On Approximating Projection Games

by

Pasin Manurangsi

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

February 2015

Author: _____

Department of Electrical Engineering and Computer Science
January 29, 2015

Certified by: _____

Prof. Dana Moshkovitz
ITT Career Development Professor in Computer Technology
Thesis Supervisor
January 29, 2015

Accepted by: _____

Prof. Albert R. Meyer
Chairman, Masters of Engineering Thesis Committee
January 29, 2015

On Approximating Projection Games

by

Pasin Manurangsi

Submitted to the Department of Electrical Engineering and Computer Science
on January 29, 2015 in Partial Fulfillment of the Requirements for
the Degree of Master of Engineering in Electrical Engineering and Computer Science

Abstract

The projection games problem (also known as LABEL COVER) is a problem of great significance in the field of hardness of approximation since almost all NP-hardness of approximation results known today are derived from the NP-hardness of approximation of projection games. Hence, it is important to determine the exact approximation ratio at which projection games become NP-hard to approximate. The goal of this thesis is to make progress towards this problem.

First and foremost, we present a polynomial-time approximation algorithm for satisfiable projection games, which achieves an approximation ratio that is better than that of the previously best known algorithm.

On the hardness of approximation side, while we do not have any improved NP-hardness result of approximating LABEL COVER, we show a polynomial integrality gap for polynomially many rounds of the Lasserre SDP relaxation for projection games. This result indicates that LABEL COVER might indeed be hard to approximate to within some polynomial factor.

In addition, we explore special cases of projection games where the underlying graphs belong to certain families of graphs. For planar graphs, we present both a subexponential-time exact algorithm and a polynomial-time approximation scheme (PTAS) for projection games. We also prove that these algorithms have tight running times. For dense graphs, we present a subexponential-time approximation algorithm for LABEL COVER. Moreover, if the graph is a sufficiently dense random graph, we show that projection games are easy to approximate to within any polynomial ratio.

Thesis Supervisor: Prof. Dana Moshkovitz
Title: ITT Career Development Professor in Computer Technology

Acknowledgments

I would like to thank my thesis advisor, Prof. Dana Moshkovitz, for her tremendous support and guidance throughout the completion of this thesis project and my previous UROP project. As my first ever research advisor, she not only guided me through the world of academia, but also spent a lot of time teaching me many fundamental theoretical knowledge; in fact, it was her who introduced me to approximation algorithms, hardness of approximation, and the projection games problem. Without the opportunity and support she has given, I would not have been able to even start doing any research in theoretical computer science. Her passion for the field of computer science and her commitment to research has also inspired me greatly. I could not have wished for a better research advisor for my Master's thesis.

Secondly, I would like to express my heartfelt gratitude towards Prof. Eden Chlamtac (Ben Gurion University) and Dr. Aravindan Vijayaraghavan (New York University); their collaborations and many discussions have significantly influenced this thesis.

I am also sincerely grateful to Bank of Thailand for their financial support for my undergraduate and Master's studies.

In addition, I am very thankful for all my friends who have always be with me during both joyful and rough periods. Without their help, my time at MIT would have been much less enjoyable and, at times, much less tolerable as well.

Lastly but importantly, I would like to thank my parents—Vichien and Vanpen—for their unfading love, understanding, and encouragement throughout my entire life.

Contents

CONTENTS

CONTENTS

1 Introduction

For many NP-hard optimization problems, we can find efficient approximation algorithms. Those are algorithms that produce an outcome, which is within some ratio, called "the approximation ratio" [1] of the optimal solution. For many such problems, the approximation algorithms known today are the best ones can hope for, i.e., approximating these problems to better ratios is NP-hard. However, for other problems, including the projection games problem that we introduce next, the best known approximation algorithm has an approximation ratio that is not known to be tight. Thus, it is natural to try to find better approximation algorithms and better hardness of approximation results for such problems in order to bridge the gap between the two sides.

1.1 Projection Games and the PCP Theorem

The projection games problem (also known as LABEL COVER) is a combinatorial optimization problem defined as follows.

[1] To avoid confusion, we define approximation ratio to be greater than one for the purpose of this thesis. In other words, an algorithm for a maximization problem has approximation ratio $\alpha > 1$ if and only if, for every input, the value of the solution output by the algorithm is at least $1/\alpha$ of the value of the optimal solution.

Projection Games

INPUT: A bipartite graph $G = (A, B, E)$, two finite sets of labels (aka alphabets) Σ_A, Σ_B, and, for each edge $e = (a, b) \in E$, a "projection" $\pi_e : \Sigma_A \to \Sigma_B$.
GOAL: Find an assignment to the vertices $\varphi_A : A \to \Sigma_A$ and $\varphi_B : B \to \Sigma_B$ that maximizes the number of edges $e = (a, b)$ that are "satisfied", i.e., $\pi_e(\varphi_A(a)) = \varphi_B(b)$.

An instance is said to be "satisfiable" or "feasible" or have "perfect completeness" if there exists an assignment that satisfies all edges. An instance is said to be "δ-nearly satisfiable" or "δ-nearly feasible" if there exists an assignment that satisfies $(1 - \delta)$ fraction of the edges. Moreover, we use n to denote $|A| + |B|$, the number of vertices in G, and k to denote $|\Sigma_A|$, the size of the alphabets.

LABEL COVER has gained much significance for approximation algorithms because of the following PCP Theorem, establishing that it is NP-hard, given a satisfiable projection game instance, to satisfy even an ε fraction of the edges:

Theorem 1.1 (Strong PCP Theorem)**.** *For every $\tilde{n}$ and $\varepsilon = \varepsilon(\tilde{n})$, there is $k = k(\varepsilon)$ and $n = n(\tilde{n}, \varepsilon)$, such that deciding* SAT *on inputs of size $\tilde{n}$ can be reduced to finding, given a satisfiable projection game on a graph of n vertices and alphabets of size k, an assignment that satisfies more than an ε fraction of the edges.*

This theorem is the starting point of the extremely successful long-code based framework for achieving hardness of approximation results [BGS98, Hås01], as well as of other optimal hardness of approximation results, e.g., for SET-COVER [LY94, Fei98, Mos12, DS13]. In fact, almost all NP-hardness of approximation results known today are based on the PCP theorem.

The aforementioned PCP theorem is essentially the hardness of approximation of LABEL COVER. There are several proofs of the strong PCP theorem that yield different parameters in Theorem 1.1 and, thus, different hardness of approximation results for projection games.

In one such proof, the parallel repetition theorem [Raz98], applied on the basic PCP Theorem [BFL91, BFLS91, AS98, ALM+98], shows a reduction from exact SAT on input size $\tilde{n}$ to LABEL COVER on input size $\tilde{n}^{O(\log 1/\varepsilon)}$. This translates to the following hardness of approximation for projection games: for any $0 < \delta < 1$, there is no polynomial-time $2^{\log^{1-\delta}(nk)}$-approximation algorithm for projection games unless NP $\not\subseteq$ DTIME$(n^{polylog(n)})$.

A different proof, based on PCP composition, has smaller blow up but larger alphabet size [MR10, DS13]. Specifically, it shows a reduction from exact SAT with input size $\tilde{n}$ to LABEL COVER with input size $\tilde{n}^{1+o(1)}poly(1/\varepsilon)$ and alphabet size $exp(1/\varepsilon)$. The corresponding hardness of approximation for projection games can be stated as followed: for any $\gamma > 0$, there is no polynomial-time $\log^{\gamma}(nk)$-approximation for projection games unless P = NP.

Moreover, although not proven, it is believed that a stronger proof of the PCP theorem with almost linear number of vertices and alphabets of sizes $poly(1/\varepsilon)$. The conjecture, formalized in [Mos12], can be stated as follows.

Conjecture 1.1 (The Projection Games Conjecture)**.** *For every $\tilde{n}$ and $\varepsilon = \varepsilon(\tilde{n})$, deciding* SAT *on inputs of size $\tilde{n}$ can be reduced to finding, given a satisfiable projection game on graph G of $\tilde{n}^{1+o(1)}poly(1/\varepsilon)$ vertices and alphabets of size $(\varepsilon)^{O(1)}$, an assignment that satisfies more than an ε fraction of the edges.*

To compare the conjecture to known hardness of approximation LABEL COVER results stated above, the Projection Game Conjecture implies that, for some constant $c > 0$, there is no polynomial-time $O((nk)^c)$-approximation algorithm for projection games unless P = NP.

It was shown in [Mos12] that, assuming the PGC, one can prove a stronger hardness of approximation result for the CLOSEST-VECTOR-PROBLEM. Moreover, [Mos12] suggested that many more hardness of approximation results could be proved based on the conjecture.

Since finding algorithms for projection games can help determine whether the PGC holds, it is therefore not only a natural pursuit in combinatorial optimization, but also a way to advance our understanding of the main paradigm for settling the approximability of optimization problems.

1.2 Previous Work on Approximating Projection Games

Even though great amount of effort has been put into understanding the hardness of approximation of the projection games problem, the approximation algorithms aspect of this problem has not been researched as much.

Prior to the author's joint work with his thesis supervisor in 2013 ([MM13]), only two papers had explicitly gave approximation algorithms for LABEL COVER. First, in 2007, Peleg presented a simple polynomial-time $O((nk)^{1/2})$-approximation algorithm for projection games [Pel07]. The approximation ratio was then improved in [CHK09] to $O((nk)^{1/3})$. In [MM13], we managed to improved this further to $O((nk)^{1/4})$, which still remains the best known polynomial-time approximation algorithm for projection games. Our algorithm will be described in full details later on in this thesis.

1.3 LP, SDP, and their Hierarchies

Linear programs are convex optimization problems that can be stated in the following form:

$$\begin{aligned} \text{maximize } & \sum_{i=1}^{n} c_i x_i \\ \text{subject to } & \sum_{i=1}^{n} a_{ij} x_i \geq b_j && \forall j \in \{1, \ldots, m\}, \\ & x_i \geq 0 && \forall i \in \{1, \ldots, n\}. \end{aligned}$$

Another form of convex optimization program, called Semidefinite program (SDP), concerns solving the following type of optimization problem:

$$\begin{aligned} \text{maximize } & \sum_{i,j=1}^{n} c_{ij} x_{ij} \\ \text{subject to } & \sum_{i,j=1}^{n} a_{ijk} x_{ij} \geq b_k && \forall k \in \{1, \ldots, n\}, \\ & x_{ij} = x_{ji} && \forall i, j \in \{1, \ldots, n\}, \\ & (x_{ij}) \text{ is positive semidefinite.} \end{aligned}$$

Recall that a real matrix $X \in \mathbb{R}^{n \times n}$ is positive semidefinite if and only if $v^T X v$ is positive for every real vector $v \in \mathbb{R}^n$. Instead of viewing semidefinite program as the above optimization problem, it is sometimes written in an equivalent form, called vector program, which is presented below.

$$\text{maximize } \sum_{i,j=1}^{n} c_{ij} (v_i \cdot v_j)$$

$$\begin{aligned} \text{subject to } \sum_{i,j=1}^{n} a_{ijk}(v_i \cdot v_j) \geq b_k \qquad & \forall k \in \{1,\ldots,n\}, \\ v_i \cdot v_i = 1 \qquad & \forall i \in \{1,\ldots,n\}, \\ v_i \in \mathbb{R}^n \qquad & \forall i \in \{1,\ldots,n\}. \end{aligned}$$

Both linear programming and semidefinite programming have been very useful in finding approximation algorithms for combinatorial optimization problems. They are used typically by rewriting a problem in terms of a linear program with x_i representing each natural variable in the problem or a vector program where each v_i represents each variable. Then, we solve the linear program or the vector program, which are known to be solvable in polynomial time in n. Since normally, for combinatorial optimization problems, we want each variable to be in a discrete set, the last step is often to perform rounding of each x_i or v_i to be a value in the set. LP and SDP techniques yield many results in approximation algorithms, such as algorithms for VERTEX-COVER, MAX-CUT, MAX-2SAT and the unique games problem on expanders [GW95, AKK$^+$08].

A more recent technique, called LP and SDP hierarchies, has been developed on top of linear programming and semidefinite programming. The main idea of the technique is that, by adding more variables and constraints, one can capture the original combinatorial problem more precisely. Using this method, one can obtain a hierarchy of semidefinite or linear programs from the simplest one to the strengthened ones. There are several systematic methods, such as the ones invented by Lovász and Schrijver [LS91], Sherali and Adams [SA90], and Lasserre [Las01a, Las01b], to add constraints and variables. Each of the method produces different hierarchies.

Among the hierarchies studied so far, the Lasserre hierarchy is known to be the

strongest one [Lau03]. It has yielded positive results in finding approximation algorithms for some problems. The most relevant such result is from [BRS11], in which the Lasserre hierarchy was used to find an approximation algorithm for the unique games problem, a problem closely related to LABEL COVER. Hence, it is essential to ask whether the Lasserre hierarchy can be used to give better approximation ratio for LABEL COVER, or even refute the Projection Games Conjecture.

1.4 Thesis Organization

Excluding this chapter, this thesis contains five chapters; while Chapter 2 only lists notations to be used throughout the thesis, each of the next four presents new results regarding approximating projection games and discusses how these results fit in the big picture.

First, in Chapter 3, we present a polynomial-time $O((nk)^{1/4})$-approximation algorithm for LABEL COVER, which is the best known polynomial-time approximation algorithm for projection games. The result is extracted from [MM13].

Second, in Chapter 4, we show negative results for using Lasserre SDP hierarchy to approximate the projection games problem. More specifically, we present LABEL COVER instances such that, even after polynomial rounds of the Lasserre SDP hierarchy, the integrality gaps of these instances remain polynomial. Note that the precise definition of integrality gap and its importance are explained at the beginning of Chapter 4.

Next, in Chapter 5, we present a polynomial-time approximation scheme (PTAS) for LABEL COVER on planar graphs. We then prove that this is the best PTAS one can hope for, i.e., no PTAS with substantially smaller running time exists. In addition,

we present a $k^{O(\sqrt{n})}$-time exact algorithm for projection games on planar graphs and prove also that this running time is tight.

In Chapter 6, we give a subexponential-time approximation algorithm for projection games on dense graphs. We also show that, if the underlying graph is a sufficiently dense random graph, then there exists a polynomial-time approximation algorithm that achieves any polynomial approximation ratio.

Finally, in the last chapter, we suggest how to improve upon our work and put forward some new directions for future work.

2 Notations

We define the following notations to be used throughout this thesis:

- Let $n_A = |A|$ denote the number of vertices in A and $n_B = |B|$ denote the number of vertices in B. Let n denote the number of vertices in the whole graph, i.e., $n = n_A + n_B$.
- Let k denote the alphabets size, i.e., $k = |\Sigma_A|$. Note that we assume without loss of generality throughout this thesis that $|\Sigma_A| \geq |\Sigma_B|$.
- Let N denote nk.
- Let V denote $A \cup B$, the set of all vertices in G.
- Let d_v denote the degree of a vertex $v \in V$.
- For a vertex u, we use $\Gamma(u)$ to denote set of vertices that are neighbors of u in G. Similarly, for a set of vertex U, we use $\Gamma(U)$ to denote the set of vertices that are neighbors of at least one vertex in U.
- For each vertex u, define $\Gamma_2(u)$ to be $\Gamma(\Gamma(u))$. This is the set of neighbors of neighbors of u.

3 Polynomial-Time Approximation Algorithm

In 2009, Charikar, Hajiaghayi and Karloff presented a polynomial-time $O((nk)^{1/3})$-approximation algorithm for LABEL COVER on graphs with n vertices and alphabets of size k [CHK09]. This improved on Peleg's $O((nk)^{1/2})$-approximation algorithm [Pel07]. We show a polynomial-time algorithm that achieves a better approximation ratio for satisfiable projection games:

Theorem 3.1. *There exists a polynomial-time $O(N^{1/4})$-approximation algorithm for satisfiable projection games.*

The result and all the proofs are extracted from [MM13]. Note here that our algorithm works in a more limited setting of projection games as both Peleg's and the CHK algorithms worked even for arbitrary constraints on the edges (not necessary functions) and possibly unsatisfiable instances.

While our result does not settle whether the Projection Games Conjecture is true, it does limit the kind of parameters we can expect for the PGC. More specifically, if the PGC is proved one day, our result would be useful to determine the best param-

eter for the number of vertices and the size of the alphabets one could get for the PGC.

3.1 Conventions

We define the following additional notations to be used in this chapter.

- Let σ_v^{OPT} be the assignment to v in an assignment to vertices that satisfies all the edges. In short, we will sometimes refer to this as "the optimal assignment". This is guaranteed to exist from our assumption that the instances considered are satisfiable.
- For any edge $e = (a, b)$, we define p_e to be $|\pi^{-1}(\sigma_b^{OPT})|$. In other words, p_e is the number of assignments to a that satisfy the edge e given that b is assigned σ_b^{OPT}, the optimal assignment. Define $\overline{p}$ to be the average of p_e over all e; that is $\overline{p} = \frac{\sum_{e \in E} p_e}{|E|}$.
- For each set of vertices S, define $E(S)$ to be the set of edges of G with at least one endpoint in S, i.e., $E(S) = \{(u, v) \in E \mid u \in S \text{ or } v \in S\}$.
- For each $a \in A$, let $h(a)$ denote $|E(\Gamma_2(a))|$. Let $h_{max} = max_{a \in A} h(a)$.

Moreover, for simplicity of the proof, we make the following assumptions in this chapter:

- G is connected. This assumption can be made without loss of generality, as, if G is not connected, we can always perform any algorithm presented below on each of its connected components and get an equally good or a better approximation ratio.
- For every $e \in E$ and every $\sigma_b \in \Sigma_B$, the number of preimages in $\pi_e^{-1}(\sigma_b)$ is the same. In particular, $p_e = \overline{p}$ for all $e \in E$.

We defer the treatment of graphs with general number of preimages to Appendix A.1.

3.2 The Algorithm

In this section, we present an improved polynomial-time approximation algorithm for projection games and prove Theorem 3.1.

To prove the theorem, we proceed to describe four polynomial-time approximation algorithms. In the end, by using the best of these four, we are able to produce a polynomial-time $O\left((n_A|\Sigma_A|)^{1/4}\right)$-approximation algorithm as desired. Next, we will list the algorithms along with its rough descriptions (see also illustrations in Figure 3.1 below); detailed description and analysis of each algorithm will follow later in this section:

1. **Satisfy one neighbor – $|E|/n_B$-approximation.** Assign each vertex in A an arbitrary assignment. Each vertex in B is then assigned to satisfy one of its neighboring edges. This algorithm satisfies at least n_B edges.

2. **Greedy assignment – $|\Sigma_A|/\overline{p}$-approximation.** Each vertex in B is assigned an assignment $\sigma_b \in \Sigma_B$ that has the largest number of preimages across neighboring edges $\sum_{a\in\Gamma(b)} |\pi^{-1}_{(a,b)}(\sigma_b)|$. Each vertex in A is then assigned so that it satisfies as many edges as possible. This algorithm works well when Σ_B assignments have many preimages.

3. **Know your neighbors' neighbors – $|E|\overline{p}/h_{max}$-approximation.** For a vertex $a_0 \in A$, we go over all possible assignments to it. For each assignment, we assign its neighbors $\Gamma(a_0)$ accordingly. Then, for each node in $\Gamma_2(a_0)$, we keep only the assignments that satisfy all the edges between it and vertices in $\Gamma(a_0)$.

   When a_0 is assigned the optimal assignment, the number of choices for each node in $\Gamma_2(a_0)$ is reduced to at most $\overline{p}$ possibilities. In this way, we can satisfy

$1/\overline{p}$ fraction of the edges that touch $\Gamma_2(a_0)$. This satisfies many edges when there exists $a_0 \in A$ such that $\Gamma_2(a_0)$ spans many edges.

4. **Divide and Conquer – $O(n_A n_B h_{max}/|E|^2)$-approximation.** For every $a \in A$ we can fully satisfy $\Gamma(a) \cup \Gamma_2(a)$ efficiently, and give up on satisfying other edges that touch $\Gamma_2(a)$. Repeating this process, we can satisfy $\Omega(|E|^2/(n_A n_B h_{max}))$ fraction of the edges. This is large when $\Gamma_2(a)$ does not span many edges for all $a \in A$.

The smallest of the four approximation factors is at most as large as their geometric mean, i.e.,

$$O\left(\sqrt[4]{\frac{|E|}{n_B} \cdot \frac{|\Sigma_A|}{\overline{p}} \cdot \frac{|E|\overline{p}}{h_{max}} \cdot \frac{n_A n_B h_{max}}{|E|^2}}\right) = O((n_A|\Sigma_A|)^{1/4}).$$

All the details of each algorithm are described below.

3.2.1 Satisfy One Neighbor Algorithm.

We will present a simple algorithm that gives $\frac{|E|}{n_B}$ approximation ratio.

Lemma 3.1. *For satisfiable instances of projection games, an assignment that satisfies at least n_B edges can be found in polynomial time, which gives the approximation ratio of $\frac{|E|}{n_B}$.*

Proof. For each node $a \in A$, pick one $\sigma_a \in \Sigma_A$ and assign it to a. Then, for each $b \in B$, pick one neighbor a of b and assign $\varphi(b) = \pi_e(\sigma_a)$ for b. This guarantees that at least n_B edges are satisfied. □

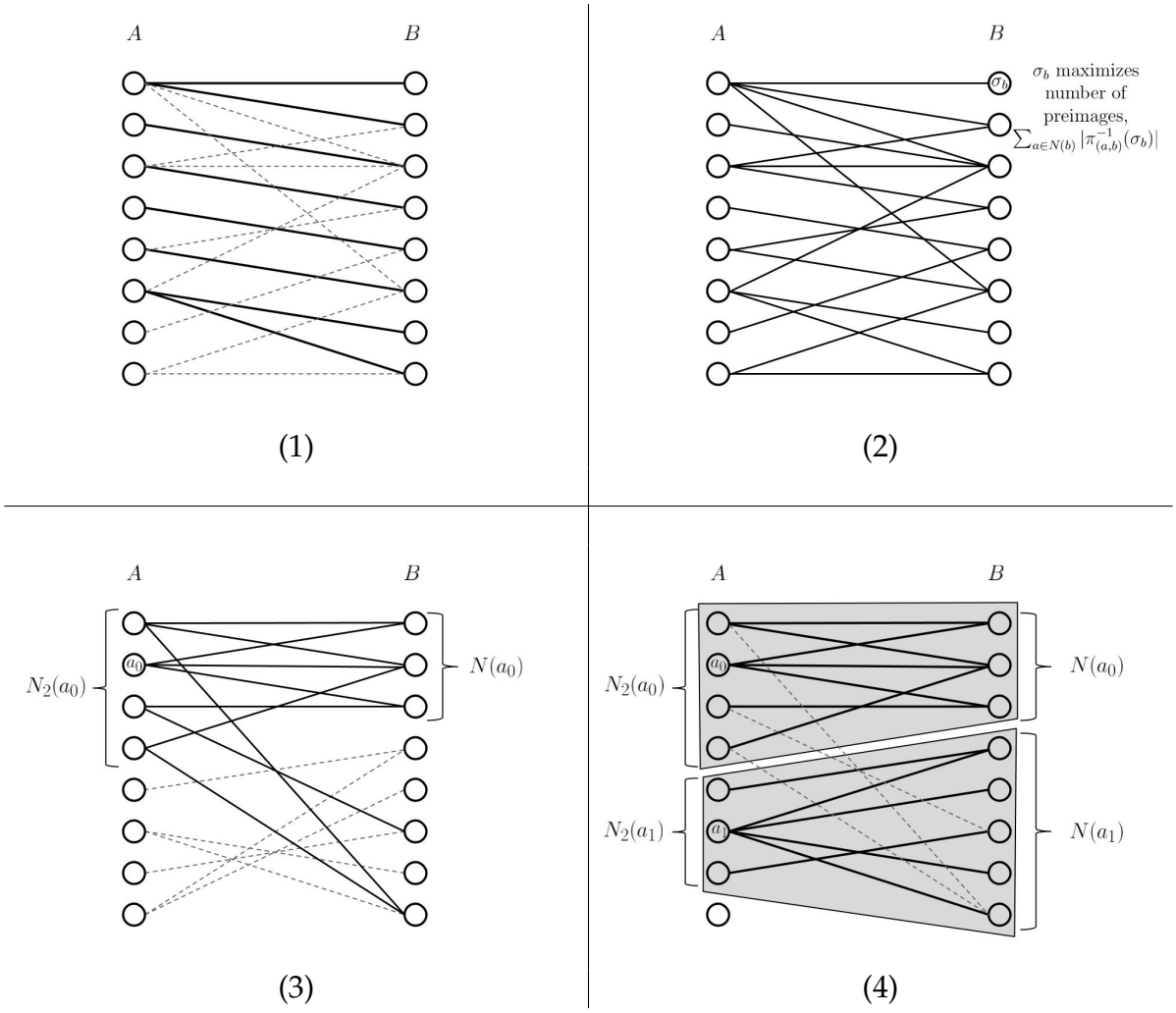

Figure 3.1: **An Overview of The Algorithms in One Figure.** Four algorithms are used to prove Theorem 3.1: (1) In *satisfy one neighbor* algorithm, each vertex in B is assigned to satisfy one of its neighboring edges. (2) In *greedy assignment* algorithm, each vertex in B is assigned with an assignment with largest number of preimages. (3) In *know your neighbors' neighbors* algorithm, for a vertex a_0, choices for each node in $\Gamma_2(a_0)$ are reduced to at most $O(\overline{p})$ possibilities so $O\left(\frac{1}{\overline{p}}\right)$ fraction of edges that touch $\Gamma_2(a_0)$ are satisfied. (4) In *divide and conquer* algorithm, the vertices are seperated to subsets, each of which is a subset of $\Gamma(a) \cup \Gamma_2(a)$, and each subset is solved separately.

3.2.2 Greedy Assignment Algorithm.

The idea for this algorithm is that if there are many assignments in Σ_A that satisfy each edge, then one satisfies many edges by guessing assignments at random. The

algorithm below is the deterministic version of this algorithm.

Lemma 3.2. *There exists a polynomial-time $\frac{|\Sigma_A|}{p}$-approximation algorithm for satisfiable instances of projection games.*

Proof. The algorithm works as follows:

1. For each b, assign it σ_b^* that maximizes $\sum_{a\in\Gamma(b)} |\pi_{(a,b)}^{-1}(\sigma_b)|$.
2. For each a, assign it σ_a^* that maximizes the number of edges satisfied, $|\{b \in \Gamma(a) \mid \pi_{(a,b)}(\sigma_a) = \sigma_b^*\}|$.

Let e^* be the number of edges that get satisfied by this algorithm. We have

$$e^* = \sum_{a\in A} |\{b \in \Gamma(a) \mid \pi_{(a,b)}(\sigma_a^*) = \sigma_b^*\}|.$$

By the second step, for each $a \in A$, the number of edges satisfied is at least an average of the number of edges satisfy over all assignments in Σ_A. This can be written as follows.

$$\begin{aligned} e^* &\geq \sum_{a\in A} \frac{\sum_{\sigma_a\in\Sigma_A} |\{b \in \Gamma(a) \mid \pi_{(a,b)}(\sigma_a) = \sigma_b^*\}|}{|\Sigma_A|} \\ &= \sum_{a\in A} \frac{\sum_{b\in\Gamma(a)} |\pi_{(a,b)}^{-1}(\sigma_b^*)|}{|\Sigma_A|} \\ &= \frac{1}{|\Sigma_A|} \sum_{a\in A} \sum_{b\in\Gamma(a)} |\pi_{(a,b)}^{-1}(\sigma_b^*)| \\ &= \frac{1}{|\Sigma_A|} \sum_{b\in B} \sum_{a\in\Gamma(b)} |\pi_{(a,b)}^{-1}(\sigma_b^*)|. \end{aligned}$$

Moreover, from the first step, we can conclude that, for each b, $\sum_{a\in\Gamma(b)} |\pi_{(a,b)}^{-1}(\sigma_b^*)| \geq \sum_{a\in\Gamma(b)} |\pi_{(a,b)}^{-1}(\sigma_b^{OPT})|$. As a result, we can conclude that

$$e^* \geq \frac{1}{|\Sigma_A|} \sum_{b\in B} \sum_{a\in\Gamma(b)} |\pi_{(a,b)}^{-1}(\sigma_b^{OPT})|$$

$$= \frac{1}{|\Sigma_A|} \sum_{e \in E} p_e$$
$$= \frac{|E|\overline{p}}{|\Sigma_A|}$$

Hence, this algorithm satisfies at least $\frac{\overline{p}}{|\Sigma_A|}$ fraction of the edges. Thus, this is a polynomial-time $\frac{|\Sigma_A|}{\overline{p}}$-approximation algorithm for satisfiable instances of projection games, which concludes our proof. □

3.2.3 Know Your Neighbors' Neighbors Algorithm

The next algorithm shows that if the neighbors of neighbors of a vertex $a_0 \in A$ expand, then one can satisfy many of the (many!) edges that touch the neighbors of a_0's neighbors.

Lemma 3.3. *For each $a_0 \in A$, there exists a polynomial-time $O\left(\frac{|E|\overline{p}}{h(a_0)}\right)$-approximation algorithm for satisfiable instances of projection games.*

Proof. To prove Lemma 3.3, we want to find an algorithm that satisfies at least $\Omega\left(\frac{h(a_0)}{\overline{p}}\right)$ edges for each $a_0 \in A$.

The algorithm works as follows:

1. Iterate over all assignments $\sigma_{a_0} \in \Sigma_A$ to a_0:
   (a) Assign $\sigma_b = \pi_{(a_0,b)}(\sigma_{a_0})$ to b for all $b \in \Gamma(a_0)$.
   (b) For each $a \in A$, find the set of plausible assignments to a, i.e., $S_a = \{\sigma_a \in \Sigma_A \mid \forall b \in \Gamma(a) \cap \Gamma(a_0), \pi_{(a,b)}(\sigma_a) = \sigma_b\}$. If for any a, the set S_a is empty, then we proceed to the next assignment without executing the following steps.

(c) For all $b \in B$, pick an assignment σ_b^* for b that maximizes the average number of satisfied edges over all assignments in S_a to vertices a in $\Gamma(b) \cap \Gamma_2(a_0)$, i.e., maximizes $\sum_{a \in \Gamma(b) \cap \Gamma_2(a_0)} |\pi_{(a,b)}^{-1}(\sigma_b) \cap S_a|$.

(d) For each vertex $a \in A$, pick an assignment $\sigma_a^* \in S_a$ that maximizes the number of satisfied edges, $|\{b \in \Gamma(a) \mid \pi_{(a,b)}(\sigma_a) = \sigma_b^*\}|$.

2. Pick an assignment $\{\sigma_a^*\}_{a \in A}$, $\{\sigma_b^*\}_{b \in B}$ from the previous step that satisfies the most edges.

We will prove that this algorithm indeed satisfies at least $\frac{h(a)}{\overline{p}}$ edges.

Let e^* be the number of edges satisfied by the algorithm. We have

$$e^* = \sum_{a \in A} |\{b \in \Gamma(a) \mid \pi_{(a,b)}(\sigma_a^*) = \sigma_b^*\}|.$$

Since in step 1, we try every possible $\sigma_{a_0} \in \Sigma_A$, we must have tried $\sigma_{a_0} = \sigma_{a_0}^{OPT}$. This means that the assignment to a_0 is the optimal assignment. As a result, the assignments to every node in $\Gamma(a_0)$ is the optimal assignment; that is $\sigma_b = \sigma_b^{OPT}$ for all $b \in \Gamma(a_0)$. Note that when the optimal assignment is assigned to a_0, we have $\sigma_a^{OPT} \in S_a$ for all $a \in A$. This means that the algorithm proceeds until the end. Thus, the solution this algorithm gives satisfies at least as many edges as when $\sigma_v = \sigma_v^{OPT}$ for all $v \in \{a_0\} \cup \Gamma(a_0)$. From now on, we will consider only this case.

Since for each $a \in A$, the assignment σ_a^* is chosen to maximize the number of edges satisfied, we can conclude that the number of edges satisfied by selecting σ_a^* is at least the average of the number of edges satisfied over all $\sigma_a \in S_a$.

As a result, we can conclude that

$$e^* \geq \sum_{a \in A} \frac{\sum_{\sigma_a \in S_a} |\{b \in \Gamma(a) \mid \pi_{(a,b)}(\sigma_a) = \sigma_b^*\}|}{|S_a|}$$

$$
\begin{aligned}
&= \sum_{a \in A} \frac{\sum_{\sigma_a \in S_a} \sum_{b \in \Gamma(a)} 1_{\pi_{(a,b)}(\sigma_a) = \sigma_b^*}}{|S_a|} \\
&= \sum_{a \in A} \frac{\sum_{b \in \Gamma(a)} \sum_{\sigma_a \in S_a} 1_{\pi_{(a,b)}(\sigma_a) = \sigma_b^*}}{|S_a|} \\
&= \sum_{a \in A} \frac{\sum_{b \in \Gamma(a)} |\pi_{(a,b)}^{-1}(\sigma_b^*) \cap S_a|}{|S_a|} \\
&= \sum_{b \in B} \sum_{a \in \Gamma(b)} \frac{|\pi_{(a,b)}^{-1}(\sigma_b^*) \cap S_a|}{|S_a|} \\
&\geq \sum_{b \in B} \sum_{a \in \Gamma(b) \cap \Gamma_2(a_0)} \frac{|\pi_{(a,b)}^{-1}(\sigma_b^*) \cap S_a|}{|S_a|}
\end{aligned}
$$

Now, for each $a \in \Gamma_2(a_0)$, consider S_a. From the definition of S_a, we have

$$
S_a = \{\sigma_a \in \Sigma_A \mid \forall b \in \Gamma(a) \cap \Gamma(a_0), \pi_{(a,b)}(\sigma_a) = \sigma_b\} = \bigcap_{b \in \Gamma(a) \cap \Gamma(a_0)} \pi_{(a,b)}^{-1}(\sigma_b).
$$

As a result, we can conclude that

$$
\begin{aligned}
|S_a| &\leq \min_{b \in \Gamma(a) \cap \Gamma(a_0)} \{|\pi_{(a,b)}^{-1}(\sigma_b)|\} \\
&= \min_{b \in \Gamma(a) \cap \Gamma(a_0)} \{|\pi_{(a,b)}^{-1}(\sigma_b^{OPT})|\} \\
&= \min_{b \in \Gamma(a) \cap \Gamma(a_0)} \{p_{(a,b)}\}.
\end{aligned}
$$

Note that since $a \in \Gamma_2(a_0)$, we have $\Gamma(a) \cap \Gamma(a_0) \neq \emptyset$. Since we assume for simplicity that $p_e = \overline{p}$ for all $e \in E$, we can conclude that $|S_a| \leq \overline{p}$.

This implies that

$$
e^* \geq \frac{1}{\overline{p}} \sum_{b \in B} \sum_{a \in \Gamma(b) \cap \Gamma_2(a_0)} |\pi_{(a,b)}^{-1}(\sigma_b^*) \cap S_a|.
$$

Since we pick the assignment σ_b^* that maximizes $\sum_{a\in\Gamma(b)\cap\Gamma_2(a_0)} |\pi_{(a,b)}^{-1}(\sigma_b^*) \cap S_a|$ for each $b \in B$, we can conclude that

$$\begin{aligned} e^* &\geq \frac{1}{\overline{p}} \sum_{b\in B} \sum_{a\in\Gamma(b)\cap\Gamma_2(a_0)} |\pi_{(a,b)}^{-1}(\sigma_b^*) \cap S_a| \\ &\geq \frac{1}{\overline{p}} \sum_{b\in B} \sum_{a\in\Gamma(b)\cap\Gamma_2(a_0)} |\pi_{(a,b)}^{-1}(\sigma_b^{OPT}) \cap S_a|. \end{aligned}$$

Since the optimal assignment satisfies every edge, we can conclude that $\sigma_a^{OPT} \in \pi_{(a,b)}^{-1}(\sigma_b^{OPT})$ and $\sigma_a^{OPT} \in S_a$, for all $b \in B$ and $a \in \Gamma(b) \cap \Gamma_2(a_0)$. This implies that

$$\begin{aligned} e^* &\geq \frac{1}{\overline{p}} \sum_{b\in B} \sum_{a\in\Gamma(b)\cap\Gamma_2(a_0)} |\pi_{(a,b)}^{-1}(\sigma_b^{OPT}) \cap S_a| \\ &\geq \frac{1}{\overline{p}} \sum_{b\in B} \sum_{a\in\Gamma(b)\cap\Gamma_2(a_0)} 1. \end{aligned}$$

The last term can be written as

$$\begin{aligned} \frac{1}{\overline{p}} \sum_{b\in B} \sum_{a\in\Gamma(b)\cap\Gamma_2(a_0)} 1 &= \frac{1}{\overline{p}} \sum_{a\in\Gamma_2(a_0)} \sum_{b\in\Gamma(a)} 1 \\ &= \frac{1}{\overline{p}}(h(a_0)) \\ &= \frac{h(a_0)}{\overline{p}}. \end{aligned}$$

As a result, we can conclude that this algorithm gives an assignment that satisfies at least $\frac{h(a_0)}{\overline{p}}$ edges out of all the $|E|$ edges. Hence, this is a polynomial-time $O\left(\frac{|E|\overline{p}}{h(a_0)}\right)$ approximation algorithm as desired. □

3.2.4 Divide and Conquer Algorithm.

We will present an algorithm that separates the graph into disjoint subgraphs for which we can find the optimal assignments in polynomial time. We shall show below that, if $h(a)$ is small for all $a \in A$, then we are able to find such subgraphs that contain most of the graph's edges.

Lemma 3.4. *There exists a polynomial-time $O\left(\frac{n_A n_B h_{max}}{|E|^2}\right)$-approximation algorithm for satisfiable instances of projection games.*

Proof. To prove Lemma 3.4, we will describe an algorithm that gives an assignment that satisfies $\Omega\left(\frac{|E|^3}{n_A n_B h_{max}}\right)$ edges.

We use $\mathcal{P}$ to represent the collection of subgraphs we find. The family $\mathcal{P}$ consists of disjoint sets of vertices. Let $V_{\mathcal{P}}$ be $\bigcup_{P \in \mathcal{P}} P$.

For any set S of vertices, define G_S to be the graph induced on S with respect to G. Moreover, define E_S to be the set of edges of G_S. We also define $E_{\mathcal{P}} = \bigcup_{P \in \mathcal{P}} E_P$.

The algorithm works as follows.

1. Set $\mathcal{P} \leftarrow \emptyset$.
2. While there exists a vertex $a \in A$ such that $|E_{(\Gamma(a) \cup \Gamma_2(a)) - V_{\mathcal{P}}}| \geq \frac{1}{4} \frac{|E|^2}{n_A n_B}$:
   (a) Set $\mathcal{P} \leftarrow \mathcal{P} \cup \{(\Gamma_2(a) \cup \Gamma(a)) - V_{\mathcal{P}}\}$.
3. For each $P \in \mathcal{P}$, find in time $poly(|\Sigma_A|, |P|)$ an assignment to the vertices in P that satisfies all the edges spanned by P.

We will divide the proof into two parts. First, we will show that when we cannot find a vertex a in step 2, $\left|E_{(A \cup B) - V_{\mathcal{P}}}\right| \leq \frac{|E|}{2}$. Second, we will show that the resulting assignment from this algorithm satisfies $\Omega\left(\frac{|E|^3}{n_A n_B h_{max}}\right)$ edges.

We will start by showing that if no vertex a in step 2 exists, then $\left|E_{(A\cup B)-V_{\mathcal{P}}}\right| \leq \frac{|E|}{2}$.

Suppose that we cannot find a vertex a in step 2. In other words, $|E_{(\Gamma(a)\cup\Gamma_2(a))-V_{\mathcal{P}}}| < \frac{1}{4}\frac{|E|^2}{n_A n_B}$ for all $a \in A$.

Consider $\sum_{a\in A}|E_{(\Gamma(a)\cup\Gamma_2(a))-V_{\mathcal{P}}}|$. Since $|E_{(\Gamma(a)\cup\Gamma_2(a))-V_{\mathcal{P}}}| < \frac{1}{4}\frac{|E|^2}{n_A n_B}$ for all $a \in A$, we have the following inequality.

$$\frac{|E|^2}{4n_B} \geq \sum_{a\in A}|E_{(\Gamma(a)\cup\Gamma_2(a))-V_{\mathcal{P}}}|.$$

Let $N^p(v) = \Gamma(v) - V_{\mathcal{P}}$ and $N_2^p(v) = \Gamma_2(v) - V_{\mathcal{P}}$. Similary, define $N^p(S)$ for a subset $S \subseteq A \cup B$. It is easy to see that $N_2^p(v) \supseteq N^p(N^p(v))$. This implies that, for all $a \in A$, we have $|E_{(N^p(a)\cup N_2^p(a))}| \geq |E_{(N^p(a)\cup N^p(N^p(a)))}|$. Moreover, it is not hard to see that, for all $a \in A - V_{\mathcal{P}}$, we have $|E_{(N^p(a)\cup N^p(N^p(a)))}| = \sum_{b\in N^p(a)}|N^p(b)|$.

Thus, we can derive the following:

$$\begin{aligned}
\sum_{a\in A}|E_{(\Gamma(a)\cup\Gamma_2(a))-V_{\mathcal{P}}}| &= \sum_{a\in A}|E_{(N^p(a)\cup N_2^p(a))}| \\
&\geq \sum_{a\in A-V_{\mathcal{P}}}|E_{(N^p(a)\cup N_2^p(a))}| \\
&\geq \sum_{a\in A-V_{\mathcal{P}}}\sum_{b\in N^p(a)}|N^p(b)| \\
&= \sum_{b\in B-V_{\mathcal{P}}}\sum_{a\in N^p(b)}|N^p(b)| \\
&= \sum_{b\in B-V_{\mathcal{P}}}|N^p(b)|^2.
\end{aligned}$$

From Jensen's inequality, we have

$$\begin{aligned}\sum_{a\in A}|E_{(\Gamma(a)\cup\Gamma_2(a))-V_{\mathcal{P}}}| &\geq \frac{1}{|B-V_{\mathcal{P}}|}\left(\sum_{b\in B-V_{\mathcal{P}}}|N^p(b)|\right)^2\\ &= \frac{1}{|B-V_{\mathcal{P}}|}\left|E_{(A\cup B)-V_{\mathcal{P}}}\right|^2\\ &\geq \frac{1}{n_B}\left|E_{(A\cup B)-V_{\mathcal{P}}}\right|^2.\end{aligned}$$

Since $\frac{|E|^2}{4n_B} \geq \sum_{a\in A}|E_{(\Gamma(a)\cup\Gamma_2(a))-V_{\mathcal{P}}}|$ and $\sum_{a\in A}|E_{(\Gamma(a)\cup\Gamma_2(a))-V_{\mathcal{P}}}| \geq \frac{1}{n_B}\left|E_{(A\cup B)-V_{\mathcal{P}}}\right|^2$, we can conclude that

$$\frac{|E|}{2} \geq \left|E_{(A\cup B)-V_{\mathcal{P}}}\right|$$

which concludes the first part of the proof.

Next, we will show that the assignment the algorithm finds satisfies at least $\Omega\left(\frac{|E|^3}{n_A n_B h_{max}}\right)$ edges. Since we showed that $\frac{|E|}{2} \geq \left|E_{(A\cup B)-V_{\mathcal{P}}}\right|$ when the algorithm terminates, it is enough to prove that $|E_{\mathcal{P}}| \geq \frac{|E|^2}{4n_A n_B h_{max}}\left(|E| - \left|E_{(A\cup B)-V_{\mathcal{P}}}\right|\right)$. Note that the algorithm guarantees to satisfy all the edges in $E_{\mathcal{P}}$.

We will prove this by using induction to show that at any point in the algorithm, $|E_{\mathcal{P}}| \geq \frac{|E|^2}{4n_A n_B h_{max}}\left(|E| - \left|E_{(A\cup B)-V_{\mathcal{P}}}\right|\right)$.

Base Case. At the beginning, we have $|E_{\mathcal{P}}| = 0 = \frac{|E|^2}{4n_A n_B h_{max}}\left(|E| - \left|E_{(A\cup B)-V_{\mathcal{P}}}\right|\right)$, which satisfies the inequality.

Inductive Step. The only step in the algorithm where any term in the inequality changes is step 2a. Let $\mathcal{P}_{old}$ and $\mathcal{P}_{new}$ be the set $\mathcal{P}$ before and after step 2a is executed, respectively. Let a be the vertex selected from step 2. Suppose that $\mathcal{P}_{old}$ satisfies the inequality.

From the condition in step 2, we have $|E_{(\Gamma(a)\cup\Gamma_2(a))-V_{\mathcal{P}_{old}}}| \geq \frac{1}{4}\frac{|E|^2}{n_A n_B}$. Since $|E_{\mathcal{P}_{new}}| = |E_{\mathcal{P}_{old}}| + |E_{(\Gamma(a)\cup\Gamma_2(a))-V_{\mathcal{P}_{old}}}|$, we have

$$|E_{\mathcal{P}_{new}}| \geq |E_{\mathcal{P}_{old}}| + \frac{1}{4}\frac{|E|^2}{n_A n_B}.$$

Now, consider $\left(|E| - |E_{(A\cup B)-V_{\mathcal{P}_{new}}}|\right) - \left(|E| - |E_{(A\cup B)-V_{\mathcal{P}_{old}}}|\right)$. We have

$$\left(|E| - |E_{(A\cup B)-V_{\mathcal{P}_{new}}}|\right) - \left(|E| - |E_{(A\cup B)-V_{\mathcal{P}_{old}}}|\right) = |E_{(A\cup B)-V_{\mathcal{P}_{old}}}| - |E_{(A\cup B)-V_{\mathcal{P}_{new}}}|$$

Since $V_{\mathcal{P}_{new}} = V_{\mathcal{P}_{old}} \cup (\Gamma_2(a) \cup \Gamma(a))$, we can conclude that

$$\left((A \cup B) - V_{\mathcal{P}_{old}}\right) \subseteq \left((A \cup B) - V_{\mathcal{P}_{new}}\right) \cup \left(\Gamma_2(a) \cup \Gamma(a)\right).$$

Thus, we can also derive

$$\begin{aligned} E_{(A\cup B)-V_{\mathcal{P}_{old}}} &\subseteq E_{\left((A\cup B)-V_{\mathcal{P}_{new}}\right)\cup(\Gamma_2(a)\cup\Gamma(a))} \\ &= E_{(A\cup B)-V_{\mathcal{P}_{new}}} \cup \{(a',b') \in E \mid a' \in \Gamma_2(a) \text{ or } b' \in \Gamma(a)\}. \end{aligned}$$

From the definition of N and N_2, for any $(a',b') \in E$, if $b' \in \Gamma(a)$ then $a' \in \Gamma_2(a)$. Thus, we have $\{(a',b') \in E \mid a' \in \Gamma_2(a) \text{ or } b' \in \Gamma(a)\} = \{(a',b') \in E \mid a' \in \Gamma_2(a)\} = E(\Gamma_2(a))$. The cardinality of the last term was defined to be $h(a)$. Hence, we can conclude that

$$\begin{aligned} |E_{(A\cup B)-V_{\mathcal{P}_{old}}}| &\leq |E_{(A\cup B)-V_{\mathcal{P}_{new}}} \cup \{(a',b') \in E \mid a' \in \Gamma_2(a) \text{ or } b' \in \Gamma(a)\}| \\ &\leq |E_{(A\cup B)-V_{\mathcal{P}_{new}}}| + |\{(a',b') \in E \mid a' \in \Gamma_2(a) \text{ or } b' \in \Gamma(a)\}| \end{aligned}$$

$$
\begin{aligned}
&= |E_{(A\cup B)-V_{\mathcal{P}_{new}}}| + |\{(a', b') \in E \mid a' \in \Gamma_2(a)\}| \\
&= |E_{(A\cup B)-V_{\mathcal{P}_{new}}}| + |E(\Gamma_2(a))| \\
&= |E_{(A\cup B)-V_{\mathcal{P}_{new}}}| + h(a) \\
&\leq |E_{(A\cup B)-V_{\mathcal{P}_{new}}}| + h_{max}.
\end{aligned}
$$

This implies that $\left(|E| - \left|E_{(A\cup B)-V_{\mathcal{P}}}\right|\right)$ increases by at most h_{max}.

Hence, since $\left(|E| - \left|E_{(A\cup B)-V_{\mathcal{P}}}\right|\right)$ increases by at most h_{max} and $|E_{\mathcal{P}}|$ increases by at least $\frac{1}{4}\frac{|E|^2}{n_A n_B}$ and from the inductive hypothesis, we can conclude that

$$
|E_{\mathcal{P}_{new}}| \geq \frac{|E|^2}{4n_A n_B h_{max}} \left(|E| - \left|E_{(A\cup B)-V_{\mathcal{P}_{new}}}\right|\right).
$$

Thus, the inductive step is true and the inequality holds at any point during the execution of the algorithm.

When the algorithm terminates, since $|E_{\mathcal{P}}| \geq \frac{|E|^2}{4n_A n_B h_{max}} \left(|E| - \left|E_{(A\cup B)-V_{\mathcal{P}}}\right|\right)$ and $\frac{|E|}{2} \geq \left|E_{(A\cup B)-V_{\mathcal{P}}}\right|$, we can conclude that $|E_{\mathcal{P}}| \geq \frac{|E|^3}{8n_A n_B h_{max}}$. Since the algorithm guarantees to satisfy every edge in $E_{\mathcal{P}}$, we can conclude that the algorithm gives $O(\frac{n_A n_B h_{max}}{|E|^2})$ approximation ratio, which concludes our proof of Lemma 3.4. □

3.2.5 Proof of the Main Theorem

Finally, we give a simple proof of the main theorem.

Proof of Theorem 3.1. Using Lemma 3.3 with a_0 that maximizes the value of $h(a_0)$, i.e., $h(a_0) = h_{max}$, we can conclude that there exists a polynomial-time $O\left(\frac{|E|\overline{p}}{h_{max}}\right)$-approximation algorithm for satisfiable instances of projection games.

Moreover, from Lemmas 3.1, 3.2 and 3.4, there exists a polynomial-time $\frac{|E|}{n_B}$-approximation algorithm, a polynomial-time $\frac{|\Sigma_A|}{\overline{p}}$-approximation algorithm and a polynomial time $O\left(\frac{n_A n_B h_{max}}{|E|^2}\right)$-approximation algorithm for satisfiable instances of projection games.

By picking the best out of these four algorithms, we can get an approximation ratio of $O\left(\min\left(\frac{|E|\overline{p}}{h_{max}}, \frac{|\Sigma_A|}{\overline{p}}, \frac{|E|}{n_B}, \frac{n_A n_B h_{max}}{|E|^2}\right)\right)$.

Since the minimum is at most the value of the geometric mean, we deduce that the approximation ratio is

$$O\left(\sqrt[4]{\frac{|E|\overline{p}}{h_{max}} \cdot \frac{|\Sigma_A|}{\overline{p}} \cdot \frac{|E|}{n_B} \cdot \frac{n_A n_B h_{max}}{|E|^2}}\right) = O\left(\sqrt[4]{n_A |\Sigma_A|}\right).$$

This concludes the proof of Theorem 3.1. □

4 Lasserre Gaps for Projection Games

Before we proceed to describe our results, we need to first describe the definition of *integrality gap* and its significance in approximation algorithms.

For a linear program or a semidefinite program that concerns a maximization problem, an integrality gap is defined to be the ratio of the value of the fractional (or vector) solution obtained from the program and the value of the optimal solution of the combinatorial optimization problem that the program tries to capture[1]. The integrality gap reflects the approximation ratio one can get from an approximation algorithm based on the program; simple approximation algorithms based on the program is not able to achieve an approximation ratio smaller than the integrality gap. Thus, a lower bound on integrality gap can be viewed as a lower bound on approximation ratio achievable by straightforward approximation algorithms based on the program.

In this chapter, we will show a couple of lower bounds on the integrality gap of a natural relaxation of projection games in the Lasserre Hierarchy as described below.

First, we show a lower bound of $N^{1/8-\varepsilon}$ on the integrality gap of $N^{\Omega(\varepsilon)}$ rounds of the Lasserre SDP hierarchy of projection games, which can be stated as follows.

[1]To avoid confusion, we only refer to integrality gaps that are larger than one in this thesis.

Theorem 4.1. *For every constant $0 < \varepsilon < 1/8$, there exists a projection game instance on a graph of n vertices and alphabets of size k such that the integrality gap of $(nk)^{\Omega(\varepsilon)}$ rounds of the Lasserre SDP hierarchy on this instance is at least $(nk)^{1/8-\varepsilon}$.*

Second, we show that, even after $N^{1-\varepsilon}$ rounds of the Lasserre SDP hierarchy, there is still a polynomial lower bound of $N^{\Omega(\varepsilon)}$ on the integrality gap as stated formally below.

Theorem 4.2. *For every constant $\varepsilon > 0$, there exists a projection game instance on a graph of n vertices and alphabets of size k such that the integrality gap of $(nk)^{1-\varepsilon}$ rounds of the Lasserre SDP hierarchy on this instance is at least $(nk)^{\Omega(\varepsilon)}$.*

Since the r-th round of the Lasserre hierarchy can be solved in time $N^{O(r)}$, our results show that the Lasserre SDP hierarchy cannot be used to refute the Projection Games Conjecture. Given that many best approximation algorithms known today, including our $O(N^{1/4})$-approximation algorithm presented in Chapter 3, can be formulated as LP or SDP hierarchies with sub-polynomial number of rounds, our results strongly suggest that the PGC is indeed true.

In addition, it is worth noting that, since the best known algorithm achieves an approximation ratio of $O(N^{1/4})$ but the integrality gap from Theorem 4.1 is $N^{1/8-\varepsilon}$, our result is not yet tight. Preferably, we would like the approximation ratio and the gap to match. For more discussion, please refer to Chapter 7

On the other hand, the gap from Theorem 4.2 does, in some sense, match with our simple subexponential-time algorithm from Chapter 6. More specifically, if we plug in $\varepsilon \leftarrow 1/N^{\varepsilon}$ in Theorem 6.1, we obtain the following corollary.

Corollary 4.1. *For every constant $\varepsilon > 0$, there exists an $\exp(O(N^{1-\varepsilon}))$-time N^{ε}-approximation algorithm for satisfiable projection games.*

It is not hard to see that the algorithm in Theorem 6.1 can be described based on LP or SDP hierarchy. Since the gap in Theorem 4.2 rules out the possibility of such algorithm that achieves $N^{\Omega(\varepsilon)}$ approximation ratio and runs in $\exp(O(N^{1-\varepsilon}))$ time, the above corollary cannot be much improved except perhaps the approximation ratio from N^{ε} to $N^{O(\varepsilon)}$ where the multiplicative constant in the exponent is smaller than one.

To prove the two main theorems in this chapter, we reduce from a lower bound of random MAX K-CSP proved in [Tul09]. The reduction and proof follow very closely from those in [BCV$^+$12], in which similar lower bound results for DENSEST k-SUBGRAPH were shown. In fact, the only main different of our proof and their proof is that we are able to prove a stronger soundness result, which ultimately leads to a better exponent in Theorem 4.1 of value $1/8 - \varepsilon$ compared to $2/53 - \varepsilon$ in [BCV$^+$12] [2]. Due to this similarity, this chapter will stick closely to the organization of [BCV$^+$12].

4.1 Conventions

In this chapter, we will assume without loss of generality that $\Sigma_A = \Sigma_B = [k]$. Moreover, when we refer to an edge (u, v) from E, it is always assumed that $u \in A$ and $v \in B$.

In addition, we define the following notations to be used in this chapter:

[2] $2/53 = 0.03773\ldots$

- For each subset $S \subseteq A \cup B$, we will view $\alpha \in [k]^S$ as a mapping from S to $[k]$. This represents an assignment to vertices in S.
- For each $S' \subseteq S \subseteq A \cup B$ and $\alpha \in [k]^S$, let $\alpha(S') \in [k]^{S'}$ denotes the restriction of α onto S'.
- For each $S_1, S_2 \subseteq A \cup B$, $\alpha_1 \in [k]^{S_1}$ and $\alpha_2 \in [k]^{S_2}$, if $\alpha_1(S_1 \cap S_2) = \alpha_2(S_1 \cap S_2)$, then define $\alpha_1 \circ \alpha_2$ by

$$\alpha_1 \circ \alpha_2(j) = \begin{cases} \alpha_1(j) & \text{if } j \in S_1, \\ \alpha_2(j) & \text{otherwise,} \end{cases}$$

  for every $j \in S_1 \cup S_2$. In other words, this is the assignment induced by both α_1 and α_2.

4.2 Lasserre SDP for Projection Games

In this section, we will write out the Lasserre SDP for projection games after r rounds. For compactness of this thesis, we will not describe the full procedure to create semidefinite programs for the Lasserre hierarchy. The reader can refer to many surveys on the topic out there, such as Rothvoß's note [Rot13], for the precise definition of the Lasserre hierarchy and its basic properties.

In the r-th round of the Lasserre SDP, for each $S \subseteq A \cup B$ with $S \leq |r|$ and $\alpha \in [k]^S$, a vector $U_{(S,\alpha)}$ induces a probability distribution over $[k]^S$ in the sense that we want $||U_{(S,\alpha)}||^2$ to represent the probability that the assignment α is chosen for S. The Lasserre SDP for projection games after r rounds is shown below.

Lasserre SDP for Projection Games

$$\text{maximize} \sum_{(u,v)\in E} \sum_{\sigma\in[k]} ||U_{(\{u,v\},\{u\to\sigma,v\to\pi_{(u,v)}(\sigma)\})}||^2$$

subjects to

$$||U_{(\emptyset,\emptyset)}|| = 1$$

$$\left\langle U_{(S_1,\alpha_1)}, U_{(S_1,\alpha_1)} \right\rangle = 0 \qquad \forall S_1, S_2, \alpha_1, \alpha_2 \text{ s.t. } \alpha_1(S_1) \neq \alpha_2(S_2)$$

$$\left\langle U_{(S_1,\alpha_1)}, U_{(S_2,\alpha_2)} \right\rangle = \left\langle U_{(S_3,\alpha_3)}, U_{(S_4,\alpha_4)} \right\rangle \qquad \forall S_1, S_2, S_3, S_4, \alpha_1, \alpha_2, \alpha_3, \alpha_4 \text{ s.t. } \alpha_1 \circ \alpha_2 = \alpha_3 \circ \alpha_4$$

$$\left\langle U_{(S_1,\alpha_1)}, U_{(S_1,\alpha_1)} \right\rangle \geq 0 \qquad \forall S_1, S_2, \alpha_1, \alpha_2$$

$$\sum_{\sigma\in[k]} ||U_{(\{v\},\sigma)}||^2 = 1 \qquad \forall v \in A \cup B$$

Note here that, since we are considering Lasserre SDP after r rounds, all sets S_1, S_2, S_3, S_4 included in the constraints are only those of size at most r. Note also that we use $\{u \to \sigma, v \to \pi_{(u,v)}(\sigma)\}$ to represents the map from $\{u, v\}$ to $[k]$ where u is mapped to σ and v is mapped to $\pi_{(u,v)}(\sigma)\}$. Similar notations later on in this chapter are also defined in similar manners.

4.3 Integrality Gap for Random MAX K-CSP Lasserre SDP from [Tul09] and [BCV$^+$12]

One of the main ingredients for our proof is the lower bound on integrality gap of the Lasserre SDP of random MAX K-CSP instances, which we will reduce from. Tulsiani proved a version of the result in [Tul09] before it was generalized by Bhaskara et al. in [BCV$^+$12]. We will use the [BCV$^+$12] version of the result for our proof.

Before we state their result, we will first review the definition of the MAX K-CSP problem:

MAX K-CSP$_q$

INPUT: An instance Φ of MAX K-CSP$_q$ consisting of

- a set of n variables $\{x_1, \ldots, x_n\}$, and
- a set of constraints $\{C_1, \ldots, C_m\}$ where each C_i is a mapping from $[q]^{T_i}$ to $\{0, 1\}$ for some ordered tuple T_i of K distinct variables.

GOAL: find an assignment $\varphi : \{x_1, \ldots, x_n\} \to [q]$ that maximizes the number of constraints C_i's that are satisfied, i.e. $C_i(\varphi|_{T_i}) = 1$ where $\varphi|_{T_i}$ is a restriction of φ to T_i. In other words, find an assignment $\varphi : \{x_1, \ldots, x_n\} \to [q]$ that maximizes $\sum_{i \in [m]} C_i(\varphi|_{T_i})$.

From now on, we will only consider q's that are prime numbers. Note, however, that, when we pick q later on in the chapter, we will not bother to check whether q is prime. Since we can always pick a prime number between $q/2$ and q, it is not hard to see that our assumption does not asymptotically affect the integrality gap and the number of rounds.

Similar to the Lasserre SDP for the projection games problem, the Lasserre SDP for the MAX K-CSP$_q$ after r rounds of the hierarchy can be written as follows.

Lasserre SDP for MAX K-CSP$_q$

maximize $\sum_{i \in [m]} \sum_{\alpha \in [q]^{T_i}} C_i(\alpha) ||V_{(T_i, \alpha)}||^2$

subjects to

$$
\begin{aligned}
&||V_{(\emptyset,\emptyset)}|| = 1 \\
&\left\langle V_{(S_1,\alpha_1)}, V_{(S_1,\alpha_1)} \right\rangle = 0 && \forall S_1, S_2, \alpha_1, \alpha_2 \text{ s.t. } \alpha_1(S_1) \neq \alpha_2(S_2) \\
&\left\langle V_{(S_1,\alpha_1)}, V_{(S_2,\alpha_2)} \right\rangle = \left\langle V_{(S_3,\alpha_3)}, V_{(S_4,\alpha_4)} \right\rangle && \forall S_1, S_2, S_3, S_4, \alpha_1, \alpha_2, \alpha_3, \alpha_4 \text{ s.t. } \alpha_1 \circ \alpha_2 = \alpha_3 \circ \alpha_4 \\
&\left\langle V_{(S_1,\alpha_1)}, V_{(S_1,\alpha_1)} \right\rangle \geq 0 && \forall S_1, S_2, \alpha_1, \alpha_2 \\
&\sum_{j \in [q]} ||V_{(\{i\},j)}||^2 = 1 && \forall i \in [n]
\end{aligned}
$$

Note here that, if we consider r rounds of the Lasserre hierarchy, all sets S_1, S_2, S_3, S_4 included in the constraints are only those of size at most r.

Now, we will define a random instance of the MAX K-CSP(C).

Random MAX K-CSP(C)

Let $C \subseteq \mathbb{F}_q^K$ be a set of codewords of any linear code of length K. A random MAX K-CSP(C) instance over n variables $x_1, \ldots, x_n$ and m constraints is a set of constraints $\{C_1, \ldots, C_m\}$ where each constraint C_i is independently sampled as follows:

- Randomly select a subset $T_i = \{x_{i_1}, \ldots, x_{i_K}\} \subseteq \{x_1, \ldots, x_n\}$ uniformly among all the subsets of size K.
- Randomly pick $(b_1^{(i)}, \ldots, b_K^{(i)}) \in \mathbb{F}_q^K$ uniformly.
- The constraint $C_i : [q]^{T_i} \to \{0,1\}$ is defined as

$$
C_i(\alpha) = \begin{cases} 1 & \text{if } (\alpha(x_{i_1}) + b_1^{(i)}, \ldots, \alpha(x_{i_K}) + b_K^{(i)}) \in C, \\ 0 & \text{otherwise,} \end{cases}
$$

for all $\alpha \in [q]^{T_i}$

In [Tul09], Tulsiani presented a Lasserre gap for random instances of the MAX K-CSP. More specifically, the result can be broken down into two parts: the vector completeness and the soundness. For the purpose of this thesis, we will only use their theorem for vector completeness, i.e., there exists a vector solution for the Lasserre SDP after some large number of rounds with perfect value. In [Tul09], the below lemma is proved for constant K, which is extended to work for superconstant K in [BCV$^+$12]. It can be stated as follows [3].

Lemma 4.1 ([BCV$^+$12])**.** *Let C be a dual code of any linear code of distance at least $D \geq 3$. For every $n, K, \beta, \eta > 0$ such that*

- *n is large enough,*
- *$10 \leq K \leq n^{1/2}$,*
- *$\eta \leq 1/(10^8(\beta K^D)^{2/(D-2)})$,*
- *$n^{\nu-1} \leq 1/(10^8(\beta K^{D+0.75})^{2/(D-2)})$ for some $\nu > 0$,*

with probability $1 - o(1)$, there exists a perfect solution to the $\eta n/16$-th round Lasserre SDP for a random MAX K-CSP *instance over n variables and $m = \beta n$ constraints. More specifically, there exists $V_{(S,\alpha)}$ for every $S \subseteq \{x_1, \ldots, x_n\}$ of size at most $\eta n/16$ and $\alpha \in [q]^S$ such that*

- *$\sum_{i \in [m]} \sum_{\alpha \in [q]^{T_i}} C_i(\alpha) ||V_{(T_i,\alpha)}||^2 = m$,*
- *$||V_{(\emptyset,\emptyset)}|| = 1$,*

[3] The lemma stated here is almost a direct quote from Theorem 4.2 in [BCV$^+$12]. The only difference is that, in Theorem 4.2 in [BCV$^+$12], η is required to be between $n^{\nu-1}$ and $1/(10^8(\beta K^{D+0.75})^{2/(D-2)})$. However, it is easy to see from the proof of the theorem that it is enough for η to be at most $1/(10^8(\beta K^D)^{2/(D-2)})$.

- $\left\langle V_{(S_1,\alpha_1)}, V_{(S_1,\alpha_1)} \right\rangle = 0 \; \forall S_1, S_2, \alpha_1, \alpha_2 \; s.t. \; \alpha_1(S_1) \neq \alpha_2(S_2)$,
- $\left\langle V_{(S_1,\alpha_1)}, V_{(S_2,\alpha_2)} \right\rangle = \left\langle V_{(S_3,\alpha_3)}, V_{(S_4,\alpha_4)} \right\rangle \; \forall S_1, S_2, S_3, S_4, \alpha_1, \alpha_2, \alpha_3, \alpha_4 \; s.t. \; \alpha_1 \circ \alpha_2 = \alpha_3 \circ \alpha_4$,
- $\left\langle V_{(S_1,\alpha_1)}, V_{(S_1,\alpha_1)} \right\rangle \geq 0 \; \forall S_1, S_2, \alpha_1, \alpha_2$,
- $\sum_{j \in [q]} ||V_{(\{i\},j)}||^2 = 1 \; \forall i \in [n]$

The code C we are going to use later is the dual code of a generalized BCH code, which gives the set of parameters as stated in the following lemma. The lemma differs slightly from Lemma 4.6 in [BCV$^+$12], where the block length of the code is $q^2 - 1$.

Lemma 4.2. *([BCV$^+$12]) For each integer $D \geq 3$ and a prime number $q \geq D$, there exists a q-ary linear code with block length $K = q - 1$, dimension $K - D + 1$ and distance at least D.*

Again, we will follow the construction of the code from [BCV$^+$12] as follows.

Proof. Let γ be a primitive root modulo q. Let the set of code words C be the set $\{(c_0, \ldots, c_{q-2}) \in \mathbb{F}_q^{q-1} \mid c(1) = c(\gamma) = \cdots = c(\gamma^{D-2}) = 0\}$ where c denote the polynomial $c(x) = c_0 + c_1 x + \cdots + c_{q-2} x^{q-2}$.

To argue about the distance of the code, consider any codeword $(c_0, \ldots, c_{q-2})$ that contains at most $D - 1$ non-zero entries. Suppose that these entires are $i_1, \ldots, i_{D-1}$.

From the definition of C, $c_{i_1}, \ldots, c_{i_{D-1}}$ must satisfies the following condition.

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ \gamma^{i_1} & \gamma^{i_2} & \cdots & \gamma^{i_{D-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma^{i_1(D-2)} & \gamma^{i_2(D-2)} & \cdots & \gamma^{i_{D-1}(D-2)} \end{bmatrix} \begin{bmatrix} c_{i_1} \\ c_{i_2} \\ \vdots \\ c_{i_{D-1}} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Since the matrix on the left is the transpose of a Vandermonde matrix, it is invertible, which means that $c_0, \ldots, c_{q-2}$ are all equal to zero. As a result, we can conclude that the distance of the code is at least D.

Moreover, note that, each condition can be written as a linear equation in $\mathbb{F}_q$. This means that our code has dimension at least $K - (D-1)$ as desired. □

Observe that, since we will choose C to be the dual of this code, C will be a code with block length K and dimension $D-1$. Note also that we will ultimately pick D to be some small constant less than 10.

4.4 Integrality Gaps for Projection Games Lasserre SDP

We devote this section to prove our two main theorems.

4.4.1 Reduction from MAX K-CSP to Projection Games

In this subsection, we present the reduction from MAX K-CSP to the projection games problem. Again, this is very similar to the reduction from MAX K-CSP to DENSEST k-SUBGRAPH in [BCV+12].

Given a random MAX K-CSP(C) instance $\Phi = \{C_1, \ldots, C_m\}$, we create a projection game instance $P = (A, B, E, \Sigma_A, \Sigma_B, \{\pi_e\}_{e \in E})$ as follows:

- We create a vertex in A for each constraint C_i , i.e., $A = \{C_1, \ldots, C_m\}$.
- We create a vertex in B for each variable x_j, i.e., $B = \{x_1, \ldots, x_n\}$.
- For each vertex $C_i \in A$ and $x_j \in B$, we create an edge between them if and only if $x_j \in T_i$, i.e., $E = \bigcup_{i \in [m]} \{(C_i, x_j) \mid x_j \in T_i\}$.
- Let Σ_A be a set of size equals to the number of codewords in C ($\Sigma_A = [|C|]$). For each $C_i \in A$, we will view each element of Σ_A as an assignment $\alpha \in [q]^{T_i}$ such that $C_i(\alpha) = 1$.
- Due to our convention here, we set Σ_B to be Σ_A. Note however that there $\pi_e^{-1}(\sigma)$ is empty for all $\sigma \notin [q]$ for every edge $e \in E$.
- For each edge (C_i, x_j), we define $\pi_{(C_i, x_j)} : \Sigma_A \to \Sigma_B$ by $\pi_{(C_i, x_j)}(\alpha) = \alpha(x_j)$.

Figure 4.1 illustrates an example of the reduction.

Observe that $|A| = m = \beta n$ and $|B| = n$ and $|\Sigma_A| = |C|$. As a result, $N = (|A| + |B|)\Sigma_A = \Theta(\beta n |C|)$.

To prove that this reduction gives us a Lasserre SDP gap, we need to show two things: the vector completeness, i.e., the instance P admits a perfect solution for Lasserre SDP, and the soundness, i.e., the optimal assignment for P satisfies not many edges. We will proceed to prove these in the following two subsections.

$$
\begin{aligned}
&q = 2, n = 3, m = 2,\\
&T_1 = \{x_1, x_2\},\\
&C_1^{-1}(1) = \{(1,1),(2,2)\},\\
&T_2 = \{x_1, x_3\},\\
&C_2^{-1}(1) = \{(1,2),(2,1)\}.
\end{aligned}
$$

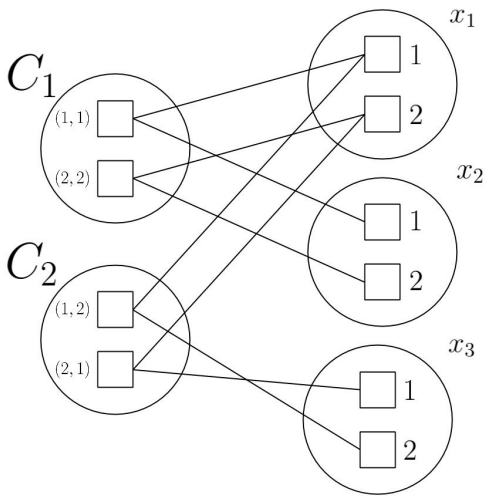

Figure 4.1: **An Example of the Reduction from** MAX K-CSP(C) **to** LABEL COVER**.** The original MAX K-CSP(C) instance is shown on the left and the resulting projection game is shown on the right. Note that, in the projection game illustration, circles represent vertices of the underlying graph whereas squares represent alphabet symbols. Each connecting line means that the corresponding projection maps the alphabet symbol on the left to the one on the right.

4.4.2 Vector Completeness

In this subsection, we will show that if the initial MAX K-CSP(C) instance Φ admits a perfect vector solution to the Lasserre SDP for MAX K-CSP$_q$, then the projection game instance P also admits a perfect vector solution (with the value mK) to the Lasserre SDP for projection games. However, the number of rounds of the later relaxation is K times smaller than that of the former. This can be stated formally as follows.

Lemma 4.3. *If the* MAX K-CSP(C) *instance* Φ *admits a perfect vector solution to the r-th round Lasserre SDP relaxation for* MAX K-CSP$_q$*, then the projection game instance P obtained from the reduction in Subsection 4.4.1 admits a perfect vector solution for the r/K-th round Lasserre SDP relaxation for projection games.*

Proof. Suppose that the MAX K-CSP(C) instance Φ admits a perfect vector solution to the r-th round Lasserre SDP relaxation. This means that there exists $V_{(S,\alpha)}$ for every $S \subseteq \{x_1, \ldots, x_K\}$ of size at most t and $\alpha \in [q]^S$ such that

- $\sum_{i\in[m]} \sum_{\alpha\in[q]^{T_i}} C_i(\alpha)||V_{(T_i,\alpha)}||^2 = m$,
- $||V_{(\emptyset,\emptyset)}|| = 1$,
- $\left\langle V_{(S_1,\alpha_1)}, V_{(S_1,\alpha_1)} \right\rangle = 0 \; \forall S_1, S_2, \alpha_1, \alpha_2$ s.t. $\alpha_1(S_1) \neq \alpha_2(S_2)$,
- $\left\langle V_{(S_1,\alpha_1)}, V_{(S_2,\alpha_2)} \right\rangle = \left\langle V_{(S_3,\alpha_3)}, V_{(S_4,\alpha_4)} \right\rangle \; \forall S_1, S_2, S_3, S_4, \alpha_1, \alpha_2, \alpha_3, \alpha_4$ s.t. $\alpha_1 \circ \alpha_2 = \alpha_3 \circ \alpha_4$,
- $\left\langle V_{(S_1,\alpha_1)}, V_{(S_1,\alpha_1)} \right\rangle \geq 0 \; \forall S_1, S_2, \alpha_1, \alpha_2$,
- $\sum_{j\in[q]} ||V_{(\{i\},j)}||^2 = 1 \; \forall i \in [n]$,

Note that all the sets S_1, S_2, S_3, S_4 above are limited only to ones with size at most r.

We will define the vector solution for r/K-th round Lasserre SDP for projection games as follows. Consider each subset $S \subseteq A \cup B$ of size at most r/K and each $\alpha \in [|C|]^S$. Suppose that $S \cap A = \{C_{i_1}, C_{i_2}, \ldots, C_{i_{a_1}}\}$ and $S \cap B = \{x_{j_1}, x_{j_2}, \cdots, x_{j_{b_1}}\}$.

First of all, if at least one of $\alpha(x_{j_1}), \ldots, \alpha(x_{j_{b_1}})$ is not in $[q]$, then we define $U_{(S,\alpha)}$ to be zero. Note that, as mentioned earlier, we treat Σ_B as $[q]$, which is why we let every set corresponding to an assignment outside of this range to be zero. It is also obvious to check that when we are dealing with this kind of "invalid" assignments, every condition in the Lasserre SDP for projection games hold. From now on, we will only consider the case where $\alpha(x_{j_1}), \ldots, \alpha(x_{j_{b_1}}) \in [q]$.

We view each $\alpha(C_{i_p})$ as an assignment to all the variables in T_{i_p} for all $p \in [a_1]$. Now, define a vector solution for projection games Lasserre SDP as follows:

$$U_{(S,\alpha)} = \begin{cases} V_{\left(T_{i_1} \cup \cdots \cup T_{i_{a_1}} \cup \{x_{j_1},\ldots,x_{j_{b_1}}\}, \alpha(C_{i_1}) \circ \cdots \circ \alpha(C_{i_{a_1}}) \circ \alpha(\{x_{j_1},\ldots,x_{j_{b_1}}\})\right)} & \text{if } \alpha(T_{i_1}), \ldots, \alpha(T_{i_{a_1}}), \\ & \alpha(\{x_{j_1}, \ldots, x_{j_{b_1}}\}) \\ & \text{are consistent,} \\ 0 & \text{otherwise,} \end{cases}$$

where "consistent" here means that no two assignments assign different values to the same variable.

Note that since $|S| \leq r/K$, we know that $|T_{i_1} \cup \cdots \cup T_{i_{a_1}} \cup \{x_{j_1}, \ldots, x_{j_{b_1}}\}| \leq r$, which means that the definition above is valid.

We will now show that each condition in the Lasserre SDP for projection games is satisfied.

- First, we have $||U_{(\emptyset,\emptyset)}|| = ||V_{(\emptyset,\emptyset)}|| = 1$ as desired.
- For every $S_1, S_2, \alpha_1, \alpha_2$ such that $\alpha_1(S_1 \cap S_2) \neq \alpha_2(S_1 \cap S_2)$, if $U_{(S_1,\alpha_1)} = 0$ or $U_{(S_2,\alpha_2)} = 0$, then $\left\langle U_{(S_1,\alpha_1)}, U_{(S_2,\alpha_2)} \right\rangle = 0$. Otherwise, if $U_{(S_1,\alpha_1)} \neq 0$ and $U_{(S_2,\alpha_2)} \neq 0$, we have $U_{(S_1,\alpha_1)} = V_{(S_1',\alpha_1')}$ and $U_{(S_2,\alpha_2)} = V_{(S_2',\alpha_2')}$ for some $S_1', S_2', \alpha_1', \alpha_2'$. Moreover, since $\alpha_1(S_1 \cap S_2) \neq \alpha_2(S_1 \cap S_2)$, we also have $\alpha_1'(S_1' \cap S_2') \neq \alpha_2'(S_1' \cap S_2')$. This implies that $\left\langle U_{(S_1,\alpha_1)}, U_{(S_2,\alpha_2)} \right\rangle = 0$ as desired.
- Consider any $S_1, S_2, S_3, S_4, \alpha_1, \alpha_2, \alpha_3, \alpha_4$ such that $\alpha_1 \circ \alpha_2 = \alpha_3 \circ \alpha_4$. It is obvious that, if $\alpha_1 \circ \alpha_2 = \alpha_3 \circ \alpha_4$ is not consistent (i.e. assignments to constraints and to variables try to set different values to the same variable), then $\left\langle U_{(S_1,\alpha_1)}, U_{(S_2,\alpha_2)} \right\rangle = 0 = \left\langle U_{(S_3,\alpha_3)}, U_{(S_4,\alpha_4)} \right\rangle$. On the other hand, if $\alpha_1 \circ \alpha_2 =$

$\alpha_3 \circ \alpha_4$ is consistent, then let α be the corresponding assignment to the variables and S be the set of all variables involved in S_1, S_2, S_3, S_4. It is obvious to see that $\left\langle U_{(S_1,\alpha_1)}, U_{(S_2,\alpha_2)} \right\rangle = ||V_{(S,\alpha)}||^2 = \left\langle U_{(S_3,\alpha_3)}, U_{(S_4,\alpha_4)} \right\rangle$.

- For every $S_1, S_2, \alpha_1, \alpha_2$, if $U_{(S_1,\alpha_1)} = 0$ or $U_{(S_2,\alpha_2)} = 0$, then $\left\langle U_{(S_1,\alpha_1)}, U_{(S_2,\alpha_2)} \right\rangle = 0$. Otherwise, if $U_{(S_1,\alpha_1)} \neq 0$ and $U_{(S_2,\alpha_2)} \neq 0$, we have $U_{(S_1,\alpha_1)} = V_{(S_1',\alpha_1')}$ and $U_{(S_2,\alpha_2)} = V_{(S_2',\alpha_2')}$ for some $S_1', S_2', \alpha_1', \alpha_2'$, which implies that $\left\langle U_{(S_1,\alpha_1)}, U_{(S_2,\alpha_2)} \right\rangle \geq 0$ as desired.

- For each $x_i \in B$, we have

$$\sum_{\sigma \in [k]} ||U_{(\{x_i\},\sigma)}||^2 = \sum_{j \in [q]} ||V_{(\{x_i\},j)}||^2 = 1$$

as desired.

For each $C_i \in A$, first observe that, since $\sum_{i \in [m]} \sum_{\alpha \in [q]^{T_i}} C_i(\alpha) ||V_{(T_i,\alpha)}||^2 = m$ and $\sum_{\alpha \in [q]^{T_i}} C_i(\alpha) ||V_{(T_i,\alpha)}||^2 \leq 1$, we can conclude that $\sum_{\alpha \in [q]^{T_i}} C_i(\alpha) ||V_{(T_i,\alpha)}||^2 = 1$ for every $i \in [m]$. As a result, we have

$$\sum_{\sigma \in [k]} ||U_{(\{C_i\},\sigma)}||^2 = \sum_{\substack{\alpha \in [q]^{T_i} \\ \text{s.t. } C_i(\alpha)=1}} ||V_{(T_i,\alpha)}||^2 = \sum_{\alpha \in [q]^{T_i}} C_i(\alpha) ||V_{(T_i,\alpha)}||^2 = 1.$$

As a result, we have $\sum_{\sigma \in [k]} ||U_{(v,\sigma)}||^2 = 1$ for every $v \in A \cup B$.

Lastly, we will show that the solution is perfect, i.e. the value of the solution is mK. We can write the value of the solution as

$$\sum_{(u,v) \in E} \sum_{\sigma \in [|C|]} ||U_{(\{u,v\},\{u \to \sigma, v \to \pi_{(u,v)}(\sigma)\})}||^2 = \sum_{(C_i,x_j) \in E} \sum_{\substack{\alpha \in [q]^{T_i} \\ \text{s.t. } C_i(\alpha)=1}} ||U_{(\{C_i,x_j\},\{C_i \to \alpha, x_j \to \alpha(x_j)\})}||^2$$

$$
\begin{aligned}
&= \sum_{i\in[m]} \sum_{\substack{\alpha\in[q]^{T_i} \\ \text{s.t. } C_i(\alpha)=1}} \sum_{x_j\in T_i} ||U_{(\{C_i,x_j\},\{C_i\to\alpha,x_j\to\alpha(x_j)\})}||^2 \\
&= \sum_{i\in[m]} \sum_{\substack{\alpha\in[q]^{T_i} \\ \text{s.t. } C_i(\alpha)=1}} \sum_{x_j\in T_i} ||V_{(T_i,\alpha)}||^2 \\
&= K \sum_{i\in[m]} \sum_{\substack{\alpha\in[q]^{T_i} \\ \text{s.t. } C_i(\alpha)=1}} ||V_{(T_i,\alpha)}||^2 \\
&= K \sum_{i\in[m]} \sum_{\alpha\in[q]^{T_i}} C_i(\alpha)||V_{(T_i,\alpha)}||^2 \\
&= Km.
\end{aligned}
$$

Thus, the defined solution is a perfect solution for the projection games Lasserre SDP as desired. □

By combining Lemma 4.1 and Lemma 4.3 directly, we get the following lemma.

Lemma 4.4. *Let C be a dual code of any linear code of distance at least $D \geq 3$. For every $n, K, \beta, \eta > 0$ such that*

- *n is large enough,*
- *$10 \leq K \leq n^{1/2}$,*
- *$\eta \leq 1/(10^8(\beta K^D)^{2/(D-2)})$,*
- *$n^{\nu-1} \leq 1/(10^8(\beta K^{D+0.75})^{2/(D-2)})$ for some $\nu > 0$,*

with probability $1 - o(1)$, there exists a perfect solution to the $\eta n/(16K)$-th round Lasserre SDP for a projection games instance created by the procedure described in Subsection 4.4.1 from a random MAX K-CSP(C) *over n variables and $m = \beta n$ constraints.*

4.4.3 Soundness

Now, we will show that, if the number of constraints $m = \beta n$ is not too small, then, with high probability, only $O(\ln q/q)$ fraction of the edges in the projection game instance can be satisfied. Note that $O(\ln q/q)$ is almost the best one can hope for since it is easy to see that one can always satisfy at least $1/q$ fraction of edges. The soundness result can be formally stated as follows.

Lemma 4.5. *Let $0 < \rho < 1$ be any constant. If $q \geq K \geq q/2, q \geq 10000/\rho, |C| \leq q^{10}$ and $\beta \geq 100q^{1+\rho}/K$, then the optimal solution of the projection game instance produced by the reduction in Subsection 4.4.1 satisfies at most $1500mK \ln q/(q\rho)$ edges with probability $1 - o(1)$.*

Proof. Let $\mathrm{Y} = \lceil 1/\rho \rceil$ and let $M = q^{1/\mathrm{Y}}$.

To prove this lemma, consider each assignment $\varphi_B : B \to [q]$. We will prove that, with probability at least $1 - \exp(-mK \ln q/(qM))$, there exists no assignment $\varphi_A : A \to [|C|]$ that, together with φ_B, satisfies more than $1500mK \ln q/(q\rho)$ edges.

To show this, consider each vertex C_i in A. For each $l = 0, \ldots, \mathrm{Y} - 1$, define X_i^l as an indicator variable whether there exists $\sigma_a \in \Sigma_A$ such that assigning σ_a to C_i satisfies more than $500KM^l \ln q/q$ edges touching the vertex. In other words, $X_i^l = 0$ if and only if, for each codeword $c = (c_1, \ldots, c_K) \in C$, $(c_1 - b_1^{(i)}, \ldots, c_K - b_K^{(i)})$ agrees with $(\varphi_B(x_{i_1}), \ldots, \varphi_B(x_{i_K}))$ on at most $500KM^l \ln q/q$ coordinates.

We will now prove an upper bound on the probability that $X_i^l = 1$. Consider each codeword $c \in C$. For each $j \in [K]$, $b_j^{(i)}$ is randomly selected from $\mathbb{F}_q$, which implies

that the probability that $c_j - b_j^{(i)} = \varphi_B(x_{i_j})$ is $1/q$. By Chernoff bound, we have

$$\begin{aligned}
&\Pr\left[\sum_{j\in[K]} 1_{c_j - b_j^{(i)} = \varphi_B(x_{i_j})} > 500KM^l \ln q/q\right] \\
&\leq \left(\frac{e(K/q)}{(500KM^l \ln q/q)}\right)^{500KM^l \ln q/q} \\
&< \exp(-500KM^l \ln q/q).
\end{aligned}$$

Hence, we can conclude that

$$\begin{aligned}
\Pr[X_i^l = 1] &= \Pr\left[\exists c \in C, \sum_{j\in[K]} 1_{c_j - b_j^{(i)} = \varphi_B(x_{i_j})} > 500KM^l \ln q/q\right] \\
\text{(Union bound)} &\leq \sum_{c\in C} \Pr\left[\sum_{j\in[K]} 1_{c_j - b_j^{(i)} = \varphi_B(x_{i_j})} > 500KM^l \ln q/q\right] \\
&\leq \sum_{c\in C} \exp(-500KM^l \ln q/q) \\
\text{(Since } |C| \leq q^{10}) &\leq q^{10} \cdot \exp(-500KM^l \ln q/q) \\
\text{(Since } q \geq K \geq q/2 \text{ and } q \geq 10000/\rho) &\leq \exp(-400KM^l \ln q/q).
\end{aligned}$$

Now, observe that, from definition of X_i^l, we can conclude that, if l is the maximum value of l such that $M_i^l = 1$, then at most $500KM^{l+1} \ln q/q$ edges touching C_i that are satisfied. In other words, the number of edges touching C_i that is satisfied is at most

$$\max\{500K \ln q/q, \max_{l=0,\ldots,Y-1}\{X_i^l 500KM^{l+1} \ln q/q\}\}.$$

As a result, for each φ_B, the maximum number of satisfied edges is at most

$$\sum_{i\in[m]} \max\{500K \ln q/q, \max_{l=0,\ldots,Y-1}\{X_i^l 500KR^{l+1} \ln q/q\}\}$$

$$\leq \sum_{i\in[m]} \left(500K \ln q/q + \sum_{l=0}^{\mathrm{Y}-1} X_i^l 500KM^{l+1} \ln q/q \right)$$
$$= 500mK \ln q/q + \sum_{l=0}^{\mathrm{Y}-1} (500KM^{l+1} \ln q/q) \cdot |\{i \in [m] \mid X_i^l = 1\}|$$

Since each C_i is sampled independently of each other, we can again use Chernoff bound to arrive at the following inequality.

$$\begin{aligned}
\Pr[|\{i \in [m] \mid X_i^l = 1\}| > m/M^{l+1}] &\leq \left(\frac{e(m \cdot \exp(-400KM^l \ln q/q))}{m/M^{l+1}} \right)^{m/M^{l+1}} \\
&= (M^{l+1} \cdot e \cdot \exp(-400KM^l \ln q/q))^{m/M^{l+1}} \\
(\text{Since } K \geq q/2 \text{ and } q \geq 10000/\rho) &\leq (\exp(-300KM^l \ln q/q))^{m/M^{l+1}} \\
&= \exp(-300mK \ln q/(qM)).
\end{aligned}$$

From this and the union bound, we can conclude that, with probability at least $1 - \mathrm{Y} \exp(-300mK \ln q/(qM)) \geq 1 - \exp(-mK \ln q/(qM))$, we have $|\{i \in [m] \mid X_i^l = 1\}| \leq m/M^{l+1}$ for all $l = 0, \ldots, \mathrm{Y} - 1$.

Observe that, if $|\{i \in [m] \mid X_i^l = 1\}| \leq m/M^{l+1}$ for all $l = 0, \ldots, \mathrm{Y} - 1$, then we have

$$\begin{aligned}
&500mK \ln q/q + \sum_{l=0}^{\mathrm{Y}-1} (500KM^{l+1} \ln q/q) \cdot |\{i \in [m] \mid X_i^l = 1\}| \\
&\leq 500mK \ln q/q + \sum_{l=0}^{\mathrm{Y}-1} (500KM^{l+1} \ln q/q) \cdot (m/M^{l+1}) \\
&= 500mK \ln q/q + \sum_{l=0}^{\mathrm{Y}-1} (500mK \ln q/q) \\
&= (500mK \ln q/q)(\mathrm{Y} + 1) \\
&\leq 1500mK \ln q/(q\rho).
\end{aligned}$$

This means that the maximum number of edges satisfied for φ_B is at most $1500mK \ln q/(q\rho)$.

There are $q^{|B|} = q^n$ possible different φ_B's. Using the union bound, we can conclude that the probability that every assignment to the vertices satisfies at most $1500mK \ln q/(q\rho)$ edges is at least

$$\begin{aligned} 1 - q^n \exp(-mK \ln q/(qM)) &= 1 - \exp(n \ln q - n\beta K \ln q/(qM)) \\ (\text{Since } \beta \geq \frac{100(qM)}{K}) &\leq 1 - \exp(n \ln q - 100n \ln q) \\ &= 1 - o(1), \end{aligned}$$

which completes our proof for the lemma. □

4.4.4 Proofs of the Two Main Theorems

Before we proceed to prove Theorem 4.1 and Theorem 4.2, we will start by combining the vector completeness result and the soundness result to get the following theorem that we will use as a basis to prove the two main theorems.

Theorem 4.3. *Let $0 < \rho < 1$ be any constant. For all n, q large enough such that $q \leq n^{10}$ and for each integer $3 \leq D \leq 10$, there exists a projection game instance of at most $n^{1+100\rho}$ vertices and alphabets of size q^{D-1} that demonstrates a gap of value $\Omega(q^{1-o(1)})$ for Lasserre SDP after $\Omega(n/(q^{(2D+2\rho)/(D-2)+1}))$ rounds of the hierarchy.*

Proof. Let C be the dual code of the code from Lemma 4.2 with $3 \leq D \leq 10$. We now have $K = q - 1$ and $|C| = q^{D-1}$. Pick β to be $100q^{1+\rho}/K$ and η to be $1/(10^8(\beta K^D)^{2/(D-2)})$.

The instance we will use is just the instance created by the reduction present in Subsection 4.4.1 from a random MAX K-CSP(C) instance.

Let $R = \eta n/(16K)$. We know from Lemma 4.4 that, after R rounds of the hierarchy, there still exists a complete solution (of value mK) to the Lasserre SDP of the instance. At the same time, we also know from Lemma 4.5 that, with probability $1 - o(1)$, every assignment can satisfy at most $1500mK \ln q/(q\rho)$ edges. This means that, after R rounds, we have a gap ratio of $mK/(1500mK \ln q/(q\rho)) = \Omega(q/\ln q) = \Omega(q^{1-o(1)})$ as desired.

Moreover, R can be written as

$$\begin{aligned}
R &= \eta n/(16K) \\
&= \frac{1}{10^8(\beta K^D)^{2/(D-2)}} \cdot \frac{n}{16K} \\
&= \Omega\left(\frac{1}{(\beta K^D)^{2/(D-2)}} \cdot \frac{n}{K}\right) \\
(\text{Since } \beta = 100q^{1+\rho}/K) &= \Omega\left(\frac{1}{(q^{1+\rho}K^{D-1})^{2/(D-2)}} \cdot \frac{n}{K}\right) \\
(\text{Since } K = q-1) &= \Omega\left(\frac{1}{(q^{D+\rho})^{2/(D-2)}} \cdot \frac{n}{q}\right) \\
&= \Omega\left(\frac{n}{q^{(2D+2\rho)/(D-2)+1}}\right).
\end{aligned}$$

Lastly, note that the number of vertices is $m + n = (\beta + 1)n \leq (400q^\rho)n = n^{1+100\rho}$ and the alphabets are of size $|C| = q^{D-1}$. □

Now, we are ready to prove the two main theorems by choosing the right N, q, D for Theorem 4.3.

Proof of Theorem 4.1. Recall that N is defined to be the number of vertices times the alphabets size for a projection game instance. This means that $N = n^{1+100\rho}q^{D-1}$. We can rewrite the number of rounds R of Lasserre SDP relaxation from Theorem 4.3 as

$$R = \Omega\left(\frac{n}{q^{(2D+2\rho)/(D-2)+1}}\right)$$

$$= \Omega\left(\frac{n^{1+100\rho}q^{D-1}}{n^{100\rho}q^{(2D+2\rho)/(D-2)+D}}\right)$$
$$= \Omega\left(\frac{N}{N^{100\rho}q^{2D/(D-2)+D+2\rho/(D-2)}}\right)$$

We select $D = 4$ to minimize $2D/(D-2) + D$, which implies that

$$R = \Omega\left(\frac{N}{N^{100\rho}q^{8+\rho}}\right).$$

By pick q to be $N^{1/8-\varepsilon/2}$, we get the gap of $\Omega(N^{1/8-\varepsilon/2-o(1)}) \geq N^{1/8-\varepsilon}$ if N is big enough. Furthermore, by picking ρ to be $\varepsilon/1000$, we have

$$R = \Omega\left(\frac{N}{N^{\varepsilon/10}N^{(1/8-\varepsilon/2)(8+\varepsilon/1000)}}\right)$$
$$= \Omega\left(N^{\varepsilon/4}\right)$$

This is $N^{\Omega(\varepsilon)}$ when N is big enough, which completes the proof of this theorem. □

Lastly, the proof of Theorem 4.2 can be seen below.

Proof of Theorem 4.2. Using Theorem 4.3 with $D = 3$, the alphabets size is equal to $q^{D-1} = q^2$. In addition, by picking $\rho = \varepsilon/1000$ we have $N = n^{1+\varepsilon/10}q^2$. Moreover, we can rewrite the number of rounds R of Lasserre SDP relaxation from Theorem 4.3 as

$$R = \Omega\left(\frac{n}{q^{(2D+2\rho)/(D-2)+1}}\right)$$
$$= \Omega\left(\frac{n}{q^{9+2\rho}}\right)$$

$$= \Omega\left(\frac{n^{1+\varepsilon/10}q^2}{n^{\varepsilon/10}q^{11+\varepsilon/500}}\right)$$
$$= \Omega\left(\frac{N}{N^{\varepsilon/10}q^{11+\varepsilon/500}}\right)$$

Select q to be $N^{\varepsilon/40}$. The number of rounds is now $\Omega(N^{1-\varepsilon/2})$, which is larger than $N^{1-\varepsilon}$ when N is big enough. Moreover, the gap is $\Omega(q^{1-o(1)}) = \Omega(N^{\varepsilon/40-o(1)}) = N^{\Omega(\varepsilon)}$, which concludes our proof. □

4.5 Note on DENSEST k-SUBGRAPH

It is not hard to see that the technique that we use in Lemma 4.5, where we consider finer boundaries for "poorly satisfied" predicates instead of categorizing every predicate into two categories as [BCV$^+$12] did, can be extended to work for DENSEST k-SUBGRAPH as well. This leads to a better exponent of $1/14 - \varepsilon$ after $n^{\Omega(\varepsilon)}$ rounds of the Lasserre relaxation of DENSEST k-SUBGRAPH instead of that of $2/53 - \varepsilon$ in [BCV$^+$12]. For a full proof of this result, please refer to Appendix A.2.

5 Projection Games on Planar Graph

As the strong PCP Theorem (Theorem 1.1) shows, LABEL COVER is NP-hard to approximate even to within subconstant ε fraction. Does LABEL COVER remain as hard when we consider special kinds of graphs?

In recent years there has been much interest in optimization problems over *planar graphs*. These are graphs that can be embedded in the plane without edges crossing each other. Many optimization problems have very efficient algorithms on planar graphs.

We show that any projection game where the underlying graph G is planar is easy to approximate by providing the following PTAS:

Theorem 5.1. *There is a polynomial time approximation scheme for projection games on planar graphs that runs in time $N^{O(1/\varepsilon)}$.*

The PTAS works via Baker's approach [Bak94] of approximating the graph by a graph with constant tree-width. Note here that the PTAS presented here is a bit different from our PTAS presented ealier in our paper ([MM13]) but the key idea is still the same; in [MM13], we used a high-level framework from [Kle05], which only yields PTAS for satisfiable instances of projection games on planar graph. Here,

however, we use the traditional Baker's approach, which works for nonsatisfiable instances as well.

We also show that this is the best running time we can get for a PTAS assuming the exponential time hypothesis (ETH) as stated below. The main idea of the proof is a reduction from GRID TILING problem introduced by Marx [Mar07, Mar12].

Theorem 5.2. *If ETH holds, then there is no PTAS for projection games on planar graphs running in time* $2^{O(1/\varepsilon)^{\gamma}} N^{O(1/\varepsilon)^{1-\delta}}$ *for any constants* $\gamma, \delta > 0$.

In addition, we give a subexponential-time exact algorithm for deciding whether a projection game on planar graph is satisfiable using divide-and-conquer technique based on a planar separator theorem of Lipton and Tarjan [LT79] and prove a running time lower bound that matches the running time of the algorithm. These results are stated below.

Theorem 5.3. *There exists an* $k^{O(\sqrt{n})}$*-time algorithm for deciding whether a projection game on planar graph with n vertices and alphabets of size k is satisfiable.*[1]

Theorem 5.4. *If there exists an algorithm that can decide whether a projection game on planar graph with n vertices and alphabets of size k is satisfiable and runs in time* $f(n)k^{o(\sqrt{n})}$ *for some function* f*, then ETH fails.*

[1]The algorithm can easily be modified to returns the exact value of the game as well.

5.1 Solving Projection Games on Planar Graphs

In this section, we present an exact algorithm for projection games on planar graphs that runs in $k^{O(\sqrt{n})}$ time and prove a tight lower bound for the running time.

5.1.1 Exact Algorithm for Projection Games on Planar Graphs

We will now give a subexponential-time exact algorithm for projection games on planar graphs based on a divide-and-conquer approach and prove Theorem 5.3.

Proof of Theorem 5.3. The algorithm is simple. Given the graph $G = (V, E)$, we first find a subset S of vertices of size $O(\sqrt{|V|})$ that $V - S = V_1 \cup V_2$ where $V_1 \cap V_2 = \emptyset$, there are no edges between V_1 and V_2, and $|V_1|, |V_2| \leq \frac{2}{3}|V|$. Due to a planar separator theorem, this subset can be found in linear time [LT79].

We then enumerate through all the possible assignments of S. For each assignment, we solve the problem for V_1 and V_2 separately, i.e., whether there exists an assignment of V_1 that satisfies every edges within $V_1 \cup S$ and similarly for V_2.

A pseudocode for the procedure can be found below. Note that SOLVE takes in three inputs: V', the concerned subset of vertices, $\phi : V \to \Sigma_A \cup \Sigma_B \cup \{\text{Null}\}$, the partial assignment so far, and G, the original graph. It returns True if and only if there is an assignment on V' that satisfies every edge within V' and between V' and vertices assigned in ϕ. Finally, SOLVE (V, $\emptyset$, G = (V, E)) tells us whether the projection game instance is satisfiable.

SOLVE (V', ϕ, $G = (V, E)$)

1 $S \leftarrow$ a mentioned separator for the graph induced by V'
2 $V_1, V_2 \leftarrow$ sets of vertices S separates V' into.
3 **for** each assignment ϕ' of all elements of S
4 $\quad$ *agree* $\leftarrow$ True
5 $\quad$ **for** each $v \in S$
6 $\quad\quad$ $\phi(v) \leftarrow \phi'(v)$
7 $\quad$ **for** each $(u, v) \in E$
8 $\quad\quad$ **if** $\phi(u), \phi(v) \neq$ Null and $\pi_{(u,v)}(\phi(u)) \neq \phi(v)$
9 $\quad\quad\quad$ *agree* $\leftarrow$ False
10 $\quad$ **if** *agree* is True and SOLVE (V_1, ϕ, G), SOLVE (V_2, ϕ, G) are True
11 $\quad\quad$ **return** True
12 **return** False

Suppose that m is the size of V' and n is the size of V. It is not hard to see that the running time $T(m)$ of this algorithm satisfies the following recurrence relation:

$$\begin{aligned} T(m) &= poly(n)k^{|S|}(T(|V_1|) + T(|V_2|)) \\ &= poly(n)k^{O(\sqrt{m})}(T(|V_1|) + T(|V_2|)) \end{aligned}$$

Since $|V_1| + |V_2| < V'$ and $|V_1|, |V_2| \leq \frac{2m}{3}$, from the master theorem, we can conclude that $T(n) = O((poly(n)^{O(\log n)})k^{O(\sqrt{n})})$. This is the running time of SOLVE (V, NULL, $G = (V, E)$), which gives us the answer whether the original projection game is satisfiable. Note that NULL represents the empty partial assignment.

For projection games, if $k = 1$, then they can be solved trivially, which means that we can assume that $k \geq 2$. This assumption implies that $poly(n)^{O(\log n)} = O(k^{\sqrt{n}})$. Hence, we can conclude that $T(n) = k^{O(\sqrt{n})}$ as desired. □

5.1.2 Exact Algorithm Running Time Lower Bound for Projection Games on Planar Graphs

In this section, we will present a reduction from GRID TILING defined in [Mar07, Mar12] to projection games on planar graph. Then, we use a running time lower bound mentioned in [Mar12] to derive Theorem 5.4.

We will start by stating the definition of GRID TILING and its running time lower bound from [Mar12].

GRID TILING **(Decision)**

The GRID TILING problem (decision version), which we will reduce from, can be defined as follows.

> GRID TILING **(Decision)**
>
> INPUT: Positive integers $\tilde{k}, \tilde{n}$ and sets $S_{i,j} \subseteq [\tilde{n}] \times [\tilde{n}]$ for each $i, j = 1, \ldots, \tilde{k}$.
>
> GOAL: Determine whether we can select $s_{i,j} \in S_{i,j}$ for every $i, j = 1 \ldots, \tilde{k}$ such that, for each i, $(s_{i,j})_1$'s are equal for all $j = 1, \ldots, \tilde{k}$, and, for each j, $(s_{i,j})_2$'s are equal for all $i = 1, \ldots, \tilde{k}$.

Note here that $(s_{i,j})_1$ represents the value in the first coordinate of $s_{i,j}$. Similar notations in this section are defined in similar manners.

By reduction from CLIQUE, Marx proved the following lemma [Mar07]. The extracted proof can be found in Appendix A.3[2].

[2]The reduction is shown in this thesis because, even though the lemma is stated explicitly in [Mar12] where Marx attributed the proof to [Mar07], the complete proof cannot be found in [Mar07].

Lemma 5.1 ([Mar07]). *If there exists an algorithm that solves* GRID TILING *in* $g(\tilde{k})\tilde{n}^{o(\tilde{k})}$ *for any function* g*, then ETH fails.*

Reduction from GRID TILING **(Decision) to Planar Projection Games**

We will now prove Theorem 5.4 by giving a reduction from GRID TILING as follows.

Proof. In order to prove the theorem, due to Lemma 5.1 it is enough to show a polynomial-time reduction from a GRID TILING instance to a projection game instance on planar graph where the alphabets are of size $O(\tilde{n}^2)$ and the number of vertices n is $O(\tilde{k}^2)$.

The projection game instance can be defined as follows:

- Let A be a set containing $\tilde{k}^2$ vertices; let us call the vertices $a_{i,j}$ for all $i, j \in [\tilde{k}]$.
- Let B be a set containing $2\tilde{k}^2 - 2\tilde{k}$ vertices; let us call the vertices $b_{i+0.5,j}$ for all $i \in [\tilde{k}-1], j = [\tilde{k}]$ and $b_{i,j+0.5}$ for all $i \in [\tilde{k}], j = [\tilde{k}-1]$.
- Let E be $\{(a_{x,y}, b_{z,t}) \in A \times B \mid |x - z| + |y - t| = 0.5\}$.
- Let Σ_A be $[\tilde{n}] \times [\tilde{n}]$.
- Let Σ_B be $[\tilde{n}] \cup \{\blacksquare, \blacklozenge\}$.
- The projections π_e's where $e = (a_{x,y}, b_{z,t})$ can be defined as follows.

$$\pi_e(s) = \begin{cases} s_1 & \text{if } s \in S_{x,y} \text{ and } x = z, \\ s_2 & \text{if } s \in S_{x,y} \text{ and } y = t, \\ \blacksquare & \text{if } s \notin S_{x,y}, x \leq z \text{ and } y \leq t, \\ \blacklozenge & \text{if } s \notin S_{x,y}, x \geq z \text{ and } y \geq t, \end{cases}$$

for all $(a_{x,y}, b_{z,t}) \in E$ and for all $s \in \Sigma_A$.

For an illustration of the reduction, please refer to Figure 5.1 below.

$$\tilde{n} = 2, \tilde{k} = 2,$$
$$S_{1,1} = \{(1,1),(2,2)\},$$
$$S_{1,2} = \{(1,2)\},$$
$$S_{2,1} = \{(2,1)\},$$
$$S_{2,2} = \{(1,1)\}.$$

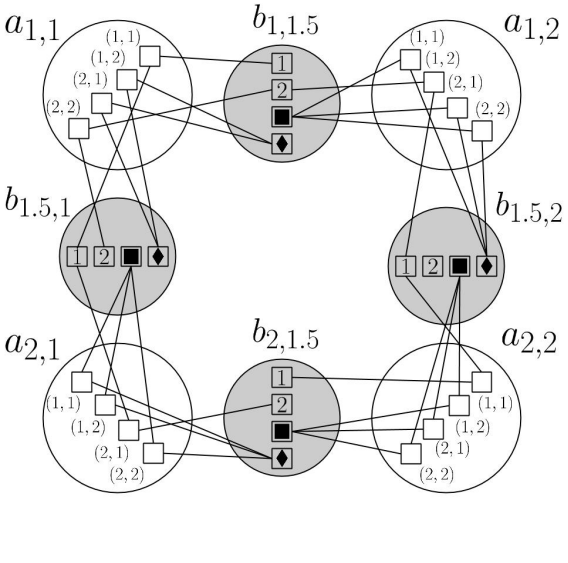

Figure 5.1: **An Example of the Reduction from** GRID TILING **to** LABEL COVER**.** The original GRID TILING instance is shown on the left and the resulting projection game is shown on the right. This reduction is used both in the proof of Theorem 5.4 and Theorem 5.2. Note that, in the projection game illustration, circles represent vertices of the underlying graph whereas squares represent alphabet symbols. The shaded circles are the vertices in B and the white circles are the vertices in A. Each connecting line between two alphabet symbols means that the corresponding projection maps the alphabet symbol from a vertex in A to the alphabet symbol from a vertex in B.

First, note that $(A \cup B, E)$ is planar since, if we place $a_{x,y}$ on the plane at (x, y) for all $a_{x,y} \in A$ and place $b_{z,t}$ at (z, t) for all $b_{z,t} \in B$, then we can see that no edges intersect each other.

Next, we will show that the defined projection game is satisfiable if and only if the original GRID TILING instance is a yes instance.

($\Rightarrow$) Suppose that the projection game is satisfiable. Let $\varphi_A : A \to \Sigma_A$ and $\varphi_B : B \to \Sigma_B$ be the assignments that satisfy every edge. First, consider $b_{i+0.5,j} \in B$ for any $i \in [\tilde{k} - 1], j \in [\tilde{k}]$.

Observe that $\pi^{-1}_{(a_{i,j}, b_{i+0.5,j})}(\blacklozenge) = \emptyset$ and $\pi^{-1}_{(a_{i+1,j}, b_{i+0.5,j})}(\blacksquare) = \emptyset$. This implies that

$\varphi_B(b_{i+0.5,j}) \neq \blacklozenge, \blacksquare$. From this, along with the definition of $\pi_{(a_{i,j},b_{i+0.5,j})}$, we can conclude that $\varphi_A(a_{i,j}) \in S_{i,j}$ and $\varphi_A(a_{i,j})_2 = \varphi_B(b_{i+0.5,j})$. Similarly, we have $\varphi_A(a_{i+1,j}) \in S_{i+1,j}$ and $\varphi_A(a_{i+1,j})_2 = \varphi_B(b_{i+0.5,j})$.

Thus, we can conclude that $\varphi_A(a_{i,j}) \in S_{i,j}$ for every $i, j \in [\tilde{k}]$ and that, for each $i \in [\tilde{k}]$, $\varphi_A(a_{i,j})_2$'s are equal for every $j \in [\tilde{k}]$.

Similarly, we also have, for each $j \in [\tilde{k}]$, $\varphi_A(a_{i,j})_1$'s are equal for every $i \in [\tilde{k}]$.

As a result, by assigning $s_{i,j} = \varphi_A(a_{i,j})$, we know that the GRID TILING instance is a yes instance.

($\Leftarrow$) Suppose that the GRID TILING instance is a yes instance, i.e., there exists $s_{i,j} \in S_{i,j}$ for all $i, j \in [\tilde{k}]$ such that, for each i, $(s_{i,j})_1$'s are equal for all $j \in [\tilde{k}]$, and, for each j, $(s_{i,j})_2$'s are equal for all $i \in [\tilde{k}]$.

By simply picking $\varphi_A(a_{i,j}) = s_{i,j}$ for every $a_{i,j} \in A$, $\varphi_B(b_{i+0.5,j}) = (s_{1,j})_2$ for every $b_{i+0.5,j} \in B$ and, $\varphi_B(b_{i,j+0.5}) = (s_{i,1})_1$ for every $b_{i,j+0.5} \in B$, we can conclude that the projection game is satisfiable.

As a result, if there is an algorithm that decides whether the projection game is satisfiable in time $f(n)k^{o(\sqrt{n})}$, then we can also decide on GRID TILING in time $f(3\tilde{k}^2 - 3\tilde{k})(\tilde{n}^2)^{o\left(\sqrt{3\tilde{k}^2-3\tilde{k}}\right)} + poly(\tilde{n}, \tilde{k})$, which contradicts Lemma 5.1. (Note here that $poly(\tilde{n}, \tilde{k})$ term comes from the time using in the reduction process.) Thus, our proof for Theorem 5.4 is complete. □

5.2 Approximating Projection Games on Planar Graphs

In this section, we provide a PTAS for projection games on planar graphs and prove the matching running time lower bound for it.

5.2.1 PTAS for Projection Games on Planar Graphs

We use the standard Baker's technique for finding PTAS for problems on planar graphs [Bak94] to construct one for instances of the projection games problem. In this subsection, we will sometimes borrow terminologies from [Kle05], which presents a subsequent work on the approach.

Although not realized in the original paper by Baker, the technique relies on the concept of *treewidth*, which we will review the definition in the next subsection. In this perspective, Baker's technique constructs an algorithm based on two main steps:

1. **Thinning Step:** Given a graph $G = (V, E)$, partition E into subsets $S_1, \ldots, S_h$ such that, for each i, we obtain a graph with bounded treewidth when all edges in S_i are deleted from the original graph.
2. **Dynamic Programming Step:** For each $i = 1, \ldots, h$, use dynamic programming to solve the problem on $(V, E - S_i)$, which has bounded treewidth. Then, output the best solution among these h solutions.

Tree Decomposition and Treewidth

Before we proceed to the algorithm, we first define *tree decomposition*. A *tree decomposition* of a graph $G = (V, E)$ is a collection of subsets $\{B_1, B_2, \ldots, B_n\}$ and a tree T whose nodes are B_i such that

1. $V = B_1 \cup B_2 \cup \cdots \cup B_n$.
2. For each edge $(u, v) \in E$, there exists B_i such that $u, v \in B_i$.
3. For each B_i, B_j, if they have an element v in common. Then v is in every subset along the path in T from B_i to B_j.

The *width* of a tree decomposition $(\{B_1, B_2, \ldots, B_n\}, T)$ is the largest size of $B_1, \ldots, B_n$ minus one. The *treewidth* of a graph G is the minimum width across all possible tree decompositions.

Thinning

Even though a planar graph itself does not necessarily have a bounded treewidth, it is possible to delete a "small" set of edges from the graph to obtain a graph with bounded treewidth; by "small", we do not refer to the size, but refer to the change in optimal solution for the input LABEL COVER instance after we delete the edges from the set.

To achieve this, we partition E into h sets such that, when deleting all edges from each set, the treewidth is bounded linearly on h. Later on, we will show that, for at least one of the sets, deleting all the edges from it affects the optimal solution of projection games by at most a factor of $1 - 1/h$.

Baker implicitly proved the following partitioning lemma in her paper [Bak94]. For a more explicit formulation, please see [Epp00]. [3]

Lemma 5.2 ([Bak94])**.** *For any planar graph $G = (V, E)$ and integer h, there is a $O(h(|V| + |E|))$-time algorithm returns an edge-sets $S_1, \ldots, S_h$ and, for each $i = 1, \ldots, h$, a tree decompositions of H_i of $(V, E - S_i)$ having width at most $O(h)$.*

Next, we will show that, for at least one of the S_i's, removing all the edges from S_i affects the optimal solution by at most a factor of $1 - 1/h$:

[3] In both [Bak94] and [Epp00], the vertices, not edges, are partitioned. However, it is obvious that the lemma works for edges as well.

Lemma 5.3. *For any projection games instance on graph $G = (V, E)$ and any partition $(S_1, \ldots, S_h)$ of E, there exists $i \in \{1, \ldots, h\}$ such that the projection game instance resulted from removing all the edges in S_i has the optimal solution that is within $1 - 1/h$ factor of the optimal of the original instance.*

Proof. Suppose that E_{sat} is the set of all the edges satisfied by the optimal assignment ϕ_{OPT} of the original instance. From the pigeonhole principle, there exists at least one $i \in \{1, \ldots, h\}$ such that $|S_i \cap E_{sat}| \leq |E_{sat}|/h$. Since ϕ_{OPT} still satisfies all the edges in $E - (S_i \cap E)$ in the projection game instance induced by $(V, E - S_i)$, we can conclude that the optimal assignment to this instance satisfies at least $(1 - 1/h)|E_{sat}|$ edges, which concludes the proof of this lemma. □

For the purpose of our algorithm, we select $h = 1 + \frac{1}{\varepsilon}$, which ensures that the treewidth of $(V, E - S_i)$ is at most $O(h) = O(1/\varepsilon)$ for each $i = 1, \ldots, h$. Moreover, from the above lemma, we can conclude that, for at least one such i, the optimal solution of the projection game instance induced on $(V, E - S_i)$ satisfies at least $1 - 1/h = 1/(1 + \varepsilon)$ times as many edges satisfied by the optimal solution of the original instance.

Dynamic Programming

In this section, we will present a dynamic programming algorithm that solves the projection game problem in a bounded treewidth bipartite graph $G' = (A', B', E')$, given its tree decomposition $(\{B_1, \ldots, B_n\}, T)$ with a bounded width w.

The algorithm works as follows. We use depth-first search to traverse the tree T. Then, at each node B_i, we solve the problem concerning only the subtree of T starting at B_i.

At B_i, we consider all possible assignments $\phi : B_i \to (\Sigma_A \cup \Sigma_B)$ of B_i. For each assignment ϕ and for each edge $(u, v) \in E'$ such that both u, v are in B_i, we check whether the condition $\pi_{(u,v)}(\phi(u)) = \phi(v)$ is satisfied or not. If not, we conclude that this assignment does not satisfy all the edges. Otherwise, we check that the assignment ϕ works for each subtree of T starting at each child B_j of B_i in T; this result was memoized when the algorithm solved the problem at B_j.

Our algorithm will fill in a two-dimensional array a so that $a[B_i][\phi]$ is true if and only if an assignment ϕ is possible considered only a subtree rooted at B_i. A pseudocode for the algorithm is shown below.

DYNAMIC-PROGRAMMING (B_i)

```
1  for each children B_j of B_i in T
2      DYNAMIC-PROGRAMMING(B_j)
3  for each assignment φ of all elements of B_i
4      a[B_i][φ] ← True
5      for each edge (u, v) ∈ E'
6          if u, v ∈ B_i and π_(u,v)(φ(u)) ≠ φ(v)
7              a[B_i][φ] ← False
8      for each children B_j of B_i in T
9          agree ← False
10         for each assignment φ' of B_j
11             if a[B_j][φ'] is True and φ(x) = φ'(x) for all x ∈ B_i ∩ B_j
12                 agree ← True
13         if agree is False
14             a[B_i][φ] ← False
```

We start by calling the function with the root of T as an input.

To analyze the runtime of the algorithm, first observe that there are $(|\Sigma_A| + |\Sigma_B|)^{|B_i|}$ possible assignments for B_i. Since the width of this tree decomposition is at most w, we can conclude that $|B_i| \leq w + 1$. Thus, for each B_i, there are at most $(|\Sigma_A| + |\Sigma_B|)^{w+1}$ assignments for it.

For each assignment ϕ, we check all the edges in the original graph whether $\pi_{(u,v)}(\phi(u)) = \phi(v)$ or not. There are $|E'|$ such edges to check.

Moreover, for each edge (B_i, B_j) in T, we need to check whether the assignment in B_i agrees with any feasible assignment in B_j or not. This means that we perform at most $(|\Sigma_A| + |\Sigma_B|)^{2w+2}$ of these checks. In addition, in each check, we check that assignments for B_i and B_j agrees for all vertices in $B_i \cap B_j$ or not. This takes at most $O(|B_i| + |B_j|) = O(w + 1) = O(w)$ time.

As a result, the overall runtime for this algorithm is $O(n|E'|(|\Sigma_A| + |\Sigma_B|)^{w+1} + nw(|\Sigma_A| + |\Sigma_B|)^{2w+2})$.

Please note that, once DYNAMIC-PROGRAMMING finishes, we can similarly use depth-first search one more time to find an assignment ϕ that satisfies the whole graph. The pseudo-code for doing this is shown below. ϕ is first initiated to be null. Then the procedure ANSWER is run on the root of T. After the program finishes, ϕ will get filled with an assignment that satisfies all the edges in E'.

ANSWER (B_i)

```
1  for each assignment φ' of all elements of B_i
2      if a[B_i][φ'] is True
3          agree ← True
4          for each v ∈ B_i
5              if φ(v) ≠ Null and φ(v) ≠ φ'(v)
6                  agree ← False
7          if agree is True
8              for each v ∈ B_i
9                  φ(v) ← φ'(v)
10 for each children B_j of B_i in T
11     ANSWER(B_j)
```

It is easy to see that the ANSWER procedure runs in $O(nw(|\Sigma_A| + |\Sigma_B|)^{w+1})$ time which is asymptotically smaller than the runtime of the DYNAMIC-PROGRAMMING procedure.

Summary

We use the dynamic programming algorithm presented above to solve the projection game instance induced by the graph $G_i = (V, E - S_i)$ for each $i = 1, \ldots, h$. We then output the solution that satisfies most edges among all i's. As shown earlier, since we select h to be $1 + \frac{1}{\varepsilon}$, at least one of the solution found on G_i's satisfies at least $\frac{1}{1+\varepsilon}$ times as many edges as the optimal solution to the original instance, which means that our algorithm is indeed an $(1 + \varepsilon)$-approximation algorithm.

Moreover, since we use the dynamic programming algorithm h times, the running time of the algorithm is $O(h(n|E'|(|\Sigma_A| + |\Sigma_B|)^{w+1} + nw(|\Sigma_A| + |\Sigma_B|)^{2w+2})) = (nk)^{O(w+h+5)} = (nk)^{O(h)} = (nk)^{O(1/\varepsilon)}$.

This gives us polynomial-time approximation scheme for satisfiable instances of the projection game problem as desired.

5.2.2 PTAS Running Time Lower Bound for Projection Games on Planar Graphs

We devote this subsection to prove Theorem 5.2. This theorem means that the PTAS presented previously is the best we can do.

The proof is again a reduction from GRID TILING problem. However, in this section, we will use an optimization version of the problem instead of the decision one defined earlier.

GRID TILING **(Optimization)**

The GRID TILING problem (optimization version) can be defined as follows.

GRID TILING **(optimization)**

INPUT: Positive integers $\tilde{k}, \tilde{n}$ and sets $S_{i,j} \subseteq [\tilde{n}] \times [\tilde{n}]$ for each $i, j = 1, \ldots, \tilde{k}$.

GOAL: Select $s_{i,j} \in S_{i,j} \cup \{\bigstar\}$ for every $i, j = 1 \ldots, \tilde{k}$ such that

- for every $i \in [\tilde{k}], j \in [\tilde{k}-1]$, if $s_{i,j}, s_{i,j+1} \neq \bigstar$, then $(s_{i,j})_1 = (s_{i,j+1})_1$, and,
- for every $i \in [\tilde{k}-1], j \in [\tilde{k}]$, if $s_{i,j}, s_{i+1,j} \neq \bigstar$, then $(s_{i,j})_2 = (s_{i+1,j})_2$

that maximizes the number of $(i, j) \in [\tilde{k}] \times [\tilde{k}]$ such that $s_{i,j} \neq \bigstar$.

Note here that the decision version mentioned above is just to decide whether the answer to the optimization problem is $\tilde{k}^2$.

In [Mar07], a running time lower bound for approximating the optimization version of GRID TILING is shown by a reduction from 3SAT. We extract the result from Lemma 2.1 in [Mar07] below.

Lemma 5.4 ([Mar07])**.** *Let $\tilde{k}$ be any positive integer. There is a reduction from an instance Φ of* 3SAT *having m clauses to an instance of* GRID TILING *with $\tilde{n} = 3^{\lceil m/\tilde{k} \rceil}$ such that*

- *If Φ is satisfiable, then the optimum of* GRID TILING *is $\tilde{k}^2$ (i.e. a yes instance in the decision version),*
- *For every $1 > \alpha > 0$, if Φ is not α-satisfiable (i.e. no assignments satisfies at least αm clauses), then the optimum of* GRID TILING *is at most $\tilde{k}^2 - \tilde{k}(1-\alpha)/2 + 1$,*

and the reduction runs in polynomial time of the size of the GRID TILING *instance.*

Reduction from GRID TILING **(Optimization) to Planar Projection Games**

In this subsection, we will show a reduction from 3SAT to a projection game on planar graph. This is the key in proving the running time lower bound for PTAS for projection games on planar graphs.

Lemma 5.5. *Let $\tilde{k}$ be any positive integer. There is a reduction from an instance Φ of* 3SAT *having m clauses to a projection game on planar graph with $n = 3\tilde{k}^2 - 2\tilde{k}$ vertices, $4\tilde{k}^2 - 4\tilde{k}$ edges, and alphabets of size $k = 3^{2\lceil m/\tilde{k} \rceil}$, such that*

- *If Φ is satisfiable, then the projection game can be fully satisfied,*
- *For every $1 > \alpha > 0$, if Φ is not α-satisfiable, then at most $4\tilde{k}^2 - 4\tilde{k} - \frac{1}{2}\left(\tilde{k}(1-\alpha)/2 - 1\right)$ edges of the projection game can be satisfied,*

and the reduction runs in polynomial time of the size of the projection game.

Proof. The reduction proceeds as follows. Starting from a 3SAT formula Φ having m clauses, we use the reduction procedure in Lemma 5.4 to reduce it to a GRID TILING instance. We then reduce the GRID TILING instance into a planar projection game instance the exact same way as we did in the proof of Theorem 5.4 in Section 5.1.2. It is clear that the reduction takes polynomial time in the size of the projection game.

Observe that, from Lemma 5.4, if Φ is satisfiable, then the GRID TILING instance is a yes instance. As a result, from the proof of Theorem 5.4, we can also conclude that the projection game instance must be satisfiable.

Next, we will show that, if Φ is not α-satisfiable, then the optimum of the projection game is at most $4\tilde{k}^2 - 4\tilde{k} - \frac{1}{2}\left(\tilde{k}(1-\alpha)/2 - 1\right)$ by contrapositive. Suppose that there is an assignment $\varphi : A \to \Sigma_A$ and $\varphi : B \to \Sigma_B$ that satisfies more than $4\tilde{k}^2 - 4\tilde{k} - \frac{1}{2}\left(\tilde{k}(1-\alpha)/2 - 1\right)$ edges.

In other words, less than $\frac{1}{2}\left(\tilde{k}(1-\alpha)/2 - 1\right)$ edges are not satisfied. Now, we will create a solution to GRID TILING instance as follows:

$$s_{i,j} = \begin{cases} \varphi_A(a_{i,j}) & \text{if all edges touching } \Gamma(a_{i,j}) \text{ are satisfied,} \\ \bigstar & \text{otherwise.} \end{cases}$$

for every $i, j \in [\tilde{k}]$. Note here that $\Gamma(a_{i,j})$ is a set containing all the neighboring vertices of $a_{i,j}$.

To see that this is indeed a solution to the GRID TILING instance, consider any i, j such that $s_{i,j} \neq \bigstar$. If $i \leq \tilde{k} - 1$, from the definition of $s_{i,j}$, we can conclude that the edges $(a_{i,j}, b_{i+0.5,j})$ and $(a_{i+1,j}, b_{i+0.5,j})$ are satisfied. This implies that $\varphi_A(a_{i,j}) \in S_{i,j}$ and $(\varphi_A(a_{i,j}))_2 = (\varphi_A(a_{i+1,j}))_2$. In other words, $s_{i,j}$ and $s_{i+1,j}$ will not contradict

each other in GRID TILING. Similarly, we can also conclude that $s_{i,j}$ does not contradict with $s_{i-1,j}, s_{i,j+1}, s_{i,j-1}$. Thus, the defined solution is a valid solution for GRID TILING.

Next, since each unsatisfying edge can touch $\Gamma(a_{i,j})$ for at most two pairs of $i, j \in [\tilde{k}]$, we can conclude that the number of ★'s in a solution to GRID TILING is at most 2 times the number of unsatisfied edges in the projection game. Thus, the solution to GRID TILING has less than $\tilde{k}(1-\alpha)/2 - 1$ ★'s. In other words, the optimum of GRID TILING instance is more than $\tilde{k}^2 - \tilde{k}(1-\alpha)/2 + 1$. Thus, by Lemma 5.4, we can conclude that Φ is α-satisfiable, which completes our proof. □

Lower Bound on PTAS Running Time for Planar Projection Games

First, note the following lemma concerning lower bound on running time of an algorithm distinguishing a satisfiable 3SAT instance and a not α-satisfiable 3SAT instance. The lemma, stated explicitly as Lemma 2.2 in [Mar07], is derived by combining the Sparsification Lemma from [IPZ01] and the PCP from [Din07].

Lemma 5.6. *There is a constant $1 > \alpha > 0$ such that, if ETH holds, then there is no algorithm that can distinguish between a satisfiable* 3SAT *instance and a not α-satisfiable* 3SAT *instance in time $2^{O(m^{1-\beta})}$ for any constant $\beta > 0$.*

Finally, we will use this lemma to prove Theorem 5.2. The proof idea is essentially the same as that of Theorem 2.3 from [Mar07].

Proof of Theorem 5.2. Suppose for the sake of contradiction that there is a PTAS for planar projection games of n vertices and alphabets size k running in time $2^{O(1/\varepsilon)^{\gamma}}(nk)^{O(1/\varepsilon)^{1-\delta}}$ for some constants $\gamma, \delta > 0$.

Let Φ be any 3SAT formula with m clauses. Using Lemma 5.5 with $\tilde{k} = \lceil m^{1/(2\gamma+1)} \rceil$. Set ε to be $(1-\alpha)/(32\tilde{k}) - 1/(16\tilde{k}^2) = \Theta(1/\tilde{k})$. From our choice of ε, we have

$$(1-\varepsilon)\left(4\tilde{k}^2 - 4\tilde{k}\right) > (1+\varepsilon)\left(4\tilde{k}^2 - 4\tilde{k} - \frac{1}{2}\left(\tilde{k}(1-\alpha)/2 - 1\right)\right).$$

In other words, by approximating the projection game instance resulted from Lemma 5.5 up to ε factor, we can also distinguish whether the 3SAT is satisfiable or not α-satisfiable. The running time of this is

$$\begin{aligned}
&2^{O(1/\varepsilon)^\gamma}(nk)^{O(1/\varepsilon)^{1-\delta}} \\
&= 2^{O(\tilde{k})^\gamma}\left((3\tilde{k}^2 - 2\tilde{k})3^{2\lceil m/\tilde{k}\rceil}\right)^{O(\tilde{k})^{1-\delta}} \\
&= \exp(O(\tilde{k})^\gamma + (O(\log\tilde{k}) + O(m/k)) \cdot O(\tilde{k})^{1-\delta}) \\
&= \exp(O(m^{\frac{\gamma}{1+2\gamma}}) + (O(\log m) + O(m^{\frac{2\gamma}{1+2\gamma}})) \cdot O(m)^{\frac{1-\delta}{1+2\gamma}}) \\
&= \exp(O(m^{\frac{\gamma}{1+2\gamma}}) + O(m^{\frac{2\gamma}{1+2\gamma}}) \cdot O(m)^{\frac{1-\delta}{1+2\gamma}}) \\
&= \exp(O(m^{\frac{1+2\gamma-\delta}{1+2\gamma}})),
\end{aligned}$$

which, from Lemma 5.6, implies that ETH fails. □

6 Projection Games on Dense Graphs

In order to prove the Projection Games Conjecture, an interesting question is what kind of projection games we should reduce SAT to. In this section, we make progress towards answering this question by showing that the average degree of the underlying graph of such projection games instances cannot be too large. More specifically, we give an subexponential time algorithm for projection games when the degree is large, as stated in the following theorem.

Theorem 6.1. *For every* $1/2 \geq \varepsilon \geq 0$*, there exists a* $k^{O\left(\frac{n_A n_B}{|E|\varepsilon}\right)} poly(n)$*-time algorithm for satisfiable projection games that always outputs an assignment that satisfies at least* ε *fraction of the edges.*

Note that $|E|/n_A$ and $|E|/n_B$ are average degrees of vertices in A and B respectively. Recall that, if the PGC is true, then we can reduce a SAT instance of size $\tilde{n}$ to approximating a satisfiable instance of projection games with $n_A, n_B = \tilde{n}^{1+o(1)} poly(1/\varepsilon)$ and $k = poly(1/\varepsilon)$ to within ε factor. Thus, assuming the exponential time hypothesis, this theorem tells us that the family of instances that SAT can be reduced to must have average degree on both sides of at most $O(n^{o(1)} poly(1/\varepsilon))$. Moreover, from Theorem 3.1 and a similar result for A, we can also deduce that the average degrees of both sides must be at least $1/\varepsilon$.

In addition to the above result, we also show that satisfiable projection games on sufficiently dense random graphs are easy to approximate, which is formulated formally as follows.

Theorem 6.2. *For every constant $\varepsilon > 0$, there exists a polynomial-time approximation algorithm for satisfiable projection games on random bipartite graph $G = \mathcal{G}(n/2, n/2, p)$*[1] *for any $p \geq 10\sqrt{\log n/n}$ that gives an assignment that satisfies at least $\Omega(1/k^{\varepsilon})$ fraction of the edges with probability $1 - o(1)$.*

In other words, the theorem says that, in polynomial time, we can approximate satisfiable projection games on random bipartite graph with expected degree at least $\Omega(\sqrt{n \log n})$ to within any polynomial ratio. Note here that, while there are works in subsampling lemmas for LABEL COVER and other similar problems (e.g., see [AdlVKK03, BHHS11, DKR12]) which can approximate projection games on dense graphs, these lemmas require the degree to be at least linear with respect to n whereas our result only requires the degree to be at least $\tilde{\Omega}(\sqrt{n})$.

6.1 Subexponential-Time Algorithm for Projection Games on Dense Graphs

We devote this section to the proof of Theorem 6.1. The key idea is that, if we can find a subset $T \subseteq A$ such that $\Gamma(T)$ touches ε fraction of the edges, then we can enumerate through all the possible assignments in T. For each assignment of T,

[1]Note that the graph $\mathcal{G}(n/2, n/2, p)$ is defined in Erdős-Rényi fashion, i.e., the graph contains $n/2$ vertices on each side and, for each pair of a vertex on the left and a vertex on the right, an edge between them is included with probability p independently from every other edge.

we assign $\Gamma(T)$ with the corresponding assignment. Finally, using choices reduction technique similar to that in Lemma 3.3, we can satisfy all the edges touching $\Gamma(T)$.

First, we will start by giving an algorithm that finds such subset T as stated formally in the lemma below.

Lemma 6.1. *For every $1/2 \geq \varepsilon \geq 0$, there is a polynomial time algorithm that, on any graph G, produce a subset T of size $O\left(\frac{n_A n_B}{|E|\varepsilon}\right)$ such that at least ε fraction of edges touch $\Gamma(T)$.*

Proof. The algorithm to find T can be described as follows.

1. Start with $T \leftarrow \emptyset, E' = E$.
2. While E' contains more than $1 - \varepsilon$ fraction of the edges, do the following:
   (a) Let $\Gamma'(v)$ denote the set of neighbors of vertex v corresponding to the graph $G = (V, E')$ and let $E'(\Gamma'(v))$ be the set of all edges in E' with at least one endpoint in $\Gamma'(v)$. Find a vertex $a^* \in A$ with maximum $|E'(\Gamma'(a^*))|$.
   (b) Set $T \leftarrow T \cup \{a\}$ and $E' \leftarrow E'(\Gamma'(a^*))$.
3. Output T.

It is obvious from the algorithm that the set T output from the algorithm satisfies the desired property that at least ε fraction of edges touches $\Gamma(T)$. Thus, we only need to show that T is of size $O\left(\frac{n_A n_B}{|E|\varepsilon}\right)$. To show this, it is enough to show that at each step $|E'(\Gamma'(a^*))|$ of the selected a^* is at least $\frac{|E|^2}{4 n_A n_B}$.

Since we select a^* to maximize $|E'(\Gamma'(a^*))|$, we can conclude that

$$|E'(\Gamma'(a^*))| \geq \frac{1}{n_A} \sum_{a \in A} |E'(\Gamma'(a^*))|.$$

We can further bound the right hand side as follows:

$$\begin{aligned}
\sum_{a\in A} |E'(\Gamma'(a^*))| &= \sum_{a\in A}\sum_{b\in\Gamma'(a)} |\Gamma'(b)| \\
&= \sum_{b\in B}\sum_{a\in\Gamma'(b)} |\Gamma'(b)| \\
&= \sum_{b\in B} |\Gamma'(b)|^2 \\
\text{(Jensen's inequality)} &\geq \frac{1}{n_B}\left(\sum_{b\in B} |\Gamma'(b)|\right)^2 \\
&= \frac{1}{n_B}|E'|^2 \\
(|E'| \geq (1-\varepsilon)|E| \geq |E|/2) &\geq \frac{|E|^2}{4n_B}.
\end{aligned}$$

As a result, we can conclude that

$$|E'(\Gamma'(a^*))| \geq \frac{|E|^2}{4n_An_B}$$

as desired.

Finally, since $|E'(\Gamma'(a^*))|$ at each step is at least $\frac{|E|^2}{4n_An_B}$, we can conclude that E' is reduced by at least the same amount. Hence, the number of iterations of the loop is $O\left(\frac{n_An_B}{|E|\varepsilon}\right)$, which means that T is of size at most $O\left(\frac{n_An_B}{|E|\varepsilon}\right)$ as desired. □

Now, we are ready to prove Theorem 6.1.

Proof of Theorem 6.1. The algorithm proceeds as follows.

1. Run the algorithm from Lemma 6.1 on graph G to get a subset T.
2. Enumerate through all possible assignments $\phi_T : T \to \Sigma_A$ of all the vertices in T. For each assignment ϕ_T, run the following steps:

(a) For each $b \in \Gamma(T)$, check whether $\pi_{(a,b)}(\phi_T(a))$'s are equal for every $a \in \Gamma(b) \cap T$. If not, skip the current ϕ_T and proceed to the next one. Otherwise, assign $\sigma_b^* = \pi_{(a,b)}(\phi_T(a))$ to b.

(b) For each $a \in \Gamma_2(T)$, find the set S_a of all possible assignments to a, i.e., $S_a = \{\sigma_a \in \Sigma_A \mid \forall b \in \Gamma(a) \cap \Gamma(T), \pi_{(a,b)}(\sigma_a) = \sigma_b\}$. If $S_a = \emptyset$, skip the current ϕ_T and proceed to the next one. Otherwise, pick σ_a^* to be any member of S_a and assign σ_a^* to a.

3. Among all assignments produced from the previous step, pick the one that satisfies maximum number of edges.

First, observe that the running time of the algorithm is $O\left(k^{|T|} poly(n,k)\right) = k^{O\left(\frac{n_A n_B}{|E|\varepsilon}\right)} poly(n)$.

To see that this algorithm produces an assignment that satisfies at least ε fraction of the edges, first observe that, from step 3, we can conclude that the output assignment satisfies at least as many edges as the assignment produced from step 2 when $\phi_T = \phi_T^{OPT}$ is the optimal assignment, i.e., $\phi_T^{OPT}(a) = \sigma_a^{OPT}$ for every $a \in T$.

Now, observe also that, if $\phi_T = \phi_T^{OPT}$ is the optimal assignment, then $\sigma_b^* = \sigma_b^{OPT}$ in step 2a for every $b \in \Gamma(T)$. Thus, it is obvious that $\sigma_a^{OPT} \in S_a$ in step 2b for every $a \in \Gamma_2(T)$. Hence, the algorithm does not skip ϕ_T^{OPT}.

Moreover, it is obvious from definition of S_a that, when we assign $\sigma_a^* \in S_a$, we satisfy all edges from a to its neighbors in $\Gamma(T)$. As a result, we can conclude that, when $\phi_T = \phi_T^{OPT}$, the assignment from step 2 satisfies all the edges that touch $\Gamma(T)$. From Lemma 6.1, the number of such edges is at least $\varepsilon|E|$, which completes our proof. □

6.2 Polynomial-Time Algorithm for Dense Random Graphs

The focus of this section is to prove Theorem 6.2. The key idea of the proof is that we will reduce the problem of approximating projection games on dense random graphs to approximating a different problem called FREEGAME defined below:

FREEGAME

INPUT: Sets A, B, a finite set of labels (aka alphabets) Σ, and, for each edge $(a, b) \in A \times B$, a predicate $C_{a,b} : \Sigma \times \Sigma \to \{0, 1\}$.

GOAL: Find an assignment to the vertices $\varphi : A \cup B \to \Sigma$ that maximizes the number of edges $e = (a, b)$ that are "satisfied", i.e., $C_{a,b}(\varphi(a), \varphi(b)) = 1$.

Note that FREEGAME can be viewed as MAX 2-CSP on complete bipartite graph.

Similar to projection games, we call an instance of FREEGAME "satisfiable" if there is an assignment that satisfies every edge. For such instance, we define σ_v^{OPT} in a similar fashion to that of projection games, i.e., σ_v^{OPT} is the assignment to vertex $v \in A \cup B$ in an assignment that satisfies every edge. Moreover, let n denote the number of vertices, $|A| + |B|$, and let k denote the size of the alphabets Σ.

First, we will show an approximation algorithm for FREEGAME. Then, we will show the reduction from projection games on random graphs to FREEGAME and, thereby, complete the proof of Theorem 6.2.

6.2.1 Approximation Algorithm for FREEGAME

Approximation algorithms for FREEGAME and lower bound on their running times have been studied in [AIM14]. However, the algorithm given in the paper does

not run in polynomial time and it focuses on small additive error, which is not our main interest[2]. As a result, we present a new algorithm to approximate satisfiable FREEGAME instances, formally stated in the lemma below.

Lemma 6.2. *For every positive integer $i > 0$, there exists an $O\left((nk)^{2i}\right)$-time algorithm that, for any satisfiable* FREEGAME *instance, produces an assignment that satisfy at least $1/k^{1/i}$ fraction of the edges.*

Proof. We will prove the lemma by induction.

Let $P(i)$ represent the following statement: there exists an $O\left((nk)^{2i}\right)$-time algorithm APPROX-FREEGAME$_i(A, B, \Sigma, \{C_{(a,b)}\}_{(a,b)\in A\times B}, \{S_b\}_{b\in B})$ that takes in a satisfiable FREEGAME instance $(A, B, \Sigma, \{C_{(a,b)}\}_{(a,b)\in A\times B})$ and a reduced alphabets set S_b for every $b \in B$ such that, if $\sigma_b^{OPT} \in S_b$ for every $b \in B$, then the algorithm outputs an assignment that satisfies at least $\frac{1}{|B|}\left(\sum_{b\in B}\frac{1}{|S_b|^{1/i}}\right)$ fraction of edges.

Note that $P(i)$ implies the lemma by setting $S_b = \Sigma$ for every $b \in B$.

Base Case. The algorithm APPROX-FREEGAME$_1(A, B, \Sigma, \{C_{(a,b)}\}_{(a,b)\in A\times B}, \{S_b\}_{b\in B})$ is a greedy algorithm that works as follows:

1. For each $a \in A$, assign it to any $\sigma_a^* \in \Sigma$ such that, for every $b \in B$, $\{\sigma_b \in \Sigma \mid C_{(a,b)}(\sigma_a^*, \sigma_b) = 1\}$ is not an empty set. If no such σ_a^* exists for some $a \in A$, abort the algorithm.

2. For each $b \in B$, assign it to $\sigma_b^* \in S_b$ that maximizes the number of edges satisfied, i.e., maximizes $\sum_{a\in A} C_{(a,b)}(\sigma_a^*, \sigma_b)$.

[2]Note that, if we are interested in subexponential-time algorithms, we can use the approximation algorithm from [AIM14] instead of our algorithm in Lemma 6.2 in the proof of Theorem 6.2. Doing so gives an $(nk)^{O(\log n)}$-time $O(1)$-approximation algorithm for projection games on sufficiently dense random graphs.

It is obvious that the algorithm runs in $O(n^2k^2)$ time as desired.

Now, we will show that, if $\sigma_b^{OPT} \in S_b$ for every $b \in B$, then the algorithm gives an assignment that satisfies at least $\frac{1}{|B|}\left(\sum_{b\in B}\frac{1}{|S_b|}\right)$ fraction of edges.

First of all, observe that, if $\sigma_b^{OPT} \in S_b$ for every $b \in B$, then the algorithm does not abort since $\sigma_a^* = \sigma_a^{OPT}$ is a valid choice for step 1. Moreover, from our choice of σ_b^*, the number of satisfied edges by the output assignment can be bounded as follows.

$$\begin{aligned}
\sum_{b\in B}\sum_{a\in A} C_{(a,b)}(\sigma_a^*,\sigma_b^*) &\geq \sum_{b\in B}\frac{1}{|S_b|}\sum_{\sigma_b\in S_b}\left(\sum_{a\in A} C_{(a,b)}(\sigma_a^*,\sigma_b)\right)\\
&= \sum_{b\in B}\frac{1}{|S_b|}\sum_{a\in A}\left(\sum_{\sigma_b\in S_b} C_{(a,b)}(\sigma_a^*,\sigma_b)\right)\\
(\text{From our choice of } \sigma_a^*) &\geq \sum_{b\in B}\frac{1}{|S_b|}\sum_{a\in A} 1\\
&= \sum_{b\in B}\frac{1}{|S_b|}|A|\\
&= |A|\sum_{b\in B}\frac{1}{|S_b|}.
\end{aligned}$$

Thus, we can conclude that the algorithm outputs an assignment that satisfies at least $\frac{1}{|B|}\left(\sum_{b\in B}\frac{1}{|S_b|}\right)$ fraction of edges. As a result, $P(1)$ is true.

Inductive Step. Let j be any positive integer. Suppose that $P(j)$ holds.

We will now describe APPROX-FREEGAME$_{j+1}$ based on APPROX-FREEGAME$_j$ as follows.

1. Define R to be $\frac{1}{|B|}\left(\sum_{b\in B}\frac{1}{|S_b|^{1/(j+1)}}\right)$, our target fraction of edges we want to satisfy.
2. For each $a \in A$, check whether there exists any $\sigma'_a \in \Sigma$ such that $\sum_{b\in B}\frac{\sum_{\sigma_b\in S_b} C_{(a,b)}(\sigma'_a,\sigma_b)}{|S_b|} \geq R|B|$. If there exists such σ'_a for every $a \in A$, then execute the following greedy algorithm.

(a) For every $a \in A$, assign the aforementioned σ'_a to a.

(b) For each $b \in B$, assign it $\sigma^*_b \in S_b$ that maximizes the number of edges satisfied, i.e., $\sum_{a \in A} C_{(a,b)}(\sigma'_a, \sigma_b)$.

3. Otherwise, if there does not exists such σ'_a for some $a \in A$, then we run the following steps instead.

(a) Let $a_0 \in A$ be a vertex that σ'_{a_0} does not exist. For each $\sigma_{a_0} \in \Sigma$, execute the following steps.

i. For each $b \in B$, set S'_b to be the set of alphabet symbols in S_b that satisfy (a_0, b) when we assign σ_{a_0} to a_0, i.e., $S'_b = \{\sigma_b \in S_b \mid C_{(a_0,b)}(\sigma_{a_0}, \sigma_b) = 1\}$.

ii. Call APPROX-FREEGAME$_j(A, B, \Sigma, \{C_{(a,b)}\}_{(a,b)\in A\times B}, \{S'_b\}_{b\in B})$.

(b) Output an assignment from the APPROX-FREEGAME$_j$ calls in the previous step that satisfies maximum number of edges.

Since every step except the APPROX-FREEGAME$_j(A, B, \Sigma, \{C_{(a,b)}\}_{(a,b)\in A\times B}, \{S_b\}_{b\in B})$ calls takes $O((nk)^2)$ time and we call APPROX-FREEGAME$_j$ only k times, we can conclude that the running time of APPROX-FREEGAME$_{j+1}$ is $O((nk)^{2j+2})$ as desired.

The only thing left to show is that the assignment output from the algorithm indeed satisfies at least $R = \frac{1}{|B|}\left(\sum_{b\in B} \frac{1}{|S_b|^{1/(j+1)}}\right)$ fraction of the edges. We will consider two cases.

First, if there exists σ'_a for every $a \in A$, then the greedy algorithm is executed. In this case, the number of satisfied edges is

$$\sum_{b\in B}\sum_{a\in A} C_{(a,b)}(\sigma'_a, \sigma^*_b) \geq \sum_{b\in B} \frac{1}{|S_b|} \sum_{\sigma_b \in S_b} \left(\sum_{a\in A} C_{(a,b)}(\sigma'_a, \sigma_b)\right)$$

$$
\begin{aligned}
&= \sum_{a\in A}\sum_{b\in B}\frac{1}{|S_b|}\left(\sum_{\sigma_b\in S_b} C_{(a,b)}(\sigma'_a,\sigma_b)\right)\\
(\text{From the definition of } \sigma'_a) &\geq \sum_{a\in A} R|B|\\
&= R|A||B|.
\end{aligned}
$$

Thus, the output assignment satisfies at least R fraction of the edges as desired.

In the second case, there does not exists σ'_a for some $a \in A$. From step 3b, we can conclude that the output assignment satisfies as many edges as the output assignment from APPROX-FREEGAME$_j(A, B, \Sigma, \{C_{(a,b)}\}_{(a,b)\in A\times B}, \{S'_b\}_{b\in B})$ call when $\sigma_{a_0} = \sigma^{OPT}_{a_0}$.

Observe that, when $\sigma_{a_0} = \sigma^{OPT}_{a_0}$, then σ^{OPT}_b remains in S'_b for every $b \in B$. As a result, from the inductive hypothesis, APPROX-FREEGAME$_j(A, B, \Sigma, \{C_{(a,b)}\}_{(a,b)\in A\times B}, \{S'_b\}_{b\in B})$ outputs an assignment that satisfies at least $R' = \frac{1}{|B|}\left(\sum_{b\in B}\frac{1}{|S'_b|^{1/j}}\right)$ fraction of edges. Moreover, we can derive the following inequalities:

$$
\begin{aligned}
(R')^j\left(\frac{1}{|B|}\sum_{b\in B}\frac{\sum_{\sigma_b\in S_b} C_{(a_0,b)}(\sigma^{OPT}_{a_0},\sigma_b)}{|S_b|}\right) &= \left(\frac{1}{|B|}\left(\sum_{b\in B}\frac{1}{|S'_b|^{1/j}}\right)\right)^j\left(\frac{1}{|B|}\sum_{b\in B}\frac{|S'_b|}{|S_b|}\right)\\
&= \frac{1}{|B|^{j+1}}\left(\sum_{b\in B}\frac{1}{|S'_b|^{1/j}}\right)^j\left(\sum_{b\in B}\frac{|S'_b|}{|S_b|}\right)\\
(\text{From definition of } S'_b) &= \frac{1}{|B|^{j+1}}\left(\sum_{b\in B}\frac{1}{|S'_b|^{1/j}}\right)^j\left(\sum_{b\in B}\frac{|S'_b|}{|S_b|}\right)\\
(\text{From Hölder's inequality}) &\geq \frac{1}{|B|^{j+1}}\left(\sum_{b\in B}\frac{1}{|S'_b|^{1/(j+1)}}\right)^{j+1}\\
&= R^{j+1}.
\end{aligned}
$$

Since there does not exists σ'_{a_0}, we can conclude that $\frac{1}{|B|}\sum_{b\in B}\frac{\sum_{\sigma_b\in S_b} C_{(a_0,b)}(\sigma^{OPT}_{a_0},\sigma_b)}{|S_b|} \leq R$. Hence, we can conclude that $R' \geq R$. In other words, the assignment output from

the algorithm satisfies at least R fraction of the edges.

Thus, we can conclude that $P(j+1)$ is true. As a result, $P(i)$ is true for every positive integer i, which completes the proof for Lemma 6.2. □

6.2.2 Reduction from Projection Games on Dense Random Graphs to FREEGAME

In this subsection, we will show the reduction from projection games on dense random graphs to FREEGAME. We then use this reduction together with the approximation algorithm for FREEGAME presented above to prove Theorem 6.2.

Before we give the reduction, we will state a couple of lemmas regarding the standard properties of random graphs. We will not give full proofs for the lemmas since they are trivial via standard Chernoff bounds.

Lemma 6.3. *When $p \geq 10\sqrt{\log n/n}$, with probability $1 - o(1)$, every vertex in $G = \mathcal{G}(n/2, n/2, p)$ has degree between $np/10$ and $10np$.*

Lemma 6.4. *In $G = \mathcal{G}(n/2, n/2, p)$ with $p \geq 10\sqrt{\log n/n}$, with probability $1 - o(1)$, every pair of vertices a, a' on the left has at least $np^2/10$ common neighbors.*

Now, we will give the reduction from projection games on dense random graphs to FREEGAME, which can be stated formally as follows.

Lemma 6.5. *With probability $1 - o(1)$, there exists a polynomial-time reduction from a satisfiable projection game instance $(A, B, E, \Sigma_A, \Sigma_B, \{\pi_e\}_{e \in E})$ where $G = (A, B, E)$ is sam-*

pled according to $\mathcal{G}(n/2, n/2, p)$ *to a satisfiable* FREEGAME *instance* $(A', B', \Sigma, \{C_{(a,b)}\}_{(a,b)\in A'\times B'})$ *such that*

1. $|A'|, |B'| \leq |A|$ *and* $|\Sigma| \leq |\Sigma_A|$, *and*
2. *For any* $1 \geq \varepsilon \geq 0$, *given an assignment* $\varphi : A' \cup B' \to \Sigma$ *that satisfies* ε *fraction of edges of the* FREEGAME *instance, one can determine an assignment* $\varphi_A : A \to \Sigma_A, \varphi_B : B \to \Sigma_B$ *for the projection game instance that satisfies* $\Omega(\varepsilon)$ *fraction of edges in polynomial time.*

Proof. The reduction from a projection game instance $(A, B, E, \Sigma_A, \Sigma_B, \{\pi_e\}_{e\in E})$ to a FREEGAME instance $(A', B', \Sigma, \{C_{(a,b)}\}_{(a,b)\in A'\times B'})$ proceeds as follows.

1. Partition A into two sets A_1, A_2 of equal sizes. Then, set $A' \leftarrow A_1$ and $B' \leftarrow A_2$.
2. Let Σ be Σ_A.
3. For each $a_1 \in A_1, a_2 \in A_2, \sigma_{a_1}, \sigma_{a_2} \in \Sigma_A$, let $C_{(a_1,a_2)}(\sigma_{a_1}, \sigma_{a_2})$ to be one if and only if these two assignments do not assign any $b \in \Gamma(a_1) \cap \Gamma(a_2)$ to different assignments. In other words, $C_{(a_1,a_2)}(\sigma_{a_1}, \sigma_{a_2}) = 1$ if and only if $\pi_{(a_1,b)}(\sigma_{a_1}) = \pi_{(a_2,b)}(\sigma_{a_2})$ for every $b \in \Gamma(a_1) \cap \Gamma(a_2)$.

It is obvious that the reduction runs in polynomial time and that the first condition holds. Moreover, observe that the resulting FREEGAME is satisfiable by assigning the corresponding optimal assignment from the original projection games instance to each vertex of the FREEGAME instance. Thus, it is enough for us to prove that, with probability $1 - o(1)$, the second condition is indeed true.

To show this, we present a simple algorithm that, given an assignment $\varphi : A' \cup B' \to \Sigma$ that satisfies ε fraction of edges of the FREEGAME instance, output an assignment $\varphi_A : A \to \Sigma_A, \varphi_B : B \to \Sigma_B$ for the projection game instance that satisfies $\Omega(\varepsilon)$ fraction of edges. The algorithm works greedily as follows.

1. For each $a \in A$, let $\varphi_A(a) \leftarrow \varphi(a)$.
2. For each $b \in B$, pick $\varphi_B(b) = \sigma_b^*$ to be the assignment to b that satisfies maximum number of edges, i.e., maximize $|\{a \in \Gamma(b) \mid \pi_{(a,b)}(\varphi_A(a)) = \sigma_b\}|$.

Trivially, the algorithm runs in polynomial time. Thus, we only need to prove that, with probability $1 - o(1)$, the produced assignment satisfies at least $\Omega(\varepsilon)$ fraction of edges of the projection game. To prove this, we will assume the properties from Lemma 6.3 and Lemma 6.4, which holds with probability $1 - o(1)$.

The number of satisfied edges can be rearranged as follows.

$$
\begin{aligned}
&\sum_{b\in B}\sum_{a\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi_A(a))=\varphi_B(b)} \\
&= \sum_{b\in B}\sum_{a\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi(a))=\sigma_b^*} \\
&= \sum_{b\in B}\left[\frac{1}{d_b}\left(\sum_{a\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi(a))=\sigma_b^*}\right) d_b\right] \\
&= \sum_{b\in B}\left[\frac{1}{d_b}\left(\sum_{a\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi(a))=\sigma_b^*}\right)\left(\sum_{\sigma_b\in\Sigma_B}\sum_{a\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi(a))=\sigma_b}\right)\right] \\
&= \sum_{b\in B}\left[\frac{1}{d_b}\sum_{\sigma_b\in\Sigma_B}\left(\sum_{a\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi(a))=\sigma_b^*}\right)\left(\sum_{a\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi(a))=\sigma_b}\right)\right] \\
(\text{From the choice of } \sigma_b^*) &\geq \sum_{b\in B}\left[\frac{1}{d_b}\sum_{\sigma_b\in\Sigma_B}\left(\sum_{a\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi(a))=\sigma_b}\right)^2\right] \\
&= \sum_{b\in B}\left(\frac{1}{d_b}\sum_{\sigma_b\in\Sigma_B}\sum_{a,a'\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi(a))=\sigma_b} 1_{\pi_{(a',b)}(\varphi(a'))=\sigma_b}\right) \\
&= \sum_{b\in B}\left(\frac{1}{d_b}\sum_{a,a'\in\Gamma(b)}\sum_{\sigma_b\in\Sigma_B} 1_{\pi_{(a,b)}(\varphi(a))=\sigma_b} 1_{\pi_{(a',b)}(\varphi(a'))=\sigma_b}\right)
\end{aligned}
$$

Observe that $\sum_{\sigma_b\in\Sigma_B} 1_{\pi_{(a,b)}(\varphi(a))=\sigma_b} 1_{\pi_{(a',b)}(\varphi(a'))=\sigma_b} = 1_{\pi_{(a,b)}(\varphi(a))=\pi_{(a',b)}(\varphi(a'))}$. Thus,

the number of satisfied edges is at least

$$\sum_{b\in B}\left(\frac{1}{d_b}\sum_{a,a'\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi(a))=\pi_{(a',b)}(\varphi(a'))}\right).$$

Moreover, from Lemma 6.3, $d_b \le 10np$ for every $b \in B$ with probability $1-o(1)$. This implies that, with probability $1-o(1)$, the output assignment satisfied at least

$$\frac{1}{10np}\sum_{b\in B}\sum_{a,a'\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi(a))=\pi_{(a',b)}(\varphi(a'))}$$

edges.

We can further reorganize this quantity as follows.

$$\begin{aligned}\frac{1}{10np}\sum_{b\in B}\sum_{a,a'\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi(a))=\pi_{(a',b)}(\varphi(a'))} &\ge \frac{1}{10np}\sum_{b\in B}\sum_{\substack{(a,a')\in A'\times B'\\ \text{s.t. } a,a'\in\Gamma(b)}} 1_{\pi_{(a,b)}(\varphi(a))=\pi_{(a',b)}(\varphi(a'))}\\ &= \frac{1}{10np}\sum_{(a,a')\in A'\times B'}\sum_{b\in\Gamma(a)\cap\Gamma(a')} 1_{\pi_{(a,b)}(\varphi(a))=\pi_{(a',b)}(\varphi(a'))}.\end{aligned}$$

Now, observe that, from its definition, if $C_{(a,a')}(\varphi(a),\varphi(a'))$ is one, then $1_{\pi_{(a,b)}(\varphi(a))=\pi_{(a',b)}(\varphi(a'))}$ is also one for every $b \in \Gamma(a)\cap\Gamma(a')$. Thus, we have

$$\begin{aligned}&\frac{1}{10np}\sum_{(a,a')\in A'\times B'}\sum_{b\in\Gamma(a)\cap\Gamma(a')} 1_{\pi_{(a,b)}(\varphi(a))=\pi_{(a',b)}(\varphi(a'))}\\ &\ge \frac{1}{10np}\sum_{(a,a')\in A'\times B'}\sum_{b\in\Gamma(a)\cap\Gamma(a')} C_{(a,a')}(\varphi(a),\varphi(a'))\\ &= \frac{1}{10np}\sum_{(a,a')\in A'\times B'} |\Gamma(a)\cap\Gamma(a')| C_{(a,a')}(\varphi(a),\varphi(a')).\end{aligned}$$

From Lemma 6.4, with probability $1-o(1)$, $|\Gamma(a)\cap\Gamma(a')| \ge np^2/10$ for every $(a,a') \in A'\times B'$. Hence, we can conclude that the above expression is, with proba-

bility $1 - o(1)$, at least

$$\frac{1}{10np} \sum_{(a,a') \in A' \times B'} \frac{np^2}{10} C_{(a,a')}(\varphi(a), \varphi(a')) = \frac{p}{100} \sum_{(a,a') \in A' \times B'} C_{(a,a')}(\varphi(a), \varphi(a')).$$

Next, note that $\sum_{(a,a') \in A' \times B'} C_{(a,a')}(\varphi(a), \varphi(a'))$ is the number of edges satisfied by φ in the FREEGAME instance, which is at least $\varepsilon |A'||B'| = \varepsilon n^2/16$. Thus, we have

$$\frac{p}{100} \sum_{(a,a') \in A' \times B'} C_{(a,a')}(\varphi(a), \varphi(a')) \geq \frac{\varepsilon n^2 p}{1600}.$$

Finally, again from Lemma 6.3, the total number of edges is at most $5n^2p$ with probability $1 - o(1)$. As a result, with probability $1 - o(1)$, the algorithm outputs an assignment that satisfies at least $\frac{\varepsilon}{8000} = \Omega(\varepsilon)$ fraction of edges of the projection game instance, which concludes the proof of this lemma. □

Finally, we give a proof for Theorem 6.2 below.

Proof of Theorem 6.2. The proof is simple. First, we use the reduction from Lemma 6.5 to transform a projection game instance on dense graph to a FREEGAME instance. Since the approximation ratio deteriorates by only constant factor with probability $1 - o(1)$ in the reduction, we can use the approximation algorithm from Lemma 6.2 with $i = \lceil 1/\varepsilon \rceil$, which gives us an assignment that satisfies at least $O(1/k^{\varepsilon})$ fraction of the edges as desired. □

7 Future Work

Even though our work presented in this thesis has helped us gain more insight into approximating projection games, the work is far from finished. The biggest question remained is whether the PGC holds. Even though we are able to achieve polynomial lower bounds on integrality gap for the Lasserre SDP after polynomial rounds, it is unlikely that the results can be easily translated to NP-hardness results. New techniques must be introduced in order to prove or disprove the Projection Games Conjecture.

A seemingly easier question is how hard LABEL COVER is under the computational model defined based on the Lasserre hierarchy. As shown in Chapter 4, after $N^{\Omega(\varepsilon)}$ rounds, the integrality gap still remains $N^{1/8-\varepsilon}$. However, our approximation algorithm only achieves an approximation ratio of $O(N^{1/4})$ in polynomial time. Ideally, we would like these the two ratios to match. In order to do so, we need to find a better polynomial-time approximation algorithm, or a better lower bound for the integrality gap, or both.

On the approximation algorithm front, Prof. Eden Chlamtac, Dr. Aravindan Vijayaraghavan, the author's advisor and the author are exploring an approach based on counting constant-size witnesses. A similar approach yielded an approximation algorithm for the DENSEST k-SUBGRAPH [BCC$^+$10], and thus we hope to get a better and simpler approximation algorithm for projection games as well.

On the integrality gap side, an interesting and most obvious direction is to try to tighten up the analysis of the completeness. More specifically, it may be possible to tighten up Lemma 4.5 from [BCV$^+$12]. It is also worth noting that the analysis for LABEL COVER instance does not need the set of variables to be random and, with some additional analysis, this likely applies for DENSEST k-SUBGRAPH as well. This means that, given K, n, β, s, it is enough for us to give explicit construction for the sets $S_1, \ldots, S_{\beta n} \subseteq [n]$, each of size K, such that the intersection of every s different sets contain more than $(K - D/2)s$ elements. Can we find such sets with s that is significantly larger than ηn specified in Lemma 4.4?

Finally, another interesting question is whether there is a deeper connection between LABEL COVER and DENSEST k-SUBGRAPH. At a glance, these two problem seems very similar, if we consider the *label-extended graph* of a projection games instance where each node is a tuple of an original vertex in G together with an assignment to that vertex and each edge is a "satisfied" relation, then projection games' goal is to find the densest n-subgraph with an additional constraint that each vertex in the subgraph must correspond to different vertex in G. Regarding the connection between the two problems, it was shown in [CHK09] that an approximation algorithm for a variant of LABEL COVER where the constraints are not limited to projections can be turned to an approximation algorithm for DENSEST k-SUBGRAPH with asymptotically as good approximation ratio. Moreover, as seen in Chapter 4, the technique to create an integrality gap in the Lasserre hierarchy translates from DENSEST k-SUBGRAPH to LABEL COVER. Furthermore, as discussed above, it does seem that a technique that is used for approximating DENSEST k-SUBGRAPH may be useful for approximating LABEL COVER too. All this evidence leads to the question of whether we can find a deeper relationship between the two problems, such as a PGC-hardness result for DENSEST k-SUBGRAPH.

Bibliography

[AAM+11] N. Alon, S. Arora, R. Manokaran, D. Moshkovitz, and O. Weinstein. Inapproximability of densest κ-subgraph from average-case hardness. Manuscript, 2011.

[AdlVKK03] N. Alon, W. F. de la Vega, R. Kannan, and M. Karpinski. Random sampling and approximation of max-csps. *J. Comput. Syst. Sci.*, 67(2):212–243, September 2003.

[AHI02] Y. Asahiro, R. Hassin, and K. Iwama. Complexity of finding dense subgraphs. *Discrete Appl. Math.*, 121(1-3):15–26, September 2002.

[AIM14] S. Aaronson, R. Impagliazzo, and D. Moshkovitz. AM with multiple Merlins. In *Computational Complexity (CCC), 2014 IEEE 29th Conference on*, pages 44–55, June 2014.

[AKK+08] S. Arora, S. A. Khot, A. Kolla, D. Steurer, M. Tulsiani, and N. Vishnoi. Unique games on expanding constraint graphs are easy: extended abstract. In *Proc. 40th ACM Symp. on Theory of Computing*, pages 21–28, 2008.

[ALM+98] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[AS98] S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.

[Bak94] B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41(1):153–180, January 1994.

[BCC+10] A. Bhaskara, M. Charikar, E. Chlamtac, U. Feige, and A. Vijayaraghavan. Detecting high log-densities: An $O(n^{1/4})$ approximation for densest k-subgraph. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC '10, pages 201–210, New York, NY, USA, 2010. ACM.

[BCV+12] A. Bhaskara, M. Charikar, A. Vijayaraghavan, V. Guruswami, and Y. Zhou. Polynomial integrality gaps for strong SDP relaxations of densest k-subgraph. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 388–405. SIAM, 2012.

[BFL91] L. Babai, L. Fortnow, and C. Lund. Nondeterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[BFLS91] L. Babai, L. Fortnow, L. A. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proc. 23rd ACM Symp. on Theory of Computing*, pages 21–32, 1991.

[BGS98] M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCPs, and nonapproximability—towards tight results. *SIAM Journal on Computing*, 27(3):804–915, 1998.

[BHHS11] B. Barak, M. Hardt, T. Holenstein, and D. Steurer. Subsampling mathematical relaxations and average-case complexity. In *Proceedings of the*

Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '11, pages 512–531. SIAM, 2011.

[BRS11] B. Barak, P. Raghavendra, and D. Steurer. Rounding semidefinite programming hierarchies via global correlation. In *Proceedings of the 2011 IEEE 52Nd Annual Symposium on Foundations of Computer Science*, FOCS '11, pages 472–481, Washington, DC, USA, 2011. IEEE Computer Society.

[CHK09] M. Charikar, M.T. Hajiaghayi, and H. Karloff. Improved approximation algorithms for label cover problems. In *In ESA*, pages 23–34. Springer, 2009.

[CHKX06] J. Chen, X. Huang, I.A. Kanj, and G. Xia. Strong computational lower bounds via parameterized complexity. *J. Comput. Syst. Sci.*, 72(8):1346–1367, December 2006.

[Din07] I. Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3):12, 2007.

[DKR12] M. Dinitz, G. Kortsarz, and R. Raz. Label cover instances with large girth and the hardness of approximating basic k-spanner. In Artur Czumaj, Kurt Mehlhorn, Andrew Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming*, volume 7391 of *Lecture Notes in Computer Science*, pages 290–301. Springer Berlin Heidelberg, 2012.

[DS13] I. Dinur and D. Steurer. Analytical approach to parallel repetition. Technical Report 1305.1979, arXiv, 2013.

[Epp00] D. Eppstein. Diameter and treewidth in minor-closed graph families. *Algorithmica*, 27(3):275–291, 2000.

[Fei98] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[Fei02] U. Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 534–543, New York, NY, USA, 2002. ACM.

[FL01] U. Feige and M. Langberg. Approximation algorithms for maximization problems arising in graph partitioning. *J. Algorithms*, 41(2):174–211, November 2001.

[FPK01] U. Feige, D. Peleg, and G. Kortsarz. The dense k -subgraph problem. *Algorithmica*, 29(3):410–421, 2001.

[GW95] M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, November 1995.

[Hås01] J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.

[Hoe63] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963.

[IPZ01] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, December 2001.

[Kho04] S. Khot. Ruling out PTAS for graph min-bisection, densest subgraph and bipartite clique. In *Proceedings of the 45th Annual IEEE Symposium*

on Foundations of Computer Science, FOCS '04, pages 136–145, Washington, DC, USA, 2004. IEEE Computer Society.

[Kle05] P. N. Klein. A linear-time approximation scheme for TSP for planar weighted graphs. In *In Proceedings, 46th IEEE Symposium on Foundations of Computer Science*, pages 146–155, 2005.

[Las01a] J. B. Lasserre. An explicit exact SDP relaxation for nonlinear 0-1 programs. In *Proceedings of the 8th International IPCO Conference on Integer Programming and Combinatorial Optimization*, pages 293–303, London, UK, UK, 2001. Springer-Verlag.

[Las01b] J. B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11:796–817, 2001.

[Lau03] M. Laurent. A comparison of the Sherali-Adams, Lovász-Schrijver, and Lasserre relaxations for 0–1 programming. *Math. Oper. Res.*, 28(3):470–496, July 2003.

[LS91] L. Lovsz and A. Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal on Optimization*, 1:166–190, 1991.

[LT79] R. Lipton and R. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.

[LY94] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41(5):960–981, September 1994.

[Mar07] D. Marx. On the optimality of planar and geometric approximation schemes. In *Foundations of Computer Science, 2007. FOCS '07. 48th Annual IEEE Symposium on*, pages 338–348, Oct 2007.

[Mar12] D. Marx. A tight lower bound for planar multiway cut with fixed number of terminals. In *Proceedings of the 39th International Colloquium Conference on Automata, Languages, and Programming - Volume Part I*, ICALP'12, pages 677–688, Berlin, Heidelberg, 2012. Springer-Verlag.

[MM13] P. Manurangsi and D. Moshkovitz. Improved approximation algorithms for projection games. In *Algorithms ESA 2013*, volume 8125 of *Lecture Notes in Computer Science*, pages 683–694. Springer Berlin Heidelberg, 2013.

[Mos12] D. Moshkovitz. The projection games conjecture and the NP-hardness of ln n-approximating set-cover. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012*, volume 7408, pages 276–287, 2012.

[MR10] D. Moshkovitz and R. Raz. Two query PCP with sub-constant error. *Journal of the ACM*, 57(5), 2010.

[Pel07] D. Peleg. Approximation algorithms for the label-cover max and red-blue set cover problems. *J. of Discrete Algorithms*, 5(1):55–64, March 2007.

[Raz98] R. Raz. A parallel repetition theorem. In *SIAM Journal on Computing*, volume 27, pages 763–803, 1998.

[Rot13] T. Rothvoß. The Lasserre hierarchy in approximation algorithms. Lecture Notes for the MAPSP 2013 Tutorial, 2013.

[RS10] P. Raghavendra and D. Steurer. Graph expansion and the unique games conjecture. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC '10, pages 755–764, New York, NY, USA, 2010. ACM.

[SA90] H.D. Sherali and W.P. Adams. A hierarchy of relaxation between the continuous and convex hull representations. *SIAM J. Discret. Math.*, 3(3):411–430, May 1990.

[SW98] A. Srivastav and K. Wolf. Finding dense subgraphs with semidefinite programming. In *Proceedings of the International Workshop on Approximation Algorithms for Combinatorial Optimization*, APPROX '98, pages 181–191, London, UK, UK, 1998. Springer-Verlag.

[Tul09] M. Tulsiani. CSP gaps and reductions in the Lasserre hierarchy. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 303–312, New York, NY, USA, 2009. ACM.

BIBLIOGRAPHY

A Appendix

A.1 Polynomial-time Approximation Algorithms for Projection Games for Nonuniform Preimage Sizes

In this section, we will describe a polynomial time $O((n_A|\Sigma_A|)^{\frac{1}{4}})$-approximation algorithm for satisfiable projection games, including those with nonuniform preimage sizes.

It is not hard to see that, if the p_e's are not all equal, then "know your neighbors' neighbors" algorithm does not necessarily end up with at least $h_{max}/\overline{p}$ fraction of satisfied edges anymore. The reason is that, for a vertex a with large $|\Gamma_2(a)|$ and any assignment $\sigma_a \in \Sigma_A$ to the vertex, the number of preimages in $\pi_e^{-1}(\pi_{(a,b)}(\sigma_a))$ might be large for each neighbor b of a and each edge e that has an endpoint b. We solve this issue, by instead of using all the edges for the algorithm, only using "good" edges whose preimage sizes for the optimal assignments are at most a particular value. However, this definition of "good" does not only depend on an edge but also on the assignment to the edge's endpoint in B, which means that we need to have some extra definitions to address the generalization of h and p as follows.

σ_b^{max} for each $b \in B$, denotes $\sigma_b \in \Sigma_b$ that maximizes the value of $\sum_{a \in \Gamma(b)} |\pi_{(a,b)}^{-1}(\sigma_b)|$.

p_e^{max} for each edge $e = (a, b)$, denotes $\left|\pi_e^{-1}(\sigma_b^{max})\right|$, the size of the preimage of e if b is assigned σ_b^{max}.

$\overline{p}^{max}$ denotes the average of p_e^{max} over all $e \in E$, i.e. $\frac{1}{|E|} \sum_{e \in E} p_e^{max}$. We will use $2\overline{p}^{max}$ as a thershold for determining "good" edges as we shall see below.

$E(S)$ for each set of vertices S, denotes the set of edges with at least one endpoint in S, i.e. $\{(u, v) \in E \mid u \in S \text{ or } v \in S\}$.

E_N^{max} denotes the maximum number of edges coming out of $\Gamma(a)$ for all $a \in A$, i.e., $max_{a \in A}\{|E(\Gamma(a))|\}$.

E' denotes the set of all edges $e \in E$ such that $p_e \leq 2\overline{p}^{max}$, i.e., $E' = \{e \in E \mid p_e \leq 2\overline{p}^{max}\}$.

G' denotes a subgraph of G with its edges being E'.

$E'(S)$ for each set of vertices S, denotes the set of all edges in E' with at least one endpoint in S, i.e., $\{(u, v) \in E' \mid u \in S \text{ or } v \in S\}$.

E'_S for each set of vertices S, denotes the set of edges with both endpoints in S, i.e. $E'_S = \{(a, b) \in E' \mid a \in S \text{ and } b \in S\}$.

$\Gamma'(u)$ for each vertex u, denotes the set of vertices that are neighbors of u in the graph G'.

$\Gamma'(U)$ for each set of vertices U, denotes the set of vertices that are neighbors of at least one vertex in U in the graph G'.

$\Gamma'_2(u)$ for each vertex u, denotes $\Gamma'(\Gamma'(u))$, the set of neighbors of neighbors of u in G'.

$\Sigma_A^*(a)$ for each $a \in A$, denotes the set of all assignments σ_a to a that, for every $b \in B$, there exists an assignment σ_b such that, if a is assigned σ_a,

b is assigned σ_b and all a's neighbors are assigned according to a, then there are still possible assignments left for all vertices in $\Gamma_2(a) \cap \Gamma(b)$, i.e., $\{\sigma_a \in \Sigma_A \mid$ for each $b \in B$, there is $\sigma_b \in \Sigma_B$ such that, for all $a' \in \Gamma_2(a) \cap \Gamma(b)$, $\left(\bigcap_{b' \in \Gamma(a') \cap \Gamma(a)} \pi^{-1}_{(a',b')}(\pi_{(a,b')}(\sigma_a))\right) \cap \pi^{-1}_{(a',b)}(\sigma_b) \neq \emptyset\}$. Note that $\sigma_a^{OPT} \in \Sigma_A^*(a)$. In other words, if we replace Σ_A with $\Sigma_A^*(a)$ for each $a \in A$, then the resulting instance is still satisfiable.

$\Gamma^*(a, \sigma_a)$ for each $a \in A$ and $\sigma_a \in \Sigma_A^*(a)$, denotes $\{b \in \Gamma(a) \mid |\pi^{-1}_{(a',b)}(\pi_{(a,b)}(\sigma_a))| \leq 2\overline{p}^{max}$ for some $a' \in \Gamma(b)\}$. Provided that we assign σ_a to a, this set contains all the neighbors of a with at least one good edge as we discussed above. Note that $\pi_{(a,b)}(\sigma_a)$ is the assignment to b corresponding to the assignment of a.

$\Gamma_2^*(a, \sigma_a)$ for each $a \in A$ and $\sigma_a \in \Sigma_A^*(a)$, denotes all the neighbors of neighbors of a with at least one good edge with another endpoint in $\Gamma(a)$ when a is assigned σ_a, i.e., $\bigcup_{b \in \Gamma^*(a,\sigma_a)}\{a' \in \Gamma(b) \mid |\pi^{-1}_{(a',b)}(\pi_{(a,b)}(\sigma_a))| \leq 2\overline{p}^{max}\}$.

$h^*(a, \sigma_a)$ for each $a \in A$ and $\sigma_a \in \Sigma_A^*(a)$, denotes $|E(\Gamma_2^*(a, \sigma_a))|$. In other words, $h^*(a, \sigma_a)$ represents how well $\Gamma_2^*(a, \sigma_a)$ spans the graph G.

$E^*(a, \sigma_a)$ for each $a \in A$ and $\sigma_a \in \Sigma_A^*(a)$, denotes $\{(a', b) \in E \mid b \in \Gamma^*(a, \sigma_a)$, $a' \in \Gamma_2^*(a, \sigma_a)$ and $|\pi^{-1}_{(a',b)}(\pi_{(a,b)}(\sigma_a))| \leq 2\overline{p}^{max}\}$. When a is assigned σ_a, this is the set of all good edges with one endpoint in $\Gamma(a)$.

h^*_{max} denotes $\max_{a \in A, \sigma_a \in \Sigma_A^*(a)} h^*(a, \sigma_a)$.

From the definitions above, we can derive two very useful observations as stated below.

Observation A.1. $|E'| \geq \frac{|E|}{2}$

Proof. Suppose for the sake of contradiction that $|E'| < \frac{|E|}{2}$. From the definition of E', this means that, for more than $\frac{|E|}{2}$ edges e, we have $p_e > 2\overline{p}^{max}$. As a result, we

can conclude that

$$\begin{aligned}
|E|\overline{p}^{max} &< \sum_{e \in E} p_e \\
&= \sum_{b \in B} \sum_{a \in \Gamma(b)} p_{(a,b)} \\
&= \sum_{b \in B} \sum_{a \in \Gamma(b)} |\pi^{-1}_{(a,b)}(\sigma_b^{OPT})| \\
&\leq \sum_{b \in B} \sum_{a \in \Gamma(b)} |\pi^{-1}_{(a,b)}(\sigma_b^{max})| \\
&= |E|\overline{p}^{max}.
\end{aligned}$$

This is a contradiction. Hence, $|E'| \geq \frac{|E|}{2}$. □

Observation A.2. *If* $\sigma_a = \sigma_a^{OPT}$, *then* $\Gamma^*(a, \sigma_a) = \Gamma'(a)$, $\Gamma_2^*(a, \sigma_a) = \Gamma_2'(a)$ *and* $E^*(a, \sigma_a) = E'(\Gamma'(a))$.

This observation is obvious since, when pluging in σ_a^{OPT}, each pair of definitions of $\Gamma^*(a, \sigma_a)$ and $\Gamma'(a)$, $\Gamma_2^*(a, \sigma_a)$ and $\Gamma_2'(a)$, and $E^*(a, \sigma_a)$ and $E'(\Gamma'(a))$ becomes the same.

Note also that from its definition, G' is the graph with good edges when the optimal assignments are assigned to B. Unfortunately, we do not know the optimal assignments to B and, thus, do not know how to find G' in polynomial time. However, directly from the definitions above, σ_b^{max}, p_e^{max}, $\overline{p}^{max}$, E_N^{max}, $\Sigma_A^*(a)$, $\Gamma^*(a, \sigma_a)$, $\Gamma_2^*(a, \sigma_a)$, $h^*(a, \sigma_a)$ and h_{max}^* can be computed in polynomial time. These notations will be used in the upcoming algorithms. Other defined notations we do not know how to compute in polynomial time and will only be used in the analyses.

For the nonuniform preimage sizes case, we use five algorithms as opposed to four algorithms used in uniform case. We will proceed to describe those five al-

gorithms. In the end, by using the best of these five, we are able to produce a polynomial-time $O\left((n_A|\Sigma_A|)^{1/4}\right)$-approximation algorithm as desired.

Now, we will list the algorithms along with their rough descriptions; detailed description and analysis of each algorithm will follow later on:

1. **Satisfy one neighbor – $|E|/n_B$-approximation.** Assign each vertex in A an arbitrary assignment. Each vertex in B is then assigned to satisfy one of its neighboring edges. This algorithm satisfies at least n_B edges.

2. **Greedy assignment – $|\Sigma_A|/\overline{p}^{max}$-approximation.** Each vertex in B is assigned an assignment $\sigma_b \in \Sigma_B$ that has the largest number of preimages across neighboring edges $\sum_{a\in\Gamma(b)} |\pi^{-1}_{(a,b)}(\sigma_b)|$. Each vertex in A is then assigned so that it satisfies as many edges as possible. This algorithm works well when Σ_B assignments have many preimages.

3. **Know your neighbors – $|E|/E_N^{max}$-approximation.** For a vertex $a_0 \in A$, pick an element of $\Sigma_A^*(a_0)$ and assign it to a_0. Assign its neighbors $\Gamma(a_0)$ accordingly. Then, for each node in $\Gamma_2(a_0)$, we find one assignment that satisfies all the edges between it and vertices in $\Gamma(a_0)$.

4. **Know your neighbors' neighbors – $|E|\overline{p}^{max}/h^*_{max}$-approximation.** For a vertex $a_0 \in A$, we go over all possible assignments in $\Sigma_A^*(a)$ to it. For each assignment, we assign its neighbors $\Gamma(a_0)$ accordingly. Then, for each node in $\Gamma_2(a_0)$, we keep only the assignments that satisfy all the edges between it and vertices in $\Gamma(a_0)$.

   When a_0 is assigned the optimal assignment, the number of choices for each node in $\Gamma_2^*(a_0)$ is reduced to at most $2\overline{p}^{max}$ possibilities. In this way, we can satisfy $1/2\overline{p}^{max}$ fraction of the edges that touch $\Gamma_2^*(a_0)$. This satisfies many edges when there exists $a_0 \in A$ such that $\Gamma_2^*(a_0)$ spans many edges.

5. **Divide and Conquer – $O(n_A n_B(h^*_{max} + E^{max}_N)/|E|^2)$-approximation.** For every $a \in A$, we can fully satisfy $\Gamma^*(a) \cup \Gamma^*_2(a)$ efficiently, and give up on satisfying other edges that touch this subset. Repeating this process, we can satisfy $\Omega(|E|^2/(n_A n_B(h^*_{max} + E^{max}_N)))$ fraction of the edges.

Aside from the new "know your neighbors" algorithm, the main idea of each algorithm remains the same as in the uniform preimage sizes case. All the details of each algorithm are described below.

A.1.1 Satisfy One Neighbor Algorithm.

The algorithm is exactly the same as that of the uniform case.

Lemma A.1. *For satisfiable instances of projection games, an assignment that satisfies at least n_B edges can be found in polynomial time, which gives the approximation ratio of $\frac{|E|}{n_B}$.*

Proof. The proof is exactly the same as that of Lemma 3.1. □

A.1.2 Greedy Assignment Algorithm.

The algorithm is exactly the same as that of the uniform case.

Lemma A.2. *There exists a polynomial-time $\frac{|\Sigma_A|}{\bar{p}^{max}}$-approximation algorithm for satisfiable instances of projection games.*

Proof. The proof of this lemma differs only slightly from the proof of Lemma 3.2.

The algorithm works as follows:

1. For each b, assign it σ^*_b that maximizes $\sum_{a \in \Gamma(b)} |\pi^{-1}_{(a,b)}(\sigma_b)|$.
2. For each a, assign it σ^*_a that maximizes the number of edges satisfied, $|\{b \in \Gamma(a) \mid \pi_{(a,b)}(\sigma_a) = \sigma^*_b\}|$.

Let e^* be the number of edges that get satisfied by this algorithm. We have

$$e^* = \sum_{a \in A} |\{b \in \Gamma(a) \mid \pi_{(a,b)}(\sigma_a^*) = \sigma_b^*\}|.$$

By the second step, for each $a \in A$, the number of edges satisfied is at least an average of the number of edges satisfied over all assignments in Σ_A. This can be written as follows.

$$\begin{aligned} e^* &= \sum_{a \in A} |\{b \in \Gamma(a) \mid \pi_{(a,b)}(\sigma_a^*) = \sigma_b^*\}| \\ &\geq \sum_{a \in A} \frac{\sum_{\sigma_a \in \Sigma_A} |\{b \in \Gamma(a) \mid \pi_{(a,b)}(\sigma_a) = \sigma_b^*\}|}{|\Sigma_A|} \\ &= \sum_{a \in A} \frac{\sum_{b \in \Gamma(a)} |\pi_{(a,b)}^{-1}(\sigma_b^*)|}{|\Sigma_A|} \\ &= \frac{1}{|\Sigma_A|} \sum_{a \in A} \sum_{b \in \Gamma(a)} |\pi_{(a,b)}^{-1}(\sigma_b^*)|. \end{aligned}$$

From the definition of σ_b^{max}, we can conclude that $\sigma_b^* = \sigma_b^{max}$ for all $b \in B$. As a result, we can conclude that

$$\begin{aligned} e^* &\geq \frac{1}{|\Sigma_A|} \sum_{a \in A} \sum_{b \in \Gamma(a)} |\pi_{(a,b)}^{-1}(\sigma_b^*)| \\ &= \frac{1}{|\Sigma_A|} \sum_{a \in A} \sum_{b \in \Gamma(a)} |\pi_{(a,b)}^{-1}(\sigma_b^{max})| \\ &= \frac{1}{|\Sigma_A|} \sum_{a \in A} \sum_{b \in \Gamma(a)} p_{(a,b)}^{max} \\ &= \frac{1}{|\Sigma_A|} |E| |\overline{p}^{max}| \\ &= \frac{\overline{p}^{max}}{|\Sigma_A|} |E|. \end{aligned}$$

Hence, this algorithm satisfies at least $\frac{\overline{p}^{max}}{|\Sigma_A|}$ fraction of the edges, which concludes our proof. □

A.1.3 Know Your Neighbors Algorithm

The next algorithm shows that one can satisfy all the edges with one endpoint in the neighbors of a vertex $a_0 \in A$.

Lemma A.3. *For each $a_0 \in A$, there exists a polynomial time $\frac{|E|}{|E(\Gamma(a_0))|}$-approximation algorithm for satisfiable instances of projection games.*

Proof. The algorithm works as follows:

1. Pick any assignment $\sigma_{a_0} \in \Sigma_A^*(a_0)$ and assign it to a_0:
2. Assign $\sigma_b = \pi_{(a_0,b)}(\sigma_{a_0})$ to b for all $b \in \Gamma(a_0)$.
3. For each $a \in \Gamma_2(a_0)$, find the set of plausible assignments to a, i.e., $S_a = \{\sigma_a \in \Sigma_A \mid \forall b \in \Gamma(a) \cap \Gamma(a_0), \pi_{(a,b)}(\sigma_a) = \sigma_b\}$. Pick one σ_a^* from this set and assign it to a. Note that $S_a \neq \emptyset$ from the definition of $\Sigma_A^*(a_0)$.
4. Assign any assignment to unassigned vertices.
5. Output the assignment $\{\sigma_a^*\}_{a\in A}$, $\{\sigma_b^*\}_{b\in B}$ from the previous step.

From step 3, we can conclude that all the edges in $E(\Gamma(a_0))$ get statisfied. This yields $\frac{|E|}{|E(\Gamma(a_0))|}$ approximation ratio as desired. □

A.1.4 Know Your Neighbors' Neighbors Algorithm

The next algorithm shows that if the neighbors of neighbors of a vertex $a_0 \in A$ expand, then one can satisfy many of the (many!) edges that touch the neighbors of

a_0's neighbors. While the core idea is similar to the uniform version, in this version, we will need to consider $\Gamma_2^*(a_0, \sigma_{a_0})$ instead of $\Gamma_2(a_0)$ in order to ensure that the number of possible choices left for each vertex in this set is at most $2\overline{p}^{max}$.

Lemma A.4. *For each $a_0 \in A$ and $\sigma_{a_0} \in \Sigma_A^*(a_0)$, there exists a polynomial-time $O\left(\frac{|E|\overline{p}^{max}}{h^*(a_0,\sigma_{a_0})}\right)$-approximation algorithm for satisfiable instances of projection games.*

Proof. To prove Lemma A.4, we first fix $a_0 \in A$ and $\sigma_{a_0} \in \Sigma_A^*(a_0)$. We will describe an algorithm that satisfies $\Omega\left(\frac{h^*(a_0,\sigma_{a_0})}{\overline{p}^{max}}\right)$ edges, which implies the lemma.

The algorithm works as follows:

1. Assign $\sigma_b = \pi_{(a_0,b)}(\sigma_{a_0})$ to b for all $b \in \Gamma(a_0)$.
2. For each $a \in A$, find the set of plausible assignments to a, i.e., $S_a = \{\sigma_a \in \Sigma_A \mid \forall b \in \Gamma(a) \cap \Gamma(a_0), \pi_{(a,b)}(\sigma_a) = \sigma_b\}$. Note that $S_a \neq \emptyset$ from the definition of $\Sigma_A^*(a_0)$.
3. For all $b \in B$, pick an assignment σ_b^* for b that maximizes the average number of satisfied edges over all assignments in S_a to vertices a in $\Gamma(b) \cap \Gamma_2^*(a_0)$, i.e., maximizes $\sum_{a \in \Gamma(b) \cap \Gamma_2^*(a_0)} |\pi_{(a,b)}^{-1}(\sigma_b) \cap S_a|$.
4. For each vertex $a \in A$, pick an assignment $\sigma_a^* \in S_a$ that maximizes the number of satisfied edges, $|\{b \in \Gamma(a) \mid \pi_{(a,b)}(\sigma_a) = \sigma_b^*\}|$ over all $\sigma_a \in S_a$.

We will prove that this algorithm indeed satisfies at least $\frac{h^*(a_0,\sigma_{a_0})}{\overline{p}^{max}}$ edges.

Let e^* be the number of edges satisfied by the algorithm. We have

$$e^* = \sum_{a \in A} |\{b \in \Gamma(a) \mid \pi_{(a,b)}(\sigma_a^*) = \sigma_b^*\}|.$$

Since for each $a \in A$, the assignment σ_a^* is chosen to maximize the number of edges satisfied, we can conclude that the number of edges satisfied by selecting σ_a^* is at least the average of the number of edges satisfied over all $\sigma_a \in S_a$.

As a result, we can conclude that

$$\begin{aligned}
e^* &\geq \sum_{a\in A} \frac{\sum_{\sigma_a\in S_a} |\{b \in \Gamma(a) \mid \pi_{(a,b)}(\sigma_a) = \sigma_b^*\}|}{|S_a|} \\
&= \sum_{a\in A} \frac{\sum_{\sigma_a\in S_a} \sum_{b\in\Gamma(a)} 1_{\pi_{(a,b)}(\sigma_a)=\sigma_b^*}}{|S_a|} \\
&= \sum_{a\in A} \frac{\sum_{b\in\Gamma(a)} \sum_{\sigma_a\in S_a} 1_{\pi_{(a,b)}(\sigma_a)=\sigma_b^*}}{|S_a|} \\
&= \sum_{a\in A} \frac{\sum_{b\in\Gamma(a)} |\pi_{(a,b)}^{-1}(\sigma_b^*) \cap S_a|}{|S_a|} \\
&= \sum_{b\in B} \sum_{a\in\Gamma(b)} \frac{|\pi_{(a,b)}^{-1}(\sigma_b^*) \cap S_a|}{|S_a|} \\
&\geq \sum_{b\in B} \sum_{a\in\Gamma(b)\cap\Gamma_2^*(a_0,\sigma_{a_0})} \frac{|\pi_{(a,b)}^{-1}(\sigma_b^*) \cap S_a|}{|S_a|}
\end{aligned}$$

From the definition of $\Gamma_2^*(a_0, \sigma_{a_0})$, we can conclude that, for each $a \in \Gamma_2^*(a_0, \sigma_{a_0})$, there exists $b' \in \Gamma^*(a_0) \cap \Gamma(a)$ such that $|\pi_{(a,b')}^{-1}(\sigma_{b'})| \leq 2\overline{p}^{max}$. Moreover, from the definition of S_a, we have $S_a \subseteq \pi_{(a,b')}^{-1}(\sigma_{b'})$. As a result, we can arrive at the following inequalities.

$$\begin{aligned}
|S_a| &\leq |\pi_{(a,b')}^{-1}(\sigma_{b'})| \\
&\leq 2\overline{p}^{max}.
\end{aligned}$$

This implies that

$$e^* \geq \frac{1}{2\overline{p}^{max}} \sum_{b\in B} \sum_{a\in\Gamma(b)\cap\Gamma_2^*(a_0,\sigma_{a_0})} |\pi_{(a,b)}^{-1}(\sigma_b^*) \cap S_a|.$$

From the definition of $\Sigma_A^*(a_0)$, we can conclude that, for each $b \in B$, there exists

$\sigma_b \in B$ such that $\pi^{-1}_{(a,b)}(\sigma_b) \cap S_a \neq \emptyset$ for all $a \in \Gamma_2(a_0) \cap \Gamma(b)$. Since $\Gamma^*_2(a_0, \sigma_{a_0}) \subseteq \Gamma_2(a_0)$, we can conclude that $|\pi^{-1}_{(a,b)}(\sigma_b) \cap S_a| \geq 1$ for all $a \in \Gamma^*_2(a_0, \sigma_{a_0}) \cap \Gamma(b)$.

Since we pick the assignment σ^*_b that maximizes $\sum_{a \in \Gamma(b) \cap \Gamma^*_2(a_0)} |\pi^{-1}_{(a,b)}(\sigma^*_b) \cap S_a|$ for each $b \in B$, we can conclude that

$$\begin{aligned} e^* &\geq \frac{1}{2\overline{p}^{max}} \sum_{b \in B} \sum_{a \in \Gamma(b) \cap \Gamma^*_2(a_0, \sigma_{a_0})} |\pi^{-1}_{(a,b)}(\sigma^*_b) \cap S_a| \\ &\geq \frac{1}{2\overline{p}^{max}} \sum_{b \in B} \sum_{a \in \Gamma(b) \cap \Gamma^*_2(a_0, \sigma_{a_0})} |\pi^{-1}_{(a,b)}(\sigma_b) \cap S_a| \\ &\geq \frac{1}{2\overline{p}^{max}} \sum_{b \in B} \sum_{a \in \Gamma(b) \cap \Gamma^*_2(a_0, \sigma_{a_0})} 1. \end{aligned}$$

The last term can be rewritten as

$$\begin{aligned} \frac{1}{2\overline{p}^{max}} \sum_{b \in B} \sum_{a \in \Gamma(b) \cap \Gamma^*_2(a_0, \sigma_{a_0})} 1 &= \frac{1}{2\overline{p}^{max}} \sum_{a \in \Gamma^*_2(a_0, \sigma_{a_0})} \sum_{b \in \Gamma(a)} 1 \\ &= \frac{1}{2\overline{p}^{max}} \sum_{a \in \Gamma^*_2(a_0, \sigma_{a_0})} d_a \\ &= \frac{h^*(a_0, \sigma_{a_0})}{2\overline{p}^{max}}. \end{aligned}$$

As a result, we can conclude that this algorithm gives an assignment that satisfies at least $\frac{h^*(a_0, \sigma_{a_0})}{2\overline{p}^{max}}$ edges out of all the $|E|$ edges. Hence, this is a polynomial-time $O\left(\frac{|E|\overline{p}^{max}}{h^*(a_0, \sigma_{a_0})}\right)$-approximation algorithm as desired. □

A.1.5 Divide and Conquer Algorithm.

We will present an algorithm that separates the graph into disjoint subgraphs for which we can find the optimal assignments in polynomial time. We shall show

below that, if $h^*(a, \sigma_a)$ is small for all $a \in A$ and $\sigma_a \in \Sigma_A^*(a)$, then we are able to find such subgraphs that contain most of the graph's edges.

Lemma A.5. *There exists a polynomial-time $O\left(\frac{n_A n_B (h^*_{max} + E_N^{max})}{|E|^2}\right)$-approximation algorithm for satisfiable instances of projection games.*

Proof. To prove this lemma, we will present an algorithm that gives an assignment that satisfies $\Omega\left(\frac{|E|^3}{n_A n_B (h^*_{max} + E_N^{max})}\right)$ edges.

We use $\mathcal{P}$ to represent the collection of subgraphs we find. The family $\mathcal{P}$ consists of disjoint sets of vertices. Let $V_{\mathcal{P}}$ be $\bigcup_{P \in \mathcal{P}} P$.

For any set S of vertices, define G_S to be the graph induced on S with respect to G. Moreover, define E_S to be the set of edges of G_S. We also define $E_{\mathcal{P}} = \bigcup_{P \in \mathcal{P}} E_P$. Note that E_S is similar to E'_S defined earlier in the appendix. The only difference is that E'_S is with respect to G' instead of G.

The algorithm works as follows.

1. Set $\mathcal{P} \leftarrow \emptyset$.

2. While there exists a vertex $a \in A$ and $\sigma_a \in \Sigma_A^*(a)$ such that

$$|E^*(a, \sigma_a) \cap E_{(A \cup B) - V_{\mathcal{P}}}| \geq \frac{1}{16} \frac{|E|^2}{n_A n_B} :$$

   (a) Set $\mathcal{P} \leftarrow \mathcal{P} \cup \{(\Gamma_2^*(a, \sigma_a) \cup \Gamma^*(a, \sigma_a)) - V_{\mathcal{P}}\}$.

3. For each $P \in \mathcal{P}$, find in time $poly(|\Sigma_A|, |P|)$ an assignment to the vertices in P that satisfies all the edges spanned by P. This can be done easily by assigning σ_a to a and $\pi_{(a,b)}(\sigma_a)$ to $b \in B \cap P$. Then assign any plausible assignment to all the other vertices in $A \cap P$.

We will divide the proof into two parts. First, we will show that when we cannot find a vertex a and an assignment $\sigma_a \in \Sigma_A^*(a)$ in step 2, $\left|E_{(A\cup B)-V_{\mathcal{P}}}\right| \leq \frac{3|E|}{4}$. Second, we will show that the resulting assignment from this algorithm satisfies $\Omega\left(\frac{|E|^3}{n_A n_B (h_{max}^* + E_N^{max})}\right)$ edges.

We will start by showing that, if no vertex a and an assignment $\sigma_a \in \Sigma_A^*(a)$ in step 2 exist, then $\left|E_{(A\cup B)-V_{\mathcal{P}}}\right| \leq \frac{3|E|}{4}$.

Suppose that we cannot find a vertex a and an assignment $\sigma_a \in \Sigma_A^*(a)$ in step 2. In other words, $|E^*(a,\sigma_a) \cap E_{(A\cup B)-V_{\mathcal{P}}}| < \frac{1}{16}\frac{|E|^2}{n_A n_B}$ for all $a \in A$ and $\sigma_a \in \Sigma_A^*(a)$.

Since $\sigma_a^{OPT} \in \Sigma_A^*(a)$ for all $a \in A$, we can conclude that

$$|E^*(a,\sigma_a^{OPT}) \cap E_{(A\cup B)-V_{\mathcal{P}}}| < \frac{1}{16}\frac{|E|^2}{n_A n_B}.$$

From Observation A.2, we have $E^*(a,\sigma^{OPT}) = E'(\Gamma'(a))$. As a result, we have

$$\begin{aligned}\frac{1}{16}\frac{|E|^2}{n_A n_B} &> |E^*(a,\sigma_a^{OPT}) \cap E_{(A\cup B)-V_{\mathcal{P}}}| \\ &= |E'(\Gamma'(a)) \cap E_{(A\cup B)-V_{\mathcal{P}}}|\end{aligned}$$

for all $a \in A$.

Since $E'(\Gamma'(a)) = E'_{\Gamma'(a)\cup\Gamma'_2(a)}$, we can rewrite the last term as

$$\begin{aligned}|E'(\Gamma'(a)) \cap E_{(A\cup B)-V_{\mathcal{P}}}| &= |E'_{\Gamma'(a)\cup\Gamma'_2(a)} \cap E_{(A\cup B)-V_{\mathcal{P}}}| \\ &= |E'_{\Gamma'(a)\cup\Gamma'_2(a)-V_{\mathcal{P}}}|.\end{aligned}$$

Consider $\sum_{a\in A} |E'_{\Gamma'(a)\cup\Gamma'_2(a)-V_{\mathcal{P}}}|$. Since $|E'_{\Gamma'(a)\cup\Gamma'_2(a)-V_{\mathcal{P}}}| < \frac{1}{16}\frac{|E|^2}{n_A n_B}$ for all $a \in A$, we

have the following inequality:

$$\frac{|E|^2}{16n_B} > \sum_{a\in A} |E'_{\Gamma'(a)\cup\Gamma'_2(a)-V_{\mathcal{P}}}|.$$

Let $N^p(v) = \Gamma'(v) - V_{\mathcal{P}}$ and $N_2^p(v) = \Gamma'_2(v) - V_{\mathcal{P}}$. Similary, define $N^p(S)$ for a subset $S \subseteq A \cup B$. It is easy to see that $N_2^p(v) \supseteq N^p(N^p(v))$. This implies that, for all $a \in A$, we have $|E'_{N^p(a)\cup N_2^p(a)}| \geq |E'_{N^p(a)\cup N^p(N^p(a))}|$. Moreover, it is easy to see that, for all $a \in A - V_{\mathcal{P}}$, we have $|E'_{N^p(a)\cup N^p(N^p(a))}| = \sum_{b\in N^p(a)} |N^p(b)|$.

Thus, the following holds:

$$\begin{aligned}
\sum_{a\in A} |E'_{(\Gamma'(a)\cup\Gamma'_2(a))-V_{\mathcal{P}}}| &= \sum_{a\in A} |E_{(N^p(a)\cup N_2^p(a))}| \\
&\geq \sum_{a\in A-V_{\mathcal{P}}} |E_{(N^p(a)\cup N_2^p(a))}| \\
&= \sum_{a\in A-V_{\mathcal{P}}} \sum_{b\in N^p(a)} |N^p(b)| \\
&= \sum_{b\in B-V_{\mathcal{P}}} \sum_{a\in N^p(b)} |N^p(b)| \\
&= \sum_{b\in B-V_{\mathcal{P}}} |N^p(b)|^2.
\end{aligned}$$

From Jensen's inequality, we have

$$\begin{aligned}
\sum_{a\in A} |E'_{(\Gamma'(a)\cup\Gamma'_2(a))-V_{\mathcal{P}}}| &\geq \frac{1}{|B-V_{\mathcal{P}}|} \left(\sum_{b\in B-V_{\mathcal{P}}} |N^p(b)| \right)^2 \\
&= \frac{1}{|B-V_{\mathcal{P}}|} \left| E'_{(A\cup B)-V_{\mathcal{P}}} \right|^2 \\
&\geq \frac{1}{n_B} \left| E'_{(A\cup B)-V_{\mathcal{P}}} \right|^2.
\end{aligned}$$

Since $\frac{|E|^2}{16n_B} \geq \sum_{a \in A} |E_{(\Gamma'(a) \cup \Gamma'_2(a)) - V_{\mathcal{P}}}|$ and $\sum_{a \in A} |E_{(\Gamma'(a) \cup \Gamma'_2(a)) - V_{\mathcal{P}}}| \geq \frac{1}{n_B} \left| E'_{(A \cup B) - V_{\mathcal{P}}} \right|^2$, we can conclude that

$$\frac{|E|}{4} \geq \left| E'_{(A \cup B) - V_{\mathcal{P}}} \right|.$$

Consider $E'_{(A \cup B) - V_{\mathcal{P}}}$ and $E_{(A \cup B) - V_{\mathcal{P}}}$. We have

$$\begin{aligned} E'_{(A \cup B) - V_{\mathcal{P}}} \cup (E - E') &\supseteq E_{(A \cup B) - V_{\mathcal{P}}} \\ \left| E'_{(A \cup B) - V_{\mathcal{P}}} \right| + \left| E - E' \right| &\geq \left| E_{(A \cup B) - V_{\mathcal{P}}} \right| \\ \frac{|E|}{4} + \left| E - E' \right| &\geq \left| E_{(A \cup B) - V_{\mathcal{P}}} \right|. \end{aligned}$$

From Observation A.1, we have $|E'| \geq \frac{|E|}{2}$. Thus, we have

$$\frac{3|E|}{4} \geq \left| E_{(A \cup B) - V_{\mathcal{P}}} \right|,$$

which concludes the first part of the proof.

Next, we will show that the assignment the algorithm finds satisfies at least $\Omega\left(\frac{|E|^3}{n_A n_B (h^*_{max} + E^{max}_N)}\right)$ edges. Since we showed that $\frac{3|E|}{4} \geq \left| E_{(A \cup B) - V_{\mathcal{P}}} \right|$ when the algorithm terminates, it is enough to prove that $|E_{\mathcal{P}}| \geq \frac{|E|^2}{16 n_A n_B (h^*_{max} + E^{max}_N)} \left(|E| - \left| E_{(A \cup B) - V_{\mathcal{P}}} \right| \right)$. Note that the algorithm guarantees to satisfy all the edges in $E_{\mathcal{P}}$.

We will prove this by using induction to show that at any point in the algorithm, $|E_{\mathcal{P}}| \geq \frac{|E|^2}{16 n_A n_B (h^*_{max} + E^{max}_N)} \left(|E| - \left| E_{(A \cup B) - V_{\mathcal{P}}} \right| \right)$.

Base Case. At the beginning, we have $|E_{\mathcal{P}}| = 0 = \frac{|E|^2}{16 n_A n_B (h^*_{max} + E^{max}_N)} \left(|E| - \left| E_{(A \cup B) - V_{\mathcal{P}}} \right| \right)$, which satisfies the inequality.

Inductive Step. The only step in the algorithm where any term in the inequality changes is step 2a. Let $\mathcal{P}_{old}$ and $\mathcal{P}_{new}$ be the set $\mathcal{P}$ before and after step 2a is executed,

respectively. Let a be the vertex selected in step 2. Suppose that $\mathcal{P}_{old}$ satisfies the inequality.

Since $|E_{\mathcal{P}_{new}}| = |E_{\mathcal{P}_{old}}| + |E_{(\Gamma^*(a,\sigma_a)\cup\Gamma_2^*(a,\sigma_a))-V_{\mathcal{P}_{old}}}|$, we have

$$\begin{aligned}|E_{\mathcal{P}_{new}}| &= |E_{\mathcal{P}_{old}}| + |E_{(\Gamma^*(a,\sigma_a)\cup\Gamma_2^*(a,\sigma_a))-V_{\mathcal{P}_{old}}}| \\ &= |E_{\mathcal{P}_{old}}| + |E_{(\Gamma^*(a,\sigma_a)\cup\Gamma_2^*(a,\sigma_a))} \cap E_{(A\cup B)-V_{\mathcal{P}_{old}}}|.\end{aligned}$$

From the condition in step 2, we have $|E^*(a,\sigma_a) \cap E_{(A\cup B)-V_{\mathcal{P}_{old}}}| \geq \frac{1}{16}\frac{|E|^2}{n_A n_B}$. Moreover, $E_{(\Gamma^*(a,\sigma_a)\cup\Gamma_2^*(a,\sigma_a))} \supseteq E^*(a,\sigma_a)$ holds. As a result, we have

$$\begin{aligned}|E_{\mathcal{P}_{new}}| &= |E_{\mathcal{P}_{old}}| + |E_{(\Gamma^*(a,\sigma_a)\cup\Gamma_2^*(a,\sigma_a))} \cap E_{A\cup B-V_{\mathcal{P}_{old}}}| \\ &\geq |E_{\mathcal{P}_{old}}| + |E^*(a,\sigma_a) \cap E_{(A\cup B)-V_{\mathcal{P}_{old}}}| \\ &\geq |E_{\mathcal{P}_{old}}| + \frac{1}{16}\frac{|E|^2}{n_A n_B}.\end{aligned}$$

Now, consider $\Big(|E| - |E_{(A\cup B)-V_{\mathcal{P}_{new}}}|\Big) - \Big(|E| - |E_{(A\cup B)-V_{\mathcal{P}_{old}}}|\Big)$. We have

$$\Big(|E| - |E_{(A\cup B)-V_{\mathcal{P}_{new}}}|\Big) - \Big(|E| - |E_{(A\cup B)-V_{\mathcal{P}_{old}}}|\Big) = |E_{(A\cup B)-V_{\mathcal{P}_{old}}}| - |E_{(A\cup B)-V_{\mathcal{P}_{new}}}|$$

Since $V_{\mathcal{P}_{new}} = V_{\mathcal{P}_{old}} \cup (\Gamma_2^*(a,\sigma_a) \cup \Gamma^*(a,\sigma_a))$, we can conclude that

$$\big((A\cup B) - V_{\mathcal{P}_{old}}\big) \subseteq \big((A\cup B) - V_{\mathcal{P}_{new}}\big) \cup \big(\Gamma_2^*(a,\sigma_a) \cup \Gamma^*(a,\sigma_a)\big).$$

Thus, we can also derive

$$\begin{aligned}E_{(A\cup B)-V_{\mathcal{P}_{old}}} &\subseteq E_{\big((A\cup B)-V_{\mathcal{P}_{new}}\big)\cup\big(\Gamma_2^*(a,\sigma_a)\cup\Gamma^*(a,\sigma_a)\big)} \\ &= E_{(A\cup B)-V_{\mathcal{P}_{new}}} \cup \{(a',b') \in E \mid a' \in \Gamma_2^*(a,\sigma_a) \text{ or } b' \in \Gamma^*(a,\sigma_a)\}.\end{aligned}$$

Moreover, we can write $\{(a', b') \in E \mid a' \in \Gamma_2^*(a, \sigma_a) \text{ or } b' \in \Gamma^*(a, \sigma_a)\}$ as $\{(a', b') \in E \mid a' \in \Gamma_2^*(a, \sigma_a)\} \cup \{(a', b') \in E \mid b' \in \Gamma^*(a, \sigma_a)\}$. Since $\Gamma^*(a, \sigma_a) \subseteq \Gamma(a)$, we can conclude that

$$\begin{aligned}\{(a', b') \in E \mid a' \in \Gamma_2^*(a, \sigma_a) \text{ or } b' \in \Gamma^*(a, \sigma_a)\} \subseteq &\{(a', b') \in E \mid a' \in \Gamma_2^*(a, \sigma_a)\} \\ &\cup \{(a', b') \in E \mid b' \in \Gamma(a)\}.\end{aligned}$$

Thus, we can conclude that

$$\begin{aligned}|\{(a', b') \in E \mid a' \in \Gamma_2^*(a, \sigma_a) \text{ or } b' \in \Gamma^*(a, \sigma_a)\}| &\leq |\{(a', b') \in E \mid a' \in \Gamma_2^*(a, \sigma_a)\}| \\ &\quad + |\{(a', b') \in E \mid b' \in \Gamma(a)\}| \\ &= h^*(a, \sigma_a) + |E(\Gamma(a))|.\end{aligned}$$

Hence, we can conclude that

$$\begin{aligned}\left|E_{(A\cup B)-V_{\mathcal{P}_{old}}}\right| &\leq \left|E_{(A\cup B)-V_{\mathcal{P}_{new}}} \cup \{(a', b') \in E \mid a' \in \Gamma_2(a) \text{ or } b' \in \Gamma(a)\}\right| \\ &\leq \left|E_{(A\cup B)-V_{\mathcal{P}_{new}}}\right| + \left|\{(a', b') \in E \mid a' \in \Gamma_2(a) \text{ or } b' \in \Gamma(a)\}\right| \\ &\leq \left|E_{(A\cup B)-V_{\mathcal{P}_{new}}}\right| + h^*(a, \sigma_a) + |E(\Gamma(a))| \\ &\leq \left|E_{(A\cup B)-V_{\mathcal{P}_{new}}}\right| + h^*_{max} + E^{max}_N.\end{aligned}$$

This implies that $\left(|E| - \left|E_{(A\cup B)-V_{\mathcal{P}}}\right|\right)$ increases by at most $h^*_{max} + E^{max}_N$.

Hence, since $\left(|E| - \left|E_{(A\cup B)-V_{\mathcal{P}}}\right|\right)$ increases by at most $h^*_{max} + E^{max}_N$ and $|E_{\mathcal{P}}|$ in-

creases by at least $\frac{1}{16}\frac{|E|^2}{n_A n_B}$ and from the inductive hypothesis, we can conclude that

$$|E_{\mathcal{P}_{new}}| \geq \frac{|E|^2}{16 n_A n_B (h^*_{max} + E_N^{max})} \left(|E| - \left| E_{(A \cup B) - V_{\mathcal{P}_{new}}} \right| \right).$$

Thus, the inductive step is true and the inequality holds at any point during the execution of the algorithm.

When the algorithm terminates, since $|E_{\mathcal{P}}| \geq \frac{|E|^2}{16 n_A n_B (h^*_{max} + E_N^{max})} \left(|E| - \left| E_{(A \cup B) - V_{\mathcal{P}}} \right| \right)$ and $\frac{3|E|}{4} \geq \left| E_{(A \cup B) - V_{\mathcal{P}}} \right|$, we can conclude that $|E_{\mathcal{P}}| \geq \frac{|E|^3}{64 n_A n_B (h^*_{max} + E_N^{max})}$. Since the algorithm guarantees to satisfy every edge in $E_{\mathcal{P}}$, it yields an $O\left(\frac{n_A n_B (h^*_{max} + E_N^{max})}{|E|^2}\right)$ approximation ratio, which concludes our proof of Lemma A.5. □

A.1.6 Proof of the Main Theorem

Proof. Using Lemma A.4 with a_0 and σ_{a_0} that maximizes the value of $h^*(a_0, \sigma_{a_0})$, i.e., $h^*(a_0, \sigma_{a_0}) = h^*_{max}$, we can conclude that there exists a polynomial-time $O\left(\frac{|E| \overline{p}^{max}}{h^*_{max}}\right)$-approximation algorithm for satisfiable instances of projection games.

Similarly, from Leamma A.3 with a_0 that maximizes the value of $E(\Gamma(a_0))$, i.e., $|E(\Gamma(a_0))| = E_N^{max}$, there exists a polynomial-time $\frac{|E|}{E_N^{max}}$-approximation algorithm for satisfiable instances of projection games.

Moreover, from Lemmas A.1, A.2 and A.5, there exists a polynomial-time $\frac{|E|}{n_B}$-approximation algorithm, a polynomial-time $\frac{|\Sigma_A|}{\overline{p}^{max}}$-approximation algorithm and a polynomial time $O\left(\frac{n_A n_B (h^*_{max} + E_N^{max})}{|E|^2}\right)$-approximation algorithm for satisfiable instances of the projection game.

Consider the following two cases.

First, if $h^*_{max} \geq E^{max}_N$, we have $O(n_A n_B(h^*_{max} + E^{max}_N)/|E|^2) = O(n_A n_B h^*_{max}/|E|^2)$. Using the best of the first, second, fourth and fifth algorithms, the smallest of the four approximation factors is at most as large as their geometric mean, i.e.,

$$O\left(\sqrt[4]{\frac{|E|}{n_B} \cdot \frac{|\Sigma_A|}{\overline{p}^{max}} \cdot \frac{|E|\overline{p}^{max}}{h^*_{max}} \cdot \frac{n_A n_B h^*_{max}}{|E|^2}}\right) = O((n_A|\Sigma_A|)^{1/4}).$$

Second, if $E^{max}_N > h^*_{max}$, we have $O(n_A n_B(h^*_{max} + E^{max}_N)/|E|^2) = O(n_A n_B E^{max}_N/|E|^2)$. We use the best answer we get from the first, second, third and fifth algorithms. The smallest of the four approximation factors is at most as large as their geometric mean, i.e.,

$$O\left(\sqrt[4]{\frac{|E|}{n_B} \cdot \frac{|\Sigma_A|}{\overline{p}^{max}} \cdot \frac{|E|}{E^{max}_N} \cdot \frac{n_A n_B E^{max}_N}{|E|^2}}\right) = O\left(\left(\frac{n_A|\Sigma_A|}{\overline{p}^{max}}\right)^{1/4}\right).$$

It is obvious that $\overline{p}^{max}$ is at least one. Thus, we can conclude that the approximation factor is at most $O((n_A|\Sigma_A|)^{\frac{1}{4}})$.

This concludes the proof of Theorem 3.1 for the nonuniform preimage sizes case.

□

A.2 Improved Lasserre Gap for DENSEST k-SUBGRAPH

Finding approximation algorithms for DENSEST k-SUBGRAPH is a topic of great interest [SW98, FL01, FPK01, AHI02, BCC$^+$10]. The best polynomial-time approximation algorithm known today is from [BCC$^+$10] which achieves an approximation ratio of $N^{1/4+\varepsilon}$ for every constant $\varepsilon > 0$. On the hardness of approximation side, however, no hardness result with approximation ratio being polynomial is known despite many attempts [AAM$^+$11, RS10, Kho04, Fei02].

APPENDIX A. APPENDIX

In absence of matching hardness of approximation, gaps on various LP and SDP relaxations have been shown as evidences that DENSEST k-SUBGRAPH is, in fact, hard to approximate to within a polynomial factor. These results are particularly interesting for DENSEST k-SUBGRAPH because the aforementioned best known algorithm from [BCC$^+$10] uses LP hierarchy. As a result, the gap instances for LP and SDP relaxations rule out the possibility of better algorithms using similar techniques.

The most relevant such result is from [BCV$^+$12], in which a DENSEST k-SUBGRAPH instances with polynomial gaps for the Lasserre and Sherali-Adams hierarchies are presented. In one of their results, Bhaskara et al. proved a lower bound of $N^{2/53-\varepsilon}$ on the Lasserre Gap of the DENSEST k-SUBGRAPH after $N^{\Omega(\varepsilon)}$ rounds. As stated in section 4.5, the technique we used to prove Lemma 4.5 can be used to improve the this lower bound. More specifically, we will prove the following theorem in this section.

Theorem A.1. *For every $0 < \varepsilon < 1/14$, there exists a* DENSEST k-SUBGRAPH *instance with N vertices such that the integrality gap of $N^{\Omega(\varepsilon)}$ rounds of the Lasserre SDP relaxation on this instance is at least $N^{1/14-\varepsilon}$.*

Note that our result is still not tight; ideally, we want the integrality gap to match the algorithm from [BCC$^+$10], i.e., we want the gap of $N^{1/4-\varepsilon}$ instead of the current $N^{1/14-\varepsilon}$.

Again, the only main different between our proof and Bhaskara et al.'s is that we are able to prove a stronger soundness result.

A.2.1 DENSEST k-SUBGRAPH and Reduction in [BCV$^+$12]

We first start by reviewing the definition of DENSEST k-SUBGRAPH and Bhaskara et al.'s reduction from random MAX K-CSP to DENSEST k-SUBGRAPH in [BCV$^+$12].

DENSEST k-SUBGRAPH problem can be defined as follows.

> DENSEST k-SUBGRAPH
>
> INPUT: A simple graph $G = (V, E)$ where the number of vertices $|V|$ is N and an integer $k \leq N$.
>
> GOAL: Find a subgraph of G of at most k vertices with maximum number of edges.

We are now ready to review the reduction from [BCV$^+$12]. Given a random MAX K-CSP(C) instance $\Phi = \{C_1, \ldots, C_m\}$, the reduction creates a DENSEST k-SUBGRAPH instance as follows.

First, we create a bipartite graph $G = (A, B, E)$ as follows:

- We create a vertex in A for each constraint C_i and each assignment $\alpha \in [q]^{T_i}$ such that $C_i(\alpha) = 1$, i.e., $A = \{(C_i, \alpha) \mid i \in [m], \alpha \in [q]^{T_i} \text{ and } C_i(\alpha) = 1\}$.
- We create a vertex in B for each variable x_j and each assignment $\sigma \in [q]$ for x_j, i.e., $B = \{(x_j, \sigma) \mid j \in [n] \text{ and } \sigma \in [q]\}$.
- For each vertex $(C_i, \alpha) \in A$ and $(x_j, \sigma) \in B$, we create an edge between them if and only if $x_j \in T_i$ and $\alpha(x_j) = \sigma$.

The instance of DENSEST k-SUBGRAPH is a graph $G' = (A', B', E')$ together with $k = 2m$ where $A' = A$, B' is a set containing β copies of B, and E' contains

an edge (a, b') if and only if a is connected to the corresponding vertex of b' in G.

Observe that $N = |A| + |B| = m|C| + \beta nq = \Theta(\beta n|C|)$.

A.2.2 Soundness

In this subsection, we will prove the soundness of the DENSEST k-SUBGRAPH instance constructed by the reduction from [BCV$^+$12], which is the only main different between our result and theirs. To show this, it is enough for us to prove the following lemma, which is an improvement over Claim 4.14 in [BCV$^+$12]. Due to the similarities between the following lemma and the claim from [BCV$^+$12], we will follow their notations here.

Lemma A.6. *Let $0 < \rho < 1$ be any constant. Fix a subset $R \subseteq B$ of size $2n$. If $q \geq K \geq q/2, q \geq 10000/\rho, |C| \leq q^{10}$ and $\beta \geq 100q^{1+\rho}|C|/K$, then, with probability at least $1 - \exp(-mK \ln q/(q^{1+\rho}|C|))$, there exists no $L \subseteq A$ of size $2m$ such that the subgraph induced by $L \cup R$ with respect to the graph G produced by the reduction in Subsection A.2.1 contains at least $4000mK \ln q/(q\rho)$ edges.*

Proof. Let $\mathrm{Y} = \lceil 1/\rho \rceil$ and let $M = q^{1/\mathrm{Y}}$.

To prove this lemma, consider each vertex C_i in A. We define $deg(T_i)$ to be the number of vertices $(x_j, \sigma) \in R$ such that $x_j \in T_i$, i.e., $deg(T_i) = |\{x_j \in T_i \mid (x_j, \sigma) \in R\}|$. For each $l = 0, \ldots, \mathrm{Y} - 1$, define Z_i^l as the indicator variable whether $deg(T_i)$ is more than $100KM^l \ln q$. We will first find an upper bound on the probability that $Z_i^l = 1$.

Since T_i is randomly selected uniformly from the set of K variables, we can conclude that the expected value of $deg(T_i)$ is $2K$. As a result, from Hoeffding's inequality [Hoe63], we can conclude that

$$\Pr[Z_i^l = 1] \leq \left(\frac{e(2K)}{100KM^l \ln q}\right)^{100KM^l \ln q/q} \leq \exp\left(-100KM^l \ln q/q\right).$$

Next, for each $\alpha \in [q]^{T_i}$ such that $C_i(\alpha) = 1$, let $agr_{T_i}(\alpha)$ be the number of neighbors of $(C_i, \alpha) \in A$ that are in R, i.e., $agr_{T_i}(\alpha) = |\{(x_j, \alpha(x_j)) \mid x_j \in T_i\} \cap R|$. Moreover, for each $l = 0, \ldots, \Upsilon - 1$, define X_i^l as an indicator variable whether there exists $\alpha \in [q]^{T_i}$ with $C_i(\alpha) = 1$ such that $arg_{T_i}(\alpha) > 1000KM^l \ln q/q$. We will now prove an upper bound on $\Pr[X_i^l = 1]$ by finding an upper bound on $\Pr[X_i^l = 1 \mid Z_i^l = 0]$.

Recall that the constraint is generated by sampling $b^{(i)} = (b_1^{(i)}, \ldots, b_K^{(i)}) \in \mathbb{F}_q^K$ and set $C_i(c - b^{(i)}) = 1$ for every codeword $c \in C$. Consider each codeword $c \in C$. If $Z_i^l = 0$, then, for each $c \in C$, we have

$$\mathbb{E}_{b^{(i)}}\left[agr_{T_i}(c - b^{(i)})\right] = deg(T_i)/q \leq 100KM^l \ln q/q.$$

Thus, for each $c \in C$, we can conclude, from the Chernoff bound, that

$$\begin{aligned}\Pr\left[agr_{T_i}(c - b^{(i)}) > 1000KM^l \ln q/q \mid Z_i^l = 0\right] &\leq \left(\frac{e(100KM^l \ln q/q)}{(1000KM^l \ln q/q)}\right)^{1000KM^l \ln q/q} \\ &\leq \exp(-1000KM^l \ln q/q).\end{aligned}$$

Hence, we can conclude that

$$\Pr[X_i^l = 1 \mid Z_i^l = 0] = \Pr\left[\left(\exists c \in C, agr_{T_i}(c - b^{(i)}) > 1000KM^l \ln q/q\right) \mid Z_i^l = 0\right]$$

$$\begin{aligned}
(\text{Union bound}) &\leq \sum_{c \in C} \Pr\left[agr_{T_i}(c - b^{(i)}) > 1000KM^l \ln q/q \mid Z_i^l = 0 \right] \\
&\leq \sum_{c \in C} \exp(-1000KM^l \ln q/q) \\
(\text{Since } |C| \leq q^{10}) &\leq q^{10} \cdot \exp(-1000KM^l \ln q/q) \\
(\text{From } K \geq q/2, q \geq 10000/\rho) &\leq \exp(-500KM^l \ln q/q).
\end{aligned}$$

As a result, we have

$$\begin{aligned}
\Pr[X_i^l = 0] &\geq \Pr[X_i^l = 0 \mid Z_i^l = 0] \Pr[Z_i^l = 0] \\
&\geq \left(1 - \exp(-500KM^l \ln q/q)\right) \left(1 - \exp(-100KM^l \ln q/q)\right) \\
&\geq 1 - \exp(-90KM^l \ln q/q).
\end{aligned}$$

Now, observe that, from definition of X_i^l, we can conclude that, if l is the maximum value of l such that $X_i^l = 1$, then (C_i, α) has at most $1000KM^{l+1} \ln q/q$ neighbors in R for every $\alpha \in C_i^{-1}(1)$. In other words, the number of neighbors of (C_i, α) in R is at most

$$\max\{1000K \ln q/q, \max_{l=0,\ldots,\Upsilon-1} \{X_i^l 1000KM^{l+1} \ln q/q\}\}.$$

As a result, for any $L \subseteq A$ of size $2m$, the number of edges in the graph induced by $L \cup R$ is at most

$$\begin{aligned}
&\sum_{(C_i,\alpha) \in L} \max\{1000K \ln q/q, \max_{l=0,\ldots,\Upsilon-1} \{X_i^l 1000KR^{l+1} \ln q/q\}\} \\
&\leq \sum_{(C_i,\alpha) \in L} \left(1000K \ln q/q + \sum_{l=0}^{\Upsilon-1} X_i^l 1000KM^{l+1} \ln q/q \right)
\end{aligned}$$

$$
\begin{aligned}
&= 2000mK\ln q/q + \sum_{l=0}^{\Upsilon-1}(1000KM^{l+1}\ln q/q)\cdot|\{(C_i,\alpha)\in L \mid X_i^l=1\}| \\
&\le 2000mK\ln q/q + \sum_{l=0}^{\Upsilon-1}(1000KM^{l+1}\ln q/q)\cdot|\{(C_i,\alpha)\in A \mid X_i^l=1\}| \\
&= 2000mK\ln q/q + \sum_{l=0}^{\Upsilon-1}(1000KM^{l+1}\ln q/q)\cdot|C|\cdot|\{i\in[m] \mid X_i^l=1\}|
\end{aligned}
$$

Since each C_i is sampled independently of each other, we can again use Chernoff bound to arrive at the following inequality.

$$
\begin{aligned}
\Pr[|\{i\in[m] \mid X_i^l=1\}| > m/(M^{l+1}|C|)] &\le \left(\frac{e(m\cdot\exp(-90KM^l\ln q/q))}{m/(M^{l+1}|C|)}\right)^{\frac{m}{M^{l+1}|C|}} \\
&= (M^{l+1}\cdot|C|\cdot e\cdot\exp(-90KM^l\ln q/q))^{\frac{m}{M^{l+1}|C|}} \\
(\text{Since } K\ge q/2, |C|\le q^{10} \text{ and } q\ge 10000/\rho) &\le (\exp(-50KM^l\ln q/q))^{\frac{m}{M^{l+1}|C|}} \\
&= \exp(-50mK\ln q/(qM|C|)).
\end{aligned}
$$

From this and the union bound, we can conclude that, with probability at least $1-\Upsilon\exp(-50mK\ln q/(qM|C|)) \ge 1-\exp(-mK\ln q/(qM|C|))$, we have $|\{i\in[m] \mid X_i^l=1\}| \le m/(M^{l+1}|C|)$ for all $l=0,\dots,\Upsilon-1$.

Observe that, if $|\{i\in[m] \mid X_i^l=1\}| \le m/M^{l+1}$ for all $l=0,\dots,\Upsilon-1$, then we have

$$
\begin{aligned}
&2000mK\ln q/q + \sum_{l=0}^{\Upsilon-1}(1000KM^{l+1}\ln q/q)\cdot|C|\cdot|\{i\in[m] \mid X_i^l=1\}| \\
&\le 2000mK\ln q/q + \sum_{l=0}^{\Upsilon-1}(1000KM^{l+1}\ln q/q)\cdot|C|\cdot(m/(M^{l+1}|C|)) \\
&= 2000mK\ln q/q + \sum_{l=0}^{\Upsilon-1}(1000mK\ln q/q) \\
&= (2000mK\ln q/q)(\Upsilon/2+1)
\end{aligned}
$$

$$\leq 4000mK \ln q/(q\rho).$$

This means that the maximum number of edges in the subgraph induced by $L \cup R$ is at most $4000mK \ln q/(q\rho)$, which completes our proof for the lemma. □

Following the exact same proof of Lemma 4.12 from [BCV$^+$12], we can conclude the following lemma, which is the desired soundness result for the DENSEST k-SUBGRAPH instance. Note that, since the proof is exactly the same as that of Lemma 4.12 from [BCV$^+$12], we will not show the full proof in this thesis.

Lemma A.7. *Let $0 < \rho < 1$ be any constant. If $q \geq K \geq q/2, q \geq 10000/\rho, |C| \leq q^{10}$ and $\beta \geq 100q^{1+\rho}|C|/K$, then, the optimal solution of the* DENSEST k-SUBGRAPH *instance produced by the reduction in Subsection A.2.1 contains at most $4000\beta mK \ln q/(q\rho)$ edges with probability at least $1 - o(1)$.*

A.2.3 Proof of the Main Theorem

We devote this section to the proof of Theorem A.1, which is the main theorem. First, we will start by stating the vector completeness result from [BCV$^+$12].

Lemma A.8 ([BCV$^+$12])**.** *Let C be a dual code of any linear code of distance at least $D \geq 3$. For every $n, K, \beta, \eta > 0$ such that*

- *n is large enough,*
- $10 \leq K \leq n^{1/2}$,
- $\eta \leq 1/(10^8(\beta K^D)^{2/(D-2)})$,
- $n^{\nu-1} \leq 1/(10^8(\beta K^{D+0.75})^{2/(D-2)})$ *for some $\nu > 0$,*

with probability $1 - o(1)$*, there exists a perfect solution to the* $\eta n/(16K)$*-th round Lasserre SDP for a* DENSEST k-SUBGRAPH *instance produced by the reduction in Subsection A.2.1 from a random* MAX K-CSP(C) *over* n *variables and* $m = \beta n$ *constraints.* [1]

Before we prove Theorem A.1, we will combine the vector completeness result and the soundness result to get the following theorem that will serve as a basis of the proof of the main theorem.

Theorem A.2. *Let* $0 < \rho < 1$ *be any constant. For all* n, q *large enough and integer* $3 \leq D \leq 10$*, there exists a* DENSEST k-SUBGRAPH *instance of* $N = \Omega(nq^{2D-2+\rho})$ *vertices that demonstrates a gap of value* $\Omega(q^{1-o(1)})$ *for Lasserre SDP after* $\Omega(n/(q^{(4D-2+2\rho)/(D-2)+1}))$ *rounds.*

Proof. Let C be the dual code of the code from Lemma 4.2 with $3 \leq D \leq 10$. We now have $K = q - 1$ and $|C| = q^{D-1}$. Pick β to be $100q^{1+\rho}|C|/K = 100q^{D+\rho}/K$ and η to be $1/(10^8(\beta K^D)^{2/(D-2)})$.

The instance we will use is just the instance created by the reduction present in Subsection A.2.1 from a random MAX K-CSP(C) instance.

Let $R = \eta n/(16K)$. We know from Lemma A.8 that, after R rounds of the Lasserre hierarchy, there still exists a complete solution (of value βmK) to the Lasserre SDP of the instance. At the same time, we also know from Lemma A.7 that, with probability $1 - o(1)$, any subgraph of the instance of $2m$ vertices contain at most $4000\beta mK \ln q/(q\rho)$ edges. This means that, after R rounds, we have a gap ratio of $\beta mK/(4000\beta mK \ln q/(q\rho)) = \Omega(q/\ln q) = \Omega(q^{1-o(1)})$ as desired.

[1] Again, the lemma stated here is a bit different from the result in [BCV+12]; the only difference is that, in [BCV+12], η is required to be between $n^{\nu-1}$ and $1/(10^8(\beta K^{D+0.75})^{2/(D-2)})$. However, it is easy to see from the proof of Theorem 4.2 from the paper that it is enough for η to be at most $1/(10^8(\beta K^D)^{2/(D-2)})$.

Moreover, R can be written as

$$\begin{aligned}
R &= \eta n/(16K) \\
&= \frac{1}{10^8(\beta K^D)^{2/(D-2)}} \cdot \frac{n}{16K} \\
&= \Omega\left(\frac{1}{(\beta K^D)^{2/(D-2)}} \cdot \frac{n}{K}\right) \\
(\text{Since } \beta = 100q^{D+\rho}/K) &= \Omega\left(\frac{1}{(q^{D+\rho}K^{D-1})^{2/(D-2)}} \cdot \frac{n}{K}\right) \\
(\text{Since } K = q-1) &= \Omega\left(\frac{1}{(q^{2D-1+\rho})^{2/(D-2)}} \cdot \frac{n}{q}\right) \\
&= \Omega\left(\frac{n}{q^{(4D-2+2\rho)/(D-2)+1}}\right).
\end{aligned}$$

Lastly, note that the number of vertices is $N = \Omega(\beta n|C|) = \Omega\left((100q^{D+\rho}/K)nq^{D-1}\right) = \Omega(nq^{2D-2+\rho})$. □

Now, we are ready to prove the main theorem by choosing the right N, q, D for Theorem A.2.

Proof of Theorem A.1. We can rewrite the number of rounds R of Lasserre SDP relaxation from Theorem 4.3 as

$$\begin{aligned}
R &= \Omega\left(\frac{n}{q^{(4D-2+2\rho)/(D-2)+1}}\right) \\
&= \Omega\left(\frac{nq^{2D-2+\rho}}{q^{(4D-2+2\rho)/(D-2)+2D-1+\rho}}\right) \\
&= \Omega\left(\frac{N}{q^{(4D-2+2\rho)/(D-2)+2D-1+\rho}}\right) \\
&= \Omega\left(\frac{N}{q^{(4D-2)/(D-2)+2D-1+\rho(1+2/(D-2))}}\right)
\end{aligned}$$

We select $D = 4$ to minimize $(4D-2)/(D-2)+2D-1$, which implies that

$$R = \Omega\left(\frac{N}{q^{14+2\rho}}\right).$$

By pick q to be $N^{1/14-\varepsilon/2}$, we get the gap of $\Omega(N^{1/14-\varepsilon/2-o(1)}) \geq N^{1/14-\varepsilon}$ if N is big enough. Furthermore, by picking ρ to be $\varepsilon/1000$, we have

$$\begin{aligned} R &= \Omega\left(\frac{N}{N^{(1/14-\varepsilon/2)(14+\varepsilon/1000)}}\right) \\ &= \Omega\left(N^{\varepsilon/4}\right) \end{aligned}$$

This is $N^{\Omega(\varepsilon)}$ when N is big enough, which completes the proof of this theorem. □

A.3 GRID TILING Running Time Lower Bound

In this section, we give a full proof of Lemma 5.1, which was stated without a complete proof in [Mar12]. The proof is a simple reduction from CLIQUE to GRID TILING.

A.3.1 CLIQUE

Before we describe the reduction, we will first review the CLIQUE problem, which can be stated as follows.

CLIQUE

INPUT: A simple graph $\acute{G} = (\acute{V}, \acute{E})$ where the number of vertices $|\acute{V}|$ is $\acute{n}$.

GOAL: Determine whether there is a clique of size $\acute{k}$.

In [CHKX06], it was proved that CLIQUE is W[1]-hard parameterized on $\acute{k}$, which implies the following lemma.

Lemma A.9. *If there exists an algorithm for* CLIQUE *that runs in $g(\acute{k})\acute{n}^{o(\acute{k})}$ for any function g, then ETH fails.*

We will not show how the lemma is proved here. However, in the following subsection, we will show how to use running time lower bound of CLIQUE to deduce a running time lower bound of GRID TILING.

A.3.2 Proof of Lemma 5.1

In this section, we will prove Lemma 5.1 by showing a reduction from CLIQUE to GRID TILING. The reduction is briefly outlined in [Mar07] and [Mar12].

Proof. Given a simple graph $\acute{G} = (\acute{V}, \acute{E})$ where $\acute{V} = [\acute{n}]$ and an integer $\acute{k}$. We will construct a GRID TILING instance as follows.

- Let $\tilde{k} = \acute{k}$ and $\tilde{n} = \acute{n}$.
- Define $S_{i,j}$ for each $i, j \in [\tilde{k}]$ as follows.

$$S_{i,j} = \begin{cases} \{(v, v) \mid v \in [\tilde{n}]\} & \text{if } i = j, \\ \{(v, u) \mid (v, u) \in E \text{ or } (u, v) \in E\} & \text{otherwise.} \end{cases}$$

Next, we will show that the answer to this GRID TILING instance is yes if and only if there exists a $\acute{k}$-clique in $\acute{G}$.

($\Rightarrow$) Suppose that the GRID TILING instance is a yes instance, i.e. there exists $s_{i,j} \in S_{i,j}$ for all $i, j \in [\tilde{k}]$ such that, for each i, $(s_{i,j})_1$'s are equal for all $j \in [\tilde{k}]$, and, for each j, $(s_{i,j})_2$'s are equal for all $i \in [\tilde{k}]$.

Observe that, from our definition, for each $i \in [\tilde{k}]$, $s_{i,i} = (u_i, u_i)$ for some $u_i \in \tilde{n}$. Thus, for each $i, j \in [\tilde{k}]$, we have $(s_{i,j})_1 = (s_{i,i})_1 = u_i$ and $(s_{i,j})_2 = (s_{j,j})_2 = u_j$. In other words, $s_{i,j} = (u_i, u_j)$, which means that there exists an edge between (u_i, u_j) in $\acute{G}$.

As a result, we can conclude that $u_1, \ldots, u_{\tilde{k}}$ forms a $\acute{k}$-clique in $\acute{G}$.

($\Leftarrow$) Suppose that there exists a $\acute{k}$-clique in G. Let the vertices in the clique be $v_1, \ldots, v_{\acute{k}} \in [\acute{n}]$. Since $v_1, \ldots, v_{\acute{k}}$ forms a clique, we know that $(v_i, v_j) \in S_{i,j}$ for every $i, j \in [\tilde{k}]$. Hence, if we select $s_{i,j} = (v_i, v_j)$ for every $i, j \in [\tilde{k}]$, we can conclude that the answer to the GRID TILING instance is yes.

Thus, from Lemma A.9, we can conclude that there exists no algorithm that decides GRID TILING in $g(\tilde{k})\tilde{n}^{o(\tilde{k})}$ for any function g. □