Inapproximability of Combinatorial Problems in Subexponential-Time

Bundit Laekhanukit

Doctor of Philosophy

School of Computer Science

McGill University

Montreal,Quebec

2014-23-06

A thesis submitted to McGill University
in partial fulfillment of the requirements of the degree of
Doctor of Philosophy

DEDICATION

This thesis is dedicated to my family, especially, my grandfather who passed away a year prior to the finishing of this dissertation.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Adrian Vetta, who has continuously supported me throughout my Ph.D. study. I would like to express my sincere gratitude to him for his help, guidance and support, and also for his patience. Without him, this thesis would have never been done. My sincere thanks also go to my former advisors, Joseph Cheriyan and Jittat Fakcharoenphol who supported and patiently supervised me during my masters (M.Math and M.Eng). I also would like to thank Bruce Shepherd for his guidance. Two of the most important people I would like to thank are my friends and colleagues, Danupon Nanongkai and Parinya Chalermsook. The results in this thesis are the product of our collaborations over many years. During Summer 2012 and while writing this thesis, I have been partly supported by Fabrizio Grandoni. I would like to thank him for his generous support. I would also like to thank the committee members and examiners, Hamed Hatami, Jochen Könemann, Luc Devroye, Sergey Norin, Alyson Fournier and Muthucumaru Maheswaran. Last but not least, I would like to thank to my family for supporting me throughout my life.

Contribution of the author

The materials in this thesis come from [17, 18], which are joint work with Parinya Chalermsook and Danupon Nanongkai. The results from [17] appear partly in Chapter 6. The results from [18] appear in Chapter 7 and Chapter 8. In [17], The author worked as a part of the team and contributed in verifying the results. In [18], the key idea in proving "new property of dispersers" came from the author. The author also fine-tuned and and simplified the analysis of the proofs in [18]. The approximation scheme in [18] also came from the observation of the author. The author recently proved new results in Chapter 5 and some results in Chapter 6 to bind the materials from the two articles together. More precisely, to the best of our knowledge, the proof of subexponential-time approximation hardness using the graph product technique has not been known prior to this thesis.

ABSTRACT

Many natural combinatorial optimization problems turn out to be NP-hard. A standard way to cope with these problems is to design an algorithm that outputs an approximate solution and still runs in polynomial-time. Although a decent solution can sometime be found, many of the natural problems that we encounter, e.g., the maximum independent set and graph coloring problems, resist almost every approximation algorithm. In fact, assuming P $\neq$ NP, there exists a barrier that prevents us to get arbitrarily close to an optimal solution. Thus, understanding the limit in which one can find an approximate solution is an important subject in theoretical computer science. This thesis studies the inapproximability of combinatorial optimization problems in the fashion of time-approximability trade-off. In particular, assuming the exponential-time hypothesis, we show that even allowing an algorithm to run in subexponential-time, some problems cannot be approximated to within some factors.

ABRÉGÉ

Beaucoup de problèmes d'optimisation combinatoire naturelle s'avèrentêtre des problèmes NP. Une méthode standard pour faire face à ces problèmes consiste à concevoir un algorithme qui fournit une solution approchée et qui fonctionne toujours en temps polynomial. Même si une solution dècente peut parfois être trouvée, de nombreux problèmes physiques que nous rencontrons, par exemple, un ensemble indépendant maximal et des problmes de coloration des graphes, résistent à presque chaque algorithme d'approximation. En fait, en supposant que P $\neq$ NP, il existe une barriére qui nous empêche d'obtenir arbitrairement proche de la solution optimale. Ainsi, en comprenant la limite dans laquelle on peut une solution approchée est un sujet important dans l'informatique théorique. Cette thèse étudie l'inapproximabilité des problèmes d'optimisation combinatoire sous la forme d'un un compromis au niveau de l'approximabilité du temps. En particulier, en nous basant sur l'hypothse exponentielle-temps, nous montrons que même en permettant à un algorithme d'être exécuté en temps sous-exponentiel, certains problèmes ne peuvent pas être estimés dans certains facteurs.

TABLE OF CONTENTS

DEDICATION ii

ACKNOWLEDGMENTS iii

ABSTRACT v

ABRÉGÉ vi

LIST OF TABLES x

LIST OF FIGURES xi

1 Introduction 1

1.1 The Approximability of Optimization Problems 5
1.2 Proving Hardness of Approximation 6

2 Preliminaries and Background 9

2.1 Preliminaries 9
2.2 Problem Definitions 11
2.3 Complexity Terminology 14
2.4 A Probabilistically Checkable Proof System 15
2.4.1 The FGLSS reduction 17

3 Exponential-Time Hypothesis and Almost Linear-Size PCPs 21

3.1 Exponential-Time Hypothesis 21
3.2 (Almost) Linear-Size PCP 22
3.3 Randomized PCP for Graph Coloring 24

4 Graph Product 30

4.1 Properties of Disjunctive Product 32
4.2 Randomized Graph Product 40

4.2.1 Independence Ratio of Randomized Graph Product 42
4.2.2 Chromatic Number of Randomized Graph Product 45

5 Subexponential-Time Approximation Hardness of Classical Results . . . 48

5.1 Overview 51
5.2 Independent Set 52
5.3 Graph Coloring 56

6 Subexponential-Time Hardness from Graph Product 61

6.1 Our Results 62
6.2 Overview 64
6.3 The Proof of Subexponential-Time Approximation Hardness . . . 66
6.4 Graph Product Inequality 69

7 Subexponential-Time Hardness of Problems on Bounded-Degree Graphs 74

7.1 Overview 74
7.1.1 Step II: independent set → induced matching 79
7.2 An Almost-Linear Size Reduction from SAT to CSP 82
7.3 FGLSS and Dispersers Replacement 86
7.4 Tight Hardness of Semi-Induced Matching 87
7.4.1 The Reduction 88
7.4.2 Analysis 92
7.4.3 Subexponential Time Approximation Hardness for the Maximum Independent Set and Induced Matching Problems 97
7.4.4 Subexponential-Time Approximation Algorithm for Induced Matching 99

8 The Hardness of Approximating k-Hypergraph Pricing 103

8.1 Overview 104
8.1.1 The Main Reduction: SAT → pricing 104
8.1.2 An Intermediate Reduction: Induced matching → Pricing . 105
8.2 The Approximation Hardness of k-Hypergraph Pricing 105
8.2.1 From Semi-Induced Matching to Pricing Problems 107
8.2.2 Intermediate Hardness 113
8.2.3 The Hardness Results (Proof of Theorem 8.2.1) 114

8.2.4 Subexponential-Time Approximation Hardness for the k-Hypergraph Pricing Problem . . . 115
8.3 Approximation Scheme for k-Hypergraph Pricing . . . 116
8.3.1 Approximation Scheme . . . 117
8.3.2 Cost Analysis . . . 118
8.3.3 Running Time Analysis . . . 121
8.3.4 Polynomial-Time $O(\sqrt{n \log n})$-Approximation Algorithm . 121
8.3.5 $O(1)$-Approximation in Time $O((\log nm)^n \text{poly}(n, m))$ . . . 122

9 Conclusion . . . 125

REFERENCES . . . 126

LIST OF TABLES

Table page

1–1 Results in this thesis 5

LIST OF FIGURES

Figure page

2–1 An example of the FGLSS reduction. 19

6–1 Example of graphs G, H, $B[G]$, $B[H]$ and $B[G \vee H]$. 70

6–2 Illustration of the sets of edges E_G and E_H. 72

CHAPTER 1
Introduction

An abundance of real-world computational tasks require the optimization of an objective function. Consequently, optimization has become a fundamental area of research in theoretical computer science.

For many optimization problems, a solution can be obtained via an efficient algorithm, that is, one that runs in time polynomial in the input size. In contrast, some problems *seem* to be solvable only using a super-polynomial amount of running time. Such problems are classified as *hard*. In particular, many natural problems are NP-hard and, to date, no polynomial-time algorithms are known that can solve an NP-hard problem optimally. Indeed, it is widely believed that no such algorithm can exist – this belief is formalized in the conjecture that $P \neq NP$.

As a result, computer scientists have studied alternative ways of solving hard problems, for example, approximation algorithms and heuristics, which output approximate rather than optimal solutions. Unfortunately, sometimes even finding an approximate solution is hard! Specifically, for some problems obtaining a solution with an approximation guarantee better than a specified factor is equivalent to solving an NP-hard problem. The most infamous example of this is the *traveling salesperson problem*: for general distance functions, finding an approximate solution is as hard as finding an exact solution. In other words, no efficient algorithm can yield any worst case approximation guarantee for the traveling salesperson problem unless

P = NP. Two other examples are the *graph coloring* problem and the *maximum independent set* problem.

In the 1970s, during the time when the theory of NP-completeness was being developed [24, 52, 49], Johnson [48] gave an $O(n/\log n)$-approximation algorithm for the graph coloring problem, where n is the number of vertices in the graph. It took two decades to improve the approximation guarantee. Even then, the improvement was very slight: the current best bound is $O(n(\log\log n)^2/(\log n)^3)$ due to Halldórsson [40]. Thus, despite a huge amount of effort little substantial progress was made on the problem. The situation was even worse for the maximum independent set problem. There, the first non-trivial approximation algorithm was only discovered in 1990 by Boppana and Halldórsson [9], and the best approximation ratio is still only $O(n(\log\log n)^2/(\log n)^3)$ by Feige [32]. The question of whether these two fundamental graph problems, graph coloring and maximum independent set, admit decent approximations was finally answered in the negative. First, in seminal work Feige et al. [33] showed that the maximum independent set problem cannot be approximated to within a factor of $2^{\log^{1-\epsilon} n}$ unless NP $\subseteq$ DTIME($2^{\text{polylog}n}$). the lower bound has since been steadily improved in a sequence of works [3, 2, 6, 8, 42]. Second, the inapproximability of the graph coloring problem was proven by Lund and Yanakakis [57]. Based upon the hardness of the maximum independent set problem, they proved that the graph coloring problem does not admit an n^δ-approximation algorithm, for some fixed constant $\delta > 0$, unless P = NP. Later works lead to a better understanding of approximation thresholds for these problems. It is now known that

both the graph coloring and the maximum independent set problems admit no $n^{1-\epsilon}$-approximation algorithm, for all $\epsilon > 0$, unless NP = ZPP. These theorems are due to Håstad [42] and Feige and Kilian [34], respectively. Furthermore, these results were derandomized by Zuckerman [70]. Thus, the required complexity assumption is now simply P $\neq$ NP. We remark that the best known approximation lower bounds of the two problems is $n/2^{\log^{3/4+\epsilon} n}$, for any $\epsilon > 0$, assuming NP $\subsetneq$ BPTIME($2^{\text{polylog}n}$), due to Khot and Ponnuswami [51]. This essentially rules out the possibility of improving upon the best known approximation guarantees for the two problems.

So, both the independent set and graph coloring problems are almost inapproximable. The lower bounds described above, however, only apply to polynomial-time algorithms. This prompts the question as to whether the approximation guarantee can be improved by allowing an algorithm to run in superpolynomial (but, subexponential) time. The answer is *yes* – it was shown that the independent set and graph coloring problems both admit r-approximation algorithms that run in time $O(2^{n/r}\text{poly}(n))$ (see [25, 11, 10]). These bounds break the approximation barrier of $n^{1-\epsilon}$ for polynomial-time algorithms. What are the limits of this approach? Specifically, are there any lower bounds on the quality of subexponential-time algorithms? This question is open but the *exponential-time hypothesis* (ETH) of Impagliazzo and Paturi [45] conjectures that 3-SAT cannot be solved in subexponential time. Assuming the ETH, any problem that has an almost linear-size reduction from 3-SAT inherits the same subexponential-time lower bound. In particular, Moshkovitz and Raz [59] recently gave an almost linear-size reduction from 3-SAT to its maximization version, Max 3-SAT, with a hardness gap of $8/7 - o(1)$. So, given that solving

3-SAT requires exponential-time, approximating Max 3-SAT to within a factor of $8/7 - o(1)$ requires exponential-time as well. One can further deduce from the result of Moshkovitz and Raz [59] that, assuming the ETH, neither independent set nor graph coloring can be approximated to within a factor of $n^{1-\epsilon}$ in subexponential-time.

In this thesis, we extend the ground breaking result of Moshkovitz and Raz [59]. To wit, we quantify the trade-off between time and approximation for subexponential-time algorithms. For example, given a running time of $O(2^{n^{1/2-\epsilon}})$, it is not possible to approximate the maximum independent set problem to within a factor of $n^{1/2-\epsilon}$, for any $\epsilon > 0$, assuming the exponential-time hypothesis. A similar hardness result also holds for the graph coloring problem. These results extend to the maximum *bipartite induced matching* problem and the *k-hypergraph pricing* problem.

Our contributions are two-fold. First, we develop a "framework" for proving subexponential-time approximation hardness. Two techniques used in the framework are (1) *graph product* and (2) *CSP-level gap amplification*. (The latter works at the level of the *constraint satisfaction* problem (CSP).) Second, we prove lower bounds on the running time of r-approximation algorithms for independent set, graph coloring, bipartite induced matching and k-hypergraph pricing as listed in Table 1. Our results for the maximum independent problem significantly improves upon that of Chitnis, Hajiaghayi and Kortsarz [22]. Furthermore, our results imply subexponential-time approximation hardness for numerous other problems whose approximation hardness are derived from maximum independent set or maximum induced matching.

Problem	**Running-Time of r-Approximation**	**Value of r**
Independent Set	$\Omega(2^{n^{1-\epsilon_1}/r^{1+\epsilon_2}})$	$r_0 \leq r \leq n^{1-\varepsilon}$
Independent Set (max deg. $= D$)	$\Omega(2^{n^{1-\epsilon_1}/r^{1+\epsilon_2}})$	$r = D^{1-\varepsilon}$
Graph Coloring	$\Omega(2^{n^{1-\epsilon_1}/r^{1+\epsilon_2}})$	$r_0 \leq r \leq n^{1-\varepsilon}$
Bip. Induced Matching	$\Omega(2^{n^{1-\epsilon_1}/r^{1+\epsilon_2}})$	$r_0 \leq r \leq n^{1-\varepsilon}$
Bip. Induced Matching (max deg. $= D$)	$\Omega(2^{n^{1-\epsilon_1}/r^{1+\epsilon_2}})$	$r = D^{1-\varepsilon}$
k-Hypergraph Pricing	polynomial time	$r = k^{1-\varepsilon}$, $k < n^{1/2}$
	polynomial time	$r = n^{1/2-\varepsilon}$
	$\Omega(2^{(\log n)^{(1-\delta-\epsilon_1)/\delta}})$	$r = n^{\delta}$, $0 < \delta < 1$

Table 1–1: The table summarizes our results. Here ε is any positive constant. The values of ϵ_1, ϵ_2 are between 0 and 1 and depend on the values of r and ε.

1.1 The Approximability of Optimization Problems

We have seen that the approximability of optimization problems is of major significance in theoretical computer science. But what exactly do we mean by approximability? To formalize this concept, consider an instance ϕ of an optimization problem Π, and let opt denote the cost of an optimal solution. Then, an *r-approximation algorithm* for the problem π is an algorithm that outputs a feasible solution I whose cost, denoted by $\text{cost}(I)$, satisfies:

$$\begin{aligned} &\text{Maximization problem:} \quad \text{cost}(I) \geq \frac{\text{opt}}{r} \\ &\text{Minimization problem:} \quad \text{cost}(I) \leq r \cdot \text{opt} \end{aligned}$$

Here the slight difference in definition for maximization and minimization problems ensures that $r > 1$. The standard notion of an r-approximation algorithm requires polynomial running time. In this thesis, we relax this requirement to include subexponential-time algorithms. We refer to an r-approximation algorithm

that runs in superpolynomial-time by a subexponential-time approximation algorithm. More specifically, we refer to a $t(n)$-time r-approximation algorithm, where $t(n)$ denotes the running time.

The existence of an r-approximation algorithm gives an upper bound on the *approximability* of the problem Π. The limit of computation at which an r-approximation algorithm does not exist is called the *hardness of approximation*. Such lower bounds are dependent upon the computational power. Thus, they rely on the complexity assumption, for example, P $\neq$ NP, NP $\neq$ ZPP, and NP $\subseteq$ DTIME($2^{\text{polylog}(n)}$).

1.2 Proving Hardness of Approximation

A basic technique in proving hardness of approximation is by reduction from an NP-complete problem. For example, there is a well-known reduction from the satisfiability (SAT) problem to graph coloring. In fact, the reduction is from 3-SAT to 3-coloring. The reduction implies that it is NP-hard to decide whether a graph is 3-colorable or requires more than 4 colors. Observe that this implies an approximation hardness of 4/3 for the graph coloring problem; an r-approximation algorithm with $r < 4/3$ would produce an algorithm for 3-coloring and, thus, show P = NP.

A powerful modern technique (which is now standard) to prove large *hardness gaps* is via a reduction from a *probabilistically checkable proof system* (PCP). A PCP is a proof system that validates whether a proof x (represented by a binary string) is correct or valid by randomly reading only a small part of the proof x. The celebrated PCP theorem [3, 2] states, informally, that one can test whether a solution x is feasible for a problem Π by reading only a constant number of bits, unless P = NP.

(In fact, the theorem is stated in term of an NP-language.) The power of the PCP theorem with respect to hardness of approximation was not totally realized until Feige et al. in [33] showed the hardness of approximating maximum independent set via a reduction from PCPs. The hardness factor of an NP-hard problem can be derived from the ratio between the *completeness* and *soundness* parameters of the PCP. (See Section 2.4 for more discussion.) Other parameters have also been shown to play an important role in proving strong approximation hardness.

Consequently, PCPs can be viewed as a reduction rather than a proof system. In particular, one may think that a PCP is a "reduction" from the *satisfiability* problem (SAT) to another problem, for example, maximum independent set. Thus, the existence of a PCP for SAT with (almost) linear size – the size of PCP verifier is almost linear in the size of SAT – means that there is an almost linear-size reduction from SAT to the designated problem Π. This view of PCPs is very important. It implies that, if there is no algorithm with running time $2^{n^{1-\epsilon}}$ to solve SAT, then it requires more than $\Omega(2^{n^{1-\epsilon}})$-time to approximate Π because the two instances have roughly the same size. The results in this thesis are based upon this observation. Namely, assuming that SAT has no subexponential-time algorithm, then maximum independent set and graph coloring have no subexponential-time algorithms either. Applying a more intricate analysis, we show that maximum independent set and graph coloring admit no r-approximation algorithm that runs in time $O(2^{n/r-\epsilon})$, where n is the number of vertices. Because these two problems are used to prove approximation lower bounds for numerous problems, similar running-time versus approximation trade-offs also hold for these other problems. We remark that it

is not clear whether such conclusions can be derived using existing results in the literature. The reason for this is that we require that the reductions used must be *almost linear*. If not, we cannot relate the designated problems to SAT.

CHAPTER 2
Preliminaries and Background

In this chapter, we discuss some backgrounds which will be needed to understand the later chapter of this thesis, and we define some notation that will be used through out.

2.1 Preliminaries

We use standard graph theory terminology; see for example Diestel's Graph Theory book [28]. The vertex and edge sets of a graph G are denoted by $E(G)$ and $V(G)$, respectively. Correspondingly, when context is clear, we may simply denote the graph by $G = (V, E)$. An *induced subgraph* H of a graph G is a subgraph of G such that, for every pair of vertices $u, v \in V(H)$, there is an edge $uv \in E(H)$ if and only if there is an edge $uv \in E(G)$.

Independent Set and Clique. A subset of vertices $S \subseteq V(G)$ is *independent* or *stable* in G if and only if there is no edge in G joining any pair of vertices in S. The size of the maximum independent set in G is called the independence (or stability) number of G, denoted by $\alpha(G)$. The ratio between the independence number and the number of vertices of a graph, that is, $\alpha(G)/|V(G)|$, is called the *independence ratio* of G.

A subset of vertices Q that induces a complete subgraph in G is called a *clique*. The clique number of G, denoted by $\omega(G)$, is the maximum size of a clique in G.

Coloring. A *proper (vertex) coloring* $\sigma : V \to L$ of G is a function mapping vertices of G to a set of colors L such that, for any edge $uv \in E(G)$, $\sigma(u) \neq \sigma(v)$. That is, any adjacent vertices in G receive different colors under σ. Typically, the set L is defined as a set of integers. A set of vertices with the same color is called a *color class*. It is easy to see that each color class forms an independent set in G. The minimum number of colors required to color a graph G is called the *chromatic number* of G and is denoted by $\chi(G)$. Since each color class is an independent set in G, an equivalent definition of the chromatic number is the minimum number of independent sets required to *cover* all the vertices of G (i.e, each vertex must be in one of the color classes).

There are many generalizations of graph coloring. An important one is *fractional coloring*, where each vertex may receive more than one color. To be precise, let $\mathcal{I}(G)$ be the collection of independent sets in G. Then a fractional coloring function of G is a function $f : \mathcal{I}(G) \to \mathbb{R}_0^+$ such that

$$\sum_{S \in \mathcal{I}(G): v \in S} f(S) \geq 1, \text{ for all vertices } v \in V(G),$$

That is, in a *proper* fractional coloring each vertex must be *covered* by independent sets with fractional value at least one. The *fractional chromatic number* of G, denoted by $\chi_f(G)$, is defined as

$$\chi_f(G) = \min \left\{ \sum_{S \in \mathcal{I}(G)} f(S) \text{ s.t. } \sum_{S \in \mathcal{I}(G): v \in S} f(S) \geq 1 \text{ for all } v \in V(G) \right\}$$

Lovász [56] proved the following relationship between the chromatic and fractional chromatic numbers of a graph.

Lemma 2.1.1 ([56]). *For any graph G,*

$$\frac{\chi(G)}{1+\ln(\alpha(G))} < \chi_f(G) \leq \chi(G).$$

□

Matching. A *matching* M is a set of edges such that no two edges share an endpoint. Thus, each vertex in M is matched to another vertex. One may think of the matching M as a graph in which each vertex has a degree exactly one. An *induced matching* M of a graph G is an induced subgraph of G that forms a matching. That is, M is a subset of edges of G that forms a matching and, for any two edges $uv, ab \in M$, the graph G does not contain any of the edges $\{ua, ub, va, vb\}$. The *induced matching number* of G is the maximum number $\mathsf{im}(G)$ such that G has an induced matching of size $\mathsf{im}(G)$.

2.2 Problem Definitions

In this thesis, we will prove the subexponential hardness of several problems. To accomplish this, we will require reductions with several additional problems. The following is a list of the problems needed for this thesis.

Maximum Independent Set. In the *maximum independent set* problem, we are given a graph G and the goal is to find a independent set S of maximum cardinality.

We will also consider a special case of the problem in which the input graph G has a maximum degree of d. This is called the *d-degree bounded independent set* problem.

Graph Coloring. In the *graph coloring* problem, we are given a graph G, and the goal is to find a proper coloring of G with minimum number of colors.

Induced Matching. In the *maximum induced matching* problem, we are given a graph G and the goal is to find an induced matching of maximum size.

Two important special cases of the maximum induced matching problem are (i) *bipartite induced matching* where the input graph G is bipartite and (ii) *d-degree bounded (bipartite) induced matching* where the input graph G has a maximum degree of d.

Semi-Induced Matching. Given any graph $G = (V, E)$ and any mapping $\sigma : V(G) \to [|V(G)|]$, we say that a matching $\mathcal{M}$ is a σ-semi-induced if and only if, for any pair of edges $uu', vv' \in \mathcal{M}$ such that $\sigma(u) < \sigma(u')$ and $\sigma(u) < \sigma(v) < \sigma(v')$, there are no edges uv', uv in E.

In the maximum σ-semi-induced matching problem, we are given a graph G and an ordering σ. The goal then is to find a σ-semi-induced matching of maximum size. We may also wish to compute

$$\mathsf{sim}(G) = \max_{\sigma} \mathsf{sim}_{\sigma}(G)$$

where $\mathsf{sim}_{\sigma}(G)$ is the maximum size of of a σ-induced matching for G.

q-Constraint Satisfaction. In the *q-constraint satisfaction problem* (q-CSP), we are given a set of variables and a collection of constraints where each constraint depends on at most q variables. The goal is to find an assignment to the variables so that all the constraints are satisfied.

The maximization version of q-CSP is *Max q-CSP* where the aim is to maximize the total number of constraints satisfied.

The most well-known special case of q-CSP is the *q-satisfiability* problem (q-SAT) where the input is a Boolean formula in *conjunctive normal form* (CNF). That is, each constraint is a Boolean formula joined by the operation OR. Consequently, the q-SAT problem is sometimes called *q-CNF*. The maximization version of q-SAT is dubbed *Max q-SAT*.

In general, our focus will be on the maximization version of q-CSP. Thus, when the context is clear, we may abuse the terminology and use q-CSP to mean *Max q-CSP*. In contrast, we will frequently use both the problems q-SAT and Max q-SAT. So, we will use this terminology to distinguish between them. Also, unless specified, the domain of the q-CSP variables is Boolean – each variable takes a value 0 or 1.

k-Hypergraph Pricing. In the *unlimited supply k-hypergraph vertex pricing* problem [4, 12], we are given a weighted n-vertex m-edge k-hypergraph (each hyperedge contains at most k vertices) modeling the situation where consumers (represented by hyperedges) with budgets (represented by weights of hyperedges) have their eyes on at most k products (represented by vertices). The goal is to find a price assignment that maximizes the revenue. In particular, there are two variants of this problem with different consumers' buying rules. In the *unit-demand pricing problem* (UDP-MIN), we assume that each consumer (represented by a hyperedge e) will buy the cheapest vertex of her interest if she can afford it. In particular, for a given hypergraph H with edge weight $w : E(H) \to \mathbb{R}_{\geq 0}$ (where $\mathbb{R}_{\geq 0}$ is the set of

non-negative reals), our goal is to find a price function $p : V(H) \to \mathbb{R}_{\geq 0}$ to maximize

$$\text{profit}_{H,w}(p) = \sum_{e \in E(H)} \text{pay}_e(p) \quad \text{where} \quad \text{pay}_e(p) = \begin{cases} \min_{v \in e} p(v) & \text{if } \min_{v \in e} p(v) \leq w(e), \\ 0 & \text{otherwise.} \end{cases}$$

The other variation is the *single-minded pricing problem* (SMP), where we assume that each consumer will buy *all* vertices if she can afford to; otherwise, she will buy nothing. Thus, the goal is to maximize

$$\text{profit}_{H,w}(p) = \sum_{e \in E(H)} \text{pay}_e(p) \quad \text{where} \quad \text{pay}_e(p) = \begin{cases} \sum_{v \in e} p(v) & \text{if } \sum_{v \in e} p(v) \leq w(e), \\ 0 & \text{otherwise.} \end{cases}$$

The pricing problem naturally arose in the area of algorithmic game theory and has important connections to algorithmic mechanism design (e.g., [5, 21]). Its general version (where k could be anything) was introduced by Rusmevichientong et al. [63, 64], and the k-hypergraph version (where k is thought of as some constant) was first considered by Balcan and Blum [4]. (The special case of $k = 2$ has also received a lot of attention [4, 16, 50, 60].) There will be two parameters of interest to us, that is, n and k. Its current approximation upper bound in terms of k is $O(k)$ [12, 4], and the upper bound in terms of n is given in this thesis.

2.3 Complexity Terminology

A *hard* problem is a problem that admits no polynomial-time algorithms. So, a problem in NP problem is hard if it is not in the class P. An *NP-hard* problem Π is a problem that may not be in the class NP, but there is a polynomial-time reduction from some NP-complete problem Ψ. By the completeness of the class NP-complete,

we may assume that the hardness of Π is derived from SAT. If the instance ϕ of Π is obtained from an instance ψ whose answer is "Yes" (e.g., a satisfiable SAT instance), then ϕ is called YES-INSTANCE. Otherwise, if the answer to ψ is "No", then ϕ is called NO-INSTANCE. In proving hardness of approximation, we will consider the gap between that the optimal value of two cases YES-INSTANCE and NO-INSTANCE. The case of YES-INSTANCE is sometimes called the *completeness* case, and the case of NO-INSTANCE is sometimes called the *soundness* case.

2.4 A Probabilistically Checkable Proof System

A *probabilistically checkable proof system* (PCP) is a proof system consisting of a verifier and a prover. The prover in this system claims that some statement is true and then submits a proof to the verifier who will check whether the proof is correct. For example, suppose the verifier claims that a Boolean formula is satisfiable. He submits an assignment to the verifier as a proof. The verifier may then check the proof (an assignment) by simply substituting the assignment to all the variables. Observe, though, that this verification process requires that she has to read all the bits.

Surprisingly, there is a way to rewrite the certificate so that the verifier has to read only a constant number of bits! This is now known as the PCP theorem.

Formally, a *probabilistically checkable proof system* (PCP) for an NP-language L is defined as follows. The system consists of a verifier and a prover. Given a bit string x, the prover claims that x is a member of the language L and provides a bit string y as a *witness* or a *proof string*. To test whether the claim is valid, the verifier

randomly reads q positions of y. If the q bits from y are sound, then the verifier accepts the claim that $x \in L$. Otherwise, the verifier rejects the claim.

It has been shown that a PCP gives a characterization of an NP problem [3]. In particular, the *PCP theorem* says that every NP-language L has a PCP verifier V such that if $x \in L$, then V accepts the proof string with probability at least c. Otherwise, V rejects the proof string with probability at least $1-s$ (that is, V accepts the proof string with probability at most s). The acceptance probability c is called the *completeness* parameter of the PCP, and the acceptance probability s is called the *soundness* parameter.

A PCP-verifier V for an NP-language L may actually involve more parameters. We shall introduce such parameters when required.

The PCP theorem is stated as follows.

Theorem 2.4.1 (PCP Theorem. [2, 3])**.** *Every NP-language L of size n has a PCP system which uses $O(\log n)$ bits of randomness, $O(1)$ queries bits in the proof, and has completeness 1 and soundness $1-\epsilon$, for some constant $\epsilon > 0$.*

The PCP theorem plays a vital role in the theory of hardness of approximation as it can be seen as a reduction from an NP-language to the *q-constraint satisfaction* problem (q-CSP) with a "hardness-gap". For example, the existence of a q-query PCP – a PCP that reads q positions of the proof – for SAT with soundness $1/2$ implies that it is NP-hard to approximate q-CSP within a factor of $1/2$. To see this, let us construct a q-CSP instance ϕ from the PCP. Let each bit of the certificate corresponds to a Boolean variable. For each random string r, the decision of the verifier depends on the q-position that she reads. So, we have one clause for each

random string r, which depends on q variables. Consequently, the number of random strings that cause the verifier to accept is the same as the number of satisfiable clauses. Now, suppose we have a q-query PCP for SAT with completeness 1 and soundness 1/2. If the SAT instance is satisfiable, then all the clauses of the q-CSP are satisfiable. Otherwise, at most half of the clauses are satisfiable. So, any 2-approximation algorithm for this q-CSP would be able to distinguish between these cases and thus solve SAT.

For clarity of presentation, we will typically view PCP as a reduction from the decision version of 3-SAT to q-CSP, and will work with an instance of q-CSP instead of a PCP.

2.4.1 The FGLSS reduction

The connection between PCPs and hardness of approximation was first discovered by Feige, Goldwasser, Lovász, Safra and Szegedy [33]. At its heart, lies a transformation that represents an instance of q-CSP, denoted by ϕ, as a graph.

We describe this representation now. First, let $x_1, \ldots, x_n$ and $C_1, \ldots, C_m$ be variables and clauses of ϕ, respectively. We say $x_i \in C_j$ if x_i is a variable occurring in C_j. For each clause C_j, the assignment restricted to variables in C_j is call a *configuration*. For example, a bit string $(0, 1, 1)$ is a configuration for the clause $x_2 \wedge \overline{x_5} \wedge x_7$. Note that, although we have n variables, the number of configurations for each clause is upper bounded by $2^q \ll 2^n$ because each clause involves at most q variables. A configuration a is an *accepting configuration* for a clause C_j if and only if a causes C_j to be "true". We say that two configurations a and a' *conflict* if and only if they assign different values to the same variable. The FGLSS reduction

transforms the CSP instance into a graph as follows. For every clause, there is a vertex for every possible accepting configurations for that clause. There is an edge uv in the graph if the configurations u and v conflict. That is, the FGLSS graph $G = (V, E)$ corresponding to the CSP instance ϕ is defined as

$$V(G) = \bigcup_{j=1}^{m} \{a : a \text{ is an accepting configuration for a clause } C_j.\}$$

$$E(G) = \{uv : u, v \in V(G), u \text{ and } v \text{ are conflicting configurations.}\}$$

Let's illustrate the FGLSS reduction using Boolean formula $C_1 \wedge C_2 := (x_1 \vee x_2) \wedge (\overline{x_2} \vee x_3)$. First, we list all the vertices of the graph as shown in the next table.

Clause	**Vertices**
$x_1 \vee x_2$	$v^1_{10*}, v^1_{11*}, v^1_{01*}$
$\overline{x_2} \vee x_3$	$v^2_{*00}, v^2_{*01}, v^2_{*11}$

Observe that, for each vertex, the superscript denotes the clause number. The subscript denotes the accepting configuration associated with the vertex. For example, v^1_{10*} assigns $x_1 = 1$ and $x_2 = 0$ but makes no assignment to x_3 as that variable is not in the clause $C_1 = (x_1 \vee x_2)$. Now there is a conflict between the two vertices v^1_{10*} and v^2_{*11} because they proscribe different values to x_2. So we list $v^1_{10*}v^2_{*11}$ as an edge of G. The graph has three other edges, namely, $v^1_{11*}v^2_{*00}, v^1_{11*}v^2_{*01}, v^1_{01*}v^2_{*00}, v^1_{01*}v^2_{*01}$. The entire FGLSS graph is illustrated in Figure 2–1.

Any independent set in the FGLSS graph G corresponds to a set of consistent partial assignments, i.e., they assign the same values to the same variables. Thus, we have the following lemma.

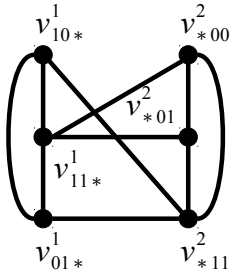


Figure 2–1: An example of the FGLSS graph of $C_1 \wedge C_2 := (x_1 \vee x_2) \wedge (\overline{x_2} \vee x_3)$.

Lemma 2.4.2. *Consider any q-CSP instance ϕ on n variables and m clauses, where each clause has at most w accepting configurations. Let β be the maximum number of clauses of ϕ that can be satisfied by any assignment. Then the FGLSS reduction, given ϕ as input, constructs a graph G such that $|V(G)| \leq w \cdot m$ and $\alpha(G) = \beta$. Moreover, if each clause has at least one accepting configuration, then $\alpha(G) \geq m/2^q$.*

Proof. For each clause, we have at most w accepting configurations. So, the number of vertices that G has is at most $m \cdot w$.

It is easy to see that $\alpha(G) \geq \beta$ because if we take an assignment x^* that satisfied β clauses, then we can pick β accepting configurations (which are partial assignments), one from each clause that x^* satisfies; these β configurations are consistent and thus correspond to an independent set in G. To show that $\alpha(G) \leq \beta$, we assume a contradiction that there is an independent set S of size $|S| > \beta$. Since vertices in S correspond to accepting configurations and they assign the same value to the same variables, we can construct from S a partial assignment $\tilde{x}$. We extend $\tilde{x}$ to be a complete assignment $\widehat{x}$ by assigning other variables arbitrary. Then we have a contradiction since $\widehat{x}$ satisfies more than β clauses. This shows that $\alpha(G) = \beta$.

Next, suppose each clause has at least one accepting configuration. Then any random assignment can satisfy each clause with probability $1/2^q$. Thus, the expected number of satisfied clauses is $m/2^q$, and there must exist an assignment that attends this value. This proves that $\alpha(G) \geq m/2^q$. □

Trevisan [68] observed that every FGLSS graph G is formed by a union of complete bipartite graphs, each corresponding to a variable of ϕ. Specifically, for each variable x_i of ϕ, we define a bipartite graph $H_i = (O_i \cup Z_i; E_i)$ where

$$O_i = \{u \in V(G) : u \text{ assigns } x_i = 1\}$$

$$Z_i = \{v \in V(G) : v \text{ assigns } x_i = 0\}$$

$$E_i = \{uv : u \in O_i, v \in Z_i\}$$

It is easy to see that E_i are edges of G because, by the construction, G has edges between every pair of vertices $u \in O_i$ and $v \in Z_i$ as they are conflicting. Hence, H_i is a complete bipartite subgraph of G. Since each vertex of G is associated with (at least one) variable, we have that

$$G = \bigcup_{i=1}^{n} H_i$$

This observation will play an important role in proving the hardness of problems on bounded degree graphs. This, in turn, will be required in proving the hardness of the k-hypergraph pricing problem.

CHAPTER 3
Exponential-Time Hypothesis and Almost Linear-Size PCPs

In this chapter, we discuss the *Exponential-Time Hypothesis* (ETH), almost linear-size PCPs and their connection to subexponential-time approximation hardness.

3.1 Exponential-Time Hypothesis

The exponential-time hypothesis (ETH) is a complexity assumption stronger than P $\neq$ NP. It states that not only can SAT not be solved in polynomial-time but that no subexponential-time algorithm can decide SAT. The hypothesis was first proposed by Impagliazzo and Paturi [46] in an attempt to understand the complexity of SAT. There they show the implication of the assumption that SAT has no subexponential-time algorithm. Recently, the ETH assumption has received a lot of attention as it has been shown to be a powerful tool in proving running-time lower bounds for many problems, especially, those in the area of *parameterized complexity*. See, e.g., [54, 27, 18]. For more discussion related to the ETH, we refer readers to a comprehensive survey by Lokshtanov et al. [55] and references therein.

The formal statement of this conjecture is as follows.

Hypothesis 3.1.1 (Exponential-Time Hypothesis (ETH))**.** *For any integer $k \geq 3$, there is a constant $0 < q_0(k) < 1$ such that there is no algorithm with a running time of 2^{qN}, for all $q < q_0(k)$, that solves k-SAT where N is the size of the instance. In particular, there is no subexponential-time algorithm that solves* 3*-SAT.*

Indeed, the ETH was first stated in terms of the number of variables. Impagliazzo, Paturi and Zane [46] showed that the statement is equivalent for all the parameters, i.e., N in the statement can be the number of variables, the number of clauses or the size of the instance. For our purpose, we state the theorem in terms of the instance size.

3.2 (Almost) Linear-Size PCP

While the ETH states the running-time lower bounds for the "decision" version of SAT, it is not clear how one can relate the assumption to the hardness of approximation. The foundation technique in the area of hardness of approximation that gives a reduction from SAT to its "maximization" version with hardness gap is known as the PCP theorem. It implicitly states that there is a polynomial-time algorithm that reduces 3-SAT to the *gap-version* of Max 3-SAT. This gives a hardness-gap of γ, for some constant $\gamma > 0$, which is sufficient in proving approximation hardness results. We may state the PCP theorem as a reduction as follows.

Theorem 3.2.1. *There is a polynomial-time reduction from an instance ϕ of* 3*-SAT (decision version) to an instance ψ of Max* 3*-SAT such that*

- *(*Yes-Instance*) If ϕ is satisfiable, then there is an assignment that satisfies all the clauses of ψ.*
- *(*No-Instance*) If ϕ is not satisfiable, then any assignment can satisfy at most a γ fraction of the clauses of ψ.*

It is easy to see that the PCP theorem (Theorem 3.2.1) implies any "polynomial-time" algorithm that approximates Max 3-SAT to within a factor of γ can solve the "decision version" of SAT in polynomial-time as well. Now, to add the ETH into

the picture, the size of the reduction must be preserved. Ideally, we wish to have a linear-size reduction from 3-SAT to Max 3-SAT. We need, in particular, a linear-size PCP for 3-SAT, which would imply the following:

$$\underbrace{\text{3-SAT of size } N}_{\text{no gap}} \implies \underbrace{\text{Max 3-SAT of size } cN}_{\text{gap} = \gamma}$$

Here c and γ are constants. To simplify the presentation, let us assume that $c = 1$. Now, suppose we have the "ideal" reduction. Then we would have that any r-approximation algorithm with $r < \gamma$ that runs in $t(N)$-time can also solve 3-SAT with the same running time. Thus, assuming the ETH, there is no such r-approximation algorithm that runs in subexponential-time.

Whilst the existence of a linear-size PCP is still a conjecture, in most of the applications, it suffices to use an almost linear-size PCP. The foundation of our subexponential-time approximation hardness results are based on the following breakthrough result by Moshokovitz and Raz [59].

Theorem 3.2.2 ([59])**.** *For every $\epsilon > 0$, there exists an alphabet σ with $\log |\sigma| \leq \text{poly}(\frac{1}{\epsilon})$, such that 3-SAT has a PCP-verifier that uses $(1 + o(1)) \cdot \log N + O(\log \frac{1}{\epsilon})$ random bits. Moreover, the PCP verifier makes two-query projection tests.*

The result of Moshokovitz and Raz can be combined with the PCP of Samorodnitsky and Trevisan [65] to obtained the PCP with an *optimal query complexity* as in Theorem 3.2.3. which also appears as Corollary 14 in [59]. In terms of CSP, optimal query complexity means that the ratio between the inverse of soundness and the maximum number of accepting configurations can be set to be arbitrarily low.

Theorem 3.2.3 ([59] + [65]). *For any sufficiently large constant $\ell \geq \omega(1)$, 3SAT on input of size N has a PCP-verifier that uses $(1+\epsilon)\log N$ random bits to pick $q = \ell^2 + 2\ell$ queries to a binary proof, such that only 2ℓ of the queries are free, i.e., for each random string, there are $2^{2\ell}$ possible satisfying assignments of the queried bits in the proof. The verifier has completeness $1-\epsilon$ and soundness error at most $2^{-\ell^2+1}$. Moreover, the acceptance predicate is linear.*

In short, the theorem says that there is a polynomial-time reduction from an instance ϕ of 3-SAT of size N to an instance of q-CSP of size $N^{1+o(1)}$ with a hardness gap $s = 2^{\ell^2-2}$ and each clause has at most $w = 2^{2\ell}$ possible accepting configurations. Moreover, each constraint (i.e., the *predicate*) is linear, e.g., $x_0 + x_1 + x_2 = 1 \pmod 2$. The above PCP is required to prove the hardness of approximating the maximum independent set problem. For the case of the graph coloring problem, we need a PCP with an additional property.

3.3 Randomized PCP for Graph Coloring

Feige and Kilian introduced the notion of the *covering parameter* of a PCP and showed that the existence of a PCP with small covering parameter implies the hardness of the graph coloring problem.

To be precise, let us define the covering parameter in terms of q-CSP. We say that a q-CSP instance has a covering parameter ρ if, in the YES-INSTANCE (i.e., the instance of is obtained from a satisfiable instance of 3-SAT), there is a set of assignments $\{A_1, A_2, \ldots, A_\rho\}$ that *covers* all the accepting configurations, i.e., each accepting configuration (which is a partial assignment) is a part of one of these assignments. If we apply the FGLSS reduction to the q-CSP instance ϕ, then we

will have a graph G with a collection of independent sets $S_1, \ldots, S_\rho$ where each set S_i corresponds to the assignment A_i, and $\bigcup_{i=1}^{\rho} S_i = V(G)$. So, each vertex of G is covered by some set S_i, thus implying that the fractional chromatic number of G is at most ρ.

Feige and Kilian also presented in [34] a general technique to construct a randomized PCP with a small covering parameter from a *long code based* PCP.

The long-code based technique is a standard way in constructing a PCP. The two important components of this kind of construction are 2-CSP, namely, the *label cover* problem, and the *long-code encoding* of an assignment to CSP.

Formally, the label cover problem is a 2-CSP on a non-Boolean domain. The input to the label cover problem consists of a bipartite graph $G = (U, W, E)$ with a set of labels (a.k.a., alphabet) $\Sigma = \Sigma_U \cup \Sigma_W$, where Σ_U and Σ_W are labels for U and W, respectively, and a projection constraint $\pi_{uw} : \Sigma_U \to \Sigma_W$ on each edge $uw \in E$. A labeling is a pair of functions $f_1 : U \to \Sigma_U$ and $f_2 : W \to \Sigma_W$ that assign a label to each vertex of G. We say that the labeling (f_1, f_2) *covers* an edge $uw \in E$ if $\pi_{uw}(f(u)) = f(w)$. The goal in the label cover problem is to find a labeling that covers the maximum number of edges. A standard scheme in constructing a PCP is to apply the long-code based technique to the label cover problem. The *long-code encoding* of a string $j \in [n]$ is the truth table of the Boolean function $f : \{0,1\}^n \to \{0,1\}$ such that $f(x_1, \ldots, x_n) = x_j$ (which is called a *dictator* function). So, the long code represents an $\log n$-bit string using 2^n bits.

The following long-code based scheme is used in constructing a q-query PCP and thus proving the hardness of approximation.

(1) Start with an instance $\mathcal{L}$ of the *label-cover* problem on the domain (or alphabet) $\Sigma = \Sigma_U \cup \Sigma_W$.

(2) Take an assignment A of the label-cover instance $\mathcal{L}$ and encode the entries of A using long code, resulting in a proof string σ.

(3) Construct a PCP-verifier that randomly picks a set of constraints from $\mathcal{L}$ and then reads q bits from σ to decide whether the proof string σ encodes an assignment that satisfies the chosen constraints.

To get the PCP with small covering parameter, Feige and Kilian add randomness to the proof string: For each label (or character) $a \in \Sigma$, we concatenate a with random strings of length r, so we have 2^r labels $(a, 1), \ldots, (a, 2^r)$ that all project to the same label a. Hence, after the process, for each assignment A, we have 2^r other assignments. This process is call *randomizing* the PCP. Feige and Kilian showed that, after alternating the label-cover instances with large enough r, provided that the construction does not requires any special structure of the underlying 2-CSP, the randomized long-code based PCP-verifier will automatically produce a CSP with a small covering parameter.

In their paper, Feige and Kilian randomize the PCP of Håstad [42] to prove tight hardness of the graph coloring problem. Later, Engebretsen and Holmerin showed in [30] that the technique of Feige and Kilian can also be applied to the PCP of Samorodnitsky and Trevisan in [65]. Both of the previous results, apply the randomization technique to the label cover problem with small hardness gap and then amplify the gap using parallel repetition [61].

With the breakthrough of Moshkovitz and Raz [59], the label cover instance in the proof of [30] can be replaced with the label cover instance of Moshkovitz and Raz. In fact, Theorem 3.2.2 is equivalent to the hardness of the label cover problem:

Theorem 3.3.1 ([59])**.** *For every n, and every $\epsilon > 0$ (that can be any function of n) the following holds. Solving 3-SAT on inputs of size n can be reduced to distinguishing between the case that a label-cover instance of size $n^{1+o(1)} \cdot \mathrm{poly}(1/\epsilon)$ and parameters $|\Sigma_U|, |\Sigma_W|$ such that $\log |\Sigma_U| \leq \mathrm{poly}(1/\epsilon)$, is completely satisfiable and the case that at most an ϵ fraction of its edges are satisfiable.*

Take an instance $\mathcal{L} = (G = (U, W, E), \Sigma_U, \Sigma_W, \{\pi_{uw}\}_{uw \in E})$ of the label cover problem in Theorem 3.3.1 where the soundness parameter ϵ of $\mathcal{L}$ is chosen to be a small enough constant. We randomize the label-cover instance $\mathcal{L}$ by alternating the set of labels and constraints on edges. Let r be a parameter. We construct a new set of labels, namely Σ':

$$\Sigma' = \{(a, i) : a \in \Sigma, i \in [2^r]\}.$$

Then we construct a new constraint Π'_{uw} on each edge $uw \in E$:

$$\Pi'_{uw}((a, i)) = (\Pi_{uw}(a), i) \quad \text{for all } a \in \Sigma, \text{ for all } i \in [2^r]$$

So, we have the altered instance $\mathcal{L}' = (G = (U, W, E), \Sigma'_U, \Sigma'_W, \{\pi'_{uw}\}_{uw \in E})$ of the label-cover problem. It is easy to see that the modification does not effect completeness and soundness. To be precise, let $\mathrm{val}(\mathcal{L})$ denote the maximum number of edges of $\mathcal{L}$ that can be covered by any labeling. We show in the following claim that $\mathrm{val}(\mathcal{L}) = \mathrm{val}(\mathcal{L}')$.

Claim 3.3.2. *The modification of a label-cover instance $\mathcal{L}$ as above outputs a label-cover instance $\mathcal{L}'$ with* $\mathrm{val}(\mathcal{L}) = \mathrm{val}(\mathcal{L}')$.

Proof. First, we show that $\mathrm{val}(\mathcal{L}) \leq \mathrm{val}(\mathcal{L}')$. Given an optimal labeling (f_1, f_2) of $\mathcal{L}$, we construct a labeling (f_1', f_2') of $\mathcal{L}'$:

$$f_1'((u,1)) = f_1(u) \quad \text{for all } u \in U$$
$$f_2'((w,1)) = f_1(w) \quad \text{for all } w \in U$$

It is easy to see that (f_1', f_2') and (f_1, f_2) cover the same number of edges.

Next, we show that $\mathrm{val}(\mathcal{L}) \geq \mathrm{val}(\mathcal{L}')$. Given an optimal labeling (f_1', f_2') of $\mathcal{L}'$, we project (f_1', f_2') to a labeling (f_1, f_2) of $\mathcal{L}$:

$$f_1'(u) = (a,i) \implies f_1(u) = a \quad \text{for all } u \in U$$
$$f_2'(w) = (b,j) \implies f_2(w) = b \quad \text{for all } w \in W$$

By construction, for every edge uw in the instance $\mathcal{L}'$ that (f_1', f_2') covers, we must have

$$\pi'_{uw}(f_1'(u)) = f_2'(w) \implies \pi'_{uw}((f_1(u), i)) = (f_2(w), i) \implies \pi_{uw}(f_1(u)) = f_2(w).$$

So, (f_1, f_2) also covers uw in the instance $\mathcal{L}$, proving that $\mathrm{val}(\mathcal{L}) \geq \mathrm{val}(\mathcal{L}')$. Therefore, $\mathrm{val}(\mathcal{L}) = \mathrm{val}(\mathcal{L}')$ as claimed. □

We recall that the long-code base PCP construction starts from a label-cover instance. Feige and Kilian showed in [34] that if the construction yields a q-CSP

whose maximum number of accepting configurations is w, then the above modification to the label-cover instance with a sufficiently large r results in a q-CSP with a covering parameter at most $O(w)$. See Lemma 5 in [34] and Lemma 10.10 in [30] for the setting of r.

Now let us combine everything together. We apply the above modification to the label-cover instance of Moshkovitz and Raz [59] in Theorem 3.3.1. Then we plug it into the PCP construction of Samorodnitsky and Trevisan [65] as in the result of Engebretsen and Holmerin [30]. We choose a parameter ϵ in Theorem 3.3.1 to be $O(2^{-\ell^2+1})$ and choose r to be a sufficiently large constant, say $r > 100\log(\Sigma)$. The change only effects the size of the PCP by a constant factor. Therefore, we have a PCP with almost linear-size, optimal query complexity and small covering parameter:

Theorem 3.3.3. *For any sufficiently large constant $\ell \geq \omega(1)$, 3SAT on input of size N has a PCP-verifier that uses $(1+\epsilon)\log N$ random bits to pick $q = \ell^2 + 2\ell$ queries to a binary proof, such that only 2ℓ of the queries are free, i.e., for each random string, there are $2^{2\ell}$ possible satisfying assignments of the queried bits in the proof. The verifier has completeness $1-\epsilon$ and soundness error at most $2^{-\ell^2+1}$. Moreover, the acceptance predicate is linear.*

CHAPTER 4
Graph Product

The study of binary operators in graph theory has been a fundamental topic of research for several decades. A basic question is whether or not a property in two graphs is maintained in their product. For example, the independence number of the *disjunctive product*[1] has been shown to amplify the independence number of graphs multiplicatively. In particular, the *k-wise disjunctive product* of a graph G has independence number $\alpha(G)^k$; a similar property holds for the chromatic number of product graphs [53].

But, why are graph products of interest in this thesis? It turns out that the multiplicative properties of graph products can be exploited to amplify hardness gaps for approximation algorithms. An early application is due to Garey and Johnson [37] who amplified the hardness gap of the graph coloring problem from 4/3 to $2 - \epsilon$. Subsequent developments illustrated many more applications of graph products; see, for example, [37, 53, 8, 31, 17, 18, 19]. In fact, the seminal $n^{1-\epsilon}$ approximation hardness results for independent set and graph coloring are, in some sense, obtained via graph product techniques.

[1] This graph product is also known as the *co-normal* product, the *inclusive* product and the *OR* product.

Numerous graph product operators have been studied in literature. For the comprehensive survey of graph products, we refer the reader to [41]. The graph product that we will use as a tool in this thesis is the disjunctive product. The term "graph product" will sometimes abuse to mean the disjunctive product as it is the only graph product operation mentioned in this thesis.

Let's now formally define the disjunctive product. Given two graphs G and H, we have:

Disjunctive Product:

$$V(G \vee H) = V(G) \times V(H)$$
$$E(G \vee H) = \{(u,x)(v,y) : (uv \in E(G)) \text{ or } (xy \in E(H)))\}$$

We will often need to apply the product operation multiple times. Specifically, we define the k-wise disjunctive product as follows.

The k-wise Disjunctive Product:

$$G^k = G^{k-1} \vee G \text{ for } k \geq 2 \quad \text{and} \quad G^1 = G$$

Important properties of the graph products that we require are:

$$\alpha(G \vee H) = \alpha(G) \cdot \alpha(H) \tag{4.1}$$
$$\chi_f(G \vee H) = \chi_f(G) \cdot \chi_f(H) \tag{4.2}$$

For completeness, in Section 4.1, we prove Equation 4.1 and Equation 4.2.

We will also use randomized versions of the disjunctive product, and this will possess similar properties to their deterministic version. A discussion on the randomized graph product is provided in Section 4.2.

4.1 Properties of Disjunctive Product

An important property of the disjunctive product is that it amplifies the independence and chromatic numbers of graphs multiplicatively. We prove in this section the following equations and an inequality.

$$\alpha(G \vee H) = \alpha(G) \cdot \alpha(H) \qquad \text{(Theorem 4.1.1)}$$

$$\chi_f(G \vee H) = \chi_f(G) \cdot \chi_f(H) \qquad \text{(Theorem 4.1.4)}$$

$$\chi_f(G) \cdot \chi(H) \leq \chi(G \vee H) \leq \chi(G) \cdot \chi(H) \qquad \text{(Theorem 4.1.3)}$$

The following theorems are standard. The independence number of the disjunctive product was proven in [31], and the proofs for the chromatic number can be seen in [31, 67].

Theorem 4.1.1 (Independence Number of Disjunctive Product)**.** *For any graphs G and H,*

$$\alpha(G \vee H) = \alpha(G) \cdot \alpha(H).$$

Thus, for any integer $k \geq 1$, $\alpha(G^k) = \alpha(G)^k$, where G^k is the k-wise disjunctive product of G.

Proof. Consider the graph $G \vee H$. First, we prove the upper bound: $\alpha(G \vee H) \leq \alpha(G) \cdot \alpha(H)$. Let S_G and S_H be maximum independent sets in G and H, respectively. Consider the set $S_{G \times H} = S_G \times S_H$. Clearly, $S_{G \times H} \subseteq V(G \vee H)$. For any two vertices $(u, x), (v, y) \in S_{G \times H}$, we know that $uv \notin E(G)$ and $xy \notin E(H)$ because S_G and

S_H are independent in G and H. Thus, there is no edge $(u,x)(v,y)$ in $G \vee H$ by the definition of the disjunctive product. Therefore, $S_{G\times H}$ is independent in $G \vee H$. Thus, the maximum independence set in $G \vee H$ has cardinality at least $|S_{G\times H}|$. The upper bound follows.

Next we prove the lower bound: $\alpha(G \vee H) \geq \alpha(G) \cdot \alpha(H)$. Let $S_{G\vee H}$ be a maximum independent set in $G \vee H$. We know that any vertex in $S_{G\vee H}$ must be of the form (u,x), where $u \in V(G)$ and $x \in V(H)$. So, we may project vertices in $S_{G\vee H}$ into vertices of H and into vertices of G as

$$S_G = \{u : (u,x) \in S_{G\vee H}\}, \quad S_H^u = \{x : (u,x) \in S_{G\vee H}\} \quad \text{for } u \in S_G.$$

By the definition of the disjunctive product, since $S_{G\vee H}$ is independent in $G \vee H$, there is no edge joining any two vertices u, v in S_G. Otherwise, we would have an edge $(u,x)(v,y)$ in $G \vee H$ for some vertices $x, y \in V(H)$. Thus, S_G is an independent set in G, implying that $|S_G| \leq \alpha(G)$. Now, consider the set S_H^u. Observe that vertices in S_H^u are projected from vertices of $G \vee H$ in $S_{G\vee H}$ whose first coordinates are the same. Thus, by definition, there is no edge between any two vertices $x, y \in S_H^u$ because $(u,x)(v,y) \notin G \vee H$. In other words, each S_H^u is independent in H. Hence, $|S_H^u| \leq \alpha(H)$ for all $u \in S_G$. Putting this together, we have

$$|S_{G\vee H}| \;=\; \sum_{u\in S_G} |S_H^u| \cdot \;\leq\; |S_G| \left(\max_{u\in S_G} |S_H^u|\right) \;\leq\; \alpha(G) \cdot \alpha(H)$$

This gives the lower bound, and so $\alpha(G \vee H) = \alpha(G) \cdot \alpha(H)$. □

It follows from Theorem 4.1.1 that any maximum independent S set in $G \vee H$ is the Cartesian product of maximum independent sets S_G and S_H in G and H, respectively.

Corollary 4.1.2. *Any independent set S in $G \vee H$ can be written in the form $S = S_G \times S_H$, where S_G and S_H are independent sets in G and H, respectively. In particular, S is a maximum independent set in $G \vee H$ if and only if S_G and S_H are maximum independent sets in G and H.*

Theorem 4.1.3 (Chromatic Number of Disjunctive Product)**.** *For any graphs G and H,*

$$\chi_f(G) \cdot \chi(H) \leq \chi(G \vee H) \leq \chi(G) \cdot \chi(H)$$

In particular, for any integer $k \geq 1$, $\frac{\chi(G)^k}{2 \log |V(G)|} \leq \chi(G^k) \leq \chi(G)^k$, where G^k is the k-wise disjunctive product of G.

Proof. First we prove the upper bound: $\chi(G \vee H) \leq \chi(G) \cdot \chi(H)$. Let $\sigma_G : V(G) \to [\chi(G)]$ and $\sigma_H : V(H) \to [\chi(H)]$ denote minimum proper colorings of G and H, respectively. We define a function

$$\sigma_{G \vee H} : V(G \vee H) \to [\chi(G)] \times [\chi(H)] \quad \text{where} \quad \sigma_{G \vee H}((u, x)) = (\sigma_G(u), \sigma_H(v)).$$

We may think that $\sigma_{G \vee H}$ maps from $V(G \vee H)$ to integers since there is a one-to-one map from $[\chi(G)] \times [\chi(H)]$ to $[\chi(G) \cdot \chi(H)]$. We claim that $\sigma_{G \vee H}$ is a proper coloring of $G \vee H$, that is, any adjacent vertices (u, x) and (v, y) in $G \vee H$ receive different colors under $\sigma_{G \vee H}$. By the definition of the disjunctive product, if (u, x) and (v, y) are adjacent in $G \vee H$, then we must have an edge $uv \in E(G)$ or an edge $xy \in E(H)$.

So either $\sigma_G(u) \neq \sigma_G(v)$ or $\sigma_H(x) \neq \sigma_H(y)$. Therefore,

$$\sigma_{G\vee H}((u,x)) = (\sigma_G(u), \sigma_H(x)) \neq (\sigma_G(v), \sigma_H(y)) = \sigma_{G\vee H}((v,y))$$

Thus, we can color $V \vee G$ by at most $\chi(G) \cdot \chi(H)$ colors. This proves the upper bound.

Next, we prove the lower bound: $\chi(G \vee H) \geq \chi_f(G) \cdot \chi(H)$. We will map a minimum proper coloring of $V \vee G$ to a "fractional" coloring of G. Let $\sigma_{G\vee H} : V(G \vee H) \to [\chi(G \vee H)]$ denote the minimum proper coloring of $G \vee H$. For each vertex $u \in V(G)$, define

$$c_u = |\{\sigma_{G\vee H}((u,x)) : x \in V(H)\}|$$

The number c_u denotes the number of different colors that $\sigma_{G\vee H}$ assigns to vertices of $G \vee H$ that have u as the first coordinate (i.e., vertices of the form (u,x)). Define

$$c^* = \min_{u\in V(G)} c_u.$$

Observe that each vertex $u \in V(G)$ inherits at least c^* colors from vertices of the form (u,x) under $\sigma_{G\vee H}$. Thus, intuitively, if we scale the value of each color by $1/c^*$, then we would have that each vertex of G receives at least "one" color from $(1/c^*) \cdot \sigma_{G\vee H}$. Hence, the scaled colors form a proper fractional coloring of G. Moreover, $c^* \geq \chi(H)$ because, for any vertex $u \in V(G)$, vertices (u,x) and (u,y) of $G \vee H$ receive different colors from $\sigma_{G\vee H}$ if x and y are adjacent in H as it implies $(u,x)(v,y) \in E(G \vee H)$, by the definition of disjunctive product.

To formalize the above discussion, we define a color class for each color $j \in [\chi(G \vee H)]$:

$$I_j = \{u \in V(G) : \text{there is } x \in V(H) \text{ such that } \sigma_{G \vee H}((u, x)) = j\}.$$

Let $\mathbb{I}(G)$ denote the collection of all independent sets in G. We define a fractional coloring function

$$\beta_G : \mathbb{I}(G) \to \mathbb{R} \text{ where } \beta_G(I_j) = \frac{1}{c^*} \quad \text{for all } j \in [\chi(G \vee H)].$$

We claim that each I_j is independent in G. Otherwise, there is a color $j \in [\chi(G \vee H)]$ such that two vertices $u, v \in I_j$ are adjacent in G. This contradicts the definition of disjunctive product since vertices (u, x) and (v, y) must be adjacent in $G \vee H$ for all $x, y \in V(H)$; thus, u, v cannot be in the same color class for $G \vee H$. Now β_G is a proper fractional coloring of G because

$$\sum_{S \in \mathbb{I}(G): u \in S} \beta_G(S) \;=\; \sum_{j \in [\chi(G \vee H)]: u \in I_j} \beta_G(I_j) \;=\; \sum_{j \in [\chi(G \vee H)]: u \in I_j} \frac{1}{c^*} \;=\; \frac{c_u}{c^*} \;\geq\; 1$$

As β_G is a proper fractional coloring of G, we have

$$\chi_f(G) \;=\; \sum_{S \in \mathbb{I}(G)} \beta_G(S) \;\leq\; \frac{1}{c^*} \cdot \chi(G \vee H) \;\leq\; \frac{\chi(G \vee H)}{\chi(H)}$$

Thus, $\chi_f(G) \cdot \chi(H) \leq \chi(G \vee H)$.

Now, consider G^k. We have

$$\chi_f(G) \cdot \chi(G^{k-1}) \leq \chi(G^k) \leq \chi(G)^k$$

By the relationship between $\chi(G)$ and $\chi_f(G)$ in Lemma 2.1.1, we then obtain

$$\frac{\chi(G)^k}{2\log|V(G)|} \leq \chi(G^k) \leq \chi(G)^k$$

□

When we only consider the fractional chromatic number, the bound is actually tight in the sense that the multiplicative bound holds in equality.

Theorem 4.1.4 (Fractional Chromatic Number of Disjunctive Product)**.** *For any graphs G and H,*

$$\chi_f(G) \cdot \chi_f(H) = \chi_f(G \vee H).$$

In particular, for any integer $k \geq 1$, $\chi_f(G^k) = \chi_f(G)^k$, where G^k is the k-wise disjunctive product of G.

Proof. The proof is similar to that for the (integral) chromatic number. First, we prove the upper bound: $\chi_f(G \vee H) \leq \chi_f(G) \cdot \chi_f(H)$. For any graph G, let $\mathbb{I}(G)$ denote the set of all independent sets in G. Let $\beta_G : \mathbb{I}(G) \to \mathbb{R}_0^+$ and $\beta_H : \mathbb{I}(H) \to \mathbb{R}_0^+$ denote minimum proper fractional colorings of G and H, respectively. We define a function

$$\beta_{G\vee H} : \mathbb{I}(G) \times \mathbb{I}(H) \to \mathbb{R}_0^+$$

where

$$\beta_{G\vee H}(S) = \beta_G(S_G) \cdot \beta_H(S_H) \text{ for all } S = S_G \times S_H, S_G \in \mathbb{I}(G), S_H \in \mathbb{I}(H).$$

We claim that $\beta_{G\vee H}$ is a proper fractional coloring of $G \vee H$. By Corollary 4.1.2, we have that $S = S_G \times S_H$ is independent in $G \vee H$. So, it only remains to show that every vertex $(u, x) \in V(G \vee H)$ is covered by $\beta_{G\vee H}$ (that is, $\sum_{S\in\mathbb{I}(G\vee H):(u,x)\in S} \beta_{(G\vee H)}(S) \geq 1$). Summing the fractional value assigned to all the independent sets in $G \vee H$ containing (u, x), we have

$$\begin{aligned}
\sum_{S\in\mathbb{I}(G\vee H):(u,x)\in S} \beta_{G\vee H}(S) &= \sum_{S_G\in\mathbb{I}(G):u\in S_G} \sum_{S_H\in\mathbb{I}(H):x\in S_H} \beta_{G\vee H}(S_G \times S_H) \\
&= \sum_{S_G\in\mathbb{I}(G):u\in S_G} \sum_{S_H\in\mathbb{I}(H):x\in S_H} \beta_G(S_G) \cdot \beta_H(S_H) \\
&= \sum_{S_G\in\mathbb{I}(G):u\in S_G} \beta_G(S_G) \cdot \left(\sum_{S_H\in\mathbb{I}(H):x\in S_H} \beta_H(S_H) \right) \\
&\geq \sum_{S_G\in\mathbb{I}(G):u\in S_G} \beta_G(S_G) \\
&\geq 1
\end{aligned}$$

The final two inequalities follow as β_G and β_H are proper fractional colorings. Thus $\chi_f(G \vee H) \leq \chi_f(G) \cdot \chi_f(H)$.

Next, we prove the lower bound: $\chi_f(G \vee H) \geq \chi_f(G) \cdot \chi_f(H)$. We will map a minimum proper fractional coloring of $G \vee H$ to fractional colorings of G and H, respectively. Let $\beta_{G\vee H} : \mathbb{I}(G \vee H) \to \mathbb{R}_0^+$ be a minimum fractional proper coloring of $G \vee H$. Define two projections $\text{proj}_H^u : V(G \vee H) \to V(H)$ and $\text{proj}_G : V(G \vee H) \to$

$V(G)$ as

$$\text{proj}_H^u(S) = \{x \in V(H) : \text{there is a vertex } (u, x) \in S.\}$$

$$\text{proj}_G(S) = \{u \in V(G) : \text{there is a vertex } (u, x) \in S.\}$$

The function proj_H^u projects vertices of $G \vee H$ whose the first coordinate is u to vertices of H, and the function proj_G projects vertices of $G \vee H$ to vertices of G.

Now consider a vertex $u \in V(G)$ and a set of vertices $S_H \subseteq V(H)$. By Corollary 4.1.2, the set $u \times S_H \in V(G \vee H)$ is independent in $G \vee H$ if and only if S_H is independent in H.

Next, define a function $\beta_u : \mathbb{I}(H) \to \mathbb{R}$ as

$$\beta_u(S_H) = \sum_{S \in \mathbb{I}(G \vee H) : \text{proj}_H^u(S) = S_H} \beta_{G \vee H}(S)$$

It can be seen that all the independent sets in $G \vee H$ containing a vertex $(u, x) \in V(G \vee H)$ are in the set $\bigcup_{S_H \in \mathbb{I}(H) : x \in S_H} \{S \in \mathbb{I}(G \vee H) : \text{proj}_H^u(S) = S_H\}$.

Thus, for any vertex $x \in V(H)$, we have

$$\sum_{S_H \in \mathbb{I}(H) : x \in S_H} \beta_u(S_H) \geq 1.$$

In other words, β_u is a proper fractional coloring of H with value $\sum_{S_H \in \mathbb{I}(H)} \beta_u(S_H)$. So, $c^* = \min_{u \in V(G)} \sum_{S_H \in \mathbb{I}(H)} f(S_H)$ upper bounds the fractional chromatic number of H.

Now, we define a fractional coloring $\beta_G : \mathbb{I}(G) \to \mathbb{R}$ of G as

$$\beta_G(S_G) = \frac{1}{c^*} \cdot \sum_{S \in \mathbb{I}(G \vee H) : \text{proj}_G(S) = S_G} \beta_{G \vee H}(S).$$

We will verify that β_G is a proper fractional coloring of G. Consider any vertex $u \in V(G)$. Observe that any set $S \in \mathbb{I}(G \vee H)$ such that $u \in \text{proj}_G(S)$ must contain the set $u \times S_H$ for some independent set $S_H \in \mathbb{I}(H)$. Thus, we have

$$\begin{aligned}
\sum_{S_G \in \mathbb{I}(G): u \in S_G} f(S_G) &= \frac{1}{c^*} \cdot \sum_{S_G \in \mathbb{I}(G): u \in S_G} \sum_{S \in \mathbb{I}(G \vee H): \text{proj}_G(S) = S_G} \beta_{G \vee H}(S) \\
&= \frac{1}{c^*} \cdot \sum_{S_H \in \mathbb{I}(H)} \sum_{S \in \mathbb{I}(G \vee H): \text{proj}_H^u(S) = S_H} \beta_{G \vee H}(S) \\
&= \frac{1}{c^*} \cdot \sum_{S_H \in \mathbb{I}(H)} \beta_u(S_H) \\
&\geq 1
\end{aligned}$$

So, β_G is a proper fractional coloring of G, which implies that

$$\begin{aligned}
\chi_f(G) &\leq \sum_{S_G \in \mathbb{I}(G)} \beta_G(S_G) \\
&= \frac{1}{c^*} \cdot \sum_{S_G \in \mathbb{I}(G)} \sum_{S \in \mathbb{I}(G \vee H): \text{proj}_G(S) = S_G} \beta_{G \vee H}(S) \\
&= \frac{1}{c^*} \cdot \sum_{S \in \mathbb{I}(G \vee H)} \beta_{G \vee H}(S) \\
&= \frac{\chi_f(G \vee H)}{c^*} \\
&\leq \frac{\chi_f(G \vee H)}{\chi_f(H)}
\end{aligned}$$

Therefore, $\chi_f(G) \cdot \chi_f(H) \leq \chi_f(G \vee H)$, as desired. □

4.2 Randomized Graph Product

The *randomized product* of two graphs is defined as a random induced subgraph of the deterministic product. Formally, a *randomized disjunctive product* Z of G and

H is an induced subgraph $Z = (G \vee H)[S]$, where S is a random subset of $V(G \vee H)$. The *k-wise randomized graph product* of G is, thus, defined as an induced subgraph $Z_k = G^k[S]$ where S is a random subset of $V(G^k)$.

The motivation for using the randomized graph product is to sparsify the graph G^k whilst preserving important properties, such as the independence ratio (that is, the ratio $\frac{\alpha(G)}{|V(G)|}$).

We may think of the randomized graph product as a function (or an algorithm) $\mathsf{RandProduct}(G, k, t)$ that returns the graph $Z_k = G^k[S]$, where S is a set of t vertices sampled uniformly at random from $V(G^k)$. A trivial implementation of $\mathsf{RandProduct}(G, k, t)$ runs in time $O(|V(G)|^{2k})$: we first construct G^k, then randomly sample a set S of t vertices from $V(G^k)$, and then output the graph $Z_k = G^k[S]$. However, the running time of this naive implementation can be huge compared to the size of the resulting graph Z_k. Observe, however, that it is not necessary to explicitly compute G^k. A vertex $(u_1, \ldots, u_k)$ of G^k can be constructed by choosing k vertices $u_1, \ldots, u_k$ uniformly at random from $V(G)$. It is easy to see that this process outputs a vertex of G^k with uniform probability. Thus, one can implement $\mathsf{RandProduct}$ to run in $O(k \cdot |S|^2)$ time, and this is linear in the size of the output graph when k is a constant. This implementation of $\mathsf{RandProduct}$ is presented in Algorithm 4.2.

This fast implementation of the randomized graph product was given by Berman and Schnitger in [8]; see also [31]. In addition, Berman and Schitger [8] showed that, for sufficiently large t, the randomized graph product $Z_k = \mathsf{RandProduct}(G, j, t)$ inherits almost the same independence ratio as G^k. A similar result was shown by

Algorithm 1 RandProduct$(G, k, t,)$

$S := \emptyset$.
for $i := 1$ to t **do**
  for $j := 1$ to k **do**
    Uniformly and independently at random pick a vertex $u_j \in V(G)$.
  end for
  Add a vertex $(u_1, u_2, \ldots, u_k)$ to S.
end for
Find the edge set F by checking if there is an edge between each pair of vertices in S.
Output the graph $Z_k = (S, F)$.

Feige [31] for the chromatic number. These proofs are based upon the Chernoff bounds. (See [58] for details on Chernoff bounds.)

Theorem 4.2.1 (Chernoff's bounds)**.** *Let $X_1, X_2, \ldots, X_n$ be independent random variables taking values 0 or 1, Let $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbf{E}[X]$. Then the following hold:*

(i) For $0 < \delta < 1$,

$$\mathbf{Pr}[X > (1+\delta)\mu] \leq \exp\left(-\frac{\delta^2\mu}{3}\right) \quad \textit{and} \quad \mathbf{Pr}[X < (1-\delta)\mu] \leq \exp\left(-\frac{\delta^2\mu}{2}\right).$$

(ii) For $R \geq 6\mu$,

$$\mathbf{Pr}[X > R] \leq 2^{-R}.$$

4.2.1 Independence Ratio of Randomized Graph Product

Now, we prove the sampling lemma for the independence ratio of the randomized graph product.

Lemma 4.2.2 (Sampling Lemma for Independence Ratio [8])**.** *Consider the k-wise disjunctive product G^k of G. Let $n = |V(G)|$ and $\alpha = \alpha(G)$, and let $Z_k =$*

$\mathsf{RandProduct}(G, k, t)$ *be the* k*-wise randomized graph product of* G *on at most* t *vertices, i.e.,* Z_k *is a random subgraph of* G *on at most* t *vertices. Then, for* $t \geq k \cdot \frac{n^k}{\alpha^{k-1}} \log n$*, with high probability,*

$$\frac{\alpha(Z_k)}{|V(Z_k)|} = \Theta\left(\frac{\alpha^k}{n^k}\right) = \Theta\left(\frac{\alpha(G^k)}{|V(G^k)|}\right)$$

Proof. The intuition behind the proof is that we may think of Z_k as being constructed from G^k by randomly removing vertices, some of which must be in a maximum independent set. Thus, the change in the size of any (maximum) independent set is in the same proportion as the change in the number of vertices of the graph. In other words, the random sampling maintains the independence ratio.

To prove the statement formally, we have to guarantee that all the independent sets become smaller after the random sampling. Note that $|V(G)| = n^k$ and, by Theorem 4.1.1, $\alpha(G^k) = \alpha^k$. It is easy to prove by a standard analysis that $|V(Z_k)| < t/2$ with very small probability. So, we may assume that $t/2 \leq |V(Z_k)| \leq t$. Let us fix any maximum independent set I in G^k. The procedure $\mathsf{RandProduct}(G, k, t)$ samples each vertex of G independently and uniformly with probability t/n^k. We may write t as

$$t = \gamma \cdot \left(k \cdot \frac{n^k}{\alpha^{k-1}} \cdot \log n\right) \text{ for some } \gamma \geq 1.$$

Thus, the expected number of vertices of I that are also in $V(Z_k)$ is $(t/n^k) \cdot |I|$. So, we have

$$\begin{aligned}
\mu = \mathbf{E}[|I \cap V(Z_k)|] &= \frac{t}{n^k} \cdot |I| \\
&= \gamma \left(k \cdot \left(\frac{n^k}{\alpha^{k-1}} \right) \cdot \log n \right) \left(\frac{|I|}{n^k} \right) \\
&\leq \gamma \left(k \cdot \left(\frac{n^k}{\alpha^{k-1}} \right) \cdot \log n \right) \left(\frac{\alpha^k}{n^k} \right) \\
&\leq \gamma \cdot k \cdot \alpha \cdot \log n
\end{aligned}$$

By Chernoff's bounds (Theorem 4.2.1 (ii)), we have

$$\mathbf{Pr}\left[|I \cap V(Z_k)| < 6\gamma \cdot k \cdot \alpha \cdot \log n\right] \leq 2^{-6\gamma \cdot k \cdot \alpha \cdot \log n} = n^{-6\gamma \cdot k \cdot \alpha}$$

Because the maximum independent set in G has cardinality α, the graph G contains at most n^α independent sets. By Corollary 4.1.2, each independent set in G^k is a Cartesian product of k independent sets in G. Therefore, there are at most $n^{-k\alpha}$ independent sets in G^k. Since $\gamma \geq 1$, we may apply the union bound over all independent sets in G^k to give, with high probability, that

$$|I \cap V(Z_k)| < 6 \cdot \gamma \cdot k \cdot \alpha \cdot \log n, \quad \text{for all independent sets } I \text{ in } G^k.$$

That is, $\alpha(Z_k) \leq 6 \cdot \gamma \cdot k \cdot \alpha \log n$ with high probability. Thus, we have the independence ratio:

$$\frac{\alpha(Z_k)}{|V(Z_k)|} \leq \frac{6 \cdot \gamma \cdot k \cdot \alpha \cdot \log n}{t/2} = \frac{12 \cdot \gamma \cdot k \cdot \alpha \cdot \log n}{\gamma \left(k \cdot \frac{n^k}{\alpha^{k-1}} \cdot \log n \right)} = \frac{12\alpha^k}{n^k}.$$

The proof for the lower bound is similar. Take a maximum independent set I in G^k. We know that the expected size of $I \cap V(Z_k)$ is

$$\mathbf{E}[|I \cap V(Z_k)|] = \gamma \cdot k \cdot \alpha \cdot \log n$$

By Chernoff's bounds (Theorem 4.2.1 (i)), we have

$$\mathbf{Pr}\left[|I \cap V(Z_k)| > \frac{1}{2}\gamma \cdot k \cdot \alpha \log n\right] \leq \exp\left(\frac{1}{8}\gamma \cdot k \cdot \alpha \log n\right)$$

That is, with high probability,

$$\frac{\alpha(Z_k)}{|V(Z_k)|} \geq \frac{\frac{1}{2} \cdot \gamma \cdot k \cdot \alpha \log n}{t} \geq \frac{\frac{1}{2} \cdot \gamma \cdot k \cdot \alpha \log n}{\gamma\left(k \cdot \frac{n^k}{\alpha^{k-1}} \cdot \log n\right)} = \frac{1}{2} \cdot \frac{\alpha^k}{n^k}$$

Combining lower and upper bounds, we have that, with high probability,

$$\frac{1}{2} \cdot \frac{\alpha(G^k)}{|V(G^k)|} \leq \frac{\alpha(Z_k)}{|V(Z_k)|} \leq 12\frac{\alpha^k}{n^k}$$

In other words, $\frac{\alpha(Z_k)}{|V(Z_k)|} = \Theta\left(\frac{\alpha(G^k)}{|V(G^k)|}\right)$ as desired. □

The randomized graph product that preserves independence ratio can be made deterministic by the method of Alon, Feige, Wigderson and Zuckerman in [1].

4.2.2 Chromatic Number of Randomized Graph Product

Next we give the sampling lemma for the chromatic number of a randomized graph product – this was was originally proven by Feige [31]. That is, we will show that with large enough $t \geq 0$, the randomized graph product $Z_k = \mathsf{RandProduct}(G, k, t)$ has approximately the same chromatic number as that of G^k. It is easy to see that the chromatic number of any subgraph $Z_k \subseteq G^k$ is $\chi(G^k)$. However, it is not clear that Z_k requires $\chi(G^k)$ colors. An alternative lower bound

used by Feige is the converse of the independence ratio of a graph. He showed that

$$\chi_f(Z_k) \geq \Theta\left(\frac{|V(G^k)|}{\alpha(G^k)}\right)$$

The above inequality, indeed, follows from the fact that any color class of G is an independent set and, thus, has cardinality at most $\alpha(G)$. So, we need at least $|V(G)|/\alpha(G)$ colors to color all the vertices. To be formal, let us prove the following claim.

Claim 4.2.3. *For any graph G, $\chi_f(G) \geq \frac{|V(G)|}{\alpha(G)}$.*

Proof. Consider any minimum proper fractional coloring β of G. By definition, β covers every vertex of G. That is, $\sum_{S \in \mathbb{I}(G): v \in S} \beta(G) \geq 1$ for all $v \in V(G)$. Thus,

$$\begin{aligned}
|V(G)| &\leq \sum_{v \in V(G)} \sum_{S \in \mathbb{I}(G): v \in S} \beta(S) \\
&= \sum_{S \in \mathbb{I}(G)} \sum_{v \in S} \beta(S) \\
&= \sum_{S \in \mathbb{I}(G)} \beta(S) \cdot |S| \\
&\leq \sum_{S \in \mathbb{I}(G)} \beta(S) \cdot \alpha(G) \\
&= \alpha(G) \cdot \sum_{S \in \mathbb{I}(G)} \beta(S) \\
&= \alpha(G) \cdot \chi_f(G)
\end{aligned}$$

Therefore, $\chi_f(G) \geq |V(G)|/\alpha(G)$ as claimed. □

Now, we apply the sampling lemma for the independence ratio to prove the sampling lemma for the chromatic number.

Lemma 4.2.4 (Sampling Lemma for Chromatic Number [31])**.** *Consider the k-wise disjunctive product G^k of G. Let $n = |V(G)|$ and $\alpha = \alpha(G)$, and let $Z_k = \mathsf{RandProduct}(G, k, t)$ be the k-wise randomized graph product of G on at most t vertices, i.e., Z_k is a random subgraph of G on at most t vertices. Then, for $t \geq k \cdot \frac{n^k}{\alpha^{k-1}} \log n$, with high probability,*

$$\Theta\left(\frac{n^k}{\alpha^k}\right) \leq \chi_f(Z_k) \leq \chi_f(G^k)$$

In particular, if $\frac{n^k}{\alpha^k} = \Theta(\chi_f(G^k))$, then $\chi_f(Z_k) = \Theta(\chi_f(G^k))$.

Proof. First, note that $|V(G^k)| = n^k$ and, by Theorem 4.1.1, $\alpha(G^k) = \alpha^k$.

The upper bound is trivial because Z_k is a subgraph of G^k, which means that we can use a fractional coloring of G^k to color Z^k. For the lower bound, we use the fact that the converse of the independence ratio of a graph gives a lower bound on the chromatic number. Specifically,

$$\chi_f(Z_k) \geq \frac{|V(Z_k)|}{\alpha(Z_k)}$$

By Lemma 4.2.2 and the choice of t, we must have that

$$\frac{|V(Z_k)|}{\alpha(Z_k)} = \Theta\left(\frac{|V(G^k)|}{\alpha(G^k)}\right) = \Theta\left(\frac{n^k}{\alpha^k}\right)$$

This completes the proof. □

CHAPTER 5
Subexponential-Time Approximation Hardness of Classical Results

Here we consider the subexponential-time approximation hardness of independent set and graph coloring. These classical problems were shown to be NP-hard in the seminal work of Karp [49] and are amongst the earliest problems studied in approximation algorithms.

First, let's discuss the maximum independent set problem. Hardness results for this problem have relied heavily upon *probabilistically checkable proof* (PCP) systems. The connection between independent set and PCPs was first discovered by Feige et al. [33]. He showed that independent set is hard to approximate to within a factor of $2^{\log^{1-\epsilon} n}$, for any $\epsilon > 0$, unless NP $\subseteq$ DTIME($n^{\text{polylog}n}$). This inapproximability result was improved by Arora and Safra [3], and then by Arora et al. [2] to show polynomial hardness.

Later, Bellare and Sudan [7] introduced the notion of *amortized free-bit complexity* of a PCP. They showed that, given a PCP with logarithmic randomness and amortized free-bit complexity f, the maximum independent set problem is hard to approximate to within a factor of $n^{1/(1+f)-\epsilon}$, for all $\epsilon > 0$, unless NP = ZPP. In particular, they [7] constructed a PCP with amortized free-bit complexity $f = 3 + \delta$, for all $\delta > 0$, thus proving a hardness of $n^{1/4-\epsilon}$ for independent set. Bellare et al. [6] then gave an improved construction with amortized free-bit complexity $f = 2 + \delta$.

Finally, Håstad [42] constructed a PCP with arbitrary small amortized free-bit complexity $f > 0$. This shows a tight hardness result (up to lower order terms) of $n^{1-\epsilon}$, for all $\epsilon > 0$, for independent set.

A PCP with optimal amortized free-bit complexity was first constructed by Samorodnitsky and Trevisan [65]. Their PCP, however, has imperfect completeness, that is, completeness of less than one. Håstad and Khot [43] constructed a PCP that has *both* perfect completeness and optimal amortized free-bit complexity. Recently, Moshkovitz and Raz [59] gave a construction of a projective 2-query PCP with almost-linear size that can be combined with the result of Samorodnitsky and Trevisan [65] to produce a PCP with almost-linear size and optimal free-bit complexity. The soundness of a PCP with optimal free-bit complexity was improved in a recent breakthrough result of Chan [20]. We remark that the complexity assumption of early tight hardness results for independent set (for example, Håstad [42]) was NP $\neq$ ZPP, because of a random process used in the PCP constructions. This process was derandomized by Zuckerman [70] to prove tight hardness under the assumption P $\neq$ NP.

Now let's turn to the complexity of the graph coloring problem. The NP-hardness of 3-coloring due to Karp [49] immediately implies that graph coloring has a hardness factor of 4/3. To see this, observe that a graph that cannot be 3-colored requires at least 4 colors. Garey and Johnson [37] amplified this hardness gap to a factor $2 - \epsilon$ using graph products. Linial and Vazirani [53] subsequently showed that the graph product technique can, in fact, amplify the gap up to a factor $2^{\log^{1-\epsilon} n}$, for any $\epsilon > 0$. Whilst the hardness of the graph coloring problem can be

obtained by gap amplification, Lund and Yannakakis [57] described an alternative method to prove hardness. They gave a reduction from a hard instance of independent set to graph coloring. Their result shows, in particular, that if independent set has an approximation hardness of n^{δ}, for some $0 < \delta < 1$, then graph coloring has a hardness of $n^{\delta'}$, for some $0 < \delta' \leq \delta$. The result was improved by Fürer in [36] where he showed that an n^{δ}-hardness of independent set implies an $n^{\frac{1}{3}\delta}$-hardness of graph coloring. More precisely, taking into account the amortized free-bit complexity of the PCP (in the hardness proof of independent set), denoted by f, Fürer's result gives a hardness of $n^{1/(\max(1+2f,2)+o(1))}$ for the graph coloring.

Obtaining a tight hardness result for graph coloring requires a more sophisticated method, due to Feige and Kilian [34], based upon both PCPs and graph products. They introduced the *covering parameter* of a PCP and proved that the *long-code* based PCP of Håstad [42] (used to obtained the tight hardness of independent set) can be modified to prove tight hardness (up to lower order terms) of graph coloring. In particular, they showed that for some constant ρ, it is hard to decide whether a graph is ρ-colorable or it has no large independent set of size $O(n/\rho)$ – and, thus, needs more than $O(\rho)$ colors. Then, using the randomized graph product of Feige [31], Feige and Kilian amplified the gap to give $n^{1-\epsilon}$-hardness for graph coloring, for any $\epsilon > 0$.

Furthermore, as shown by Engebretsen and Holmerin [30], the method of Feige and Killian can also be applied to the PCP of Samorodnitsky and Trevisan [65]. Consequently, one can combine this result with that of Moshkovitz and Raz [59] on

the almost linear-size 2-query PCP to obtain subexponential-time hardness for graph coloring.

5.1 Overview

We now give an overview of our proofs. Precise calculations will be deferred to the next section.

The classical results on the hardness of the independent set and graph coloring in [42] and [34] were obtained via a reduction from q-CSP combined with a randomized graph product. Both results follow by the same reduction. Starting from an instance of q-CSP on N variables and M clauses with small hardness gap, say r_0, and with a relatively small number of accepting configurations, say $w \ll r_0$, we apply the FGLSS reduction to obtain a base graph G_0. The reduction guarantees that $n = |V(G_0)| \leq wM$ and $\alpha = \alpha(G_0) \geq r_0 M$. We then apply the randomized graph product to obtain the graph $G = \mathsf{RandProduct}(G_0, k, kn/r_0^k)$ (the parameters roughly satisfy the conditions in Lemma 4.2.2). The hardness gap we then obtain is approximately $1/r$ where $r = r_0^k$. The number of vertices of the graph G is kn/r. By taking $k = t \log n$, for any $t > 0$, we have a gap of n^t and the graph G with n^{t+1} vertices. Thus, this gives a $|V(G)|^{1-\epsilon}$-hardness since t can be any large constant. The hardness of the graph coloring problem follows analogously.

For subexponential-time approximation hardness, we consider the q-CSP instance obtained by applying the PCP in Theorem 3.2.3. This gives a reduction from the decision version of 3-SAT of size N to an instance of q-CSP of size $N' = N^{1+o(1)}$. Thus, we have a q-CSP instance with at most N' clauses. For clarity of presentation, assume for now that $N' = N$. We then apply the reduction from q-CSP to independent set (respectively, graph coloring). The size of the instance in each step evolves

as follows:

$$\underbrace{\text{SAT on } N \text{ vars}}_{\text{no gap}} \implies \underbrace{q\text{-CSP on } N \text{ vars}}_{\text{small gap } r_0} \implies \underbrace{\text{Ind.Set } G_0}_{\text{small gap } r_0} \implies \underbrace{\text{Ind.Set } G \;\&\; |V(G)| = rN}_{\text{gap } r}$$

The final graph G has rN vertices. So, if there is an r-approximation algorithm for independent set (respectively, graph coloring) that runs in time $2^{(|V(G)|/r)^{1-\epsilon}}$, for some $\epsilon > 0$, then 3-SAT can be solved in time $2^{N^{1-\epsilon}}$, contradicting the Exponential-Time Hypothesis.

5.2 Independent Set

We now present a formal proof of the subexponential-time approximation hardness for the maximum independent set problem. We apply a reduction from an instance ψ of 3-SAT of size N to a q-CSP instance ϕ of size $N^{1+o(1)}$ as in Theorem 3.2.3. Moreover, for a given parameter $\ell > 0$, ϕ has the following properties:

Parameters	
Number of variables	$n = N^{1+o(1)}$
Number of clauses	$m = N^{1+o(1)}$
Number of variables in each clause	$q = \ell^2 + 2\ell$
Completeness (ϕ is satisfiable.)	$c = 1 - o(1)$
Soundness (ϕ is not satisfiable.)	$s \leq 2^{-\ell^2+1}$
Maximum number of accepting configurations for each clause	$w \leq 2^{2\ell}$

Because each clause has at most w configurations, the FGLSS reduction gives a graph G_0 with $|V(G_0)| = m \cdot w$ vertices. In addition,

- **Completeness:** If the SAT instance ψ is satisfiable then $\alpha(G_0) \geq c \cdot m$.

- **Soundness:** If the SAT instance ψ is not satisfiable then $\alpha(G_0) \leq s \cdot m$.

Thus, we have a hardness gap of c/s. Note that w can be arbitrarily smaller than $1/s$, and $1/c$ can be arbitrary smaller than w. This is the important property of the q-CSP instance obtained from Theorem 3.2.3. Indeed, we may ignore the precise values of c, s and w and assume that $(1/c) \ll w \ll (1/s)$.

For future use, we state the following theorem, which follows from Theorem 3.2.3 and Lemma 2.4.2 (FGLSS Reduction). We may assume that each clause has at least one accepting configuration; otherwise, we remove the clause.

Theorem 5.2.1 (Corollary of Theorem 3.2.3)**.** *For any constant $k > 0$, let $c = 1 - o(1)$, $s = 2^{-\ell^2+1}$, $w = 2^{2\ell}$ and $q = 2^{\ell^2+2\ell}$. Thus, $w \leq (c/s)^{1/\ell}$. Then there is a polynomial-time reduction that, given a SAT instance ϕ of size N with $m = N^{1+o(1)}$, outputs a graph G on $wm = wN^{1+o(1)}$ vertices. The graph G has $\alpha(G) \geq m/2^q$, and moreover,*

- YES-INSTANCE*: If ϕ is satisfiable, then $\alpha(G) \geq cm = cN^{1+o(1)}$.*
- NO-INSTANCE*: If ϕ is not satisfiable, then $\alpha(G) \leq sm = sN^{1+o(1)}$.*

□

Here we use the disjunctive product as the default graph product. We then apply the randomized graph product to obtain the graph:

$$G = \mathsf{RandProduct}\left(G_0, k, k \cdot (wm \log wm) \cdot (2^q w)^{k-1}\right)$$

First, let's verify that these parameters satisfy the conditions in Lemma 4.2.2. Since we use the disjunctive product, the condition $\alpha(G^k) = \alpha(G)^k$ holds by Theorem 4.1.1. For the size condition, the graph G must have at least $k \cdot \frac{|V(G_0)|^k}{\alpha(G_0)^{k-1}} \cdot \log |V(G_0)|$ vertices.

By substituting the parameters from Theorem 5.2.1, we have

$$k \cdot \frac{|V(G_0)|^k}{\alpha(G_0)^{k-1}} \cdot \log |V(G_0)| \leq k \cdot \frac{(wm)^k}{(2^{-q}m)^{k-1}} \cdot \log wm$$
$$= k \cdot (wm) \log wm \cdot (2^q w)^{k-1}.$$

Thus, for both the cases of YES-INSTANCE and NO-INSTANCE, the conditions in Lemma 4.2.2 are satisfied, which implies that G and G_0^k have almost the same independence ratio, i.e.,

$$\frac{\alpha(G)}{|V(G)|} = \Theta\left(\frac{\alpha(G_0^k)}{|V(G_0^k)|}\right) = \Theta\left(\frac{\alpha(G_0)^k}{|V(G_0)|^k}\right)$$

Hardness Gap. Now we analyze the hardness gap. As discussed, Lemma 4.2.2 implies that

$$\alpha(G) = \Theta\left(|V(G)| \cdot \frac{\alpha(G_0)^k}{|V(G_0)|^k}\right) = \Theta\left(\frac{|V(G)|}{|V(G_0)|^k}\right) \cdot \alpha(G_0)^k$$

Since $|V(G)|/|V(G_0)|^k$ are the same for both a YES-INSTANCE and a NO-INSTANCE, the gap only depends on $\alpha(G_0)^k$.

- YES-INSTANCE: If the SAT instance is satisfiable, then

$$\alpha(G_0) \geq cm \implies \alpha(G) = \Theta\left(\frac{|V(G)|}{|V(G_0)|^k}\right) \cdot \alpha(G_0)^k \geq \Theta\left(\frac{|V(G)|}{|V(G_0)|^k}\right) \cdot (cm)^k$$

- NO-INSTANCE: If the SAT instance is not satisfiable, then

$$\alpha(G_0) \leq sm \implies \alpha(G) = \Theta\left(\frac{|V(G)|}{|V(G_0)|^k}\right) \cdot \alpha(G_0)^k \leq \Theta\left(\frac{|V(G)|}{|V(G_0)|^k}\right) \cdot (sm)^k$$

Thus, the hardness gap is $r = \Theta((c/s)^k)$.

Subexponential-Time Approximation Hardness. To prove subexponential-time hardness, we have to calculate the number of vertices of the graph G and compare it to the size of the SAT instance. Let us first calculate $|V(G)|$. By Lemma 4.2.2, with high probability, we have

$$\begin{aligned}
|V(G)| &\leq k \cdot (wm \log wm) \cdot (2^q w)^{k-1} \\
&\leq k \cdot wm \cdot (\log w + \log m) \cdot (2^q w)^{k-1} \\
&\leq 2k \cdot wm \cdot \log m \cdot (2^q w)^{k-1} \\
&\leq k \cdot m \log m \cdot (2^q w)^k
\end{aligned}$$

We set the range of the parameter k to be $1 \leq k \leq O(\log m)$. We will write $(2^q w)^k$ in terms of $(c/s)^k$, which is the hardness gap. The values of c, s, w, q in Theorem 5.2.1 (the same values as in Theorem 3.2.3) are $c = 1 - o(1), s = 2^{-\ell^2+1}, w = 2^{2\ell}, q = 2^{\ell^2+2\ell}$. Hence, for $\ell \geq 2$,

$$\begin{aligned}
2^q w &= 2^{\ell^2+2\ell} \cdot 2^{2\ell} \\
&= 2^{\ell^2+4\ell} \\
&\leq (2^{-1} \cdot 2^{\ell^2-1}) \cdot (2^{-10/\ell} \cdot 2^{10\ell-10/\ell}) \\
&= (2^{-1} \cdot 2^{\ell^2-1}) \cdot (2^{-1} \cdot 2^{\ell^2-1})^{10/\ell} \\
&\leq \left(\frac{c}{s}\right) \cdot \left(\frac{c}{s}\right)^{10/\ell}
\end{aligned}$$

So, $(2^q w)^k \leq (c/s)^{k(1+10/\ell)}$, for $\ell \geq 2$. Since $r = \Theta((c/s)^k)$, we may bound $(2^q w)^k$ in terms of r as $(2^q w)^k \leq r^{1+\varepsilon}$, where ε can be any small positive real number. Also,

we write $|V(G)|$ in terms of r and m (recall that $m = N^{1+o(1)}$):

$$|V(G)| \leq (k \cdot m \log m) \cdot (2^q w) = O(m \log^2 m \cdot r^{1+\varepsilon}) = N^{1+o(1)} \cdot r^{1+\varepsilon}$$

Thus, for any $\varepsilon > 0$, there is a gap instance G of the maximum independent set problem such that $|V(G)|/r^{1+\varepsilon} \leq N^{1+o(1)}$, where N is the size of the input 3-SAT instance. Recall that the ETH (Hypothesis 3.1.1) asserts that 3-SAT does not admit a $2^{N^{1-\epsilon}}$-time algorithm, for any constant $\epsilon > 0$. Thus, assuming the ETH, the above construction rules out $2^{|V(G)|^{1-\epsilon_1}/r^{1+\epsilon_2}}$-time algorithms for the maximum independent set problem, for any constants $\epsilon_1, \epsilon_2 > 0$.

Theorem 5.2.2. *Let G be any graph. Unless the ETH is false, for any constants $\varepsilon_1, \epsilon_2, \epsilon_3 > 0$, any r-approximation algorithm for the maximum independent set problem must run in time at least $2^{|V(G)|^{1-\epsilon_1}/r^{1+\epsilon_2}}$, where r ranges from some sufficiently large constant to $|V(G)|^{1-\epsilon_3}$.*

5.3 Graph Coloring

The proof of the subexponential-time approximation hardness for the graph coloring problem follows the same reduction as that for the maximum independent set problem except the q-CSP instances we choose are different. We require a q-CSP instance with a small *covering parameter*, which can be obtained from Theorem 3.3.3. Recall the definition of the covering parameter as defined in Section 3.3. We say that a q-CSP instance has a covering parameter ρ if, in the Yes-Instance (i.e., the instance of is obtained from a satisfiable instance of 3-SAT), there is a set of assignments $\{A_1, A_2, \ldots, A_\rho\}$ that *covers* all the accepting configurations, i.e., each

accepting configuration (which is a partial assignment) is a part of one of these assignments.

More precisely, Theorem 3.3.3. says that there is a polynomial-time reduction from an instance ψ of 3-SAT of size N to a q-CSP instance ϕ of size $N^{1+o(1)}$. Moreover, for a given parameter $\ell > 0$, ϕ has the following properties:

Parameters	
The number of variables	$n = N^{1+o(1)}$
The number of clauses	$m = N^{1+o(1)}$
The number of variables in each clause	$q = \ell^2 + 2\ell$
Completeness (ϕ is satisfiable.)	$c = 1 - o(1)$
Soundness (ϕ is not satisfiable.)	$s \leq 2^{-\ell^2+1}$
The maximum number of accepting configurations for each clause	$w \leq 2^{2\ell}$
Covering parameter	$\rho = O(w)$

The FGLSS reduction transforms each accepting configuration into a vertex. Thus, the graph G_0 has $|V(G_0)| = m \cdot w$ vertices. Notice that each assignment corresponds to an independent set in G_0. By the definition of the covering parameter, in Yes-Instance, there is a set of $O(w)$ assignments that covers all the accepting configurations. So, the fractional chromatic number of G_0 of the Yes-Instance is ρ. For the No-Instance, the fractional chromatic number of G_0 can be lower bounded by the independence ratio. Therefore,

- **Completeness:** If the SAT instance ψ is satisfiable, then

$$\chi(G_0) \leq O(\rho) = O(w).$$

- **Soundness:** If the SAT instance ψ is not satisfiable, then

$$\chi(G_0) \geq \frac{|V(G_0)|}{\alpha(G_0)} \geq \frac{w \cdot m}{s \cdot m} = \frac{w}{s}.$$

Thus, we have a hardness gap of $\Omega(1/s)$. Recall w can be arbitrary smaller than $1/s$. That is, $w \leq (1/s)^{3/\ell}$, for any $\ell \geq 1$. So, $w \leq (1/s)^{\epsilon}$, for a chosen parameter $\epsilon > 0$.

For future use, we state the following theorem, which follows from Theorem 3.3.3 and Lemma 2.4.2 (FGLSS Reduction).

Theorem 5.3.1 (Corollary of Theorem 3.3.3)**.** *For any constant $k > 0$, let $c = 1 - o(1)$, $s = 2^{-\ell^2+1}$, $w = 2^{2\ell}$, $\rho = O(w)$ and $q = 2^{\ell+2\ell}$. Thus, $w \leq (c/s)^{1/\ell}$. There is a polynomial-time reduction that, given a SAT instance ϕ of size N and letting $m = N^{1+o(1)}$, outputs a graph G on $wm = wN^{1+o(1)}$ vertices. The graph G has $\alpha(G) \geq m/2^q$, and moreover,*

- YES-INSTANCE*: If ϕ is satisfiable, then $\chi(G) \leq \rho$.*
- NO-INSTANCE*: If ϕ is not satisfiable, then $\alpha(G) \geq w/s$.*

□

Next, we apply the same reduction as that for the maximum independent set problem. That is, we apply the randomized graph product with respect to the disjunctive product to obtain the graph:

$$G = \mathsf{RandProduct}(G_0, k, k \cdot (wm \log wm) \cdot (2^q w)^{k-1})$$

We use the same parameters as before. So, it is easy to see that all the parameters satisfy the conditions in Lemma 4.2.4. Also, by Lemma 4.1.4, $\chi_f(G_0^k) = \chi_f(G_0)^k$.

Thus, we have the graph G such that

$$\frac{\alpha(G)}{|V(G)|} = \Theta\left(\frac{\alpha(G_0)^k}{|V(G_0)|^k}\right) \leq \chi_f(G) \leq \chi_f(G_0)^k$$

Hardness Gap. Now we analyze the hardness gap.

- YES-INSTANCE: If the SAT instance is satisfiable, then

$$\chi_f(G_0) \leq \rho = O(w) \implies \chi_f(G) \leq \chi_f(G_0)^k = z^k w^k \quad \text{for some constant } z > 0.$$

- NO-INSTANCE: If the SAT instance is not satisfiable, then

$$\alpha(G_0) \leq s \cdot m \implies \chi_f(G) \geq \Theta\left(\frac{|V(G_0)|^k}{\alpha(G_0)^k}\right) = \Theta\left(\frac{(w \cdot m)^k}{(s \cdot m)^k}\right) = \Theta\left(\left(\frac{w}{s}\right)^k\right)$$

Thus, the hardness gap is $r = \Omega(1/(zs)^k)$.

Subexponential-Time Approximation Hardness. Now we calculate the ratio $|V(G)|$ to the hardness gap and compare it to the size of the SAT instance. As before, the value of $|V(G)|$ is

$$|V(G)| \leq (k \cdot m \log m) \cdot (2^q w)^k$$

We set the range of the parameter k to be $1 \leq k \leq O(\log m)$. The values of s, w, q in Theorem 5.3.1 (the same values as in Theorem 3.3.3) are $s = 2^{-\ell^2+1}, w =$

$2^{2\ell}, q = 2^{\ell^2+2\ell}$. Hence, for $\ell \geq \max\{2, \log z\}$,

$$\begin{aligned} 2^q w &= 2^{\ell^2+2\ell} \cdot 2^{2\ell} \\ &= 2^{\ell^2+4\ell} \\ &\leq (z^{-1} \cdot 2^{\ell^2-1}) \cdot (z^{-10/\ell} \cdot 2^{10\ell-10/\ell}) \\ &= (z^{-1} \cdot 2^{\ell^2-1}) \cdot (z^{-1} \cdot 2^{\ell^2-1})^{10/\ell} \\ &= \left(\frac{1}{zs}\right) \cdot \left(\frac{1}{zs}\right)^{10/\ell} \end{aligned}$$

So, $(2^q w)^k \leq (1/zs)^{k(1+10/\ell)}$, for $\ell \geq \max\{2, \log z\}$. Since $r = \Omega((1/zs)^k)$, we may write $(2^q w)^k$ in terms of r as $(2^q w)^k = r^{1+\varepsilon}$, where ε can be any small positive real number. Also, we write $|V(G)|$ in terms of r and m (recall that $m = N^{1+o(1)}$):

$$|V(G)| \leq k \cdot m \log m \cdot (2^q w)^k = k \cdot m \log m r^{1+\varepsilon} \leq O(m \log^2 m) r^{1+\varepsilon} = N^{1+o(1)} \cdot r^{1+\varepsilon}.$$

The last equality follows because $m = N^{1+o(1)}$. Assume the ETH (Hypothesis 3.1.1). That is, 3-SAT does not admit a $2^{N^{1-\epsilon}}$-time algorithm, for any constant $\epsilon > 0$. Then the above construction rules out $2^{|V(G)|^{1-\epsilon}/r^{1+\varepsilon}}$-time algorithms for the graph coloring problem, for any constant $\epsilon, \varepsilon > 0$.

Theorem 5.3.2. *Let G be any graph. Unless the ETH is false, for any constants $\epsilon_1, \epsilon_2, \epsilon_3 > 0$, any r-approximation algorithm for the graph coloring problem must run in time at least $2^{|V(G)|^{1-\epsilon_1}/r^{1+\epsilon_2}}$, where r ranges from some sufficiently large constant to $|V(G)|^{1-\epsilon_3}$.*

CHAPTER 6
Subexponential-Time Hardness from Graph Product

In this chapter, we present a subexponential-time approximation hardness result obtained from the randomized graph product. The problem considered in this chapter is the maximum bipartite induced matching problem.

Our reduction is simple and is based on the well-known graph transformations, namely the *bipartite double cover*, denoted by $B[G]$, and the *extended bipartite double cover*, denoted by $B_e[G]$.

We construct $B[G]$ by making two copies of each vertex $v \in V(G)$, namely, $(v, 1)$ and $(v, 2)$. Then, for each edge $uv \in E(G)$, we add edges $(u, 1)(v, 2)$ and $(u, 2)(v, 1)$. So, $B[G]$ is a bipartite graph with bi-partition V_1, V_2, where $V_i = \{(v, i) : v \in V(G)\}$ for $i = 1, 2$. To obtain $B_e[G]$, we simply add to $B[G]$ an edge joining the two copies of each vertex $v \in V(G)$. That is, we add an edge between $(v, 1)$ and $(v, 2)$ for all $v \in V(G)$.

This transformation might sound familiar to many readers. The *bipartite double cover* and has been used repeatedly as a natural way to transform any graph into a bipartite graph; for example, one can use this transformation to reduce the problem of computing cycle covers to the maximum bipartite matching problem.

6.1 Our Results

The hardness result in this chapter is obtained by the graph product technique that we have developed in [17]. The key theorem that we use in the proof is the graph product inequality:

Theorem 6.1.1 (Graph Product Inequality)**.** *For any graphs G and H,*

$$\mathsf{im}(B[G \vee H]) \leq \mathsf{im}(B[G]) + \mathsf{im}(B[H]). \tag{6.1}$$

Also, we need the following lemma implicitly proved in [29] and [44]. In fact, the theorem follows from the observation that edges in $B_e[G]$ can be partitioned into two sets, those that correspond to vertices of G and those that correspond to edges.

Lemma 6.1.2 (Bipatite Double Cover Inequality [29])**.** *For any graphs G and H,*

$$\alpha(G) \leq \mathsf{im}(B_e[G]) \leq \mathsf{im}(B[G]) + \alpha(G). \tag{6.2}$$

Theorem 6.1.1 shows the change in the graph measures after applying graph products. Lemma 6.1.2 says that we can partition edges in the extended bipartite double cover $B_e[G]$ into two sets, those that correspond to vertices of G and those that correspond to edges, which are in $B[G]$. To see this, let us partition edges of $B_e[G]$ into two sets E_1 and E_2 where

$$E_1 = \{(v,1)(v,2) : v \in V(G)\}$$
$$E_2 = \{(v,1)(w,2) : v, w \in V(G) \text{ and } v \neq w\}$$

Each edge $(v,1)(v,2) \in E_1$ corresponds to the vertex $v \in V(G)$ while edges $(v,1)(w,2)$ and $(w,1)(v,2)$ correspond to the edge $vw \in E(G)$. It is easy to see that, for any

induced matching M in $B_e[G]$, the set $E_1 \cap M$ maps to an independent set in G. But, it is not clear whether we can map $E_2 \cap M$ to an independent set in G. We thus consider $E_2 \cap M$ as the "gabage" term. Since edges in E_2 are in $B[G]$, it suffices to consider the gabage term in the graph $B[G]$.

The inequality in Theorem 6.1.1 implies the tight approximation hardness for the bipartite induced matching problem. However, the original construction does not imply the subexponential-time approximation hardness because of the blow-up in the size of the instances. To be more precise, the original construction outputs the graph $B_e[G^k]$, for some large enough constant k. The graph $B_e[G^k]$ has approximately N^k vertices, where N is the size of the SAT instance that is the source of the hardness. However, the hardness gap is only $r = \gamma^k$ for some constant $\gamma > 0$. Thus, $|V(G^k)|/r \approx N^k/r \approx N^{k-\epsilon}$, for $\epsilon > 0$, meaning that the running time of $2^{|V(G)|/r}$ is large enough to solve the SAT instance even if the ETH is true. We extend the technique by applying the random sampling technique (i.e., applying the randomized graph product) to reduce the size of the instance and thus obtain the subexponential-time approximation hardness result as stated below.

Theorem 6.1.3. *Let G be any graph. Unless the ETH is false, for any constants $\varepsilon_1, \epsilon_2, \epsilon_3 > 0$, any r-approximation algorithm for the maximum bipartite induced matching problem must run in time at least $2^{|V(G)|^{1-\epsilon_1}/r^{1+\epsilon_2}}$, where r ranges from some sufficiently large constant to $|V(G)|^{1-\epsilon_3}$.*

The organization of this chapter are as follows. We start by giving the overview of our proofs in Section 6.2. Then we prove the subexponential-time approximation

hardness for the bipartite induced matching problem in Section 6.3. The proof for the graph product inequality is deferred to Section 6.4.

6.2 Overview

In this section, we give the overview of the proofs. The presentation consists of two subsections. The first part overviews the hardness proof using the graph product technique that we have developed in [17]. The second part overviews the proof of the subexponential-time approximation hardness of the bipartite induced matching problem.

Part 1: Using Graph Product Inequality

We first sketch the idea of the $n^{1-\epsilon}$-hardness of the bipartite induced matching problem, where n is the number of vertices. The subadditivity property of the inequality in Theorem 6.1.1 plays an important role in proving the hardness of approximation. We build on the following connection between independence and induced matching numbers of a graph, which was implicitly shown by Elbassioni, Raman, Ray and Sitters in [29].

$$\alpha(G) \leq \mathsf{im}(B_e[G]) \leq \mathsf{im}(B[G]) + \alpha(G) \tag{6.3}$$

If $\mathsf{im}(B[G])$ is relatively small, i.e., $\mathsf{im}(B_e[G]) = O(\alpha(G))$, then we will already have the hardness of $n^{1-\epsilon}$ using the hardness of approximating the independence number (e.g., [42]). However, $\mathsf{im}(B_e[G])$ could be as large as $|V(G)|$, and in such case, we do not get any hardness result (not even NP-hardness). To remedy this, we apply Equation (6.1) in Theorem 6.1.1 repeatedly to show that

$$\mathsf{im}(B[G^k]) \leq \mathsf{im}(B[G^{k-1}]) + \mathsf{im}(B[G]) \leq \ldots \leq k \cdot \mathsf{im}(B[G]) \tag{6.4}$$

where $G^k = G \vee G \vee \ldots \vee G$ is a k-wise disjunctive product. Combining Inequality (6.3) and Inequality (6.4) with the lower bound $\alpha(G^k) \leq \mathsf{im}(B_e[G^k])$, we have

$$\alpha(G^k) \leq \mathsf{im}(B_e[G^k]) \leq k \cdot \mathsf{im}(B[G]) + \alpha(G^k)\,.$$

We have from Theorem 4.1.1 that $\alpha(G^k) = (\alpha(G))^k$. So, the term $\alpha(G^k)$ grows exponentially in terms of k while the term $\mathsf{im}(B[G])$ only grows linearly. Therefore, for large enough k, the induced matching number and the independence number coincide. That is, $\mathsf{im}(B_e[G^k]) \approx \alpha(G^k)$.

In more detail, take a hard instance of the maximum independent set problem with a parameter $\epsilon > 0$, where the Yes-Instance has an independent set of size $|V(G)|^{1-\epsilon}$ and the No-Instance has no independent set of size $|V(G)|^{\epsilon}$. Then setting $k \geq 1/\epsilon$ gives the graph $B_e[G^k]$ that has $\alpha(G^k) \leq B_e[G^k] \leq 2\alpha(G^k)$. Now, any hardness of approximating the independence number implies immediately roughly the same hardness of approximating the induced matching number. This implies the $n^{1-\epsilon}$-hardness for the induced matching problem, where $n = |V(B_e[G^k])|$ is the number of vertices of the input graph. Moreover, the input graph is bipartite by the definition of $B_e[G^k]$.

Part II: Subexponential-Time Approximation Hardness

The proof from the first part as used in [17] gives the tight approximation hardness for the bipartite induced matching problem, but the size of the instance blows up by the power of k. To obtain the subexponential time hardness, we start with a hard instance that has a small gap which is obtained from an almost linear time reduction from the decision version of SAT. Then we amplify the gap using the

graph product technique. However, the size of G^k is too large compared to the size of the SAT instance, which means that we cannot apply the ETH assumption. So, we invoke the randomized graph product to get a smaller graph Z_k with $|V(Z_k)| \approx rN$, where r is the hardness gap and N is the size of the SAT instance. The size of instances in each step are as below:

$$\underbrace{N\text{-var SAT}}_{\text{no gap}} \implies \underbrace{\text{Ind.Set } G\text{: } |V(G)| \approx N}_{\text{small gap}} \implies \underbrace{\text{Ind.Matching } Z_k\text{: } |V(Z_k)| \approx rN}_{\text{gap} = r}$$

So, any approximation algorithm that gives an approximation ratio of r must run in time at least $2^{(|V(Z_k)|/r)^{1-\epsilon}} = 2^{N^{1-\epsilon}}$; otherwise, we would break the ETH.

6.3 The Proof of Subexponential-Time Approximation Hardness

In this section, we give the formal proof of the subexponential-time approximation hardness of the maximum bipartite induced matching problem.

The maximum induced matching problem is closely related to the maximum independent set problem as the goal is to find a subset of edges that are, in some sense, "independent".

The hardness of the induced matching problem in general graphs can be derived easily from the maximum independent set problem: Take a hard instance of the maximum independent set problem, namely H. Then construct from H a graph G by adding an edge vv' hanging out from each vertex $v \in V(H)$. That is, for each vertex $v \in V(H)$, we add an auxiliary vertex v' and join v to v' by an edge vv'. So, any independent set S in H corresponds to an induced matching $M = \{vv' : v \in S\}$ in G. Conversely, given an induced matching M in G, one can pick one vertex from each edge $e \in M$, remove auxiliary vertices and map them to a set of vertices S in

H. It is not hard to check that S is independent in H. (See [23] for more detail.) So, $\alpha(H) = \mathsf{im}(G)$, showing that both the maximum induced matching problem is at least as hard to approximate as the maximum independent set problem. However, the graph G is not bipartite. To obtain the hardness result on bipartite graphs, especially when proving hardness in the regime of subexponential-time, the construction needs to be more involved.

We require Theorem 5.2.1 from Chapter 5 in the proof.

Theorem 5.2.1 *For any constant $k > 0$, let $c = 1 - o(1)$, $s = 2^{-\ell^2+1}$, $w = 2^{2\ell}$ and $q = 2^{\ell^2+2\ell}$. Thus, $w \leq (c/s)^{1/\ell}$. Then there is a polynomial-time reduction that, given a SAT instance ϕ of size N with $m = N^{1+o(1)}$, outputs a graph G on $wm = wN^{1+o(1)}$ vertices. The graph G has $\alpha(G) \geq m/2^q$, and moreover,*

- YES-INSTANCE*: If ϕ is satisfiable, then $\alpha(G) \geq cm = cN^{1+o(1)}$.*
- NO-INSTANCE*: If ϕ is not satisfiable, then $\alpha(G) \leq sm = sN^{1+o(1)}$.*

□

Take an instance of the maximum independent set problem as in Theorem 5.2.1. Denote the graph obtained from Theorem 5.2.1 by G. For $k \geq 2$, construct a graph Z_k by the randomized graph product:

$$Z_k = \mathsf{RandProduct}\left(G, k, k \cdot (wm \log wm) \cdot (2^q w)^{k-1}\right)$$

Then we construct the graph $\mathcal{G} = B_e[Z_k]$ as an instance of the maximum bipartite induced matching problem.

Analysis. First, we compute the lower bound on $\alpha(Z_k)$ by applying Lemma 4.2.2. Let us verify that the chosen parameters satisfy the conditions in the lemma. By

Theorem 4.1.1, we have that $\alpha(G^k) = \alpha(G)^k$, For the parameter $t = |V(Z_k)|$, we have by substituting the parameters from Theorem 5.2.1 that

$$\begin{aligned} k \cdot \frac{|V(G)|^k}{\alpha(G)^{k-1}} \cdot \log |V(G)| &\leq k \cdot \frac{(wm)^k}{(2^{-q}m)^{k-1}} \cdot \log wm \\ &= k \cdot (wm) \log wm \cdot (2^q w)^{k-1} \\ &= t = |V(Z_k)|. \end{aligned}$$

So, the graph G_k has independence ratio $\Theta(|\alpha(G)|^k/|V(G)|^k)$, and for $k \geq 2$, we have

$$\begin{aligned} \alpha(Z_k) &= \Theta(\frac{\alpha(G)^k}{|V(G)|^k} \cdot |V(Z_k)|) \\ &\geq \Theta\left(\frac{\alpha(G)^k}{|V(G)|^k} \cdot k \cdot \frac{|V(G)|^k}{\alpha(G)^{k-1}} \cdot \log |V(G)|\right) \\ &= \Theta(k \cdot \alpha(G) \cdot \log |V(G)|) \\ &\geq k \cdot (2^{-q}m) \cdot \log wm \\ &> k \cdot (w \cdot m) \qquad (\log m > w, q \text{ because } w, q \text{ are constants.}) \\ &\geq k \cdot |V(G)|. \end{aligned}$$

Next, we bound the value of $\mathsf{im}(\mathcal{G}) = \mathsf{im}(B[Z_k])$. Consider Inequality (6.1).

$$\mathsf{im}(B[G \vee H]) \leq \mathsf{im}(B[G]) + \mathsf{im}(B[H])$$

It is easy to show by induction that

$$\mathsf{im}(B[G^k]) \leq \mathsf{im}(B[G^{k-1}]) + \mathsf{im}(B[G]) \leq k \cdot \mathsf{im}(B[G]) \leq k|V(G)|$$

The last inequality follows because $B[G]$ has $2|V(G)|$ vertices, and the size of any matching in G can be at most $|V(G)|/2$. Observe that any induced matching M in $B[Z_k]$ is also an induced matching in $B[G^k]$ because Z_k is an induced subgraph of G^k (that is, $B[Z_k][M] = B[G^k][M] = M$). Thus,

$$\mathsf{im}(B[Z_k]) \leq k|V(G)| \text{ while } \alpha(Z_k) > k|V(G)|.$$

By Inequality (6.2) in Lemma 6.1.2, we have

$$\alpha(Z_k) \leq \mathsf{im}(\mathcal{G}) = \mathsf{im}(B_e[Z_k])) \leq \mathsf{im}(B[Z_k]) + \alpha(Z_k) \leq 2\alpha(Z_k)$$

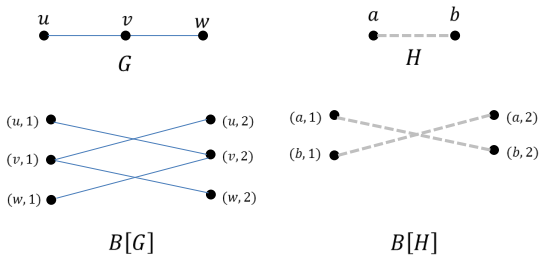
In the construction of Z_k, we use the same parameters as those used for the maximum independent set problem in Chapter 5. So, we can follow the same proof but with $k \geq 2$. Indeed, we may take Z_k as the hard instance in Theorem 5.2.2. Since $|V(\mathcal{G})| = 2|V(Z_k)|$ and $\mathsf{im}(\mathcal{G}) = \Theta(\alpha(Z_k))$, we obtain the subexponential-time approximation hardness for the maximum bipartite induced matching problem, thus proving Theorem 6.1.3.

6.4 Graph Product Inequality

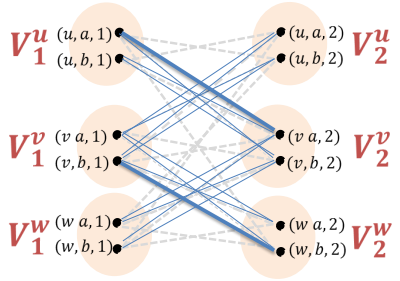
Now, we will prove the graph product inequality in Theorem 6.1.1:

$$\mathsf{im}(B[G \vee H]) \leq \mathsf{im}(B[G]) + \mathsf{im}(B[H])$$

Let V_1 and V_2 be the two partitions of vertices in $B[G \vee H]$ and $\mathcal{M}$ be an induced matching in $B[G \vee H]$. Recall that each edge in $B[G \vee H]$ is of the form $(u, a, 1)(v, b, 2)$, where $u, v \in V(G)$ and $a, b \in V(H)$, and it appears in $B[G \vee H]$



(a) G, H, $B[G]$ and $B[H]$



(b) $B[G \vee H]$, super vertices V_i^x and an induced matching $\mathcal{M}_G$

Figure 6–1: Example of graphs G, H, $B[G]$, $B[H]$ and $B[G \vee H]$, as well as super vertices V_i^x, set of edges E_G and induced matching $\mathcal{M}_G$ (defined in Section 6.4). Bold edges are in $\mathcal{M}_G$. Solid edges (in blue, including bold edges) are edges assigned to E_G, and dashed edges (in gray) are edges assigned to E_H. Observe that if we view V_i^x as a vertex (by unifying vertices in them) and consider only edges in E_G, then the graph looks exactly like $B[G]$. Moreover, the induced matching $\mathcal{M}_G$ becomes an induced matching $\{V_1^u V_2^v, V_1^v V_2^w\}$ in this graph of super vertices. This is the main fact we use to prove Inequality (6.1).

if and only if at least one of the following conditions holds: (1) $uv \in E(G)$ or (2) $ab \in E(H)$. Our strategy is to consider edges satisfying each condition separately.

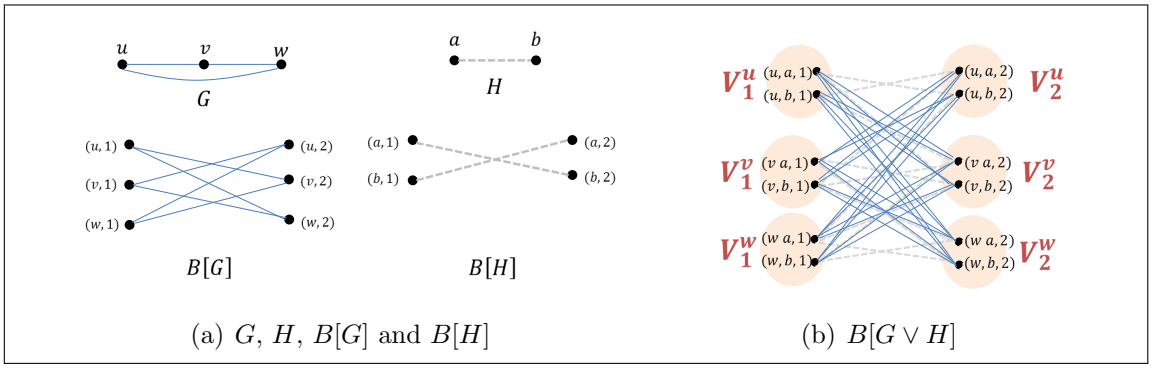
In particular, we let $E(B[G \vee H]) = E_G \cup E_H$, where E_G and E_H consist of edges $(u, a, 1)(v, b, 2)$ that satisfy the first and second condition, respectively. That is, $E_G = \{(u, a, 1)(v, b, 2) : uv \in E(G)\}$ and $E_H = \{(u, a, 1)(v, b, 2) : ab \in E(H)\}$. For example, in Figure 6–1(b), E_G consists of solid edges (in blue) and E_H consists of dashed edges (in gray). Note that some edges, e.g., edge $(u, a, 1)(v, b, 2)$ in Figure 6–1(b), are in both E_G and E_H. We also partition the induced matching $\mathcal{M}$ into $\mathcal{M} = \mathcal{M}_G \cup \mathcal{M}_H$ where $\mathcal{M}_G = \mathcal{M} \cap E_G$ and $\mathcal{M}_H = \mathcal{M} \cap E_H$. Obviously, $|\mathcal{M}| \leq |\mathcal{M}_G| + |\mathcal{M}_H|$. Our goal is to show that $|\mathcal{M}_G| \leq \mathsf{im}(B[G])$ and $|\mathcal{M}_H| \leq \mathsf{im}(B[H])$. We will only show the former claim because the latter can be argued similarly.

To prove this claim, we partition vertices in V_1 and V_2 according to which vertices in G they "inherit" from. That is, for any vertex $u \in V(G)$, we let $V_1^u = \{(u, a, 1) : a \in V(H)\}$ and $V_2^u = \{(u, a, 2) : a \in V(H)\}$ (e.g., see Figure 6–1(b)).

We can think of each set V_i^u as a "super vertex" corresponding to a vertex (u, i) in $B[G]$ in the sense that if we unify all vertices in V_i^u into one vertex, for all $u \in V(G)$ and $i \in V(K_2)$, and remove duplicate edges, then we will get the graph $B[G]$. In fact, we can show more than this. We can show that if we look at $\mathcal{M}_G$ in the graph of super vertices, then we will get an induced matching of $B[G]$ having the same size as $\mathcal{M}_G$! For example, in Figure 6–1(b) the induced matching $\mathcal{M}_G$ in $B[G \vee H]$ consisting of bold edges becomes a set of two edges $\{V_1^u V_2^v, V_1^v V_2^w\}$ in the graph of super vertices, which is still an induced matching.

The key idea in proving this fact is an observation that, for any pair of super vertices V_1^u and V_2^v, either there is no edge between any pair of vertices in V_1^u and V_2^v, or there will be edges between all pairs of vertices in V_1^u and V_2^v. For example, in Figure 6–1(b), there is no edge between any pair of vertices $x \in V_1^u$ and $y \in V_2^w$ while there is an edge between every pair of vertices $x \in V_1^u$ and $y \in V_2^v$. Using this observation, we can easily prove the two lemmas below. The first lemma says that $\mathcal{M}_G$ becomes a matching in the graph of super vertices, and the second one says that this matching is, in fact, an induced matching.

Before proceeding to the proofs, recall that we write the edge set of $B[G \vee H]$ as $E(B[G \vee H]) = E_G \cup E_H$, where $E_G = \{(u, a, 1)(v, b, 2) : uv \in E(G)\}$ and $E_H = \{(u, a, 1)(v, b, 2) : ab \in E(H)\}$.



(a) G, H, $B[G]$ and $B[H]$ (b) $B[G \vee H]$

Figure 6–2: Example of graphs G, H, $B[G]$, $B[H]$ and $B[G \vee H]$, as well as super vertices V_i^x. Solid edges (in blue) are edges assigned to E_G, and dashed edges (in gray) are edges assigned to E_H. (E_G and E_H are defined in Section 6.4.)

Lemma 6.4.1. *For any $u \in V(G)$ and $i \in \{1, 2\}$, V_i^u contains an endpoint of at most one edge in $\mathcal{M}_G$.*

Proof. For the sake of contradiction, assume that there is a vertex $u \in V(G)$ such that V_1^u contains two endpoints of two edges in $\mathcal{M}_G$, say $(u, a, 1)(v, b, 2)$ and $(u, a', 1)(v', b', 2)$. (The case of V_2^u is proved analogously.) Since $(u, a, 1)(v, b, 2)$ is in E_G (recall that $\mathcal{M}_G = \mathcal{M} \cap E_G$), we have that $uv \in E(G)$ and thus $(u, 1)(v, 2)$ is in $E(B[G])$. This fact then implies that there is an edge between $(u, a', 1)$ and $(v, b, 2)$ in E_G as well, contradicting the fact that $\mathcal{M}_G$ (and thus $\mathcal{M}$) is an induced matching. □

Example. Here we illustrate the proof of Lemma 6.4.1. Consider Figure 6–2(b) and let us say that $\mathcal{M}_G$ contains edges $(u, a, 1)(v, b, 2)$ and $(u, b, 1)(v, a, 2)$ which means that V_1^u contains endpoints of two edges in $\mathcal{M}_G$. Having the first edge in E_G

means that $uv \in E(G)$ and thus $(u,1)(v,2)$ is in $E(B[G])$ (as witnessed in Figure 6–2(a)). But then it means that the edge $(u,a,1)(v,a,2)$ must be in E_G as well, making $\mathcal{M}_G$ (and thus $\mathcal{M}$) not an induced matching.

Lemma 6.4.2. *For any $u, u', v, v' \in V(G)$, if $\mathcal{M}_G$ contains an edge between a pair of vertices in V_1^u and V_2^v and an edge between another pair of vertices in $V_1^{u'}$ and $V_2^{v'}$, then there must be no edge between vertices in V_1^u and $V_2^{v'}$ in E_G.*

Proof. Assume for a contradiction that $\mathcal{M}_G$ contains both edges $(u,a,1)(v,b,2)$ and $(u',a',1)(v',b',2)$, and there is an edge, say $(u,c,1)(v',d,2)$ in $E(G)$. Since the edge $(u,c,1)(v',d,2)$ is in E_G, we have $uv' \in E(G)$ and thus $(u,1)(v',2) \in E(B[G])$. This implies that $(u,a,1)(v',b',2)$ is in E_G, which contradicts the fact that $\mathcal{M}$ is an induced matching in $B[G \vee H]$. □

Example. Here we illustrate the proof of Lemma 6.4.2. Consider Figure 6–2(b), and let us say that the matching $\mathcal{M}_G$ contains $(v,a,1)(u,a,2)$ and $(u,a,1)(w,a,2)$ and there is an edge $(v,b,1)(w,b,2)$ which prevents $\mathcal{M}_G$ from being an induced matching in the graph of super vertices. Having the last edge in E_G implies that $(v,1)(w,2) \in E(B[G])$ which in turns implies that $(v,a,1)(w,a,2) \in E_G$, making $\mathcal{M}_G$ (and thus $\mathcal{M}$) not an induced matching in $B[G \vee H]$.

CHAPTER 7
Subexponential-Time Hardness of Problems on Bounded-Degree Graphs

We now discuss the subexponential-time approximation hardness of problems on bounded degree graphs. Specifically, we prove the hardness of independent set, semi-induced matching and bipartite induced matching and on d-degree bounded graphs. (The hardness of the bipartite induced matching will follow directly from the hardness of the semi-induced matching.) The basic method for the proof is similar to that used in Chapter 5. We begin with a hard instance of q-CSP that has a small hardness-gap. Then we boost the gap multiplicatively by a gap-amplification scheme. In Chapter 5 and Chapter 6, we used the randomized graph product as a gap-amplification scheme. In this chapter, however, the gap-amplification scheme we use runs at the CSP level, that is, we amplify the hardness gap before applying the FGLSS reduction.

7.1 Overview

Our proofs involve many parameters. To motivate the importance of each parameter and the need for each tool, we give a top-down overview of our proofs in an informal manner. As in the other chapters, this initial presentation will be imprecise, but we hope it will aid the reader in understanding the technical proofs that follow.

Our reduction consists of two steps. In the first step, we reduce N-variable 3-SAT (decision version) to maximum independent set on a Δ-degree bounded graph.

In the second step, we reduce the maximum independent set to induced matching on a Δ-degree bounded bipartite graph.

$$N\text{-var 3-SAT} \quad \Longrightarrow \quad \underbrace{\Delta\text{-deg. induc. matching with } |V(G)| = \Delta N}_{\text{hardness gap } \Delta} \tag{7.1}$$

$$\underbrace{\Delta\text{-deg. indep. set with } |V(G)| = \Delta N}_{\text{hardness gap } \Delta} \quad \Longrightarrow \quad \underbrace{\Delta\text{-deg. induc. matching with } |V(G)| = \Delta N}_{\text{hardness gap } \Delta} \tag{7.2}$$

The subexponential-time approximation hardness for the maximum independent set and bipartite induced matching problems in graphs with bounded degree follow directly from the above connection. Suppose there is a $2^{o(|V(G)|/\Delta)}$-time $o(\Delta)$-approximation algorithm for either the independent set or bipartite induced matching on Δ-degree bounded graphs. We can then invoke this algorithm to solve 3-SAT in time $2^{o(N)}$ because the reduction in Equation (7.1) gives a hardness gap of Δ with $|V(G)| = N\Delta$. Such an algorithm cannot exist, assuming the ETH.

Step I: SAT $\rightarrow$ Independent Set (Eq. (7.1))

Our reduction from 3-SAT to maximum independent set is simply a sequence of standard reductions; see, for example, [33] and [68]. We start with a 3-SAT instance ψ, which is NP-complete but has no hardness gap. We then construct a PCP for ψ, which outputs a CSP instance ϕ_1 with some small gap. (Note that one may think of a PCP as a reduction from the SAT instance with no gap to a CSP instance with some gap.) Then we apply a *gap-amplification and sparsification scheme* to boost up

the hardness-gap to (some value) k and reduce the number of clauses to roughly the same as the number of variables. This produces another CSP instance ϕ_2. Finally, we apply the *FGLSS reduction* to obtain an instance G of maximum independent set with hardness-gap k. These reductions give a $|V(G)|$-hardness for independent set. To obtain a hardness result for Δ-bounded-degree graphs, we follow the technique of Trevisan [68]. He showed that by modifying the FGLSS reduction with *disperser replacement*, one can obtain a graph with maximum degree $\Delta \approx k$, thus, implying a Δ-hardness. This reduction is summarized below.

$$\underbrace{N\text{-var 3-SAT } \psi}_{\text{(no hardness-gap)}} \xRightarrow{\text{PCP}} \underbrace{N\text{-var CSP } \phi_1}_{\text{(hardness-gap} = \gamma\text{)}} \tag{7.3}$$

$$\xRightarrow{\text{gap amplifi. + sparsi.}} \underbrace{N\text{-var } (kN)\text{-clause CSP } \phi_2}_{\text{(hardness-gap} = k \text{ for all } k \geq \gamma\text{)}} \tag{7.4}$$

$$\xRightarrow{\text{FGLSS + dispers. replac.}} \underbrace{\text{Indep. Set on } G \text{ with } |V(G)| = N\Delta \text{ and } \Delta = k}_{\text{(hardness-gap} = \Delta = k\text{)}} \tag{7.5}$$

Observe that all tools we employ have been used previously. The FGLSS reduction with gap amplification and sparsification technique (with no disperser replacement) was used in [7] to prove the hardness of approximating the maximum independent set problem. The combination of FGLSS and disperser replacement (but with no gap amplification and sparsification) was used in [68]. Our main contribution in Step I is to show how to combine all these tools together and how to tune the parameters appropriately to get the desired subexponential-time hardness of independent set. We emphasize that a key component of our method (that has not

been used before in the context of independent set) is the almost-linear size PCP of Moshkovitz and Raz [59] and the PCP with optimal query complexity of Samorodnitsky and Trevisan [65]. Our reduction in a more detailed form is illustrated below.

$$\underbrace{N\text{-var 3-SAT } \psi}_{\text{(no hardness-gap)}} \xRightarrow{\text{PCP (MR'10 [59] + ST'00 [65])}} \underbrace{N\text{-var CSP } \phi_1\text{, acc. conf. } w_1 = \gamma^{o(1)}}_{\text{(hardness-gap} = \gamma)} \qquad ((7.3)^*)$$

$$\xRightarrow{\text{gap amplification + sparsification}} \underbrace{N\text{-var } (kN)\text{-clauses CSP } \phi_2\text{, acc. conf. } w_2 = k^{o(1)}}_{\text{(hardness-gap} = k = \gamma^t \text{ for any } t)} \qquad ((7.4)^*)$$

$$\xRightarrow{\text{FGLSS + disperser}} \underbrace{\text{Indep. Set on } G\text{: } \Delta = k \text{ and } |V(G)| = (kN)w_2}_{\text{(hardness-gap} = k)} \qquad ((7.5)^*)$$

Here "acc. conf." is an abbreviation for the maximum number of accepting configurations. We note that, amongst the numerous parameters in this reduction, two that play particularly important roles are γ and t. We now explain the above equations. Further details behind Eq. $((7.3)^*)$ and Eq.$((7.4)^*)$ can also be found in Section 7.2 (Eq. $((7.3)^*)$ and Eq. $((7.4)^*)$ together give a almost-linear size sparse PCP with small free-bit complexity and large degree). The details behind Eq. $((7.5)^*)$ can be found in Section (7.4).

Eq. $((7.5)^*)$**.** For the additional details from Eq. (7.5) to Eq. $((7.5)^*)$, we have a parameter w_2 denoting the *maximum number of satisfying assignments for each clause* (known as the *maximum number of accepting configurations*). It is well-known that if we start with an N-variable, M-clause CSP instance ϕ_2 with maximum number of accepting configurations w_2, then we obtain a graph G with $|V(G)| = Mw_2$ with

the properties of Eq. (7.5), after applying the FGLSS reduction and the disperser replacement (for more details, see Eq. 7.3). Since $M = Nk$, it follows that to obtain a reduction as in Eq. (7.5), we need w_2 to be small. For intuition, it is sufficient to imagine that $w_2 = k^{o(1)}$. This makes $|V(G)| \approx (Nk)k^{o(1)} \approx Nk$ (note, again, that we are being imprecise in our calculations). This leads to the refinement of Eq. (7.5) as in Eq. ((7.5)*). Our next job is to obtain ϕ_2 with the properties of Eq. ((7.5)*). That is, for any $k \geq \gamma$, it has N variables, kN clauses, and maximum number of accepting configurations $w_2 = k^{o(1)}$.

Eq. ((7.4)*). We have not yet formalized the precise relationship between γ and k in Eq. (7.4) nor how to obtain $w_2 = k^{o(1)}$. To do this, we introduce yet another parameter t, called the *gap amplification parameter*. By using the standard gap amplification technique with parameter t, we can transform a CSP instance ϕ_1 with hardness gap γ to an instance ϕ_2 with hardness gap $k = \gamma^t$ with the same number of variables. This gap amplification process increases the number of clauses exponentially in t but we can use the standard sparsification technique to bound the number of clauses by $\gamma^t N$. Moreover, if the maximum number of accepting configurations in ϕ_1 is w_1, then we will get $w_2 = w_1^t$ as the maximum number of accepting configurations in ϕ_2. We will show that we can obtain ϕ_1 with $w_1 = \gamma^{o(1)}$, which implies that $w_2 = \gamma^{o(t)} = k^{o(1)}$. This leads to the refinement of Eq. (7.4) as Eq. ((7.4)*).

Eq. ((7.3)*). Our last job is to obtain ϕ_1 with the desired properties. That is, it has N variables and the maximum number of accepting configurations is $w_1 = \gamma^{o(1)}$ for a hardness gap of γ. After this, the combination of the almost-linear size PCP of

Moshkovitz and Raz [59] and the PCP with low free-bit complexity of Samorodnitsky and Trevisan [65], as stated in [59, Corollary 14], gives us exactly what we need. Any 3-SAT formula of size N can be turned into an $N^{1+o(1)}$-variable CSP with hardness gap γ and $w_1 = \gamma^{o(1)}$. This gives a refined version of Eq. (7.3) as Eq. ((7.3)*).

7.1.1 Step II: independent set $\rightarrow$ induced matching

We next reduce independent set to bipartite induced matching. The key idea is to prove a new property of the disperser. The main result is that if we construct a hardness instance of independent set using disperser replacement [68] (as outlined in the previous section) then the hardness of independent set and bipartite induced matching are essentially the same.

Theorem 7.1.1 (Informal)**.** *Let G be a graph (not necessarily bipartite) constructed by disperser replacement. There is a bipartite graph H of roughly the same size as G such that $\mathsf{im}(H) \approx \alpha(G)$.*

The graph H is obtained by an operation, called *extended bipartite double cover*, which we used in Chapter 6. Recall that a bipartite double cover of a graph G, denoted by $B_e[G]$, is a bipartite graph $H = (X, Y, E)$ where X and Y are copies of $V(G)$, and any vertices $x \in X$ and $y \in Y$ are adjacent in H if and only if x and y are adjacent in G. In the extended bipartite double cover, we also have an edge joining two vertices x, y that originate from the same vertex in G. These two operations are natural transformations that are frequently used in transforming a graph G into a bipartite graph. Many of its properties have been studied, mostly in graph theory (see, for example, [14, 66, 35, 47]). In this thesis, we require a new property of this transformation when applied to a disperser.

A New Property of an Old Disperser

A *disperser* is a bipartite graph $G = (X, Y, E)$ with an "expanding property", in that it contains no large *balanced bipartite independent set*. Informally we have (a formal definition can be found in Definition 7.4.3):

Definition 7.1.2 (Disperser (informal))**.** *Let $G = (X, Y, E)$ be a disperser with some parameter $k > 0$. Then G has* no *independent set S such that $|S \cap X| = |S \cap Y| \geq k$.*

In this case, we say the balanced bipartite independent set number of G, denoted by $\mathsf{BBIS}(G)$ is at most k. (For intuition, think of k as $k = \delta \cdot |X|$ for some small value δ.) The disperser plays an important role in proving hardness of approximation; see, for example, the use of dispersers in Trevisan's construction [68] for proving the hardness of approximation of independent set in a bounded-degree graph. Therefore, it is natural to study the properties of dispersers with respect to independent set and related problems (maximum induced matching, in particular).

We show that if G is a disperser, then there is a tight connection between its balanced bipartite independence number $\mathsf{BBIS}(G)$, its induced matching number, and the induced matching number of its extended bipartite double cover $\mathsf{im}(B_e[G])$.

Lemma 7.1.3 (Disperser Lemma (Informal))**.** *If $G = (X, Y, E)$ is a disperser, then*

$$\mathsf{im}(B[G]) = O(\mathsf{im}(G)) = O(\mathsf{BBIS}(G)).$$

(The formal version of this lemma is given in Lemma 7.4.5.) In fact, Lemma 7.1.3 holds for any bipartite graph G, but we only need to apply it to dispersers. This lemma is crucial in our construction. Its proof is simple so we can sketch it here. First, we show that $\mathsf{im}(G) = O(\mathsf{BBIS}(G))$. Let $M = \{x_1y_1, \ldots, x_ty_t\}$ be any induced

matching of size t in G, where $x_i \in X$ and $y_i \in Y$ for all i. Observe that the set $S = \{x_1, \ldots, x_{\lfloor t/2 \rfloor}\} \cup \{y_{\lfloor t/2 \rfloor + 1}, \ldots, y_t\}$ is a balanced bipartite independent set. So, it follows that $\mathsf{BBIS}(G) \geq \lfloor \mathsf{im}(G)/2 \rfloor$ and thus $\mathsf{im}(G) = O(\mathsf{BBIS}(G))$ as desired. Observe that our proof simply exploits the fact that G is bipartite.

Next, we prove that $\mathsf{im}(B[G]) = O(\mathsf{im}(G))$. As the equation looks natural, one might wonder if this holds for *any graph*. Unfortunately, this is not the case as there is a simple non-bipartite counter-example. (Consider a graph H with vertex set $\{x_1, \ldots, x_t\} \cup \{y_1, \ldots, y_t\}$ where edges are of the form $x_i y_i$, $x_i x_j$ and $y_i y_j$, for all i and j. In other words, the graph H consists of two equal-sized cliques that are connected by a perfect matching. It can easily be seen that $\mathsf{im}(H) = 2$ whereas $\mathsf{im}(B[G]) \geq t$.) So, our proof must exploit the fact that G is bipartite. Take an induced matching M in $B[G]$. Let U' and U'' be the two bipartitions of $B[G]$. For any vertex x in the bipartition X of G, denote its copies in U' and U'' as x' and x'', respectively. Similarly, for every vertex $y \in Y$. Observe that edges in $B[G]$ are of the form $x'y''$ or $x''y'$, for some $x \in X$ and $y \in Y$. It follows that M (an induced matching in $B[G]$) must be of the form $\{x'_1 y''_1, \ldots, x'_t y''_t\} \cup \{x''_{t+1} y'_{t+1}, \ldots, x''_{t'} y'_{t'}\}$ where, for all i, $x_i \in X$, $y_i \in Y$. We can use M to construct two induced matchings in G: $M_1 = \{x_1 y_1, \ldots, x_t y_t\}$ and $M_2 = \{x_{t+1} y_{t+1}, \ldots, x_{t'} y_{t'}\}$ (it is not hard to show that M_1 and M_2 are induced matchings). Thus, $\mathsf{im}(G) \geq \mathsf{im}(B[G]|)$, implying that $\mathsf{im}(B[G]) = O(\mathsf{im}(G))$, as desired.

The property of dispersers, stated in Lemma 7.1.3, when plugged into the construction of Trevisan [68], immediately implies Theorem 7.1.1. Combining Theorem 7.1.1 with the hardness of independent set gives the hardness of induced matching in a bipartite graph. Further details can be found in Section 7.4.

7.2 An Almost-Linear Size Reduction from SAT to CSP

This section implements the reductions in Eq. ((7.3)*) and Eq.((7.4)*) as outlined in the overview section.

Informally, given an instance of SAT on N variables, we need to construct a CSP with the following properties:

(1) The *size* of the CSP instance is almost linear on N: the number of variables and clauses are $N^{1+o(1)}$.

(2) The *gap* between completeness and soundness is large.

(3) The *number of accepting configurations* for each clause is small.

(4) The *degree of CSP* – the number of occurrences of each variable – is large.

To obtain an almost linear-size reduction, we start our reduction from the PCP in Theorem 3.2.3. This gives us a reduction from an instance ψ of 3-SAT of size N to an instance φ of q-CSP with the parameters:

Parameters	
Number of variables	$n \leq qN^{1+\epsilon}$
Number of clauses	$m \leq N^{1+\epsilon}$
Number of variables in each clause	$q = \ell^2 + 2\ell$
Completeness (ϕ is satisfiable.)	$1 - o(1)$
Soundness (ϕ is not satisfiable.)	$\leq 2^{-\ell^2+1}$
Number of accepting configurations for each clause	$\leq 2^{2\ell}$

We amplify the ratio between completeness and soundness further by a gap amplification scheme (see, e.g., [69]):

For $t \geq 2$, let $M = 100q2^{t(\ell^2+1)}N^{1+\epsilon+\delta}$. We construct from φ a new CSP instance ϕ on M clauses. For $i = 1, 2, \ldots, M$, we create a clause C_i of ϕ by independently and uniformly at random choosing t clauses from the formula φ and join them by the operation "AND". Thus, we have a CSP instance ϕ on n variables and M clauses. Moreover, each clause of ϕ depends on at most qt variables and has at most w^t accepting configurations.

The following theorem summarizes the properties of our (qt)-CSP ϕ'.

Theorem 7.2.1 (CSP large gap, small accepting configurations and large degree)**.** *Let ℓ, t be parameters and $\epsilon = 1/\ell$. Also, let $\delta > 0$ be any parameter. There is a polynomial-time algorithm that transforms an instance of 3-SAT of size N to a (tq)-CSP instance ϕ, where $q = \ell^2 + 2\ell$, that satisfies the following properties:*

Parameters	
Number of variables	$n \leq N^{1+\epsilon}$
Number of clauses	$100 \cdot q \cdot 2^{t(\ell^2+1)} \cdot N^{1+\epsilon+\delta}$.
Number of variables in each clause	$qt = t(\ell^2 + 2\ell)$
Completeness (ϕ is satisfiable.)	$\geq c = 1/2^t$
Soundness (ϕ is not satisfiable.)	$\leq s = 2^{t(-\ell^2+2)}$
Number of accepting configurations for each clause	$\leq w = 2^{t(2\ell)}$
Number of clauses each variable occurs	$\geq N^{\delta} 2^{t(\ell^2+1)}$.

Moreover, the CSP is balanced in the sense that the number of accepting configurations that involve $x_j = 1$ and those that involve $x_j = 0$ are the same.

Proof. We prove that the CSP constructed from the above gap-amplification scheme satisfies the aforementioned properties.

- Completeness: In a Yes-Instance, there is an (optimal) assignment σ of ϕ that satisfies a $1 - o(1) \geq 1/2$ fraction of the clauses in ϕ. Note that ϕ and ϕ' have the same set of variables. Consider the random process that constructs ϕ'. The probability that each randomly generated clause of ϕ' is satisfied by σ (i.e., the probability that all the t clauses are satisfied by σ) is at least $1/2^t$. So, the expected number of clauses satisfied by σ is $2^{-t}M \geq 100qN^{1+\delta+\epsilon}$ (by our choice of M). By Chernoff's bound, the probability that σ satisfies less than a $1/2^{t+1}$ fraction of clauses in ϕ' is less than $1/N$.

- Soundness: In a No-Instance, any assignment σ satisfies at most a $2^{-\ell^2+1}$ fraction of the clauses of ϕ. Fix an assignment σ. The expected number of

clauses in ϕ' satisfied by σ is at most $2^{-t(\ell^2-1)}M \leq 100qN^{1+\delta+\epsilon}$. By Chernoff's bound, the probability that such assignment satisfies more than $2^{-t(\ell^2-2)}M$ clauses is at most $2^{-10qN^{1+\delta+\epsilon}}$. Since there are at most $2^{qN^{1+\delta+\epsilon}}$ such assignments (because the number of variables is $qN^{1+\delta+\epsilon}$), applying the union bound gives the claimed soundness.

- Degree of CSP: Consider a variable x_j. The probability that each random clause contains a variable x_j is at least $1/m$. So, the expected number of clauses that contain x_j is at least

$$\frac{1}{m} \cdot M \geq \frac{1}{qN^{1+\epsilon}} \cdot 100q2^{t(\ell^2+1)}N^{1+\delta+\epsilon} \geq 100N^{\delta}2^{t(\ell^2+1)}$$

  By applying Chernoff's bound, the probability that we have less than $N^{\delta}2^{t(\ell^2+1)}$ clauses containing x_j is at most $1/N$.

- Balanceness: Consider a variable x_j and a clause C'_i involving x_j. Let $C_{i,1}, \ldots, C_{i,z}$ be clauses of ϕ that included in C'_j. Since each constraint is linear, the number of accepting configurations of each $C_{i,k}$ with $x_j = 1$ and those with $x_j = 0$ are the same. Now, as the C'_i are formed by "AND" of clauses from ϕ, the number of accepting configurations of C'_i with $x_j = 1$ and those with $x_j = 0$ are also the same.

The random process can be derandomized by the result of Zuckerman in [70]. This concludes the proof of Theorem 7.2.1. □

7.3 FGLSS and Dispersers Replacement

The FGLSS reduction has another important property, observed by Trevisan [68]: the FGLSS graph G is formed by a union of N' bipartite cliques where N' is the number of variables of ϕ_2, that is, $G = \bigcup_{i=1}^{N'} G_i$. The dispersers replacement technique exchanges each bipartite clique $G_i = (A_i, B_i, E)$ with a d-regular bipartite graph (with a certain property). This technique involves the following parameters of ϕ_2.

(1) Linearity: Each constraint of ϕ_2 is linear.

(2) Min-Degree δ: The minimum number of clauses each variable participates in.

(3) Clause-Size q: The number of literals in each clause.

Let $H = \bigcup_{i=1}^{N'} H_i$, where $H_i = (A_i, B_i, F_i)$ is a disperser, denote the graph obtained after the disperser replacement. Then the following describes how the are parameters are transformed.

Properties of ϕ_2		**Properties of $G = \bigcup_{i=1}^{N'} G_i$**		**Properties of $H = \bigcup_{i=1}^{N'} H_i$**
Linear Constraints	$\Rightarrow$	$\forall i\, G_i = (A_i, B_i, E_i) : \lvert A_i\rvert = \lvert B_i\rvert$	$\Rightarrow$	$\forall i\, H_i = (A_i, B_i, F_i) : \lvert A_i\rvert = \lvert B_i\rvert$
Min CSP Degree δ	$\Rightarrow$	$\min_{i=1}^{N'} \lvert V(G_i)\rvert \geq \delta$	$\Rightarrow$	$\min_{i=1}^{N'} \lvert V(H_i)\rvert \geq \delta$
Clause-Size q	$\Rightarrow$	$\Delta(G) \leq q \cdot \max_{i=1}^{N'} \Delta(G_i)$	$\Rightarrow$	$\Delta(H) \leq q \cdot d$

We can set d such that $\Delta(H) \approx k$ to obtain Δ-hardness. The linearity of ϕ_2 is required because we need each bipartite clique G_i to be balanced. Also, because the construction of dispersers is randomized, we need $\delta \geq N^{\epsilon}$, for some constant $\epsilon : 0 < \epsilon < 1$, to guarantee that each disperser can be constructed with high probability for all hardness parameters k. This allows us to apply the union bound to show the success probability of the entire construction. (When $k = O(1)$ or $k = \text{poly}(N)$, the

property is not needed.) To guarantee that $\delta \geq N^{\epsilon}$ for all k, we modify the CSP instance ϕ_1 by making N^{ϵ} copies of each clause in Step (4).

7.4 Tight Hardness of Semi-Induced Matching

In this section, we prove the (almost) tight hardness result of the semi-induced matching problem on a Δ-degree bounded bipartite graph. What we prove is actually stronger than the hardnesses of the induced and semi-induced matching problems themselves: We show that the completeness case has a large induced matching while the soundness case has no large semi-induced matching. The formal statement is encapsulated in the following theorem.

Theorem 7.4.1 (Hardness of Δ-Degree Bounded Bipartite Semi-induced Matching)**.** *Let $\epsilon > 0$ be any constant and $t > 0$ be a positive integer. There is a randomized algorithm that transforms a SAT formula ϕ of input size N into a Δ-degree bounded bipartite graph, where $\Delta = 2^{t(\frac{1}{\epsilon^2}+O(\frac{1}{\epsilon}))}$ such that:*

- *(*YES-INSTANCE*:) If ϕ is satisfiable, then $\mathsf{im}(G) \geq |V(G)|/\Delta^{\epsilon}$.*
- *(*NO-INSTANCE*:) If ϕ is not satisfiable, then $\mathsf{sim}(G) \leq |V(G)|/\Delta^{1-\epsilon}$.*

The construction size is $|V(G)| \leq N^{1+\epsilon}\Delta^{1+\epsilon}$, and the running time is $\mathrm{poly}(N, \Delta)$*. Moreover, as long as $t \leq 5\epsilon^2 \log N$, the reduction is guaranteed to be successful with high probability.*

Theorem 7.4.2 (Hardness of d-Degree-Bounded Maximum Independent Set)**.** *Let $\epsilon > 0$ be any sufficiently small constant and $t > 0$ be a positive integer. There is a randomized algorithm that transforms a SAT formula ϕ of input size N into a d-degree-bounded graph G, where $d = 2^{t(\frac{1}{\epsilon^2}+O(\frac{1}{\epsilon}))}$ such that:*

- *(*YES-INSTANCE*:) If ϕ is satisfiable, then $\alpha(G) \geq |V(G)|/d^{\epsilon}$.*

- *(*No-Instance*:) If ϕ is not satisfiable, then $\alpha(G) \leq |V(G)|/d^{1-\epsilon}$.*

The construction size is $|V(G)| = N^{1+\epsilon}d^{1+\epsilon}$, and the running time is $\mathrm{poly}(N, d)$*. Moreover, as long as $t \leq 5\epsilon^2 \log N$, the reduction is guaranteed to be successful with high probability.*

7.4.1 The Reduction

Our reduction is precisely described as follows. Take an instance ϕ of (qt)-CSP as in Theorem 7.2.1 that has N variables and M clauses.

The FGLSS Graph $\widehat{G}$ with Disperser Replacement. First, we construct from ϕ a graph $\widetilde{G}$ by the FGLSS construction. Then the graph $\widetilde{G}$ will be transformed to a graph $\widehat{G}$ by the disperser replacement step. For each clause ϕ_j of ϕ, and for each possible satisfying assignment C of ϕ_j, we create in $\widetilde{G}$ a vertex $v(j, C)$ representing the fact that "ϕ_j is satisfied by assignment C". Then we create an edge $v(j, C)v(j', C') \in E(\widetilde{G})$ if there is a *conflict* between the partial assignments C and C', that is, there is a variable x_i appearing in clauses ϕ_j and $\phi_{j'}$ such that C assigns $x_i = 0$ whereas C' assigns $x_i = 1$. Therefore, the total number of vertices is $|V(\widetilde{G})| = w \cdot M$. The independence number of $\widetilde{G}$ corresponds to the number of clauses of ϕ that can be satisfied. In particular, we can choose at most one vertex from each clause ϕ_j (otherwise, we would have a conflict between $v(j, C)$ and $v(j, C')$), and we can choose two vertices $v(j, C), v(j', C') \in V(\widetilde{G})$ if and only if the assignments C and C' have no conflicts between variables. Thus, the number of satisfiable clauses of ϕ is the same as the independence number $\alpha(\widetilde{G})$. Hence, in a Yes-Instance, we have $\alpha(\widetilde{G}) \geq c \cdot M$, and in a No-Instance, we have $\alpha(\widetilde{G}) \leq s \cdot M$. This gives a hard instance of independent set. Notice that the degree of $\widetilde{G}$ can be very high.

Next, in order to reduce the degree of $\widetilde{G}$, we apply the disperser replacement step as in [68]. Consider an additional property of $\widetilde{G}$. For each variable x_i in ϕ, let O_i and Z_i denote the set of vertices $v(j, C)$ corresponding to the (partial) assignments for which $x_i = 1$ and $x_i = 0$, respectively. It can be deduced from Theorem 7.2.1 that $|O_i| = |Z_i| = M_i/2 \geq 2^{t(\ell^2+1)}N^{\delta}$, for some constant $\delta > 0$.

Since there is a conflict between every vertex of O_i and Z_i, these two sets define a complete bipartite subgraph of $\widetilde{G}$, namely $\widetilde{G}_i = (O_i, Z_i, \widetilde{E}_i)$, where $\widetilde{E}_i = \{uw : u \in O_i, w \in Z_i\}$. Now, if we replace each subgraph $\widetilde{G}_i$ of G by a d-degree bounded bipartite graph, then the degree of a vertex in the resulting graph reduces to qtd. To see this, we may think of each vertex u of $\widetilde{G}$ as a vector with qt coordinates (since it corresponds to an assignment to some clause ϕ_j which has qt related variables). For each coordinate ℓ of u corresponding to a variable x_i, there are d neighbors of u having a conflict at coordinate ℓ (since the conflict forming in each coordinate are edges in $\widetilde{G}_i$, and we replace $\widetilde{G}_i$ by a d-degree bounded bipartite graph). Thus, each vertex u has at most qtd neighbors. However, to preserve the independence number of G, that is, $\alpha(\widehat{G}) \approx \alpha(\widetilde{G})$, we require the d-degree bounded graph to have some additional properties. In particular, we construct the graph $\widehat{G}$ by replacing each subgraph $\widetilde{G}_i$ of $\widetilde{G}$ by a (d, γ)-*disperser* $H_i = (O_i, Z_i, E_i)$, defined below.

Definition 7.4.3 (Disperser)**.** *A (d, γ)-disperser $H = (U', W', E')$ is a d-degree bounded bipartite graph on $n' = |U'| = |W'|$ vertices such that, for all $X \subseteq U', Y \subseteq W'$, if $|X|, |Y| \geq \gamma n'$, then there is an edge $xy \in E'$ joining a pair of vertices $x \in X$ and $y \in Y$.*

Intuitively, the important property of the disperser H_i is that any independent set S in H_i cannot contain a large number of vertices from both O_i and Z_i; otherwise, we would have an edge joining two vertices in S.

The idea of using a disperser to "sparsify" a graph was used by Trevisan [68] to prove the hardness of the bounded degree maximum independent set problem. The key observation that makes this construction work for our problem is that a similar property that holds for the size of a maximum independent set also holds for the size of a maximum σ-semi-induced matching in $B_e[H_i]$. Specifically, $B_e[H_i]$ cannot contain a large σ-semi-induced matching, for any permutation σ.

Now, we proceed to make the intuition above precise. A (d, γ)-disperser can be constructed by a randomized algorithm, as shown in the following lemma. In the special case where d is constant, we may actually construct a (d, γ)-disperser by a deterministic algorithm in [62] with running time exponential in d.

Lemma 7.4.4. *For all $\gamma > 0$ and $n \geq (1/\gamma)\text{polylog}(1/\gamma)$, there is a randomized algorithm that, with success probability $1-e^{-n\gamma(\log(1/\gamma)-2)}$, outputs a d-regular bipartite graph $H = (O, Z, E), |O| = |Z| = n$, where $d = (3/\gamma)\log(1/\gamma)$ such that, for all $X \subseteq Z, Y \subseteq O$, if $|X|, |Y| \geq \gamma n$, there is an edge $(x, y) \in E$ joining some pair of vertices $x \in X$ and $y \in Y$.*

The condition that n_i is sufficiently large is satisfied because $|O_i| = |Z_i| \geq M_i \geq N^\delta 2^{t\ell^2}$ for all i (since each variable x_i appears in M_i clauses, and for each such clause, there is at least one accepting configuration for which $x_i = 0$ and one for which $x_i = 1$.) Also, since the success probability in constructing each disperser is high (at least 2^{N^δ}), we can guarantee that all the dispersers are successfully constructed with

high probability. By selecting an appropriate value for γ (which we will do later) and following the analysis of [68], we get the following completeness and soundness parameters with high probability:

- (YES-INSTANCE:) $\alpha(\widehat{G}) \geq 2^{-t}M$
- (NO-INSTANCE:) $\alpha(\widehat{G}) \leq 2^{-t(\ell^2+2)}M + \gamma qt(wM)$

The Final Graph G. We construct the final graph G by transforming $\widehat{G}$ into a bipartite graph as follows: first create two copies V' and V'' of vertices of $\widehat{G}$. That is, each vertex $u \in V(G)$ has two corresponding copies $u' \in V'$ and $u'' \in V''$. We create an edge joining two vertices $u' \in V'$ and $w'' \in V''$ if and only if there is an edge $uw \in E(\widehat{G})$ or $u = w$. Thus, the formal definition can be written as $G = B_e[\widehat{G}] = (U \cup W, E = E_1 \cup E_2)$ where

$$\begin{aligned} U &= \{(u,1) : u \in V(\widehat{G})\}, \\ W &= \{(w,2) : w \in V(\widehat{G})\} \\ E_1 &= \{(u,1)(u,2) : u \in V(\widehat{G})\}, \\ E_2 &= \{(u,1)(w,2) : u,w \in V(\widehat{G}) \wedge (uw \in E(\widehat{G}))\} \end{aligned}$$

The graph G is a $(2qtd+1)$-degree bounded bipartite graph with $2|V(\widehat{G})|$ vertices. Observe that the edges in G of the form $(u,1)(u,2)$ correspond to a vertex in $\widehat{G}$. Thus, a (semi) induced matching in G whose edges are of this type corresponds to an independent set in $\widehat{G}$. Although this is not the case for every (semi) induced matching $\mathcal{M}$ in G, we will show that we can extract a (semi) induced matching $\mathcal{M}'$ from $\mathcal{M}$ in such a way that $\mathcal{M}'$ maps to an independent set in G, and $|\mathcal{M}'| \geq \Omega(|\mathcal{M}|)$.

7.4.2 Analysis

We now analyze our reduction. Before proceeding, we prove some useful properties of dispersers. The next lemma gives bounds on the size of a σ-semi-induced matching in a disperser.

Lemma 7.4.5 (Disperser Lemma)**.** *Every (d, γ)-disperser $H = (O, Z, E)$ on $2n$ vertices has the following properties.*

- *For any independent set S of H, S cannot contain more than γn vertices from both O and Z, i.e.,*

$$\min(|S \cap O|, |S \cap Z|) \leq \gamma n$$

- *For any permutation (ordering) σ of the vertices of H, the graph $B[H] = (U, W, F)$ obtained by transforming H into a bipartite graph (using only edges of type E_2) contains no σ-semi induced matching of size more than $4\gamma n$, i.e.,*

$$\mathsf{sim}_{B_e[H]}(\sigma) \leq 4\gamma n$$

Proof. The first property follows from the definition of the (d, γ)-disperser H. That is, letting $X = S \cap O$ and $Y = S \cap Z$, if $|X|, |Y| > \gamma n$, then we must have an edge $xy \in E(H)$ joining some vertex $x \in X$ to some vertex $y \in Y$. This contradicts the fact that S is an independent set in H.

Next, we prove the second property. Consider the set of edges $\mathcal{M}$ that form a σ-semi-induced matching in $B[G]$. We claim that $|\mathcal{M}| \leq 4\gamma n$. By way of contradiction, assume that $|\mathcal{M}| > 4\gamma n$. Observe that, for each edge $(u, 1)(v, 2) \in \mathcal{M}$, either (1) $u \in O$ and $v \in Z$ or (2) $v \in O$ and $u \in Z$. Since the two cases are symmetric,

we consider only the first case, denoted by $\widehat{\mathcal{M}}$. Also, we assume wlog that at least half of the edges of $\mathcal{M}$ are in $\widehat{\mathcal{M}}$; thus, $|\widehat{\mathcal{M}}| \geq |\mathcal{M}|/2 > 2\gamma n$.

Let us denote by $V(\widehat{\mathcal{M}})$ the set of vertices that are adjacent to some edges in $\widehat{\mathcal{M}}$. To get a contradiction, we prove the following claim.

Claim 7.4.6. *There are two subsets $X \subseteq U \cap V(\widehat{\mathcal{M}}) : |X| = \gamma n$ and $Y = W \cap V(\widehat{\mathcal{M}}) : |Y| \geq \gamma n$ such that $\sigma(x) < \sigma(y)$, for any $x \in X$ and $y \in V(\widehat{\mathcal{M}}) \setminus X$. Moreover, there is no $\widehat{\mathcal{M}}$-edge between vertices in X and Y.*

We first argue that the second property follows from Claim 7.4.6: If there were two such sets X and Y, then we can define the "projection" of X and Y onto the graph H by $X' = \{u \in V(H) : (u, 1) \in X\}$ and $Y' = \{v \in V(H) : (v, 2) \in Y\}$. It must be the case that $X' \subseteq O$ and $Y' \subseteq Z$ (due to the definition of $\widehat{\mathcal{M}}$), so by the property of the disperser, there is an edge in $E(H)$ joining some $x \in X'$ and $y \in Y'$. This implies that there must be an edge $(x, 1)(y, 2) \in E(B[H])$ where $x \in X$ and $y \in Y$. Also, there are edges $(x, 1)(x', 2) \in \widehat{\mathcal{M}}$ and $(y', 1)(y, 2) \in \widehat{\mathcal{M}}$. This contradicts the fact that $\mathcal{M}$ is a σ-semi-induced matching. Thus, it only remains to prove the claim.

Proof of Claim 7.4.6. Recall that the ordering σ is defined on the vertices of $B[H]$, not on the vertices of H. We construct X and Y as follows. Order the vertices in $U \cap V(\widehat{\mathcal{M}})$ according to the ordering σ and define X to be the first γn vertices according to this ordering. We obtain $X \subseteq U \cap V(\widehat{\mathcal{M}})$ with the property that for any $x \in X$ and $y \in V(\widehat{\mathcal{M}}) \setminus X$, $\sigma(x) < \sigma(y)$.

Now, we define $Y \subseteq W \cap V(\widehat{\mathcal{M}})$ as the set of vertices that are not matched by $\widehat{\mathcal{M}}$ with any vertices in X. Since $|X| = \gamma n$, the number of vertices in $W \cap V(\widehat{\mathcal{M}})$ that are matched by $\widehat{\mathcal{M}}$ is only γn, so we can choose arbitrary γn vertices that are not matched as our set Y. □

Therefore, the lemma follows. □

As a corollary of Lemma 7.4.5, we relate the independence number of the FGLSS graph $\widetilde{G}$ to the final graph G. Recall that $\widetilde{G}$ is the graph of the independent set instance, $\widehat{G}$ is the graph after the disperser replacement, and G is the final construction obtained after applying the extended bipartite double cover. Recall, also, that γ is the parameter of the disperser construction.

Corollary 7.4.7. *Let $\widetilde{G}$ and G be the graphs constructed as above. Then, for any permutation (ordering) σ of vertices of G,*

$$\alpha(\widehat{G}) \leq \mathsf{sim}_G(\sigma) \leq \alpha(\widehat{G}) + 4\gamma|V(\widehat{G})|$$

Proof. Recall that the edges of G are $E = E_1 \cup E_2$. To prove the inequality on the left-hand-side, consider the set of edges E_1. Observe that edges of $E_1 = \{(v,1)(v,2) : u \in V(\widehat{G})\}$ correspond to vertices of $\widehat{G}$, as G and $\widehat{G}$ share the same vertex set. Let S be an independent set in $\widehat{G}$. We claim that the set $E_S = \{(u,1)(u,2) : u \in S\}$ must be an induced matching in G. This immediately implies the first inequality. To see the claim, assume there is an edge $(u,1)(v,2) \in E(G)$ for some $(u,1)(u,2), (v,1)(v,2) \in E_S$. So we have that $u, v \in S$ and that $uv \in E(\widehat{G}) \subseteq E(\widehat{G})$. This contradicts the fact that S is an independent set.

Next, we prove the inequality on the right-hand-side. Let $\mathcal{M}$ be a σ-semi-induced matching in G. We decompose $\mathcal{M}$ into $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2$. Now $|\mathcal{M}_1| \leq \alpha(\widehat{G})$. Too see this, from the set $\mathcal{M}_1$, we can define a set $S \subseteq V(\widetilde{G})$ by $S = \{u \in V(\widetilde{G}) : (u,1)(u,2) \in \mathcal{M}_1\}$. S must be an independent set in $\widehat{G}$; otherwise, if there is an edge $uv \in E(\widehat{G})$ for $u, v \in S$, then there are edges $(u,1)(v,2), (v,1)(u,2) \in E(G)$, contradicting the fact that $\mathcal{M}_1$ is a σ-semi-induced matching.

It is sufficient to show that $|\mathcal{M}_2| \leq 4\gamma|V(\widehat{G})|$. We do so by partitioning $\mathcal{M}_2$ into $\mathcal{M}_2 = \bigcup_{j=1}^{N} \mathcal{M}_2^j$ where $\mathcal{M}_2^j = \{(u,1)(v,2) \in \mathcal{M}_2, uv \in E(H_j)\}$ (since $\widehat{G}$ is the union of edges of the subgraphs H_j). Each set $\mathcal{M}_2^j$ must be a σ_j-induced matching for the ordering σ_j obtained by projecting σ onto the vertices of $B[H_j]$. Thus, we can invoke Lemma 7.4.5 to bound the size of $\mathcal{M}_2^j$, that is, $|\mathcal{M}_2^j| \leq 4\gamma n_j$. Summing over all j, we have

$$|\mathcal{M}_2| \leq \sum_{j=1}^{N} |\mathcal{M}_2^j| \leq \sum_{j=1}^{N} 4\gamma n_j \leq 4\gamma qt|V(\widehat{G})|$$

The last inequality follows by a basic counting argument. Each vertex belongs to exactly qt subgraphs H_j, so if we sum n_j over all $j = 1, 2, \ldots, N$, we get $\sum_{j=1}^{N} n_j = qt|V(\widehat{G})|$. □

Completeness and Soundness. The completeness and soundness proofs are now straightforward. In a YES-INSTANCE, $\alpha(\widehat{G}) \geq c \cdot M$ implies that $\mathsf{sim}_G(\sigma) \geq c \cdot M$, and in a NO-INSTANCE, we have $\alpha(\widehat{G}) \leq s \cdot M + \gamma qtwM$. This implies that $\mathsf{sim}_G(\sigma) \leq s \cdot M + 5\gamma qtwM$.

Now, we choose $\gamma = s/(5qtw)$, and this gives

$$d = O\left(\frac{1}{\gamma}\log\frac{1}{\gamma}\right) = O\left(\left(\frac{wqt}{s}\right)\log\left(\frac{wqt}{s}\right)\right).$$

Therefore, the final graph G has the following properties:

Number of Vertices	**Degree**	**Hardness Gap**
$n = 2wM$	$\Delta = (2dq+1)$	$g = \dfrac{c \cdot M}{s \cdot M + 5\gamma \cdot wM} \geq \dfrac{c}{2s}$

Substituting c, s, w, q, M as in Theorem 7.2.1, we get

- The degree $\Delta = O(t^2\ell^4 2^{t(\ell^2+2\ell-1)}) = 2^{t(\ell^2+\Theta(\ell))} = 2^{t(1/\epsilon^2+\Theta(1/\epsilon))}$
- The number of vertices $|V(G)| = 2^{t(\ell^2)}N^{1+O(\epsilon)} = \Delta^{1+O(\epsilon)}N^{1+O(\epsilon)}$
- The hardness gap $g \geq 2^{t(\ell^2-1)} \geq \Delta^{1-O(\epsilon)}$

Remark. Notice that we cannot set $1/\gamma$ to be larger than the size of a complete bipartite subgraph (that we replace by a (d,γ)-disperser). The chosen parameters are feasible only because the degree of CSP, which implies the size of the complete bipartite subgraph, is large enough. That is, each complete bipartite subgraph $\widetilde{G}_i$ has size

$$N_i \geq N^{\delta}2^{t(\ell^2+1)} \geq \frac{1}{\gamma} \geq 2^{t(\ell^2-2\ell+2)}$$

Success probability of the disperser construction. Notice that the failure probability of the disperser construction given in Lemma 7.4.4 is large when $N_i\gamma$ is small. In our case, we have $N_i \geq 2^{t\ell^2}N^{\delta}$ and $\gamma \geq 2^{-t(\ell^2+O(\ell))}$. Thus, we are guaranteed that $N_i\gamma \geq N^{\delta}2^{-O(t\ell)} = 2^{\delta\log N - O(t\ell)}$. As long as $t \leq O(\delta\epsilon\cdot\log N)$, we are guaranteed that $N_i\gamma \geq N^{\delta/2}$, so the failure probability in Lemma 7.4.4 is at most $2^{-N^{\delta/2}}$. This allows us to apply the union bound over all variables x_j in the CSP and conclude

that the construction is successful with high probability. If we appropriately pick $\delta = \Theta(\epsilon)$ and $t \leq 5\epsilon^2 \log N$, then we obtain Theorem 7.4.1.

7.4.3 Subexponential Time Approximation Hardness for the Maximum Independent Set and Induced Matching Problems

We now present hardness results that show a trade-off between running-time and approximation-ratio. Roughly speaking, we obtain the following results under the Exponential Time Hypothesis: any algorithm that guarantees an approximation ratio of r for the maximum independent set problem and the maximum bipartite induced matching problem on bipartite graphs, for any $r \geq r_0$ for some constant r_0, must run in time at least $2^{n^{1-\epsilon}/r^{1+\epsilon}}$. This almost matches the upper bound of $2^{n/r}$ given by Cygan et al. [26] for independent set and by our simple algorithm, given in Section 7.4.4, for bipartite induced matching. These results are obtained as by-products of the proof in Section 7.4.

Theorem 7.4.2 implies almost immediately the following corollary.

Theorem 7.4.8. *Consider the maximum independent set problem on an input graph $G = (V, E)$. For any $\epsilon > 0$ and sufficiently large $r \leq |V(G)|^{1/2-\epsilon}$, every algorithm that guarantees an approximation ratio of r must run in time at least $2^{|V(G)|^{1-2\epsilon}/r^{1+4\epsilon}}$ unless the ETH is false.*

Proof. The intuition is very simple. Theorem 7.4.2 can be seen as a reduction from a SAT instance of size N to the independent set problem whose instance size is, roughly, $|V(G)| = Nr$ where r is the approximability gap for the independent set problem (ignoring the small exponent ϵ in the theorem). (In fact, the graph resulting from the reduction is an r-degree-bounded graph as we will set $r \approx d$.) It is immediate

that getting a running time of $2^{o(|V(G)|/r)}$ for the maximum independent set problem is equivalent to getting a running time of $2^{o(N)}$ for SAT (since $N \approx |V(G)|/r$), contradicting the ETH. Below we give a formal proof.

Assume for a contradiction that there is an algorithm $\mathcal{A}$ that obtains an r-approximation in time $2^{|V(G)|^{1-2\epsilon}/r^{1+4\epsilon}}$, for some $r \leq |V(G)|^{1/2-\epsilon}$ and a small constant $\epsilon > 0$. Then we can use the algorithm $\mathcal{A}$ to decide the satisfiability of a given SAT formula ϕ as follows. First, we invoke the reduction in Theorem 7.4.2 on the SAT formula ϕ to construct a graph $G = (V, E)$ with parameters t and ϵ, such that $d = 2^{t(1/\epsilon^2+\Theta(1/\epsilon))} \leq r^{1+3\epsilon}$. Notice, the value of t is at most $t \leq 2\epsilon^2 \log r \leq 5\epsilon^2 \log N$, so the reduction is guaranteed to be successful with high probability.

Since $d = r^{1+3\epsilon}$, we have $r < d^{1-\epsilon}$, which means we can use the algorithm $\mathcal{A}$ to distinguish between a Yes-Instance and a No-Instance in time $2^{|V(G)|^{1-2\epsilon}/r^{1+4\epsilon}} < 2^{N^{1-\epsilon}}$. Plugging in the values $|V(G)| \leq N^{1+\epsilon}d^{1+\epsilon}$ and $r^{1+4\epsilon} \geq d$ gives a violation of the ETH. □

Note that, since t in Theorem 7.4.2 cannot be chosen beyond $5\epsilon^2 \log N$, we have no flexibility of making r arbitrary close to $|V(G)|^{1-\epsilon}$. However, this can be easily fixed by slightly modifying the proof of Theorem 7.4.2 and leaving some flexibility in the choice of parameter δ, as discussed in Section 7.2. Since this is not necessary for the hardness of the k-hypergraph pricing problem and will make the proof in Section 7.2 more complicated, the details are omitted.

The subexponential time hardness of approximating the maximum induced matching problem can be proved analogously, but we need Theorem 7.4.1 instead of Theorem 7.4.2.

Theorem 7.4.9. *Consider the maximum induced matching problem on a bipartite graph $G = (U, V, E)$. For any $\epsilon > 0$ and sufficiently large $r \leq |V(G)|^{1/2-\epsilon}$, every algorithm that guarantees an approximation ratio of r must run in time at least $2^{|V(G)|^{1-2\epsilon}/r^{1+4\epsilon}}$ unless the ETH is false.*

The proof of Theorem 7.4.9 is similar to that of Theorem 7.4.8, so we omit the details.

7.4.4 Subexponential-Time Approximation Algorithm for Induced Matching

In this section, we present r-approximation algorithms which run in time $2^{n/r} \cdot \text{poly}(n)$ in bipartite graphs and time $2^{(n/r)\log\Delta} \cdot \text{poly}(n)$ in non-bipartite graphs, where Δ is the maximum degree. These running times are almost tight, except that the latter one incurs an extra $O(\log\Delta)$ factor in the exponent. We leave the question of whether this extra term is necessary as an open problem.

Bipartite Graphs. For the case of bipartite graphs, we prove the following theorem.

Theorem 7.4.10 (Algorithm on Bipartite Graphs)**.** *For any $r \geq 1$, there is an r-approximation algorithm for the maximum bipartite induced matching problem that runs in time $2^{n/r}\text{poly}(n)$ where n is the size of the input graph.*

To prove Theorem 7.4.10, we will need the following lemma, which says that we can compute the maximum induced matching in a bipartite graph in time $2^{n'}$ where n' is the cardinality of the smaller partition in the graph.

Lemma 7.4.11. *For any bipartite graph $G = (U, W, E)$, there is an algorithm that returns a maximum induced matching in G and runs in time $2^{\min(|U|,|W|)} \cdot \text{poly}|V(G)|$.*

Proof. We assume without loss of generality that $|U| \leq |W|$. We first need to characterize the existence of an induced matching in terms of *good neighbors*, defined as follows. Given a subset $U' \subseteq U$ and a fixed $u \in U'$, we say that $w \in W$ is a *U'-good neighbor of u* if there is no other $u' \in U'$ (where $u' \neq u$) such that $u'w \in E$. Observe that $U' \subseteq U$ forms end-vertices of some induced matching $\mathcal{M}'$ if and only if every vertex in U' has a U'-good neighbor in W. To see this, if $U' \subseteq U$ is a set of end-vertices of a matching $\mathcal{M}'$, then it is clear that for each $uw \in \mathcal{M}$, the vertex w is a U'-good neighbor. For the converse, for any $u \in U'$, let $w_u \in W$ be a U'-good neighbor of u. Then $\{uw_u : u \in U'\}$ must form an induced matching.

Applying this observation, we compute a maximum induced matching in G as follows. For each possible subset $U' \subseteq U$, we check whether the vertices in U' can be end-vertices of any induced matching. This can be done in time $\text{poly}(|V(G)|)$ (simply by checking the existence of U'-good neighbors). Finally we return the maximum-cardinality subset U' and its corresponding induced matching $\mathcal{M}'$. □

From this lemma, given an input graph $G = (U, W, E)$, we partition the vertices of U into r sets $U_1, \ldots, U_r$ in a balanced manner and define $G_i = G[U_i \cup W]$. That is, G_i is an induced subgraph on vertices $U_i \cup W$. Our algorithm simply invokes the lemma on each graph G_i to obtain an induced matching $\mathcal{M}_i$, and finally we return the $\mathcal{M}_{i^*}$ with maximum cardinality among $\mathcal{M}_1, \ldots, \mathcal{M}_r$. Since $|U_i| \leq \lceil n/r \rceil$, the running time of our algorithm is at most $2^{\lceil n/r \rceil}\text{poly}n$. The following lemma implies that $\mathcal{M}_{i^*}$ is an r-approximation and is feasible in G, thus completing the proof.

Lemma 7.4.12. *The following holds on G and its subgraphs G_i.*

- *Any induced matching $\mathcal{M}_i$ in G_i is also an induced matching in G.*

- $\sum_{i=1}^{r} \mathsf{im}(G_i) \geq \mathsf{im}(G)$

Proof. Let's prove the first fact. If $\mathcal{M}_i$ is not an induced matching in G, then there must be two edges $uv, ab \in \mathcal{M}_i$ that are joined by some edge e in G. But, since $u, v, a, b \in U_i \cup W$, the edge e must also be present in G_i, contradicting the fact that $\mathcal{M}_i$ is an induced matching in G_i.

Next, we prove the second fact. Let $\mathcal{M}$ be a maximum induced matching in G. For $i = 1, 2, \ldots, r$, define $\mathcal{M}_i = \mathcal{M} \cap E(G_i)$. It is clear that each $\mathcal{M}_i$ is an induced matching in G_i. Thus, it follows immediately that $\sum_{i=1}^{r} \mathsf{im}(G_i) \geq \sum_{i=1}^{r} |\mathcal{M}_i| = |\mathcal{M}|$. □

Non-Bipartite Graphs. We note that almost the same running time can be obtained for the case of non-bipartite graphs, except that we have an extra $\log \Delta$ factor in the exponent, where Δ is the maximum degree.

Theorem 7.4.13 (Algorithm on Non-Bipartite Graphs)**.** *For any $r \geq 1$, there is an r-approximation algorithm for the maximum induced matching problem that runs in time $2^{(n/r)\log\Delta}\mathrm{poly}(n)$, where n is the size of the input graph and Δ is the maximum degree.*

To prove Theorem 7.4.13, we give the following algorithm. Our algorithm takes as input a graph $G = (V, E)$ on n vertices and a parameter r. We first partition V arbitrarily into $V = \bigcup_{i=1}^{r} V_i$ such that the sizes of the V_is are roughly equal, namely, $|V_i| = \lfloor n/r \rfloor$ or $|V_i| = \lfloor n/r \rfloor + 1$. For each $i = 1, \ldots, r$, we find a maximum-cardinality subset of edges M_i such that M_i is an induced matching in G and every edge in M_i has at least one end-vertex in V_i. We implement this step by checking every possible subset of edges: We choose one edge incident to each vertex in V_i or choose none and

then check whether the set of chosen edges F is an induced matching in G. We select the set F that passes the test with maximum cardinality as the set M_i. Finally, we choose as output the set M_i that has maximum-cardinality over all $i = 1, 2, \ldots, r$.

It can be seen that the running time of our algorithm is $O(\Delta^{n/r} \cdot \text{poly}(n)) = O(2^{(n/r)\log\Delta}\text{poly}(n))$, where Δ is the maximum degree of G. For the approximation guarantee, it suffices to show that

$$\mathsf{im}(G) \leq \sum_{i=1}^{r} |M_i| \leq r \cdot \mathsf{im}(G).$$

We shall complete the proof of Theorem 7.4.13 by proving the above inequalities as in the following decomposition lemma.

Lemma 7.4.14. *Consider any graph $G = (V, E)$. Let $V_1 \cup V_2 \cup \ldots \cup V_r$ be any partition of V. For $i = 1, 2, \ldots, r$, let M_i be a set of edges with maximum-cardinality such that M_i is an induced matching in G and every edge in M_i has at least one end-vertex in V_i. Then $\mathsf{im}(G) \leq \sum_{i=1}^{r} |M_i| \leq r \cdot \mathsf{im}(G)$.*

Proof. Let $\mathcal{M}$ be any maximum induced matching in G. Then, clearly, $|M_i| \leq |\mathcal{M}| = \mathsf{im}(G)$ for all $i = 1, 2, \ldots, r$ because M_i is an induced matching in G. Thus, $\sum_{i=1}^{r} |M_i| \leq r \cdot \mathsf{im}(G)$, proving the second inequality.

For $i = 1, 2, \ldots, r$, define $\mathcal{M}_i$ to be a subset of $\mathcal{M}$ such that each edge in $\mathcal{M}_i$ has at least one end-vertex in V_i, and $\mathsf{im}(G) = \sum_{i=1}^{r} |\mathcal{M}_i|$. By the maximality of M_i, we have $|\mathcal{M}_i| \leq |M_i|$, for all $i = 1, 2, \ldots, r$. Thus, $\mathsf{im}(G) \leq \sum_{i=1}^{r} |M_i|$. □

CHAPTER 8
The Hardness of Approximating k-Hypergraph Pricing

In this chapter, we prove approximation hardness of the k-hypergraph pricing problem. Assuming the ETH, we show that within polynomial-time there is no approximation algorithm that yields an approximation ratio better than $\tilde{o}(\sqrt{n})$, where n is the number of the vertices of the hypergraph. In contrast, there exists a quasi-polynomial-time approximation algorithm has an approximation guarantee of $\tilde{O}(n^{\delta})$, for any chosen constant $\delta > 0$. We also show an almost matching lower bound for the running time of n^{δ}-approximation algorithms, for every parameter $\delta > 1/2$. Consequently, our results show that, assuming the ETH, there is a line that separate between the computation power of the polynomial and quasi-polynomial-time algorithms, which were believed to have equivalent computation power.

We will use the following equivalent formulation of the k-hypergraph pricing problem: The pricing instance is given by two sets $(\mathcal{C}, \mathcal{I})$ where $\mathcal{C}$ and $\mathcal{I}$ are the sets of consumers and items, respectively. Each consumer $c \in \mathcal{C}$ is associated with a budget B_c and an item set $S_c \subseteq \mathcal{I}$. We have an additional constraint that $|S_c| = k$. It is easy to see that this formulation is equivalent to the hypergraph formulation, i.e., each vertex corresponds to an item and each edge corresponds to a consumer, and the additional constraint $|S_c| = k$ ensures that the size of each hyperedge is k. The reason we use this formulation is because we will be dealing with other graph problems, which might cause confusion to readers.

8.1 Overview

We begin with an informal overview of our method. Our presentation will show the translation of each parameter from SAT to the k-hypergraph pricing problem. As an intermediate problem, we require the hard instance of the bipartite induced matching problem on bounded degree graphs

8.1.1 The Main Reduction: SAT $\rightarrow$ pricing

At a very high level, our proof makes the following connection between 3-SAT and the h-hypergraph pricing problem. We give a reduction that transforms an N-variable SAT formula into a h-hypergraph pricing instance $(\mathcal{C}, \mathcal{I})$ with the following parameters: the number of items is $|\mathcal{I}| = Nh$, the number of consumers is $|\mathcal{C}| = 2^h \text{poly}(N)$, and the hardness gap is h. The following specifies the relationships between the parameters.

$$\underbrace{N\text{-var 3-SAT}}_{\text{no gap}} \quad \Longrightarrow \quad \underbrace{h\text{-hypergraph pricing with } |\mathcal{I}| = Nh,\ |\mathcal{C}| = 2^h \text{poly} N}_{\text{hardness gap } h} \tag{8.1}$$

The running time of our reduction is small, i.e., poly($|\mathcal{C}|$), and thus can be ignored for now. The hardness gap of h means that any algorithm that could solve the pricing problem with an approximation factor of $o(h)$ would be able to solve 3-SAT (optimally) within the same running time, thus breaking the ETH. Consequently, we obtain a hardness of h for any $h = o(N)$, assuming the ETH. This hardness can be written as $\min(h, \sqrt{\mathcal{I}})$ since $|\mathcal{I}| = Nh$. The reduction goes via an intermediate problem: the maximum induced matching problem on a bipartite graph where the

input graph has maximum degree Δ. The hardness of this problem was shown in Chapter 7.

8.1.2 An Intermediate Reduction: Induced matching $\rightarrow$ Pricing

We reduce bipartite induced matching to hypergraph pricing. Given a bipartite graph $G = (U, W, E)$, we create an instance of hypergraph pricing by viewing the left vertices U as consumers (with different budgets) and the right vertices W as items. The basic idea is that the price of each item will determine which consumer it should be matched to (to produce a solution to the bipartite induced matching problem). That is, if an item I has price p, then it should be matched to the consumer with budget roughly p. To make this idea work, the budgets of the consumers must differ *geometrically*, as must the number of consumers of each type. More precisely, we will convert each vertex $u \in U$ into $2^i(u)$ consumers in such a way that any two vertices u and u' sharing the same neighbor v must receive different exponents. Intuitively, we will require a blow-up of $2^{O(\Delta)}$ because each vertex $v \in V$ has degree Δ. This is how an exponential blow-up arises in the reduction.

8.2 The Approximation Hardness of k-Hypergraph Pricing

We now prove the hardness of the k-hypergraph pricing problem. Throughout this section, we use n and m to denote the number of items and consumers, respectively. Note the difference between n, the number of items in the pricing instance, and N, the size of the 3SAT formula.

Theorem 8.2.1. *Unless* NP = ZPP*, for any $\epsilon > 0$, there is a universal constant k_0 (depending on ϵ) such that the k-hypergraph pricing problem for any constant $k > k_0$*

is $k^{1-\epsilon}$ hard to approximate. Assuming the ETH, for any $\epsilon > 0$, the k-hypergraph pricing problem is hard to approximate to within a factor of $\min(k^{1-\epsilon}, n^{1/2-\epsilon})$.

Proof Overview and Organization. For any k-hypergraph pricing instance $(\mathcal{C}, \mathcal{I})$, we denote by $\text{opt}(\mathcal{C}, \mathcal{I})$ the optimal possible revenue that can be collected by any price function. The key to proving Theorem 8.2.1 is the connection between the hardness of semi-induced matching and k-hypergraph pricing. This is formalized in the following lemma, whose proof will be given in Section 8.2.1.

Lemma 8.2.2 (From Semi-induced Matching to Pricing)**.** *There is a randomized reduction that, given a bipartite graph $G = (U, V, E)$ with maximum degree d, outputs an instance $(\mathcal{C}, \mathcal{I})$ of the k-hypergraph pricing problem such that, with high probability,*

$$(6 \ln d / \ln \ln d)\mathsf{sim}(G) \geq \text{opt}(\mathcal{C}, \mathcal{I}) \geq \mathsf{im}(G)$$

The number of consumers is $|\mathcal{C}| = |U|d^{O(d)}$ and the number of items is $|\mathcal{I}| = |V|$. Moreover, each consumer $c \in \mathcal{C}$ satisfies $|S_c| = d$. The running time of this reduction is $\text{poly}(|\mathcal{C}|, |\mathcal{I}|)$.

We remark that using the upper bound for $\text{opt}(\mathcal{C}, \mathcal{I})$ in terms of $\mathsf{sim}(G)$ instead of $\mathsf{im}(G)$ appears to be necessary. Specifically, obtaining a similar reduction with a bound of $\text{opt}(\mathcal{C}, \mathcal{I}) = \tilde{O}(\mathsf{im}(G))$ may not be possible.

Combining the above reduction in Lemma 8.2.2 with the hardness of induced and semi-induced matching in Theorem 7.4.1 leads to the following intermediate hardness result which, in turn, leads to all the hardness results stated in Theorem 8.2.1.

Lemma 8.2.3 (Intermediate Hardness)**.** *Let $\epsilon > 0$ be any constant. There is a universal constant $d_0 = d_0(\epsilon)$ such that the following holds. For any function $d(\cdot)$*

such that $d_0 \leq d(N) \leq N^{1-\epsilon}$, there is a randomized algorithm that transforms an N-variable 3SAT formula ϕ to a k-hypergraph pricing instance $(\mathcal{C}, \mathcal{I})$ such that:

- *For each consumer c, $|S_c| = d(N)$.*
- *The algorithm runs in time* $\text{poly}(|\mathcal{C}|, |\mathcal{I}|)$.
- $|\mathcal{C}| \leq d^{O(d)} N^{1+\epsilon}$ *and* $|\mathcal{I}| \leq N^{1+\epsilon} d^{1+\epsilon}$.
- *There is a value Z such that (*YES-INSTANCE*) if ϕ is satisfiable, then* $\text{opt}(\mathcal{C}, \mathcal{I}) \geq Z$*; and (*NO-INSTANCE*) if ϕ is not satisfiable, then* $\text{opt}(\mathcal{C}, \mathcal{I}) \leq Z/d^{1-\epsilon}$.

8.2.1 From Semi-Induced Matching to Pricing Problems

Here we prove Lemma 8.2.2 by showing a reduction from semi-induced matching on d-degree bounded bipartite graphs to the *k-hypergraph pricing*. The reduction is randomized and is guaranteed to be successful with constant probability.

The Reduction

Let $G = (U, V, E)$ be a bipartite graph with maximum degree d. Assume without loss of generality that $|U| \leq |V|$. (This assumption will be important in our analysis). Observe we always have $\mathsf{sim}(G) \geq \mathsf{im}(G) \geq |U|/d$. For each vertex u of G, we use $N_G(u)$ to denote the set of neighbors of u in G. If the choice of a graph G is clear from the context, then we will omit the subscript G. Our reduction consists of two phases.

Phase 1: Coloring. We color each vertex $u \in U$ of G by uniformly and independently choosing a random color from $\{1, 2, \ldots, d\}$. We denote by $U_i \subseteq U$, for each $i = 1, 2, \ldots, d$, the set of left vertices that are assigned color i. We say that a right vertex $v \in V$ is *highly congested* if there is some $i \in [d]$ such that $|N_G(v) \cap U_i| \geq 3 \ln d / \ln \ln d$. That is, v has at least $3 \ln d / \ln \ln d$ neighbors of the

same color. Let $V_{high} \subseteq V$ be a subset of all vertices that are highly congested and set $V' = V \setminus V_{high}$. Therefore V' is the set of vertices in V with highly congested vertices removed. Now let G' be the subgraph of G induced by (U, V', E). We will require the following fact for the analysis in Section 8.2.1.

Lemma 8.2.4. *With probability at least* $1/2$*,*

$$\mathsf{im}(G') \geq (1 - 2/d)\mathsf{im}(G) \quad \textit{and} \quad \mathsf{sim}(G') \geq (1 - 2/d)\mathsf{sim}(G).$$

In particular, for $d \geq 4$*,* $\mathsf{im}(G') \geq \mathsf{im}(G)/2$ *and* $\mathsf{sim}(G') \geq \mathsf{sim}(G)/2$ *with probability at least 1/2.*

Proof. First, consider any vertex $v \in V$. We claim that vertex v is highly congested with probability at most $1/d$. To see this, we can view the coloring of neighbors of v as a *balls and bins* problem: let each $u \in N_G(v)$ be a ball b_u and let each color c be a bin B_c; coloring a vertex $u \in N_G(v)$ with color c corresponds to putting a ball v to a bin c. It is well-known (e.g., see [58, Section 5.2] and [38]) that, with probability at least $1 - 1/d$, all bins contain at most $3 \ln d / \ln \ln d$ balls. That is, $|N_G(v) \cap U_i| \leq 3 \ln d / \ln \ln d$. Thus, v is highly congested with probability at most $1/d$, as claimed.

Now consider a maximum induced matching $M = (A, B, F)$ of G, where $A \subseteq U$, $B \subseteq V$ and $F \subseteq E$. So, $|A| = |B| = |F| = \mathsf{im}(G)$. Let $M' = (A', B', F')$ be the subgraph of M obtained by removing the vertices of V_{high}. Therefore, $A' = A$ and $B' = B \setminus V_{high}$. Note that the edges in M' give an induced matching of G' of size $|B'|$. It follows that

$$\mathsf{im}(G') \geq |B'|.$$

Because every vertex $v \in V$ is in V_{high} with probability at most $1/d$, we have $\mathbb{E}[|B \cap V_{high}|] \leq |B|/d$. Applying Markov's inequality then gives

$$\mathbb{P}[|B \cap V_{high}| \geq 2|B|/d] \leq 1/2.$$

Consequently,

$$\mathbb{P}[|B'| \leq (1 - 2/d)|B|] \leq 1/2.$$

So $\mathsf{im}(G') \geq |B'| \geq (1-2/d)|B| = (1-2/d)\mathsf{im}(G)$ with probability at least $1/2$. This gives the first inequality in the statement. Proving the second inequality uses exactly the same argument, except we take M to be a maximum semi-induced matching, and note that edges in M' then give a semi-induced matching of G' of size $|B'|$. □

Phase 2: Finishing. We now have a coloring of the left vertices U of G with the desired properties. We then construct an instance of the k-hypergraph pricing problem in both the Udp-Min and Smp models as follows. For each vertex $v \in V'$, we have an item $I(v)$. For each vertex $u \in U_i$, we create d^{3i} consumers; we denote this set of consumers by $\mathcal{C}(u)$. We define the budget of each consumer $c \in \mathcal{C}(u)$, where $u \in U_i$, to be $B_c = d^{-3i}$. We define $S_c = \{I(v) : v \in N_G(u)\}$; it is immediate that $|S_c| \leq d$. To recap, we have

- The set of items $\mathcal{I} = \{I(v) : v \in V'\}$.
- The set of consumers $\mathcal{C} = \bigcup_{u \in U} \mathcal{C}(u)$, where $|\mathcal{C}(u)| = d^{3i}$ for $u \in U_i$.
- A budget $B_u = \dfrac{1}{d^{3i}}$ for each consumer $u \in U_i$.
- A set of desired items $S_c = \{I(v) : v \in N_G(u)\}$ for each customer $c \in \mathcal{C}$. (Note that $|S_c| \leq d$.)

This completes the description of our reduction. Notice that, in the k-hypergraph formulation, we have $\mathcal{I}$ as a set of vertices, $\mathcal{C}$ as a set of hyperedges, and $k = d$ (since $|S_c| \leq d$ for all $c \in \mathcal{C}$).

The Analysis

Completeness. We will show that the profit we can collect is at least $\mathsf{im}(G') \geq \mathsf{im}(G)/2$ (by Lemma 8.2.4). Let $\mathcal{M}$ be any induced matching in the graph G'. For each item $I(v)$ with $uv \in \mathcal{M}$ and $u \in U_i$, we set its price to be $p(I(v)) = 1/d^{3i}$. For any other item, we set its price to be ∞ for UDP-MIN and to be 0 for SMP. Observe that, for each $u \in U_i$ that belongs to $\mathcal{M}$, any consumer $c \in \mathcal{C}(u)$ sees only one item of finite price (that is, $1/d^{3i}$). So, for UDP-MIN the consumer c must buy the item $I(v)$ and thus contributes $1/d^{3i}$ to the total profit. Similarly, for SMP, the consumer c can afford to buy the whole set S_c of total cost $1/d^{3i}$. Since $|\mathcal{C}(u)| = d^{3i}$, the total profit contributed by each set of consumers $\mathcal{C}(u)$ is 1. This implies that the total profit we obtain from this price function is $|\mathcal{M}|$.

Soundness. Now, suppose that an *optimal* price function p yields a profit of r (for either UDP-MIN or SMP). We will show that $\mathsf{sim}(G) \geq r \log\log d/(12 \log d)$. The proof has two parts. First, we identify a collection of "tight consumers" who roughly correspond to those consumers that pay a sufficiently large fraction of their budgets. Second, we construct a large semi-induced matching using these tight consumers. We say that a consumer $c \in \mathcal{C}$ is *tight* if she spends at least $1/4d$ fraction of her budget for her desired item. A vertex $u \in U$ is tight if its set of consumers $\mathcal{C}(u)$ contains a tight consumer. Let $\mathcal{C}'$ be the set of tight consumers.

Claim 8.2.5. *The profit made by tight consumers only is at least $r/2$.*

Proof. Observe that *profitable non-tight* consumers contribute at most $|U|/4d$ to the total profit. Since we proved that $r \geq \mathsf{im}(G') \geq \mathsf{im}(G)/2 \geq |U|/2d$ (by the assumption that $|U| \leq |V|$ and Lemma 8.2.4), the revenue made from non-tight consumers is at most $r/2$. □

Now, we construct from the set of tight consumers $\mathcal{C}'$, a σ-semi-induced matching in G for some total order σ. We define σ so that vertices in U are ordered by their colors (increasingly for the case of UDP-MIN and decreasingly for the case of SMP).

Definition 8.2.6. *Let σ be a total order of vertices such that vertices in U_i always precede vertices in U_j if $i < j$ for* UDP-MIN *(and $i > j$ for* SMP*). (Recall that U_i is the set of vertices in U of color i.)*

Let $U' = \{u \in U : \mathcal{C}(u) \cap \mathcal{C}' \neq \emptyset\}$ be the set of left vertices whose $\mathcal{C}(u)$ contains a tight consumer. Note that $|U'| \geq r/2$ by Theorem 8.2.5. For UDP-MIN, we define a set of edges $\mathcal{M}$ to be such that an edge uv is in $\mathcal{M}$ if $u \in U'$ and a tight consumer in $\mathcal{C}(u)$ buys an item $I(v)$. For SMP, $\mathcal{M}$ is defined to contain uv such that $u \in U'$ and $I(v)$ is the most expensive item for a consumer in $\mathcal{C}(u)$. Note that

$$|\mathcal{M}| \geq |U'| \geq r/2\,.$$

This collection $\mathcal{M}$ may not be a σ-semi-induced matching and may not even be a matching. So, we have to remove some edges from $\mathcal{M}$ to ensure the resulting set is a σ-semi-induced matching. To be precise, we extract from $\mathcal{M}$ a set of edges $\mathcal{M}' \subseteq \mathcal{M}$ that is a σ-semi-induced matching with cardinality $|\mathcal{M}'| \geq r \log\log d/6 \log d$. This implies that $\mathsf{sim}(G) \geq r \log\log d/6 \log d$.

Our intention is to construct $\mathcal{M}'$ one edge at a time. (Clearly, we will not add an edge from $\mathcal{M}$ to $\mathcal{M}'$ if it would not produce a σ-semi-induced matching). The order we add edges from $\mathcal{M}$ depends *reversely* on σ (and, clearly, we will not add an edge to $\mathcal{M}'$ if it would not produce a σ-semi-induced matching). The process will also be applied separately for each color of the left vertices. Specifically, we partition $\mathcal{M}$ into $\mathcal{M}_1 \cup \mathcal{M}_2 \cup \ldots \cup \mathcal{M}_d$, where

$$\mathcal{M}_i = \{uv \in \mathcal{M} : u \in U_i\}.$$

contains those edges uv whose endpoint u is colored i. Then we construct from each set $\mathcal{M}_i$ a set of edges $\mathcal{M}'_i$ as follows. We process each edge $uv \in \mathcal{M}_i$ in the *reverse* order of σ. That is, an edge uv is processed before an edge $u'v'$ if $\sigma(u) > \sigma(u')$. For each edge $uv \in \mathcal{M}_i$, we remove from $\mathcal{M}_i$ all edges $u'v'$ such that u' is adjacent to v. Then we add uv to the set $\mathcal{M}'_i$ and proceed to the next edge in $\mathcal{M}_i$. Notice that, each time we add an edge uv to $\mathcal{M}'_i$, we remove at most $3\log d/\log\log d$ edges from $\mathcal{M}_i$. This is because such an endpoint is not highly congested by the construction of $\mathcal{M}_i$. Thus, $|\mathcal{M}'_i| \geq |\mathcal{M}_i| \log\log d/3\log d$. Moreover, it can be seen by the construction that $\mathcal{M}'_i$ is a σ-semi-induced matching. Finally, define $\mathcal{M}' = \bigcup_{i=1}^{d} \mathcal{M}'_i$. Then we have that

$$|\mathcal{M}'| \geq \frac{|\mathcal{M}| \log\log d}{3\log d}.$$

We claim that $\mathcal{M}'$ is a σ-semi-induced matching. Suppose not. Then there is a pair of edges $uv, u'v' \in \mathcal{M}'$ such that $\sigma(u) < \sigma(u')$ and $uv' \in E(G)$. We need two cases to distinguish between the two models of SMP and UDP-MIN.

- For UDP-MIN, by construction, the two vertices u and u' must belong to different color classes U_i and U_j, respectively, where $i < j$. Since $uv' \in E(G)$, consumers in $\mathcal{C}(u)$ are interested in item $I(v')$, whose prices are $1/d^{3j}$ (which is strictly less than $1/2d^{i+1}$) because u' is a tight index. But, then u would have never been tight, a contradiction.
- For SMP, the two vertices u and u' belong to U_i and U_j respectively where $i > j$. Since $uv' \in E(G)$, consumers in $\mathcal{C}(u)$ are interested in $I(v')$, whose prices are $1/2d^{3j+1} > 1/d^{3i}$. Then consumers in $\mathcal{C}(u)$ would not have sufficient budget to buy their item sets, contradicting the fact that they are tight.

Thus, we have

$$\mathsf{sim}(G') \geq |\mathcal{M}'| \geq \frac{|\mathcal{M}| \log\log d}{3 \log d} \geq \frac{r \log\log d}{6 \log d}$$

as desired.

8.2.2 Intermediate Hardness

We prove Theorem 8.2.3 using Theorem 7.4.1 and Theorem 8.2.2.

To see how to prove Theorem 8.2.3 by combining Theorem 7.4.1 with Theorem 8.2.2, we start with an N-bit 3SAT formula ϕ and invoke Theorem 7.4.1 to obtain a d-degree bounded bipartite graph $G = (U, V, E)$. We then apply a reduction as in Theorem 8.2.2 to obtain an instance $(\mathcal{C}, \mathcal{I})$ of UDP-MIN or SMP with $|\mathcal{C}| = |V(G)|d^{O(d)} = N^{1+O(\epsilon)}d^{O(d)}$ and $|\mathcal{I}| = N^{1+O(\epsilon)}d^{1+O(\epsilon)}$. It is immediate that the gap between a YES-INSTANCE and a NO-INSTANCE is $d^{1-2\epsilon}$ for all values of d.

Notice that our reduction gives (nearly) tight hardness results for all values of d. The complexity assumptions that we make, however, differ for different values of d.

(Note that $d = 2^{t(1/\epsilon^2+O(1/\epsilon))}$ is a function of the parameters t and ϵ.) For example, if d is constant, then our complexity assumption is NP $\neq$ ZPP, and if $d = \text{polylog}N$, then our assumption is NP $\not\subseteq$ ZPTIME($2^{\text{polylog}N}$). To see this, consider the size (which also implies the running time) of our reduction. Our instance for Udp-Min (resp., Smp) has the number of consumers $m = |\mathcal{C}| = N^{1+O(\epsilon)}d^{O(d)} \leq N^{1+O(\epsilon)}2^{d^{1+\epsilon}}$ and the number of items $n = |\mathcal{I}| = N^{1+O(\epsilon)}d^{1+O(\epsilon)}$. Suppose now we have an algorithm with a running time of poly(n, m). Then we also have a randomized algorithm with a running time of poly$(N^{1+O(\epsilon)}, 2^{d^{1+O(\epsilon)}})$ that solves SAT exactly.

8.2.3 The Hardness Results (Proof of Theorem 8.2.1)

From the above discussion, we can see the complexity assumption we require is NP $\not\subseteq$ ZPTIME(poly$(N, 2^{O(d)})$). Thus, if t is constant, then d is also a constant. That is, $d = 2^{O(1/\epsilon^2)}$, and the corresponding complexity assumption is, indeed, NP $\neq$ ZPP. In this case, we get the hardness of the k-hypergraph pricing problem when k is constant (note that $k = d$). More precisely, we have proved that, for any $\epsilon > 0$, there is a constant k_0 that depends on ϵ such that k-hypergraph pricing is $k^{1-\epsilon}$-hard for any $k \geq k_0$.

We note that our reduction also implies a hardness of $\Omega(\log^{1-\epsilon} m)$, as proved by Chalermsook et al. [15]. In this case, if the value of k is chosen to be polylog$N$, then the complexity assumption becomes NP $\not\subseteq$ ZPTIME($2^{\text{polylog}N}$). In particular, we can plug in $k = \log^{1/\epsilon} N$ to give $m = N^{\log^{1/\epsilon} N}$ and a hardness factor of $k^{1-\epsilon} = \log^{1/\epsilon-1} N = \log^{1-O(\epsilon)} m$.

Now, let's incorporate the ETH into our hardness result. Thus, we assume there is no exponential-time (randomized) algorithm that solves 3SAT. We choose

$t = (\epsilon^2 - O(\epsilon)) \log N$. So we have $k = 2^{t(1/\epsilon^2 - O(1/\epsilon)} = 2^{(1-O(\epsilon)) \log N} = N^{1-O(\epsilon)}$. Moreover, $|\mathcal{I}| = N^{2+O(\epsilon)}$, and the size of the resulting pricing instance (as well as the running time) is dominated by $k^{O(k)} \leq 2^{N^{1-\epsilon}}$. This is fine (still subexponential time) because we assume the ETH. Writing k in terms of the number of items, we have that $k = N^{1-O(\epsilon)} = n^{1/2-O(\epsilon)}$. Consequently, our $k^{1-\epsilon}$-hardness result rules out a polynomial-time algorithm with an $n^{1/2-\epsilon}$-approximation guarantee for Udp-Min (resp., Smp), assuming the ETH.

8.2.4 Subexponential-Time Approximation Hardness for the k-Hypergraph Pricing Problem

We now present an approximability/running time trade-off for the pricing problems. We note that this hardness result is in a slightly different form than those for maximum independent set and maximum induced matching. Our inapproximability result shows that any n^δ-approximation algorithm for the k-hypergraph pricing problem (both Udp-Min and Smp) must run in time at least $2^{(\log m)^{\frac{1-\delta-\epsilon}{\delta}}}$ for any constant $\delta, \epsilon > 0$. This almost matches the running time of $O\left(2^{(\log m)^{\frac{1-\delta}{\delta}} \log\log m} \text{poly}(n, m)\right)$ presented in Section 8.3.

Theorem 8.2.7. *Consider the k-hypergraph pricing problem (with either* Smp *or* Udp-Min *buying rule). For any $\delta > 0$ and a sufficiently small $\epsilon < \epsilon_0(\delta)$, every n^δ approximation algorithm for* Udp-Min *(resp.,* Smp*) must run in time at least $2^{(\log m)^{\frac{1-\delta-\epsilon}{\delta}}}$ unless the ETH is false.*

For example, plug in $\delta = 1/3$. Then, an $n^{1/3}$ approximation for the pricing problem requires running time at least $m^{\log^{1-\epsilon} m}$, assuming the ETH.

Proof. Fix $\delta < 1/2$. Let ϵ be as in Theorem 8.2.3. Assume (for contradiction) that we have an n^δ approximation algorithm $\mathcal{A}$ for Udp-Min (resp., Smp) that runs in time $2^{(\log m)^{\frac{1-\delta-100\epsilon}{\delta}}}$. We apply the reduction in Lemma 8.2.3 with $d = n^{\delta(1+10\epsilon)}$. Therefore, the algorithm $\mathcal{A}$ can distinguish between a Yes-Instance and a No-Instance, thus deciding the satisfiability of SAT. It remains to analyze the running time of the algorithm and show that the algorithm runs in time $O(2^{N^{1-\epsilon}})$, which will contradict the ETH.

Now $n \leq d^{1+\epsilon}N^{1+\epsilon}$. So, plugging in $d = n^{\delta(1+10\epsilon)}$, we obtain $n \leq n^{\delta(1+20\epsilon)}N^{1+\epsilon}$. This implies that $n \leq N^{1+\delta+40\epsilon}$. But, if we also plug the value of d into $m \leq 2^{d^{1+\epsilon}}N^{1+\epsilon}$, we get

$$m \leq 2^{n^{\delta(1+20\epsilon)}} \leq 2^{N^{\delta(1+\delta+40\epsilon)}}$$

Hence, we have $\log m \leq N^{\delta(1+\delta+40\epsilon)}$, implying the running time of $\log^{\frac{1-\delta-100\epsilon}{\delta}} m \leq N^{1-10\epsilon}$. This is subexponential in the size of SAT instance, contradicting the ETH.

□

8.3 Approximation Scheme for k-Hypergraph Pricing

In this section, we present an approximation scheme for the k-hypergraph pricing problem, which works for both Udp-Min and Smp buying rules. Throughout, we denote by n and m the number of items and the number of consumers, respectively. For any parameter δ, our algorithm gives an approximation ratio of $O(n^\delta)$ and runs in time $O(2^{(\log m)^{\frac{1-\delta}{\delta}}} \log\log m \text{poly}(n, m))$.

In the underlying mechanism, we employ as subroutines an $O(\log m)$-approximation algorithm for Udp-Min (resp., Smp) and an $O((\log m)^n)$-time constant-approximation algorithm for Udp-Min (resp., Smp) as stated in the following two lemmas.

Lemma 8.3.1 ([39]). *There is an $O(\log m)$-approximation algorithm for* UDP-MIN *(resp.,* SMP*), where $m = |\mathcal{C}|$ is the number of consumers.*

Lemma 8.3.2. *There is a constant-factor approximation algorithm for* UDP-MIN *(resp.,* SMP*) that runs in time $O((\log nm)^n \text{poly}(n, m))$.*

For the sake of presentation flow, we defer the proof of Lemma 8.3.2 to Section 8.3.5. Now, we present our approximation scheme and its analysis.

8.3.1 Approximation Scheme

We exploit a trade-off between the approximation ratio and the running time. In particular, the $O(\log m)$-approximation algorithm from Lemma 8.3.1, denoted by $\mathcal{A}_1$, always runs in polynomial time but yields a bad approximation ratio in terms of n when $n^\delta \ll \log m$. In contrast, the $O((\log nm)^n \text{poly}(n, m))$-time $O(1)$-approximation algorithm from Lemma 8.3.2, denoted by $\mathcal{A}_2$, has a slow running time but always gives a good approximation ratio. So, we take advantage of the trade-off between the running time and approximation ratio by selecting one of these two algorithms according to the values of n^δ and $\log m$. To be precise, our approximation scheme takes as input a set of consumers $\mathcal{C}$, a set of items $\mathcal{I}$ and a parameter $\delta : 0 < \delta < 1$. If the number of items is large, i.e., $n^\delta > \log m$, then we apply the $O(\log m)$-approximation algorithm $\mathcal{A}_1$. Otherwise, we partition the set of m items into n^δ (almost) equal subsets, namely, $\mathcal{I}_1, \ldots, \mathcal{I}_{n^\delta}$, and we apply the algorithm $\mathcal{A}_2$ to each subinstance $(\mathcal{C}, \mathcal{I}_i)$, for $i = 1, \ldots, n^\delta$. We then sell to consumers the set $\mathcal{I}_{i^*}$ that yields a maximum revenue over all $i = 1, \ldots, n^\delta$. Here the key idea is that one of the sets $\mathcal{I}_i$ gives a revenue of at least opt/n^δ in the optimal pricing, where opt is the optimal revenue. So, by choosing the set that maximizes a revenue, we would get

a revenue of at least $O(\mathrm{opt}/n^{\delta})$ (because $\mathcal{A}_2$ is an $O(1)$-approximation algorithm). Since we have two different buying rules, UDP-MIN and SMP, there is some detail that we need to adjust. When we assign the prices to all the items, we need to ensure that the consumers will (and can afford to) buy the set of items we choose. So, we price items in the set $\mathcal{I}_{i^*}$ by a price function returned from the algorithm $\mathcal{A}_2$, and we apply two different rules for filling the prices of items in $\mathcal{I} \setminus \mathcal{I}_{i^*}$ for the cases of UDP-MIN and SMP. In UDP-MIN, we price items in $\mathcal{I} \setminus \mathcal{I}_{i^*}$ by ∞ to guarantee that no consumers will buy items outside $\mathcal{I}_{i^*}$. In *Smp*, we price items in $\mathcal{I} \setminus \mathcal{I}_{i^*}$ by 0 to guarantee that consumers can afford to buy the whole set of items that they desire (although we get no profit from items outside $\mathcal{I}_{i^*}$). The running time of the algorithm $\mathcal{A}_2$ in general is large, but since $n^{\delta} < \log m$ and each subinstance contains at most $n^{1-\delta}$ items, we are able to guarantee the desired running time. Our approximation scheme is summarized in Algorithm 8.3.1.

8.3.2 Cost Analysis

First, we analyze the approximation guarantee of our algorithm. If $n^{\delta} > \log m$, then our algorithm immediately gives $O(n^{\delta})$-approximation. So, we assume that $n^{\delta} \geq O(\log m)$. We will use the following two lemmas.

Lemma 8.3.3. *For any instance $(\mathcal{C}, \mathcal{I})$ of* UDP-MIN *(resp.* SMP*), let $\mathcal{I}'$ be any subset of $\mathcal{I}$. Let p' be a price function that collects a revenue of r from $(\mathcal{C}, \mathcal{I}')$. Then, the price function $p : \mathcal{I} \to \mathbb{R}$ obtained by setting $p(i) = \infty$ (resp. $p(i) = 0$) for $i \in \mathcal{I} \setminus \mathcal{I}'$ and $p(i) = p'(i)$ for $i \in \mathcal{I}'$ gives revenue at least r for the instance $(\mathcal{C}, \mathcal{I})$.*

Proof. We first prove the lemma for UDP-MIN. Consider the price function p' that collects a revenue of r. Notice that, under the price p, each customer $c \in \mathcal{C}$ who has

Algorithm 2 Pricing($\mathcal{C}$,$\mathcal{I}$,δ)

if $n^\delta > \log m$ **then**
    Apply an $O(\log m)$ approximation algorithm for UDP-MIN (resp., SMP) from Lemma 8.3.1.
    return The price function p obtained by the $O(\log m)$-approximation algorithm.
else
    Partition $\mathcal{I}$ into n^δ equal sets, namely $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_{n^\delta}$. So, each $\mathcal{I}_j$ has size $|\mathcal{I}_j| \leq n^{1-\delta}$.
    for $j = 1$ to n^δ **do**
        Apply Lemma 8.3.2 on the instance $\Pi_j = (\mathcal{C}, \mathcal{I}_j)$, i.e., restricting the set of items to $\mathcal{I}_j$.
    end for
    Choose an instance Π_{j^*} that maximizes the revenue over all $j = 1, 2, \ldots, n^\delta$.
    Let p be the price function obtained by solving an instance Π_{j^*}.
    For UDP-MIN (resp. SMP), set the prices of all the items in $\mathcal{I} \setminus \mathcal{I}_{j^*}$ to ∞ (resp. 0).
    return the price function p.
end if

positive payment in p' also pays for the same amount in p (since items in $S_c \cap (\mathcal{I} \setminus \mathcal{I}')$ have infinite prices).

For SMP, for each customer $c \in \mathcal{C}$ who pays positive price in p', we have by construction that $\sum_{i \in S_c} p(i) = \sum_{i \in S_c \cap \mathcal{I}'} p'(i)$. So, the customer c can still afford the set and pays the same amount as in the subinstance $(\mathcal{C}, \mathcal{I}')$. □

The above lemma allows us to focus on analyzing the revenue obtained from the subinstance $(\mathcal{C}, \mathcal{I}_{j^*})$. Since we apply a constant-factor approximation algorithm to the instance $(\mathcal{C}, \mathcal{I}_{j^*})$, it suffices to show that $\text{opt}(\mathcal{C}, \mathcal{I}_{j^*}) \geq \text{opt}(\mathcal{C}, \mathcal{I})/n^{\delta}$, which follows from the following lemma.

Lemma 8.3.4. *Let q be any positive integer. For any set of consumers $\mathcal{C}$ and any partition of $\mathcal{I}$ into $\mathcal{I} = \bigcup_{j=1}^{q} \mathcal{I}_j$, the following holds for* UDP-MIN *(resp.,* SMP*)*

$$\text{opt}(\mathcal{C}, \mathcal{I}) \leq \sum_{j=1}^{q} \text{opt}(\mathcal{C}, \mathcal{I}_j)$$

Proof. Consider an optimal price function p^* for $(\mathcal{C}, \mathcal{I})$. Fix some optimal assignment of items to customers with respect to p^*. Now for each $j = 1, \ldots, q$, let r_j be the revenue obtained by function p^* from items in $\mathcal{I}_j$, so we can write $\sum_{j=1}^{q} r_j = \text{opt}(\mathcal{C}, \mathcal{I})$. Notice that, in each sub-instance $(\mathcal{C}, \mathcal{I}_j)$, we can restrict the price function p^* onto the set $\mathcal{I}_j$ and obtain the same revenue. This means that $\text{opt}(\mathcal{C}, \mathcal{I}_j) \geq r_j$, implying that

$$\sum_{j=1}^{q} \text{opt}(\mathcal{C}, \mathcal{I}_j) \geq \sum_{j=1}^{q} r_j = \text{opt}(\mathcal{C}, \mathcal{I})$$

as desired. □

8.3.3 Running Time Analysis

If $n^\delta > \log m$, then our algorithm runs in polynomial-time (since we apply a polynomial-time $O(\log m)$-approximation algorithm). So, we assume that $n^\delta \leq \log m$. In this case, we run an algorithm from Lemma 8.3.2 on n^δ sub-instances having $n^{1-\delta}$ items each. It follows that the running time of this algorithm is

$$O\left(n^\delta (\log nm)^{n^{1-\delta}} \text{poly}(n^{1-\delta}, m)\right) = O\left(2^{(\log m)^{\frac{1-\delta}{\delta}} \log\log nm} \text{poly}(n, m)\right).$$

The equality follows since $\log m \geq n^\delta$ implies that $n^{1-\delta} \leq (\log m)^{(1-\delta)/\delta}$. Thus, for any constant $\delta > 0$, our algorithm runs in quasi-polynomial time.

8.3.4 Polynomial-Time $O(\sqrt{n \log n})$-Approximation Algorithm

Now, we will set δ so that our approximation scheme runs in polynomial-time. To be precise, we set δ so that $n^\delta = \sqrt{n \log n}$. It follows that our algorithm yields an approximation guarantee of $O(\sqrt{n \log n})$. The running time of our algorithm is (note that $n^{1-\delta} = \sqrt{\frac{n}{\log n}}$)

$$\begin{aligned} O\left(n^\delta \cdot (\log nm)^{n^{1-\delta}} \text{poly}(n, m)\right) &= O\left(2^{\frac{\sqrt{n}}{\sqrt{\log n}} \log\log nm} \text{poly}(n, m)\right) \\ &= O\left(2^{\frac{\sqrt{n \log n}}{\log n} \log\log nm} \text{poly}(n, m)\right) \end{aligned}$$

If $\sqrt{n \log n} \leq \log nm / \log\log nm$, then we are done because the running time of the algorithm will be $O(2^{\log nm} \text{poly}(n, m)) = \text{poly}(n, m)$. Thus, we assume that $\sqrt{n \log n} > \log nm / \log\log nm$. So, we have

$$\log n > \log\left(\frac{\log nm}{\log\log nm}\right) = \log\log nm - \log\log\log nm \geq \frac{1}{2} \log\log nm$$

This means that $\log n / \log\log nm \geq 1/2$. Thus, the running time of our algorithm is

$$\begin{aligned} O\left(2^{\frac{\sqrt{n\log n}}{\log n}\log\log nm}\text{poly}(n,m)\right) &\leq O\left(2^{2\sqrt{n\log n}}\text{poly}(n,m)\right) \\ &\leq O(2^{\log m}\text{poly}(n,m)) \\ &= \text{poly}(n,m) \end{aligned}$$

The last inequality follows since $n^\delta = \sqrt{n\log n} \leq \log m$. Thus, in polynomial-time, our approximation scheme yields an approximation ratio of $O(\sqrt{n\log n})$ for both Udp-Min and Smp.

8.3.5 $O(1)$-Approximation in Time $O((\log nm)^n \text{poly}(n,m))$

In this section, we present an $O(1)$-approximation algorithm for Udp-Min and Smp that runs in $O((\log nm)^n\text{poly}(n,m))$ time. That is, we prove Lemma 8.3.2). Our algorithm reads as input an instance $(\mathcal{C},\mathcal{I})$ of Smp (resp., Udp-Min) and a parameter $\alpha > 1$. Let W be the largest budget of the consumers in $\mathcal{I}$, and define a set

$$\mathcal{P} = \left\{W, \frac{W}{\alpha^1}, \frac{W}{\alpha^2}, \ldots, \frac{W}{\alpha^{\lceil \log_\alpha(\alpha nm)\rceil}}, 0\right\}$$

Our algorithm tries all the possible price functions that take values from the set $\mathcal{P}$ and returns as output a price function p that maximizes the revenue (over all the sets of price functions $p : \mathcal{I} \to \mathcal{P}$). It is easy to see that the running of the algorithm is $O(\lceil \log_\alpha nm + 3\rceil^n \text{poly}(n,m))$. Thus, we can set $\alpha = 2+\epsilon$ for some $\epsilon > 0$ so that the running time is $O((\log nm)^n\text{poly}(n,m))$. We claim that our algorithm

gives an approximation ratio of $\alpha^2/(\alpha-1)$ (proved in Lemma 8.3.5), thus yielding a constant-factor approximation.

Cost Analysis

We will focus on the case of SMP. The case of UDP-MIN can be analyzed analogously. Fix any optimal price function p^*, which yields a revenue of opt. We will construct from p^* a price function p' that takes values from the set $\mathcal{P}$ by rounding down each $p^*(i)$ to its closest value in $\mathcal{P}$. Since $p'(i) \in \mathcal{P}$ for all $i \in \mathcal{I}$, we can use p' to lower bound the revenue that we could obtain from our algorithm. For the ease of analysis, we will do this in two steps. First, we define a price function p_1 by setting

$$p_1(i) = \begin{cases} 0 & \text{if } p^*(i) < \frac{W}{\alpha nm} \\ p^*(i) & \text{otherwise} \end{cases}$$

In this step, we lose a revenue of at most opt$/\alpha$. This is because we have m consumers, and each consumer wants at most n items. So, the revenue loss is at most $nm \cdot W/(\alpha nm) \leq$ opt$/\alpha$ (since opt $\geq W$). That is, p_1 yields a revenue of at least $(1 - 1/\alpha)$opt. Next, we define a price function p_2 from p_1 by setting

$$p_2(i) = \begin{cases} 0 & \text{if } p_1(i) = 0 \text{ (i.e., } p^*(i) < \frac{W}{\alpha nm}) \\ \frac{W}{\alpha^j}, \text{ for some } j : \frac{W}{\alpha^{j+1}} < p^*(i) \leq \frac{W}{\alpha^j} & \text{otherwise} \end{cases}$$

So, $p_2(i) = p_1(i)$ for all i such that $p^*(i) < W/(\alpha nm)$. Observe that $p_2(i) \geq p_1(i)/\alpha$ for all $i \in \mathcal{I}$. Hence, the revenue obtained from p_2 is within a factor of $1/\alpha$ of the revenue obtained from p_1. Thus, p_2 yields a revenue of at least $(1/\alpha - 1/\alpha^2)$opt. Thus, the approximation ratio of our algorithm is $O(\alpha^2/(\alpha-1))$, for all $\alpha > 1$. In particular, by setting $\alpha = 2+\epsilon$ for $\epsilon > 0$, we have a $(4+\epsilon^2)$-approximation algorithm.

Remark: An Exact-Algorithm for Udp-Min

In this section, we observe that for the case of Udp-Min, an optimal solution can be obtained in time $O(n!\text{poly}(n, m))$ by using a *price ladder constraint*. To be precise, the price ladder constraint says that, given a permutation σ of items, the price of an item $\sigma(i)$ must be at most the price of an item $\sigma(i+1)$, i.e., $p(\sigma(i)) \leq p(\sigma(i+1))$, for all $i = 1, \ldots, n-1$. Briest and Krysta [13] showed that Udp-Min with the price ladder constraint (i.e., the permutation is additionally given as an input) is polynomial-time solvable. Thus, we can solve any instance of Udp-Min optimally in time $O((n!)\text{poly}(n, m))$ by trying all possible permutations of the items.

CHAPTER 9
Conclusion

This thesis studies hardness of approximation in the regime of subexponential-time algorithms. As the tools were limited, prior to the breakthrough result of Moshkovitz and Raz [59], the approximation lower bounds pertain only to polynomial-time algorithms. With the new tools developed, we step to the area beyond polynomial-time and extend the result of Moshkovitz and Raz to prove subexponential-time approximation hardness of several fundamental problems – independent set, graph coloring, bipartite induced matching, semi-induced matching and k-hypergraph pricing. Our aim is to understand the computation power of subexponential-time algorithms in the form of time-approximation trade-off. As this area is relatively new, there will be more open problems coming up. We hope that our results will be building blocks and foundation for others to explore the area of approximation algorithms when superpolynomial running-time algorithms are concerned.

REFERENCES

[1] N. Alon, U. Feige, A. Wigderson, and D. Zuckerman. Derandomized graph products. *Computational Complexity*, 5(1):60–75, 1995. 45

[2] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998. 2, 6, 16, 48

[3] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998. 2, 6, 16, 48

[4] M.-F. Balcan and A. Blum. Approximation algorithms and online mechanisms for item pricing. *Theory of Computing*, 3(1):179–195, 2007. 13, 14

[5] M.-F. Balcan, A. Blum, J. D. Hartline, and Y. Mansour. Mechanism design via machine learning. In *FOCS*, pages 605–614, 2005. 14

[6] M. Bellare, O. Goldreich, and M. Sudan. Free bits, pcps, and nonapproximability-towards tight results. *SIAM J. Comput.*, 27(3):804–915, 1998. 2, 48

[7] M. Bellare and M. Sudan. Improved non-approximability results. In *STOC*, pages 184–193, 1994. 48, 76

[8] P. Berman and G. Schnitger. On the complexity of approximating the independent set problem. *Inf. Comput.*, 96(1):77–94, 1992. 2, 30, 41, 42

[9] R. B. Boppana and M. M. Halldórsson. Approximating maximum independent sets by excluding subgraphs. *BIT*, 32(2):180–196, 1992. 2

[10] N. Bourgeois, B. Escoffier, and V. T. Paschos. Approximation of min coloring by moderately exponential algorithms. *Inf. Process. Lett.*, 109(16):950–954, 2009. 3

[11] N. Bourgeois, B. Escoffier, and V. T. Paschos. Approximation of max independent set, min vertex cover and related problems by moderately exponential algorithms. *Discrete Applied Mathematics*, 159(17):1954–1970, 2011. 3

[12] P. Briest and P. Krysta. Single-minded unlimited supply pricing on sparse instances. In *SODA*, pages 1093–1102, 2006. 13, 14

[13] P. Briest and P. Krysta. Buying cheap is expensive: Approximability of combinatorial pricing problems. *SIAM J. Comput.*, 40(6):1554–1586, 2011. 124

[14] R. A. Brualdi, F. Harary, and Z. Miller. Bigraphs versus digraphs via matrices. *Journal of Graph Theory*, 4(1):51–73, 1980. 79

[15] P. Chalermsook, J. Chuzhoy, S. Kannan, and S. Khanna. Improved hardness results for profit maximization pricing problems with unlimited supply. In *APPROX-RANDOM*, pages 73–84, 2012. 114

[16] P. Chalermsook, S. Kintali, R. J. Lipton, and D. Nanongkai. Graph pricing problem on bounded treewidth, bounded genus and k-partite graphs. *Chicago J. Theor. Comput. Sci.*, 2013, 2013. 14

[17] P. Chalermsook, B. Laekhanukit, and D. Nanongkai. Graph products revisited: Tight approximation hardness of induced matching, poset dimension and more. In *SODA*, pages 1557–1576, 2013. iv, 30, 62, 64, 65

[18] P. Chalermsook, B. Laekhanukit, and D. Nanongkai. Independent set, induced matching, and pricing: Connections and tight (subexponential time) approximation hardnesses. In *FOCS*, pages 370–379, 2013. iv, 21, 30

[19] P. Chalermsook, B. Laekhanukit, and D. Nanongkai. Coloring graph powers: Graph product bounds and hardness of approximation. In *LATIN*, pages 409–420, 2014. 30

[20] S. O. Chan. Approximation resistance from pairwise independent subgroups. In *STOC*, pages 447–456, 2013. 49

[21] S. Chawla, J. D. Hartline, and R. D. Kleinberg. Algorithmic pricing via virtual valuations. In *ACM Conference on Electronic Commerce*, pages 243–251, 2007. 14

[22] R. H. Chitnis, M. Hajiaghayi, and G. Kortsarz. Fixed-parameter and approximation algorithms: A new look. In *IPEC*, pages 110–122, 2013. 4

[23] M. Chlebík and J. Chlebíková. Complexity of approximating bounded variants of optimization problems. *Theor. Comput. Sci.*, 354(3):320–338, 2006. 67

[24] S. A. Cook. The complexity of theorem-proving procedures. In *STOC*, pages 151–158, 1971. 2

[25] M. Cygan, L. Kowalik, M. Pilipczuk, and M. Wykurz. Exponential-time approximation of hard problems. *CoRR*, abs/0810.4934, 2008. 3

[26] M. Cygan, L. Kowalik, M. Pilipczuk, and M. Wykurz. Exponential-time approximation of hard problems. *CoRR*, abs/0810.4934, 2008. 97

[27] M. Cygan, M. Pilipczuk, and M. Pilipczuk. Known algorithms for EDGE CLIQUE COVER are probably optimal. In *SODA*, pages 1044–1053, 2013. 21

[28] R. Diestel. *Graph Theory*. Graduate Texts in Mathematics, Volume 173. Springer-Verlag, Heidelberg, 4th edition, July 2010. 9

[29] K. M. Elbassioni, R. Raman, S. Ray, and R. Sitters. On the approximability of the maximum feasible subsystem problem with 0/1-coefficients. In *SODA*, pages 1210–1219, 2009. 62, 64

[30] L. Engebretsen and J. Holmerin. Towards optimal lower bounds for clique and chromatic number. *Theor. Comput. Sci.*, 1-3(299):537–584, 2003. 26, 27, 29, 50

[31] U. Feige. Randomized graph products, chromatic numbers, and the lovász vartheta-funktion. *Combinatorica*, 17(1):79–90, 1997. 30, 32, 41, 42, 45, 47, 50

[32] U. Feige. Approximating maximum clique by removing subgraphs. *SIAM J. Discrete Math.*, 18(2):219–225, 2004. 2

[33] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *J. ACM*, 43(2):268–292, 1996. 2, 7, 17, 48, 75

[34] U. Feige and J. Kilian. Zero knowledge and the chromatic number. *J. Comput. Syst. Sci.*, 57(2):187–199, 1998. 3, 25, 28, 29, 50, 51

[35] Y.-Q. Feng, K. Kutnar, A. Malnic, and D. Marusic. On 2-fold covers of graphs. *J. Comb. Theory, Ser. B*, 98(2):324–341, 2008. 79

[36] M. Fürer. Improved hardness results for approximating the chromatic number. In *FOCS*, pages 414–421, 1995. 50

[37] M. R. Garey and D. S. Johnson. The complexity of near-optimal graph coloring. *J. ACM*, 23(1):43–49, 1976. 30, 49

[38] G. H. Gonnet. Expected length of the longest probe sequence in hash code searching. *J. ACM*, 28(2):289–304, 1981. 108

[39] V. Guruswami, J. D. Hartline, A. R. Karlin, D. Kempe, C. Kenyon, and F. McSherry. On profit-maximizing envy-free pricing. In *SODA*, pages 1164–1173, 2005. 117

[40] M. M. Halldórsson. A still better performance guarantee for approximate graph coloring. *Inf. Process. Lett.*, 45(1):19–23, 1993. 2

[41] R. Hammack, W. Imrich, and S. Klavžar. *Handbook of Product Graphs, Second Edition*. Discrete Mathematics and Its Applications. Taylor & Francis, 2011. 31

[42] J. Håstad. Clique is hard to approximate within $n^{1\text{-epsilon}}$. In *FOCS*, pages 627–636, 1996. 2, 3, 26, 49, 50, 51, 64

[43] J. Håstad and S. Khot. Query efficient PCPs with perfect completeness. *Theory of Computing*, 1(1):119–148, 2005. 49

[44] R. Hegde and K. Jain. The hardness of approximating poset dimension. *Electron. Notes Discrete Math.*, 29:435–443, 2007. 62

[45] R. Impagliazzo and R. Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. 3

[46] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. 21, 22

[47] W. Imrich and T. Pisanski. Multiple kronecker covering graphs. *Eur. J. Comb.*, 29(5):1116–1122, 2008. 79

[48] D. S. Johnson. Worst case behavior of graph coloring algorithms. In *Proceedings of the Fifth Southeastern Conference on Combinatorics, Graph Theory and Computing (Florida Atlantic Univ., Boca Raton, Fla., 1974)*, pages 513–527. Congressus Numerantium, No. X, Winnipeg, Man., 1974. Utilitas Math. 2

[49] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972. 2, 48, 49

[50] R. Khandekar, T. Kimbrel, K. Makarychev, and M. Sviridenko. On hardness of pricing items for single-minded bidders. In *APPROX-RANDOM*, pages 202–216, 2009. 14

[51] S. Khot and A. K. Ponnuswami. Better inapproximability results for maxclique, chromatic number and min-3lin-deletion. In *ICALP (1)*, pages 226–237, 2006. 3

[52] L. A. Levin. Universal enumeration problems. *Problemy Peredači Informacii*, 9(3):115–116, 1973. (Russian). 2

[53] N. Linial and U. V. Vazirani. Graph products and chromatic numbers. In *FOCS*, pages 124–128, 1989. 30, 49

[54] D. Lokshtanov, D. Marx, and S. Saurabh. Known algorithms on graphs on bounded treewidth are probably optimal. In *SODA*, pages 777–789, 2011. 21

[55] D. Lokshtanov, D. Marx, and S. Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011. 21

[56] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13(4):383 – 390, 1975. 10, 11

[57] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41(5):960–981, 1994. 2, 50

[58] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA, 2005. 42, 108

[59] D. Moshkovitz and R. Raz. Two-query PCP with subconstant error. *J. ACM*, 57(5), 2010. 3, 4, 23, 24, 27, 29, 49, 50, 77, 79, 125

[60] P. Popat and Y. Wu. On the hardness of pricing loss-leaders. In *SODA*, pages 735–749, 2012. 14

[61] R. Raz. A parallel repetition theorem. *SIAM J. Comput.*, 27(3):763–803, 1998. 26

[62] O. Reingold, S. P. Vadhan, and A. Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *FOCS*, pages 3–13, 2000. 90

[63] P. Rusmevichientong, S. U. D. of Management Science, and Engineering. *A Nonparametric Approach to Multi-product Pricing Theory and Application*. Stanford University, 2003. 14

[64] P. Rusmevichientong, B. V. Roy, and P. W. Glynn. A nonparametric approach to multiproduct pricing. *Operations Research*, 54(1):82–98, 2006. 14

[65] A. Samorodnitsky and L. Trevisan. A PCP characterization of NP with optimal amortized query complexity. In *STOC*, pages 191–199, 2000. 23, 24, 26, 29, 49, 50, 77, 79

[66] E. Sampathkumar. On tensor product graphs. *Journal of the Australian Mathematical Society (Series A)*, 20:268–273, 11 1975. 79

[67] E. Scheinerman and D. Ullman. *Fractional graph theory: a rational approach to the theory of graphs*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1997. 32

[68] L. Trevisan. Non-approximability results for optimization problems on bounded degree instances. In *STOC*, pages 453–461, 2001. 20, 75, 76, 79, 80, 82, 86, 89, 90, 91

[69] D. Zuckerman. On unapproximable versions of NP-complete problems. *SIAM J. Comput.*, 25(6):1293–1304, 1996. 83

[70] D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(1):103–128, 2007. 3, 49, 85