifying a uniform edge in the graph

In this section, we give an algorithm that returns an edge sample from a distribution that is close to the uniform distribution. Our algorithm extends Algorithm EDGE-ESTIMATOR and is based on the following idea. Suppose we know the degrees of all the vertices, denoted by  $d(v) \forall v \in V$ , in the graph. In order to sample a uniform edge, we can sample a vertex  $v$  with probability  $d(v)/\sum_{w \in V} d(w)$  and return a uniform neighbor among the neighbors of  $v$ . We can observe that the probability that an edge  $e = (v, u)$  is sampled is  $d(v)/\sum_{w \in V} d(w) \cdot 1/d(v) + d(u)/\sum_{w \in V} d(w) \cdot 1/d(u) = 1/m$ .

In order to extend the above idea to our setting, there are two challenges. First, we do not know the degrees (or approximate degrees) of all the vertices. This is because the set of recovered vertices, i.e., with  $(1 \pm \epsilon)$ -approximate degree estimates known is a subset of the sampled vertices at each level of sampling. Secondly, each vertex is recovered at a different level and is therefore sampled with different probabilities. In order to return a uniform edge based on the previously discussed idea, we must return

---

**Algorithm 23** SAMPLING: Non-adaptive algorithm for sampling uniform edges
 

---

**Input:**  $V$  set of  $n$  vertices and  $\epsilon > 0$  error parameter.

**Output:** Edge of the graph sampled from a (near)-uniform distribution.

- 1: Scale  $\epsilon \leftarrow \frac{\epsilon}{600 \log_{1/\epsilon} \log n}$  and initialize  $\gamma \leftarrow 1/(1 - \epsilon)$  and  $B \leftarrow 2/\epsilon$ .
  - 2: Let  $s$  be an integer selected uniformly at random from the interval  $[0, B)$ .
  - 3: Let  $\mu(j) \leftarrow -s + j \cdot B$  for every integer  $j$  in the interval  $[0, \frac{1}{B} \cdot \log_\gamma n]$ .
  - 4: Initialize  $S_0 \leftarrow V$  and construct  $S_1$  by sampling vertices in  $S_0$  with probability  $1/\gamma^{\mu(1)}$ .
  - 5: Construct  $S_1 \supseteq S_2 \dots \supseteq S_L$  for  $L = \frac{1}{B} \cdot \log_\gamma n$  where each  $S_j$  is obtained by sampling vertices in  $S_{j-1} \forall j \geq 2$ , independently with probability  $1/\gamma^B$ .
  - 6: **for**  $j = 0, 1, \dots, L$  **do**
  - 7: Run UNIFORM-NEIGHBOR (Algorithm 22) on  $S_j$ , to obtain the degree estimates  $\hat{d}_j(v)$  satisfying  $(1 - \epsilon)d(v) \leq \hat{d}_j(v) \leq d(v) + \frac{c_1 \epsilon^3 \cdot m}{\log n \cdot \gamma^{\mu(j)}}$  for all  $v \in S_j$ .
  - 8: Let  $\mathcal{U}(v)$  denote the neighbor returned by Algorithm 22 for vertex  $v \in S_j$ .
  - 9: **end for**
  - 10: Let  $\bar{m}_0$  be the  $O(\log n)$ -approximate estimate from the Algorithm COARSEESTIMATOR in Beame et al. [22] on a random partition of  $V$ .
  - 11: Set  $\bar{m}_0 \leftarrow \max\{2, 16 \log n \cdot \bar{m}_0\}$ , so that we have  $m \leq \bar{m}_0 \leq (64 \log^2 n) \cdot m$ .
  - 12: **for**  $t = 1, 2, \dots, T = 2 \log_{1/\epsilon} \log n$  **do**
  - 13:  $\bar{m}_t$  is assigned the output of REFINE-ESTIMATE that takes as input approximate degree values  $\hat{d}_j(v) \forall v \in S_j \forall j \in [L]$ , the previous estimate  $\bar{m}_{t-1}$  and the iteration  $t$ .
  - 14: **end for**
  - 15: For every  $v$  recovered, let  $\hat{\ell}(v)$  denote the level at which  $v$  was recovered by  $T$ th iteration of REFINE-ESTIMATE. Include all the recovered vertices in  $\mathcal{R}$ .
  - 16: Let  $v_{\text{sampled}}$  be the vertex drawn from the distribution such that a vertex  $v$  in  $\mathcal{R}$  is selected with probability proportional to  $\gamma^{\mu(\hat{\ell}(v))} \cdot \hat{d}_{\hat{\ell}(v)}(v)$ .
  - 17: **return** edge  $(v_{\text{sampled}}, \mathcal{U}(v_{\text{sampled}}))$ .
- 

a single vertex among the set of recovered vertices with probability proportional to its degree.

We address these two challenges by, amongst the recovered vertices, returning vertex  $v$  with probability proportional to  $\gamma^{\hat{\ell}(v)} \cdot \hat{d}(v)$  where  $\hat{\ell}(v)$  is the level at which it is recovered, and  $\hat{d}_{\hat{\ell}(v)}(v)$  is the degree estimate at the level of recovery. From Section 5.6.3, we know that our estimator  $\sum_{v \text{ is recovered}} \hat{X}(v) = \sum_{v \text{ is recovered}} \gamma^{\hat{\ell}(v)} \cdot \hat{d}(v)$  is concentrated around  $2m$  (See Lemma 5.6.21) and we will be able to return a *near*-uniform sample.

**Overview of Algorithm SAMPLING.** Our algorithm is an extension of Algorithm EDGE-ESTIMATOR (Algorithm 20) and the differences are highlighted in blue. During the process of constructing the edge estimator, by repeated refinements, let  $\widehat{\ell}(v)$  denote the level at which a vertex  $v$  is recovered at the last, i.e.,  $T = O(\log_{1/\epsilon} \log n)^{th}$  refinement iteration, and the corresponding degree estimate  $\widehat{d}_{\widehat{\ell}(v)}(v)$ . Let  $\mathcal{R}$  denote the set of all recovered vertices, i.e.,  $\widehat{X}(v) \neq 0$ , in the last refinement iteration. Then, we draw a vertex  $v$  from the distribution such that it is selected with probability proportional to  $\gamma^{\mu(\widehat{\ell}(v))} \cdot \widehat{d}_{\widehat{\ell}(v)}(v)$ . From our earlier discussion, this approach will result in a *near*-uniform sample.

### 5.7.3.1 Proof of Theorem 5.3.2

**Claim 5.7.6.** *With probability  $1 - 2\epsilon$ , for all levels  $j \in \{1, 2, \dots, L\}$ , we have:*

$$d(S_j) \leq \frac{m \cdot L}{\epsilon \gamma^{\mu(j)}}.$$

*Proof.* As every vertex is included in  $S_j$  with probability  $1/\gamma^{\mu(j)}$ , we get:

$$\mathbf{E}[d(S_j)] = \frac{\sum_{v \in V} d(v)}{\gamma^{\mu(j)}} = \frac{2m}{\gamma^{\mu(j)}}$$

Therefore, by Markov's Inequality,  $\Pr[d(S_j) \geq m \cdot L/\epsilon \gamma^{\mu(j)}] \leq 2\epsilon/L$ . Taking a union bound over all the levels, with probability at least  $1 - 2\epsilon$ ,

$$d(S_j) \leq m \cdot L/\epsilon \gamma^{\mu(j)} \leq m \cdot \log n/\epsilon \gamma^{\mu(j)} \text{ for every level } j \in [L].$$

□

By setting  $\lambda = O(\epsilon^{-4} \log^2 n)$ , a multiplicative factor of  $1/\epsilon$  more than that in section 5.6, we ensure that the exact guarantees hold with probability  $1 - \epsilon$ . Combining Claim 5.7.6 and Lemma 5.7.5, for sufficiently large  $n$ , we have:

**Corollary 5.7.7.** *The degree estimates returned by Algorithm 22 for each sampling level  $j \in [L]$ , satisfy the following with probability at least  $1 - \epsilon$ :*

$$(1 - \epsilon)d(v) \leq \widehat{d}_j(v) \leq d(v) + \frac{c_1 \epsilon^3 m}{\gamma^{\mu(j)} \log n} \quad \forall v \in S_j.$$

For the remaining portion of this section, we will condition on the event that Corollary 5.7.7 is satisfied. In the proof of the main Theorem 5.7.10, we account for the failure probability of this event.

In the next lemma, we show that if a vertex is recovered at level  $j$  (recall the threshold value for recovery from Section 5.6.3), the neighbor returned by Algorithm 22, given by  $\mathcal{U}_j(v)$  is equal to any neighbor of  $v$  with probability  $1/\widehat{d}(v)$ . From Section 5.6.3, we know that approximate degree of  $v$  obtained from the set  $S$ , denoted by  $\widehat{d}_j(v)$ , when  $\ell(v) = j$  is a  $(1 \pm \epsilon)$ -approximation of  $d(v)$ , therefore, we have returned a neighbor of  $v$  with probability  $(1 \pm \epsilon)/d(v)$ .

**Lemma 5.7.8.** *Suppose  $v \in S_j$  satisfies the following:  $\widehat{d}(v) \geq \frac{m}{\gamma^{\mu(j)}} \cdot \frac{c_2 \epsilon^2}{\log n}$ . Then, for every  $w \in \Gamma(v)$ , we have with probability  $1 - \epsilon$ :*

$$(1 - \epsilon) \frac{1}{\widehat{d}(v)} \leq \Pr[\mathcal{U}_j(v) = w] \leq (1 + \epsilon) \frac{1}{\widehat{d}(v)}.$$

*Proof.* From Lemma 5.6.3, we know that:

$$(1 - \epsilon)|\Gamma(S^{t_{\min}(v)a}) \cap V \setminus S^{t_{\min}(v)a}| \leq \eta_{\text{est}}(S^{t_{\min}(v)a}) \leq (1 + \epsilon)|\Gamma(S^{t_{\min}(v)a}) \cap V \setminus S^{t_{\min}(v)a}|.$$

Moreover, our degree estimates in Algorithm 22 are obtained by  $\widehat{d}(v) = \eta_{\text{est}}(S^{t_{\min}(v)a})$ .

Consider a vertex  $w \in \Gamma(v)$ . If  $w \in \Gamma(v) \setminus S^{t_{\min}(v)a}$ , we have:

$$\Pr[\mathcal{U}_j(v) = w] = \frac{1}{|\Gamma(S^{t_{\min}(v)a}) \cap V \setminus S^{t_{\min}(v)a}|}$$

$$(1 - \epsilon) \cdot \frac{1}{\widehat{d}(v)} \leq \Pr[\mathcal{U}_j(v) = w] \leq (1 + \epsilon) \cdot \frac{1}{\widehat{d}(v)}$$

If  $w \in \Gamma(v) \cap S^{t_{\min}(v)^a}$ , then, it is never returned. In iteration  $t_{\min}(v) \in [T]$ ,  $w \in S$  is assigned to one of the  $\lambda$  random partitions, we observe that such an event happens with probability:

$$\Pr[w \in S^{t_{\min}(v)^a} \cap \Gamma(v)] = \frac{1}{\lambda} = \frac{\epsilon^4}{c \log^2 n} \leq \epsilon, \text{ for some constant } c > 1.$$

It is possible that  $\mathcal{U}_j(v) \notin \Gamma(v) \cap (V \setminus S^{t_{\min}(v)^a})$ , which is a failure event for us, as no neighbor of  $v$  will be returned. We will argue that probability for such an event occurring is small.

From the analysis in Lemma 5.6.4 and Corollary 5.7.7, with probability at least  $1 - 1/n^3$  and for an appropriate choice of  $c_2$ , we have:

$$d(S^{t_{\min}(v)^a} \setminus \{v\}) \leq \frac{c_1 m \cdot \epsilon^3}{\gamma^{\mu(j)} \log n} \leq \epsilon \widehat{d}(v) \quad (\text{as we are given } \widehat{d}(v) \geq \frac{m}{\gamma^{\mu(j)}} \cdot \frac{c_2 \epsilon^2}{\log n}).$$

$$\begin{aligned} \Pr[\mathcal{U}_j(v) \notin \Gamma(v) \cap (V \setminus S^{t_{\min}(v)^a})] &\leq \frac{|\Gamma(S^{t_{\min}(v)^a} \setminus \{v\}) \cap V \setminus S^{t_{\min}(v)^a}|}{|\Gamma(S^{t_{\min}(v)^a}) \cap V \setminus S^{t_{\min}(v)^a}|} \\ &\leq \frac{d(S^{t_{\min}(v)^a} \setminus \{v\})}{|\Gamma(S^{t_{\min}(v)^a}) \cap V \setminus S^{t_{\min}(v)^a}|} \\ &\leq (1 + \epsilon) \cdot \frac{d(S^{t_{\min}(v)^a} \setminus \{v\})}{\widehat{d}(v)} \leq \epsilon(1 + \epsilon) = 2\epsilon. \end{aligned}$$

By union bound, failure probability is at most  $1/n^3 + \epsilon \leq 2\epsilon$ . Scaling  $\epsilon$  appropriately, gives us the lemma.  $\square$

In the next lemma, we show that if a vertex is not in  $V_{\text{boundary}}$ , then, it is recovered with the required probability of  $\widehat{d}(v)/2m$  (upto  $1 \pm \epsilon$  factor). Otherwise, we argue that the probability of returning it is not too large.

**Lemma 5.7.9.** *For any vertex  $v$ , with probability at least  $1 - \epsilon$ , we have:*

1.  $(1 - \epsilon) \frac{\widehat{d}_{\widehat{\ell}(v)}(v)}{2m} \leq \Pr[v_{\text{sampled}} = v] \leq (1 + \epsilon) \frac{\widehat{d}_{\widehat{\ell}(v)}(v)}{2m}$  if  $v \notin V_{\text{boundary}}$ .
2.  $\Pr[v_{\text{sampled}} = v] \leq \frac{15(1+\epsilon)\widehat{d}_{\widehat{\ell}(v)}(v)}{2m}$  if  $v \in V_{\text{boundary}}$ .

*Proof.* We condition on the event that the  $(1-\epsilon)2m \leq \sum_{v \in \mathcal{R}} \widehat{X}(v) \leq (1+\epsilon)2m$ , which happens with probability at least  $1 - \epsilon$  (follows from Corollary 5.7.7, Lemma 5.6.21, and Theorem 5.6.5).

From Claim 5.6.12, we know that a vertex not lying at the boundary will be recovered at level  $\ell(v)$ , i.e.,  $\widehat{\ell}(v) = \ell(v)$ . Consider a vertex  $v \in V \setminus V_{\text{boundary}}$ , we have:

$$\Pr[v \in \mathcal{R}] = \Pr[v \in S_{\widehat{\ell}(v)}] = \frac{1}{\gamma^{\mu(\widehat{\ell}(v))}}$$

From construction, for any vertex  $v \in \mathcal{R}$ , we have that

$$\Pr[v_{\text{sampled}} = v \mid v \in \mathcal{R}] = \frac{\gamma^{\mu(\widehat{\ell}(v))} \widehat{d}_{\widehat{\ell}(v)}(v)}{\sum_{w \in \mathcal{R}} \gamma^{\mu(\widehat{\ell}(v))} \widehat{d}_{\widehat{\ell}(w)}(w)} \leq (1 + 2\epsilon) \frac{\gamma^{\mu(\widehat{\ell}(v))} \widehat{d}_{\widehat{\ell}(v)}(v)}{2m}.$$

Similarly, we get:

$$\Pr[v_{\text{sampled}} = v \mid v \in \mathcal{R}] \geq (1 - \epsilon) \frac{\gamma^{\mu(\widehat{\ell}(v))} \widehat{d}_{\widehat{\ell}(v)}(v)}{2m}.$$

Combining both the above statements and scaling  $\epsilon = \epsilon/2$ , we get:

$$(1 - \epsilon) \frac{\widehat{d}_{\widehat{\ell}(v)}(v)}{2m} \leq \Pr[v_{\text{sampled}} = v] \leq (1 + \epsilon) \frac{\widehat{d}_{\widehat{\ell}(v)}(v)}{2m}.$$

Now, consider a vertex  $v \in V_{\text{boundary}}$ . From Claim 5.6.14, if included in we know that  $\widehat{\ell}(v) \in \{\ell(v) - 1, \ell(v), \ell(v) + 1\}$ , provided it is included in their corresponding set  $S_{\widehat{\ell}(v)}$ .

$$\begin{aligned}
\Pr[v \in \mathcal{R}] &\leq \sum_{\widehat{\ell}(v) \in \{\ell(v)-1, \ell(v), \ell(v)+1\}} \Pr[v \in \mathcal{S}_{\widehat{\ell}(v)}] \leq \frac{3}{\gamma^{\mu(\ell(v)-1)}} \\
\Pr[v_{\text{sampled}} = v] &= \Pr[v_{\text{sampled}} = v \mid v \in \mathcal{R}] \cdot \Pr[v \in \mathcal{R}] \leq \frac{3(1+2\epsilon)\gamma^{\mu(\widehat{\ell}(v))}}{\gamma^{\mu(\ell(v)-1)}} \cdot \frac{\widehat{d}_{\widehat{\ell}(v)}(v)}{2m} \\
&\leq \frac{15(1+2\epsilon)\widehat{d}_{\widehat{\ell}(v)}(v)}{2m}
\end{aligned}$$

Hence, the lemma.  $\square$

**Theorem 5.7.10.** *Given a graph  $G$  with  $n$  nodes,  $m$  edges, and edge set  $E$ , there is an algorithm that makes  $O(\epsilon^{-4} \log^6 n \log(\epsilon^{-1} \log n) + \epsilon^{-6} \log^5 n \log^6(\log n) \log(\epsilon^{-1} \log n))$  non-adaptive BIS queries which, with probability at least  $1 - \epsilon$ , outputs an edge from a probability distribution  $P$  satisfying  $(1 - \epsilon)/m \leq P(e) \leq (1 + \epsilon)/m$  for every  $e \in E$ .*

*Proof.* Consider an edge  $e = (v, u)$ . Then, the edge  $e = (v, u)$  can be returned by Algorithm 23 if either  $v$  or  $u$  is the vertex sampled  $v_{\text{sampled}}$  and the other vertex is the neighbor returned by Algorithm 22. From Lemmas 5.7.8 and 5.7.9, we have:

$$\begin{aligned}
&\Pr[e \text{ is returned by Algorithm 23}] \\
&= \Pr[v_{\text{sampled}} = v] \cdot \Pr[\mathcal{U}(v) = u] + \Pr[v_{\text{sampled}} = u] \cdot \Pr[\mathcal{U}(u) = v] \\
\Rightarrow \Pr[v_{\text{sampled}} = v] &= \Pr[v_{\text{sampled}} = v \mid v \in V \setminus V_{\text{boundary}}] \Pr[v \in V \setminus V_{\text{boundary}}] \\
&\quad + \Pr[v_{\text{sampled}} = v \mid v \in V_{\text{boundary}}] \Pr[v \in V_{\text{boundary}}] \\
&\leq (1 + \epsilon)^2 \frac{\widehat{d}_{\widehat{\ell}(v)}(v)}{2m} + \epsilon \cdot \frac{15(1 + 2\epsilon)\widehat{d}_{\widehat{\ell}(v)}(v)}{2m} \\
&\leq (1 + O(\epsilon)) \cdot \frac{\widehat{d}_{\widehat{\ell}(v)}(v)}{2m}
\end{aligned}$$

Therefore, we have:

$$\begin{aligned}
& \Pr[e \text{ is returned by Algorithm 23}] \\
& \leq (1 + O(\epsilon)) \cdot \frac{\widehat{d}_{\widehat{\ell}(v)}(v)}{2m} \cdot \frac{1}{\widehat{d}_{\widehat{\ell}(v)}(v)} + (1 + O(\epsilon)) \cdot \frac{\widehat{d}_{\widehat{\ell}(u)}(u)}{2m} \cdot \frac{1}{\widehat{d}_{\widehat{\ell}(u)}(u)} \\
& \leq (1 + O(\epsilon)) \cdot \frac{1}{m}.
\end{aligned}$$

The total number of additional (other than those used for edge estimation) BIS queries used is  $O(\log n \cdot Q)$  where  $Q$  is the queries used by Algorithm 22 to return a (near) uniform sample. From Claim 5.7.5, we have that  $Q = O(\epsilon^{-4} \log^5 n \log(\epsilon^{-1} \log n))$ . In Algorithm 22, as we partition each sampled subset  $S_j$ , for every  $j \in L$ , into an additional  $1/\epsilon$  factor many partitions as compared to Algorithm 19. Therefore, we use a total of  $O(\epsilon^{-4} \log^6 n \log(\epsilon^{-1} \log n) + \epsilon^{-6} \log^5 n \log^6(\log n) \log(\epsilon^{-1} \log n))$  BIS queries for edge estimation.

Using union bound, we have that the failure probability in Lemma 5.7.9 and Corollary 5.7.7 is at most  $O(\epsilon)$ . Scaling the  $\epsilon$  appropriately, gives us a failure probability of  $\epsilon$ . Hence, the theorem.  $\square$

## 5.8 Graph Connectivity

In this section, we present Algorithm CONNECTIVITY-BIS that uses 2-rounds of adaptivity to determine the connectivity of an input graph  $G$ . This improves upon a prior three-round algorithm of [16]. In particular, the algorithm of [16] selects  $O(\log^2 n)$  random neighbors per vertex, and contracts the connected components of this random graph into *supernodes*. This random sampling step can be performed using one round of  $\widetilde{O}(n)$  BIS queries. They prove that in the contracted graph on the supernodes, there are at most  $O(n \log n)$  edges. Using this fact, they then show how to identify whether all the supernodes are connected using  $\widetilde{O}(n)$  BIS queries and two additional rounds of adaptivity.

We follow the same basic approach: using a first round of  $\tilde{O}(n)$  queries to randomly sample  $O(\log^2 n)$  neighbors per vertex and contract the graph into supernodes. Once this is done, we observe that we have BIS query access to the contracted graph simply by always grouping together the set of nodes in each supernode. So, we can directly apply the non-adaptive sampling algorithm of Theorem 5.3.2 to sample edges from the contracted graph. By a coupon collecting argument, drawing  $O(n \log^2 n)$  near-uniform edge samples (with replacement) from the contracted graph suffices to recover all  $O(n \log n)$  edges in the graph, and thus determine connectivity of the contracted graph, and, in turn, the original graph.

**Algorithm** CONNECTIVITY-BIS:

1. For every node  $v \in V$ , sample  $O(\log^2 n)$  edges uniformly with replacement, from the neighborhood  $\Gamma(v)$ , using Lemma 5.7.3. Let the resulting set of edges sampled be denoted by  $E' \subseteq E$  and the connected components in the subgraph  $G(V, E')$  be  $S_1, S_2, \dots, S_p$ .
2. Let  $G^{\text{sup}}(V^{\text{sup}}, E^{\text{sup}})$  where  $E^{\text{sup}} \subseteq V^{\text{sup}} \times V^{\text{sup}}$  denotes the supergraph obtained from  $G(V, E')$  by collapsing the connected components  $S_1, S_2, \dots, S_p$  into single supernodes  $s_1, s_2, \dots, s_p$  respectively, given by:

$$V^{\text{sup}} = \{s_i \mid S_i \text{ where } i \in [p] \text{ is a connected component in } G(V, E')\}$$

$$E^{\text{sup}} = \{(s_i, s_j) \mid \exists x \in S_i, y \in S_j \text{ where } i \neq j \text{ such that } (x, y) \in E\}.$$

3. Run Algorithm 23 (with any constant value for  $\epsilon$ ) on  $G^{\text{sup}}$  to draw  $T = O(n \log^2 n)$  uniform superedge samples with replacement, from  $E^{\text{sup}}$ . If the resulting graph  $G^{\text{sup}}$  is connected, output ‘Yes’. Otherwise, output ‘No’.

**Theorem 5.8.1.** *Given a graph  $G$  with  $n$  nodes, there is a 2-round adaptive algorithm that determines if  $G$  is connected with probability at least  $1 - 1/n$  using  $\tilde{O}(n \log^8 n)$  BIS queries, where  $\tilde{O}(\cdot)$  ignores the  $\log^{O(1)} \log n$  dependencies.*

*Proof.* We have:  $|E^{\text{sup}}| = O(n \log n)$  (see Lemma 6.5 in [16]). From Theorem 5.7.10, we have for constant  $\epsilon < 0.5$ :

$$\begin{aligned} \Pr[(s_i, s_j) \in E^{\text{sup}} \text{ is returned}] &\geq (1 - \epsilon) \cdot \frac{(1 - \epsilon)}{|E^{\text{sup}}|} \geq (1 - 2\epsilon) \cdot c \cdot \frac{1}{n \log n} \\ \Rightarrow \Pr[(s_i, s_j) \text{ is not returned}] &= (1 - \Pr[(s_i, s_j) \in E^{\text{sup}} \text{ is returned}])^T \\ &\leq e^{-\frac{2c}{3} \cdot \frac{1}{n \log n} \cdot T} \leq \frac{1}{n^4}. \end{aligned}$$

By union bounding over at most  $O(n \log n)$  many superedges, the total failure probability is at most  $1/n^2$ . Similarly, union bounding over the failure probability of recovering  $O(n \log^2 n)$  edges in the first step, we have that the failure probability is at most  $1/n^2$ . Therefore, Algorithm CONNECTIVITY-BIS recovers all the edges in  $E^{\text{sup}}$  with probability at least  $1 - 1/n$ .

From Lemma 5.7.3, the total number of BIS queries required in the first round of our algorithm is  $O(n \cdot \log^2 n \cdot \log^3 n) = O(n \log^5 n)$ . Setting  $\epsilon$  to be any constant value, from Theorem 5.7.10, the total number of BIS queries required is  $O(n \log^5 n) + \tilde{O}(n \log^2 n \cdot \log^6 n) = \tilde{O}(n \log^8 n)$ , where  $\tilde{O}(\cdot)$  ignores the  $\log \log n$  dependencies. Hence, the theorem.  $\square$

## 5.9 Conclusion

In this chapter, we presented non-adaptive algorithms for edge estimation and sampling using BIS queries. It would be interesting to see if there is a better dependence on  $\epsilon$  than that obtained by our algorithms, when we consider non-adaptive algorithms for edge estimation. Using Independent Set (IS) queries, *adaptive* algorithms for edge estimation with optimal query complexity  $O(\min\{\sqrt{m}, n/\sqrt{m}\} \cdot \text{poly}(\log n, 1/\epsilon))$  were obtained only recently [40, 22]. It would be interesting to see if we can extend our techniques to study *non-adaptive* algorithms for edge estimation using IS queries or in the standard adjacency list model. We believe that adaptivity

plays a role similar to that of number of passes for streaming algorithms, and optimizing for the rounds of adaptivity could be a good future direction, even for problems that might already have query optimal adaptive sub-linear time algorithms.

## BIBLIOGRAPHY

- [1] Abasi, Hasan, and Nader, Bshouty. On learning graphs with edge-detecting queries. In *In 30th International Conference on Algorithmic Learning Theory (ALT)* (2019), pp. 3–30.
- [2] Acharya, Jayadev, Bhattacharyya, Arnab, Daskalakis, Constantinos, and Kandasamy, Saravanan. Learning and testing causal models with interventions. In *Advances in Neural Information Processing Systems 31 (NeurIPS)* (2018), pp. 9469–9481.
- [3] Adamic, Lada A, and Huberman, Bernardo A. Power-law distribution of the world wide web. *Science* 287, 5461 (2000), 2115–2115.
- [4] Addanki, Raghavendra, and Kasiviswanathan, Shiva. Collaborative causal discovery with atomic interventions. *Advances in Neural Information Processing Systems 34 (NeurIPS) 34* (2021).
- [5] Addanki, Raghavendra, Kasiviswanathan, Shiva, McGregor, Andrew, and Musco, Cameron. Efficient intervention design for causal discovery with latents. In *Proceedings of the 37th International Conference on Machine Learning (ICML)* (2020), PMLR, pp. 63–73.
- [6] Addanki, Raghavendra, McGregor, Andrew, and Musco, Cameron. Intervention efficient algorithms for approximate learning of causal graphs. In *The 32nd International Conference on Algorithmic Learning Theory International Conference on Algorithmic Learning Theory (ALT)* (2021), PMLR, pp. 151–184.
- [7] Addanki, Raghavendra, McGregor, Andrew, and Musco, Cameron. Non-adaptive edge counting and sampling via bipartite independent set queries. *30th Annual European Symposium on Algorithms (ESA)* (2022).
- [8] Agarwal, Arpit, Agarwal, Shivani, Assadi, Sepehr, and Khanna, Sanjeev. Learning with limited rounds of adaptivity: Coin tossing, multi-armed bandits, and ranking from pairwise comparisons. In *Proceedings of the 30th Annual Conference on Computational Learning Theory (COLT)* (2017), PMLR, pp. 39–75.
- [9] Ahn, Kook Jin, Guha, Sudipto, and McGregor, Andrew. Analyzing graph structure via linear measurements. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2012), SIAM, pp. 459–467.

- [10] Aliakbarpour, Maryam, Biswas, Amartya Shankha, Gouleakis, Themis, Peebles, John, Rubinfeld, Ronitt, and Yodpinyanee, Anak. Sublinear-time algorithms for counting star subgraphs via edge sampling. *Algorithmica* 80, 2 (2018), 668–697.
- [11] Alon, Noga, Seymour, Paul, and Thomas, Robin. A separator theorem for graphs with an excluded minor and its applications. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC)* (1990), pp. 293–299.
- [12] Angluin, Dana, and Chen, Jiang. Learning a hidden graph using  $o(\log n)$  queries per edge. *Journal of Computer and System Sciences* 74, 4 (2008), 546–556.
- [13] Aronov, Boris, and Har-Peled, Sariel. On approximating the depth and related problems. *SIAM Journal on Computing* 38, 3 (2008), 899–921.
- [14] Arora, Sanjeev, and Chlamtac, Eden. New approximation guarantee for chromatic number. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC)* (2006), pp. 215–224.
- [15] Ashtiani, Hassan, Kushagra, Shrinu, and Ben-David, Shai. Clustering with same-cluster queries. In *Advances in Neural Information Processing Systems 29 (NeurIPS)* (2016).
- [16] Assadi, Sepehr, Chakrabarty, Deeparnab, and Khanna, Sanjeev. Graph connectivity and single element recovery via linear and OR queries. In *29th Annual European Symposium on Algorithms, ESA 2021* (2021), Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 7:1–7:19.
- [17] Awasthi, Pranjali, Blum, Avrim, and Sheffet, Or. Center-based clustering under perturbation stability. *Information Processing Letters* 112, 1-2 (2012), 49–54.
- [18] Balkanski, Eric, and Singer, Yaron. The adaptive complexity of maximizing a submodular function. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC)* (2018), pp. 1138–1151.
- [19] Bardenet, Rémi, and Maillard, Odalric-Ambrym. Concentration inequalities for sampling without replacement. *Bernoulli* 21, 3 (Aug 2015), 1361–1385.
- [20] Bareinboim, Elias, Brito, Carlos, and Pearl, Judea. Local characterizations of causal bayesian networks. In *Graph Structures for Knowledge Representation and Reasoning*. Springer, 2012, pp. 1–17.
- [21] Bareinboim, Elias, and Pearl, Judea. Causal inference and the data-fusion problem. *Proceedings of the National Academy of Sciences* 113, 27 (2016), 7345–7352.
- [22] Beame, Paul, Har-Peled, Sariel, Ramamoorthy, Sivaramakrishnan Natarajan, Rashtchian, Cyrus, and Sinha, Makrand. Edge estimation with independent set oracles. *ACM Transactions on Algorithms (TALG)* 16, 4 (2020), 1–27.

- [23] Behnezhad, Soheil. Time-optimal sublinear algorithms for matching and vertex cover. In *Proceedings of the 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)* (2021), IEEE, pp. 873–884.
- [24] Bhattacharya, Anup, Bishnu, Arijit, Ghosh, Arijit, and Mishra, Gopinath. Hyperedge estimation using polylogarithmic subset queries. [arXiv:1908.04196](#) (2019).
- [25] Bhattacharya, Anup, Bishnu, Arijit, Ghosh, Arijit, and Mishra, Gopinath. On triangle estimation using tripartite independent set queries. *Theory of Computing Systems* (2021), 1–28.
- [26] Bhattacharya, Anup, Bishnu, Arijit, Ghosh, Arijit, and Mishra, Gopinath. Faster counting and sampling algorithms using colorful decision oracle. In *Proceedings of the 39th International Symposium on Theoretical Aspects of Computer Science (STACS)* (2022).
- [27] Bhattacharyya, Arnab, Gayen, Sutanu, Kandasamy, Saravanan, Maran, Ashwin, and Vinodchandran, NV. Efficiently learning and sampling interventional distributions from observations. *Proceedings of the 37th International Conference on Machine Learning (ICML)* (2020).
- [28] Bhattacharyya, Arnab, Gayen, Sutanu, Kandasamy, Saravanan, Raval, Vedant, and Vinodchandran, NV. Efficient interventional distribution learning in the pac framework. *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS)* (2022).
- [29] Bishnu, Arijit, Ghosh, Arijit, Kolay, Sudeshna, Mishra, Gopinath, and Saurabh, Saket. Parameterized query complexity of hitting set using stability of sunflowers. In *29th International Symposium on Algorithms and Computation* (2018).
- [30] Bishnu, Arijit, Ghosh, Arijit, Mishra, Gopinath, and Paraashar, Manaswi. Efficiently sampling and estimating from substructures using linear algebraic queries. [arXiv:1906.07398](#) (2019).
- [31] Biswas, Amartya Shankha, Eden, Talya, and Rubinfeld, Ronitt. Towards a decomposition-optimal algorithm for counting and sampling arbitrary motifs in sublinear time. In *Proceedings of the 25th International Workshop on Randomization and Computation (RANDOM)* (2021).
- [32] Blum, Avrim, and Karger, David. An  $O(n^{3/14})$ -coloring algorithm for 3-colorable graphs. *Information processing letters* 61, 1 (1997), 49–53.
- [33] Cabello, Sergio, and Jejčič, Miha. Shortest paths in intersection graphs of unit disks. *Computational Geometry* 48, 4 (2015), 360–367.
- [34] Canonne, Clément L, Diakonikolas, Ilias, Kane, Daniel M, and Stewart, Alistair. Testing conditional independence of discrete distributions. In *2018 Information Theory and Applications Workshop (ITA)* (2018), IEEE, pp. 1–57.

- [35] Canonne, Clément L, and Gur, Tom. An adaptivity hierarchy theorem for property testing. *computational complexity* 27, 4 (2018), 671–716.
- [36] Chakrabarti, Amit, and Stoeckl, Manuel. The element extraction problem and the cost of determinism and limited adaptivity in linear queries. *arXiv:2107.05810* (2021).
- [37] Chan, Siu-On, Diakonikolas, Ilias, Valiant, Paul, and Valiant, Gregory. Optimal algorithms for testing closeness of discrete distributions. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2014), SIAM, pp. 1193–1203.
- [38] Chekuri, Chandra, and Quanrud, Kent. Parallelizing greedy for submodular set function maximization in matroids and beyond. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)* (2019), ACM, pp. 78–89.
- [39] Chen, Chao L, and Swallow, William H. Using group testing to estimate a proportion, and to test the binomial model. *Biometrics* (1990), 1035–1046.
- [40] Chen, Xi, Levi, Amit, and Waingarten, Erik. Nearly optimal edge estimation with independent set queries. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2020), SIAM, pp. 2916–2935.
- [41] Chickering, David Maxwell. Optimal structure identification with greedy search. *Journal of machine learning research* 3, Nov (2002), 507–554.
- [42] Colombo, Diego, Maathuis, Marloes H, Kalisch, Markus, and Richardson, Thomas S. Learning high-dimensional directed acyclic graphs with latent and selection variables. *The Annals of Statistics* (2012), 294–321.
- [43] Cormen, Thomas H, Leiserson, Charles E, Rivest, Ronald L, and Stein, Clifford. *Introduction to Algorithms*. MIT press, 2009.
- [44] Cormode, Graham, and Muthukrishnan, Shan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms* 55, 1 (2005), 58–75.
- [45] Czumaj, Artur, Shapira, Asaf, and Sohler, Christian. Testing hereditary properties of nonexpanding bounded-degree graphs. *SIAM Journal on Computing* 38, 6 (2009), 2499–2510.
- [46] Dell, Holger, and Lapinskas, John. Fine-grained reductions from approximate counting to decision. *ACM Transactions on Computation Theory (TOCT)* 13, 2 (2021), 1–24.
- [47] Dell, Holger, Lapinskas, John, and Meeks, Kitty. Approximately counting and sampling small witnesses using a colourful decision oracle. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2020), SIAM, pp. 2201–2211.

- [48] Dorfman, Robert. The detection of defective members of large populations. *The Annals of Mathematical Statistics* 14, 4 (1943), 436–440.
- [49] Du, Dingzhu, Hwang, Frank K, and Hwang, Frank. *Combinatorial group testing and its applications*, vol. 12. World Scientific, 2000.
- [50] Eberhardt, Frederick. Causation and intervention. *PhD Thesis, Carnegie Mellon University* (2007).
- [51] Eberhardt, Frederick, and Scheines, Richard. Interventions and causal inference. *Philosophy of Science* 74, 5 (2007), 981–995.
- [52] Eden, Talya, Ron, Dana, and Seshadhri, C. On approximating the number of  $k$ -cliques in sublinear time. *SIAM Journal on Computing* 49, 4 (2020), 747–771.
- [53] Eden, Talya, and Rosenbaum, Will. On sampling edges almost uniformly. In *1st Symposium on Simplicity in Algorithms (SOSA)* (2018).
- [54] Feige, Uriel. On sums of independent random variables with unbounded variance and estimating the average degree in a graph. *SIAM Journal on Computing* 35, 4 (2006), 964–984.
- [55] Feige, Uriel, Goldwasser, Shafi, Lovász, Laszlo, Safra, Shmuel, and Szegedy, Mario. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM (JACM)* 43, 2 (1996), 268–292.
- [56] Fishkin, Aleksei V. Disk graphs: A short survey. In *International Workshop on Approximation and Online Algorithms* (2003), Springer, pp. 260–264.
- [57] Frank, Andras. Some polynomial algorithms for certain graphs and hypergraphs. In *Proceedings of the 5th British Combinatorial Conference* (1975).
- [58] Gates, Kathleen M, and Molenaar, Peter CM. Group search algorithm recovers effective connectivity maps for individuals in homogeneous and heterogeneous samples. *NeuroImage* 63, 1 (2012), 310–319.
- [59] Ghassami, AmirEmad, Salehkaleybar, Saber, and Kiyavash, Negar. Interventional experiment design for causal structure learning. [arXiv:1910.05651](https://arxiv.org/abs/1910.05651) (2019).
- [60] Ghassami, AmirEmad, Salehkaleybar, Saber, Kiyavash, Negar, and Bareinboim, Elias. Budgeted experiment design for causal structure learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)* (2018), PMLR, pp. 1724–1733.
- [61] Glymour, Clark, Zhang, Kun, and Spirtes, Peter. Review of causal discovery methods based on graphical models. *Frontiers in genetics* 10 (2019), 524.
- [62] Goldreich, Oded, Goldwasser, Shafi, and Ron, Dana. Property testing and its connection to learning and approximation. *Journal of the ACM (JACM)* 45, 4 (1998), 653–750.

- [63] Goldreich, Oded, and Ron, Dana. Approximating average parameters of graphs. *Random Structures & Algorithms* 32, 4 (2008), 473–493.
- [64] Hassidim, Avinatan, Kelner, Jonathan A, Nguyen, Huy N, and Onak, Krzysztof. Local graph partitions for approximation and testing. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)* (2009), pp. 22–31.
- [65] Hauser, Alain, and Bühlmann, Peter. Characterization and greedy learning of interventional markov equivalence classes of directed acyclic graphs. *Journal of Machine Learning Research* 13, Aug (2012), 2409–2464.
- [66] Hauser, Alain, and Bühlmann, Peter. Two optimal strategies for active learning of causal models from interventional data. *International Journal of Approximate Reasoning* 55, 4 (2014), 926–939.
- [67] He, Yang-Bo, and Geng, Zhi. Active learning of causal networks with intervention experiments and optimal designs. *Journal of Machine Learning Research* 9, Nov (2008), 2523–2547.
- [68] Heinze-Deml, Christina, Maathuis, Marloes H, and Meinshausen, Nicolai. Causal structure learning. *Annual Review of Statistics and Its Application* 5 (2018), 371–391.
- [69] Hoeffding, Wassily. Probability inequalities for sums of bounded random variables. In *The Collected Works of Wassily Hoeffding*. Springer, 1994, pp. 409–426.
- [70] Hoyer, Patrik O, Janzing, Dominik, Mooij, Joris M, Peters, Jonas, and Schölkopf, Bernhard. Nonlinear causal discovery with additive noise models. In *Advances in Neural Information Processing Systems 22 (NeurIPS)* (2009), pp. 689–696.
- [71] Hyttinen, Antti, Eberhardt, Frederick, and Hoyer, Patrik O. Experiment selection for causal discovery. *The Journal of Machine Learning Research* 14, 1 (2013), 3041–3071.
- [72] Hyttinen, Antti, Hoyer, Patrik O, Eberhardt, Frederick, and Jarvisalo, Matti. Discovering cyclic causal models with latent variables: A general sat-based procedure. [arXiv:1309.6836](https://arxiv.org/abs/1309.6836) (2013).
- [73] Indyk, Piotr, Ngo, Hung Q, and Rudra, Atri. Efficiently decodable non-adaptive group testing. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2010), SIAM, pp. 1126–1142.
- [74] Indyk, Piotr, Price, Eric, and Woodruff, David P. On the power of adaptivity in sparse recovery. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)* (2011), IEEE, pp. 285–294.

- [75] Jaber, Amin, Kocaoglu, Murat, Shanmugam, Karthikeyan, and Bareinboim, Elias. Causal discovery from soft interventions with unknown targets: Characterization and learning. *Advances in Neural Information Processing Systems 33 (NeurIPS)* (2020).
- [76] Joffe, Michael, Gambhir, Manoj, Chadeau-Hyam, Marc, and Vineis, Paolo. Causal diagrams in systems epidemiology. *Emerging themes in epidemiology* 9, 1 (2012), 1–18.
- [77] Jukna, Stasys. *Extremal combinatorics: with applications in computer science*. Springer Science & Business Media, 2011.
- [78] Kalisch, Markus, Mächler, Martin, Colombo, Diego, Maathuis, Marloes H, and Bühlmann, Peter. Causal inference using graphical models with the r package pcalg. *Journal of Statistical Software* 47, 11 (2012), 1–26.
- [79] Kamath, Akshay, and Price, Eric. Adaptive sparse recovery with limited adaptivity. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2019), pp. 2729–2744.
- [80] Karger, David, Motwani, Rajeev, and Sudan, Madhu. Approximate graph coloring by semidefinite programming. In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science (FOCS)* (1994), IEEE, pp. 2–13.
- [81] Karloff, Howard, Suri, Siddharth, and Vassilvitskii, Sergei. A model of computation for mapreduce. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2010), SIAM, pp. 938–948.
- [82] Katona, Gyula. On separating systems of a finite set. *Journal of Combinatorial Theory* 1, 2 (1966), 174–194.
- [83] Katz, Dmitriy, Shanmugam, Karthikeyan, Squires, Chandler, and Uhler, Caroline. Size of interventional markov equivalence classes in random dag models. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)* (2019), PMLR.
- [84] Khalil, Lidiya Khalidah binti, and Konrad, Christian. Constructing large matchings via query access to a maximal matching oracle. In *Proceedings of the 40th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)* (2020).
- [85] Kisvölcsy, Ákos. Flattening antichains. *Combinatorica* 1, 26 (2006), 65–82.
- [86] Kocaoglu, Murat, Dimakis, Alex, and Vishwanath, Sriram. Cost-optimal learning of causal graphs. In *Proceedings of the 34th International Conference on Machine Learning (ICML)* (2017), pp. 1875–1884.

- [87] Kocaoglu, Murat, Jaber, Amin, Shanmugam, Karthikeyan, and Bareinboim, Elias. Characterization and learning of causal graphs with latent variables from soft interventions. In *Advances in Neural Information Processing Systems 32 (NeurIPS)* (2019), pp. 14346–14356.
- [88] Kocaoglu, Murat, Shanmugam, Karthikeyan, and Bareinboim, Elias. Experimental design for learning causal graphs with latent variables. In *Advances in Neural Information Processing Systems 30 (NeurIPS)* (2017), pp. 7018–7028.
- [89] Kruskal, Joseph B. The number of simplices in a complex. *Mathematical Optimization Techniques 10* (1963), 251–278.
- [90] Lauritzen, Steffen L, and Spiegelhalter, David J. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society: Series B (Methodological)* 50, 2 (1988), 157–194.
- [91] Lindgren, Erik, Kocaoglu, Murat, Dimakis, Alexandros G, and Vishwanath, Sriram. Experimental design for cost-aware learning of causal graphs. In *Advances in Neural Information Processing Systems 31 (NeurIPS)* (2018), pp. 5279–5289.
- [92] Loh, Po-Ling, and Bühlmann, Peter. High-dimensional learning of linear causal networks via inverse covariance estimation. *The Journal of Machine Learning Research* 15, 1 (2014), 3065–3105.
- [93] MacWilliams, Florence Jessie, and Sloane, Neil James Alexander. *The theory of error-correcting codes*, vol. 16. Elsevier, 1977.
- [94] Mao-Cheng, Cai. On separating systems of graphs. *Discrete Mathematics* 49, 1 (1984), 15–20.
- [95] McGregor, Andrew. Graph stream algorithms: a survey. *ACM SIGMOD Record* 43, 1 (2014), 9–20.
- [96] Montanaro, Ashley, and Shao, Changpeng. Quantum algorithms for learning graphs and beyond. [arXiv:2011.08611](https://arxiv.org/abs/2011.08611) (2020).
- [97] Nakos, Vasileios, Shi, Xiaofei, Woodruff, David P, and Zhang, Hongyang. Improved algorithms for adaptive compressed sensing. In *Proceedings of the 45th International Colloquium on Automata, Languages and Programming (ICALP)* (2018).
- [98] Neykov, Matey, Balakrishnan, Sivaraman, and Wasserman, Larry. Minimax optimal conditional independence testing. *The Annals of Statistics* 49, 4 (2021), 2151–2177.
- [99] Nguyen, Dang Trinh, Duong, Quoc Bao, Zamai, Eric, and Shahzad, Muhammad Kashif. Fault diagnosis for the complex manufacturing system. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability* 230, 2 (2016), 178–194.

- [100] Nisan, Noam. The demand query model for bipartite matching. *Proceedings of the 32nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2021).
- [101] Onak, Krzysztof, Ron, Dana, Rosen, Michal, and Rubinfeld, Ronitt. A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2012), SIAM, pp. 1123–1131.
- [102] Parviainen, Pekka, and Koivisto, Mikko. Ancestor relations in the presence of unobserved variables. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (2011), Springer, pp. 581–596.
- [103] Pearl, Judea. Causal diagrams for empirical research. *Biometrika* 82, 4 (1995), 669–688.
- [104] Pearl, Judea. *Causality: models, reasoning and inference*, vol. 29. Springer, 2000.
- [105] Pearl, Judea. *Causality: Models, Reasoning, and Inference*. Cambridge university press, 2009.
- [106] Ramsey, JD, Spirtes, Peter, and Glymour, Clark. On meta-analyses of imaging data and the mixture of records. *NeuroImage* 57, 2 (2011), 323–330.
- [107] Rashtchian, Cyrus, Woodruff, David P, and Zhu, Hanlin. Vector-matrix-vector queries for solving linear algebra, statistics, and graph problems. In *Proceedings of the 24th International Workshop on Randomization and Computation (RANDOM)* (2020).
- [108] Richardson, Thomas, Spirtes, Peter, et al. Ancestral graph markov models. *The Annals of Statistics* 30, 4 (2002), 962–1030.
- [109] Ron, Dana. Algorithmic and analysis techniques in property testing. In *Now Publishers Inc* (2010).
- [110] Ron, Dana. Sublinear-time algorithms for approximating graph parameters. In *Computing and Software Science*. Springer, 2019, pp. 105–122.
- [111] Ron, Dana, and Tsur, Gilad. The power of an example: Hidden set size approximation using group queries and conditional sampling. *ACM Transactions on Computation Theory (TOCT)* 8, 4 (2016), 1–19.
- [112] Rubinfeld, Aviad, Schramm, Tselil, and Weinberg, S. Matthew. Computing Exact Minimum Cuts Without Knowing the Graph. In *Proceedings of the 9th Conference on Innovations in Theoretical Computer Science (ITCS)* (2018), vol. 94, pp. 39:1–39:16.

- [113] Saeed, Basil, Panigrahi, Snigdha, and Uhler, Caroline. Causal structure discovery from distributions arising from mixtures of dags. In *Proceedings of the 37th International Conference on Machine Learning (ICML)* (2020), PMLR, pp. 8336–8345.
- [114] Seshadhri, C. A simpler sublinear algorithm for approximating the triangle count. *arXiv:1505.01927* (2015).
- [115] Shanmugam, Karthikeyan, Kocaoglu, Murat, Dimakis, Alexandros G, and Vishwanath, Sriram. Learning causal graphs with small interventions. In *Advances in Neural Information Processing Systems 28 (NeurIPS)* (2015), pp. 3195–3203.
- [116] Shimizu, Shohei, Hoyer, Patrik O, Hyvärinen, Aapo, and Kerminen, Antti. A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research* 7, Oct (2006), 2003–2030.
- [117] Shpitser, Ilya, and Pearl, Judea. Identification of joint interventional distributions in recursive semi-markovian causal models. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA* (2006), pp. 1219–1226.
- [118] Silva, Ricardo, Scheine, Richard, Glymour, Clark, and Spirtes, Peter. Learning the structure of linear latent variable models. *Journal of Machine Learning Research* 7, Feb (2006), 191–246.
- [119] Spirtes, Peter, Glymour, Clark N, Scheines, Richard, Heckerman, David, Meek, Christopher, Cooper, Gregory, and Richardson, Thomas. *Causation, prediction, and search*. MIT press, 2000.
- [120] Spirtes, Peter, Meek, Christopher, and Richardson, Thomas. An algorithm for causal inference in the presence of latent variables and selection bias. *Computation, causation, and discovery* 21 (1999), 1–252.
- [121] Stockmeyer, Larry. The complexity of approximate counting. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing (STOC)* (1983), pp. 118–126.
- [122] Stockmeyer, Larry. On approximation algorithms for  $\#$  p. *SIAM Journal on Computing* 14, 4 (1985), 849–861.
- [123] Sussex, Scott, Uhler, Caroline, and Krause, Andreas. Near-optimal multi-perturbation experimental design for causal structure learning. *Advances in Neural Information Processing Systems 34 (NeurIPS)* 34 (2021).
- [124] Tian, Jin, and Pearl, Judea. Causal discovery from changes. In *Proceedings of the 17th Annual Conference on Uncertainty in Artificial Intelligence (UAI)* (2001), pp. 512–521.

- [125] Tian, Jin, and Shpitser, Ilya. On the identification of causal effects. In *Citeseer* (2003).
- [126] Tětek, Jakub, and Thorup, Mikkel. Sampling and counting edges via vertex accesses. *Proceedings of the 54th Annual ACM Symposium on Theory of Computing (STOC)* (2022).
- [127] Verma, Thomas, and Pearl, Judea. An algorithm for deciding if a set of observed independencies has a causal explanation. In *Proceedings of the 8th Annual Conference on Uncertainty in Artificial Intelligence (UAI)* (1992), Elsevier, pp. 323–330.
- [128] Wigderson, Avi. Improving the performance guarantee for approximate graph coloring. *Journal of the ACM (JACM)* 30, 4 (1983), 729–735.
- [129] Williamson, David P, and Shmoys, David B. *The design of approximation algorithms*. Cambridge university press, 2011.
- [130] Yang, Karren, Katcoff, Abigail, and Uhler, Caroline. Characterizing and learning equivalence classes of causal dags under interventions. In *Proceedings of the 35th International Conference on Machine Learning (ICML)* (2018), PMLR, pp. 5541–5550.
- [131] Yao, Andrew Chi-Chin. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th Annual IEEE Symposium on Foundations of Computer Science (FOCS)* (1977), IEEE, pp. 222–227.
- [132] Zhang, Jiji. Causal reasoning with ancestral graphs. *Journal of Machine Learning Research* 9, Jul (2008), 1437–1474.
- [133] Zhang, Jiji. On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. *Artificial Intelligence* 172, 16-17 (2008), 1873–1896.
- [134] Zhang, Kun, Peters, Jonas, Janzing, Dominik, and Schölkopf, Bernhard. Kernel-based conditional independence test and application in causal discovery. In *Proceedings of the 27th Annual Conference on Uncertainty in Artificial Intelligence (UAI)* (2011), AUAI Press, pp. 804–813.