# *Improved Approximation Algorithms for Projection Games*

# Improved Approximation Algorithms for Projection Games

Pasin Manurangsi and Dana Moshkovitz

Massachusetts Institute of Technology, Cambridge MA 02139, USA,
{pasin,dmoshkov}@mit.edu *

**Abstract.** The projection games (aka Label-Cover) problem is of great importance to the field of approximation algorithms, since most of the NP-hardness of approximation results we know today are reductions from Label-Cover. In this paper we design several approximation algorithms for projection games:

1. A polynomial-time approximation algorithm that improves on the previous best approximation by Charikar, Hajiaghayi and Karloff [7].
2. A sub-exponential time algorithm with much tighter approximation for the case of smooth projection games.
3. A PTAS for planar graphs.

**Keywords:** Label-Cover, projection games

## 1 Introduction

The projection games problem (also known as Label Cover) is defined as follows.

Input: A bipartite graph $G = (A, B, E)$, two finite sets of labels $\Sigma_A, \Sigma_B$, 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 "$\delta$-nearly satisfiable" or "$\delta$-nearly feasible" if there exists an assignment that satisfies $(1 - \delta)$ fraction of the edges. In this work, we focus on satisfiable instances of projection games.

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 $\varepsilon$ fraction of the edges:

* This material is based upon work supported by the National Science Foundation under Grant Number 1218547.

**Theorem 1 (Strong PCP Theorem).** *For every $n$, $\varepsilon = \varepsilon(n)$, there is $k = k(\varepsilon)$, such that deciding* SAT *on inputs of size $n$ can be reduced to finding, given a satisfiable projection game on alphabets of size $k$, an assignment that satisfies more than an $\varepsilon$ fraction of the edges.*

This theorem is the starting point of the extremely successful long-code based framework for achieving hardness of approximation results [6,11], as well as of other optimal hardness of approximation results, e.g., for SET-COVER [9,16,8].

We know several proofs of the strong PCP theorem that yield different parameters in Theorem 1. The parallel repetition theorem [19], applied on the basic PCP Theorem [5,4,3,2], yields $k(\varepsilon) = (1/\epsilon)^{O(1)}$. Alas, it reduces exact SAT on input size $n$ to LABEL COVER on input size $n^{O(\log 1/\varepsilon)}$. Hence, a lower bound of $2^{\Omega(n)}$ for the time required for solving SAT on inputs of size $n$ only implies a lower bound of $2^{n^{\Omega(1/\log 1/\varepsilon)}}$ for LABEL COVER via this theorem. This bound deteriorates as $\epsilon$ approaches zero; for instance, if $\epsilon = (1/n)^{O(1)}$, then the bound is $\Omega(1)$, which gives us no information at all.

A different proof is based on PCP composition [17,8]. It has smaller blow up but larger alphabet size. Specifically, it shows a reduction from exact SAT with input size $n$ to LABEL COVER with input size $n^{1+o(1)}poly(1/\varepsilon)$ and alphabet size $exp(1/\varepsilon)$.

One is tempted to conjecture that a PCP theorem with both a blow-up of $n^{1+o(1)}poly(1/\varepsilon)$ and an alphabet size $(1/\varepsilon)^{O(1)}$ holds. See [16] for a discussion of potential applications of this "Projection Games Conjecture".

Finding algorithms for projection games is therefore both a natural pursuit in combinatorial optimization, and also a way to advance our understanding of the main paradigm for settling the approximability of optimization problems. Specifically, our algorithms help make progress towards the following questions:

1. Is the "Projection Games Conjecture" true? What is the tradeoff between the alphabet size, the blow-up and the approximation factor?
2. What about even stronger versions of the strong PCP theorem? E.g., Khot introduced "smooth" projection games [13] (see discussion below for the definition). What kind of lower bounds can we expect to get via such a theorem?
3. Does a strong PCP theorem hold for graphs of special forms, e.g., on planar graphs?

## 2 Our Results

### 2.1 Better Approximation in Polynomial Time

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 $k$ [7]. This improved on Peleg's $O((nk)^{1/2})$-approximation algorithm [18]. Both Peleg's and the CHK algorithms worked in the more general setting of arbitrary constraints on the edges and possibly unsatisfiable instances.

We show a polynomial-time algorithm that achieves a better approximation for satisfiable projection games:

**Theorem 2.** *There is a polynomial-time algorithm that, given a satisfiable instance of projection games on a graph of size $n$ and alphabets of size $k$, finds an assignment that satisfies $O((nk)^{1/4})$ edges.*

### 2.2 Algorithms for Smooth Projection Games

Khot introduced "smooth" projection games in order to obtain new hardness of approximation results, e.g., for coloring problems [13]. In a smooth projection game, for every vertex $a \in A$, the assignments projected to $a$'s neighborhood by the different possible assignments $\sigma_a \in \Sigma_A$ to $a$, differ a lot from one another (alternatively, form an error correcting code with high relative distance). More formally:

**Definition 1.** *A projection game instance is $\mu$-smooth if for every $a \in A$ and any distinct assignments $\sigma_a, \sigma'_a \in \Sigma_A$, we have*

$$Pr_{b \in N(a)}[\pi_{(a,b)}(\sigma_a) = \pi_{(a,b)}(\sigma'_a)] \leq \mu$$

where $N(a)$ is the set of neighbors of $a$.

Intuitively, smoothness makes the projection games problem easier, since knowing only a small fraction of the assignment to a neighborhood of a vertex $a \in A$ determines the assignment to $a$.

Smoothness can be seen as an intermediate property between projection and uniqueness, with uniqueness being 0-smoothness. Khot's Unique Games Conjecture [14] is that the Strong PCP Theorem holds for unique games on nearly satisfiable instances for any constant $\varepsilon > 0$.

The Strong PCP Theorem (Theorem 1) is known to hold for $\mu$-smooth projection games with $\mu > 0$. However, the known reductions transform SAT instances of size $n$ to instances of smooth LABEL COVER of size at least $n^{O((1/\mu)\log(1/\varepsilon))}$ [13,12]. Hence, a lower bound of $2^{\Omega(n)}$ for SAT only translates into a lower bound of $2^{n^{\Omega(\mu/\log(1/\varepsilon))}}$ for $\mu$-smooth projection games.

Interestingly, the efficient reduction of Moshkovitz and Raz [17] inherently generates instances that are not smooth. Moreover, for unique games it is known that if they admit a reduction from SAT of size $n$, then the reduction must incur a blow-up of at least $n^{1/\delta^{\Omega(1)}}$ for $\delta$-almost satisfiable instances. This follows from the sub-exponential time algorithm of Arora, Barak and Steurer [1].

Given this state of affairs, one wonders whether a large blow-up is necessary for smooth projection games. We make progress toward settling this question by showing:

**Theorem 3.** *For any constant $c \geq 1$, the following holds: there is a randomized algorithm that given a $\mu$-smooth satisfiable projection game in which all vertices in $A$ have degrees at least $\frac{c \log |A|}{\mu}$, finds an optimal assignment in time $exp(O(\mu |B| \log |\Sigma_B|))poly(|A|, |\Sigma_A|)$ with probability $2/3$.*

*There is also a deterministic $O(1)$-approximation algorithm for $\mu$-smooth satisfiable projection games of any degree. The deterministic algorithm runs in time $exp(O(\mu|B|\log|\Sigma_B|))poly(|A|,|\Sigma_A|)$ as well.*

The algorithms work by finding a subset of fraction $\mu$ in $B$ that is connected to all, or most, of the vertices in $A$ and going over all possible assignments to it.

Theorem 3 essentially implies that a blow-up of $n/\mu$ is necessary for any reduction from SAT to $\mu$-smooth LABEL COVER, no matter what is the approximation factor $\varepsilon$.

### 2.3 PTAS For Planar Graphs

As the strong PCP Theorem shows, LABEL COVER is NP-hard to approximate even to within very small $\varepsilon$. 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 while projection games remain NP-hard to solve *exactly* on planar graphs, when it comes to approximation, they admit a PTAS:

**Theorem 4.** *The following hold:*

1. *Given a satisfiable instance of projection games on a planar graph, it is NP-hard to find a satisfying assignment.*
2. *There is a polynomial time approximation scheme for satisfiable instances of projection games on planar graphs.*

The NP-hardness of projection games on planar graphs is based on a reduction from 3-colorability problem on planar graphs. The PTAS works via Klein's approach [15] of approximating the graph by a graph with constant tree-width.

## 3 Conventions

We define the following notation to be used in the paper.

- 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 $d_v$ denote the degree of a vertex $v \in A \cup B$.
- For a vertex $u$, we use $N(u)$ to denote set of vertices that are neighbors of $u$. Similarly, for a set of vertex $U$, we use $N(U)$ to denote the set of vertices that are neighbors of at least one vertex in $U$.
- For each vertex $u$, define $N_2(u)$ to be $N(N(u))$. This is the set of neighbors of neighbors of $u$.

– Let $\sigma_v^{OPT}$ 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 $\sigma_b^{OPT}$, 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 $|E(N_2(a))|$. Let $h_{max} = max_{a \in A} h(a)$.

For simplicity, we make the following assumptions:

– $G$ 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 only make use of the assumptions in the algorithms for proving Theorem 2. We defer the treatment of graphs with general number of preimages to the appendix.

## 4 Polynomial-time Approximation Algorithms for Projection Games

In this section, we present an improved polynomial time approximation algorithm for projection games and prove Theorem 2.

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 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 $\sum_{a \in N(b)} |\pi_{(a,b)}^{-1}(\sigma_b)|$. Each vertex in $A$ is then assigned so that it satisfies as many edges as possible. This algorithm works well when $\Sigma_B$ 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 $N(a_0)$ accordingly. Then, for each node in $N_2(a_0)$, we keep only the assignments that satisfy all the edges between it and vertices in $N(a_0)$.
   When $a_0$ is assigned the optimal assignment, the number of choices for each node in $N_2(a_0)$ is reduced to at most $\overline{p}$ possibilities. In this way, we can satisfy $1/\overline{p}$ fraction of the edges that touch $N_2(a_0)$. This satisfies many edges when there exists $a_0 \in A$ such that $N_2(a_0)$ 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 $N(a) \cup N_2(a)$ efficiently, and give up on satisfying other edges that touch $N_2(a)$. Repeating this process, we can satisfy $\Omega(|E|^2/(n_A n_B h_{max}))$ fraction of the edges. This is large when $N_2(a)$ 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.

**Satisfy One Neighbor Algorithm.** We will present a simple algorithm that gives $\frac{|E|}{n_B}$ approximation ratio.

**Lemma 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.

**Greedy Assignment Algorithm.** The idea for this algorithm is that if there are many assignments in $\Sigma_A$ 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 2.** *There exists a polynomial-time $\frac{|\Sigma_A|}{\overline{p}}$-approximation algorithm for satisfiable instances of projection games.*

*Proof.* The algorithm works as follows:

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

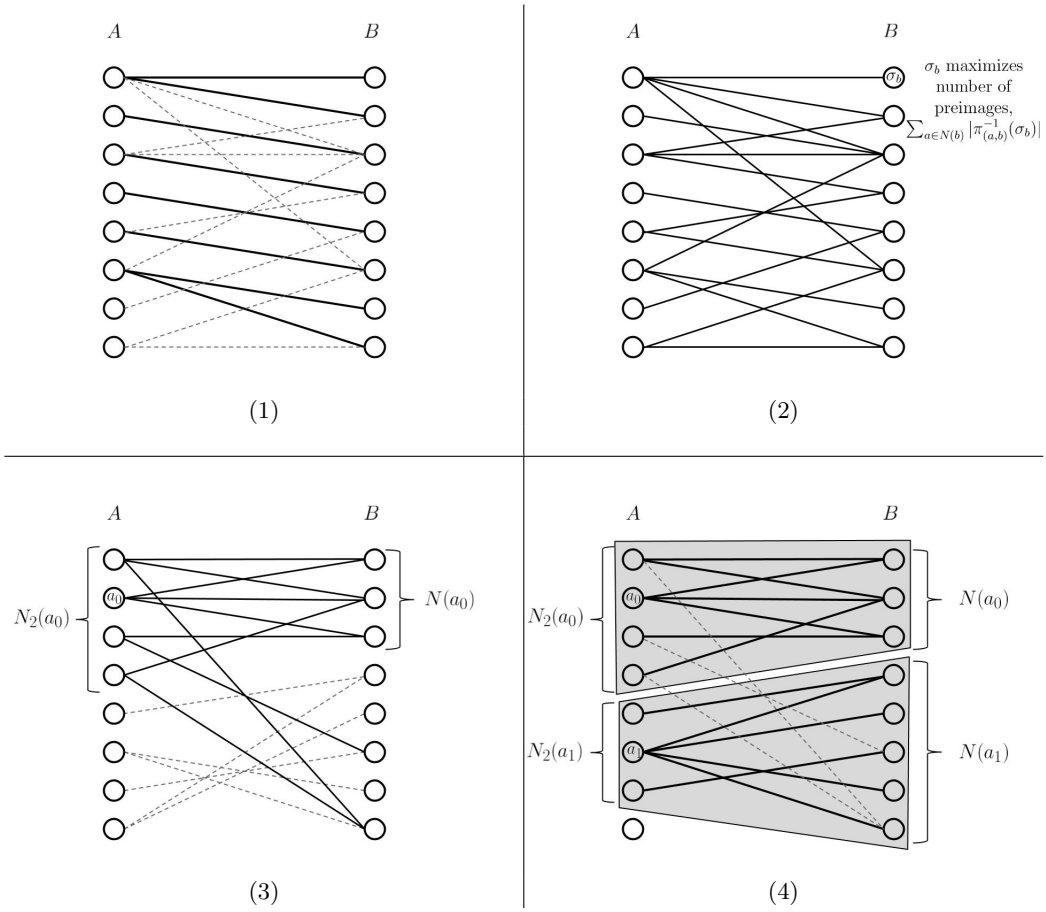


**Fig. 1. An Overview of The Algorithms in One Figure.** Four algorithms are used to prove Theorem 2: (1) In *satisfy one neighbor* algorithm, each vertex in $B$ is assigned to satisfy one of its neighboring edges. (2) In *greedy assignment* algorithm, each vertex in $B$ is assigned with an assignment with largest number of preimages. (3) In *know your neighbors' neighbors* algorithm, for a vertex $a_0$, choices for each node in $N_2(a_0)$ are reduced to at most $O(\overline{p})$ possibilities so $O\left(\frac{1}{\overline{p}}\right)$ fraction of edges that touch $N_2(a_0)$ are satisfied. (4) In *divide and conquer* algorithm, the vertices are seperated to subsets, each of which is a subset of $N(a) \cup N_2(a)$, and each subset is solved separately.

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

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

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 $\Sigma_A$. This can

be written as follows.

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

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

$$\begin{aligned} e^* &\geq \frac{1}{|\Sigma_A|} \sum_{b \in B} \sum_{a \in N(b)} |\pi_{(a,b)}^{-1}(\sigma_b^{OPT})| \\ &= \frac{1}{|\Sigma_A|} \sum_{e \in E} p_e \\ &= \frac{|E|\overline{p}}{|\Sigma_A|} \end{aligned}$$

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.

**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.** *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, 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 $b \in N(a_0)$.

(b) For each $a \in A$, find the set of plausible assignments to $a$, i.e., $S_a = \{\sigma_A \in \Sigma_A \mid \forall b \in N(a) \cap N(a_0), \pi_{(a,b)}(\sigma_A) = \sigma_b\}$. 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 $\sigma_b^*$ for $b$ that maximizes the average number of satisfied edges over all assignments in $S_a$ to vertices $a$ in $N(b) \cap N_2(a_0)$, i.e., maximizes $\sum_{a \in N(b) \cap N_2(a_0)} |\pi_{(a,b)}^{-1}(\sigma_b) \cap S_a|$.

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

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

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

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 $N(a_0)$ is the optimal assignment; that is $\sigma_b = \sigma_b^{OPT}$ for all $b \in N(a_0)$. 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 $v \in \{a_0\} \cup N(a_0)$. From now on, we will consider only this case.

Since for each $a \in A$, the assignment $\sigma_a^*$ is chosen to maximize the number of edges satisfied, we can conclude that the number of edges satisfied by selecting $\sigma_a^*$ 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

$$\begin{aligned}
e^* &\geq \sum_{a \in A} \frac{\sum_{\sigma_a \in S_a} |\{b \in N(a) \mid \pi_{(a,b)}(\sigma_a) = \sigma_b^*\}|}{|S_a|} \\
&= \sum_{a \in A} \frac{\sum_{\sigma_a \in S_a} \sum_{b \in N(a)} 1_{\pi_{(a,b)}(\sigma_a) = \sigma_b^*}}{|S_a|} \\
&= \sum_{a \in A} \frac{\sum_{b \in N(a)} \sum_{\sigma_a \in S_a} 1_{\pi_{(a,b)}(\sigma_a) = \sigma_b^*}}{|S_a|} \\
&= \sum_{a \in A} \frac{\sum_{b \in N(a)} |\pi_{(a,b)}^{-1}(\sigma_b^*) \cap S_a|}{|S_a|} \\
&= \sum_{b \in B} \sum_{a \in N(b)} \frac{|\pi_{(a,b)}^{-1}(\sigma_b^*) \cap S_a|}{|S_a|} \\
&\geq \sum_{b \in B} \sum_{a \in N(b) \cap N_2(a_0)} \frac{|\pi_{(a,b)}^{-1}(\sigma_b^*) \cap S_a|}{|S_a|}
\end{aligned}$$

Now, for each $a \in N_2(a_0)$, consider $S_a$. From the definition of $S_a$, we have

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

As a result, we can conclude that

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

Note that since $a \in N_2(a_0)$, we have $N(a) \cap N(a_0) \neq \emptyset$. 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

$$e^* \geq \frac{1}{\overline{p}} \sum_{b \in B} \sum_{a \in N(b) \cap N_2(a_0)} |\pi^{-1}_{(a,b)}(\sigma^*_b) \cap S_a|.$$

Since we pick the assignment $\sigma^*_b$ that maximizes $\sum_{a \in N(b) \cap N_2(a_0)} |\pi^{-1}_{(a,b)}(\sigma^*_b) \cap S_a|$ for each $b \in B$, we can conclude that

$$\begin{aligned} e^* &\geq \frac{1}{\overline{p}} \sum_{b \in B} \sum_{a \in N(b) \cap N_2(a_0)} |\pi^{-1}_{(a,b)}(\sigma^*_b) \cap S_a| \\ &\geq \frac{1}{\overline{p}} \sum_{b \in B} \sum_{a \in N(b) \cap N_2(a_0)} |\pi^{-1}_{(a,b)}(\sigma^{OPT}_b) \cap S_a|. \end{aligned}$$

Since the optimal assignment satisfies every edge, we can conclude that $\sigma^{OPT}_a \in \pi^{-1}_{(a,b)}(\sigma^{OPT}_b)$ and $\sigma^{OPT}_a \in S_a$, for all $b \in B$ and $a \in N(b) \cap N_2(a_0)$. This implies that

$$\begin{aligned} e^* &\geq \frac{1}{\overline{p}} \sum_{b \in B} \sum_{a \in N(b) \cap N_2(a_0)} |\pi^{-1}_{(a,b)}(\sigma^{OPT}_b) \cap S_a| \\ &\geq \frac{1}{\overline{p}} \sum_{b \in B} \sum_{a \in N(b) \cap N_2(a_0)} 1. \end{aligned}$$

The last term can be written as

$$\begin{aligned} \frac{1}{\overline{p}} \sum_{b \in B} \sum_{a \in N(b) \cap N_2(a_0)} 1 &= \frac{1}{\overline{p}} \sum_{a \in N_2(a_0)} \sum_{b \in N(a)} 1 \\ &= \frac{1}{\overline{p}} (h(a_0)) \\ &= \frac{h(a_0)}{\overline{p}}. \end{aligned}$$

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.

**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 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 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 $|E_{(N(a)\cup N_2(a))-V_{\mathcal{P}}}| \geq \frac{1}{4}\frac{|E|^2}{n_A n_B}$:
   (a) Set $\mathcal{P} \leftarrow \mathcal{P} \cup \{(N_2(a) \cup N(a)) - V_{\mathcal{P}}\}$.
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, $|E_{(N(a)\cup N_2(a))-V_{\mathcal{P}}}| < \frac{1}{4}\frac{|E|^2}{n_A n_B}$ for all $a \in A$.

Consider $\sum_{a\in A} |E_{(N(a)\cup N_2(a))-V_{\mathcal{P}}}|$. Since $|E_{(N(a)\cup N_2(a))-V_{\mathcal{P}}}| < \frac{1}{4}\frac{|E|^2}{n_A n_B}$ for all $a \in A$, we have the following inequality.

$$\frac{|E|^2}{4n_B} \geq \sum_{a\in A} |E_{(N(a)\cup N_2(a))-V_{\mathcal{P}}}|.$$

Let $N^p(v) = N(v) - V_{\mathcal{P}}$ and $N_2^p(v) = N_2(v) - V_{\mathcal{P}}$. 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:

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

From Jensen's inequality, we have

$$\begin{aligned}
\sum_{a \in A} |E_{(N(a) \cup N_2(a)) - V_{\mathcal{P}}}| &\geq \frac{1}{|B - V_{\mathcal{P}}|} \left( \sum_{b \in B - V_{\mathcal{P}}} |N^p(b)| \right)^2 \\
&= \frac{1}{|B - V_{\mathcal{P}}|} \left| E_{(A \cup B) - V_{\mathcal{P}}} \right|^2 \\
&\geq \frac{1}{n_B} \left| E_{(A \cup B) - V_{\mathcal{P}}} \right|^2.
\end{aligned}$$

Since $\frac{|E|^2}{4n_B} \geq \sum_{a \in A} |E_{(N(a) \cup N_2(a)) - V_{\mathcal{P}}}|$ and $\sum_{a \in A} |E_{(N(a) \cup N_2(a)) - V_{\mathcal{P}}}| \geq \frac{1}{n_B} \left| E_{(A \cup B) - V_{\mathcal{P}}} \right|^2$, 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 $|E_{(N(a)\cup N_2(a))-V_{\mathcal{P}_{old}}}| \geq \frac{1}{4}\frac{|E|^2}{n_A n_B}$. Since $|E_{\mathcal{P}_{new}}| = |E_{\mathcal{P}_{old}}| + |E_{(N(a)\cup N_2(a))-V_{\mathcal{P}_{old}}}|$, 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 $V_{\mathcal{P}_{new}} = V_{\mathcal{P}_{old}} \cup (N_2(a) \cup N(a))$, we can conclude that

$$((A \cup B) - V_{\mathcal{P}_{old}}) \subseteq ((A \cup B) - V_{\mathcal{P}_{new}}) \cup (N_2(a) \cup N(a)).$$

Thus, we can also derive

$$\begin{aligned} E_{(A\cup B)-V_{\mathcal{P}_{old}}} &\subseteq E_{((A\cup B)-V_{\mathcal{P}_{new}})\cup(N_2(a)\cup N(a))} \\ &= E_{(A\cup B)-V_{\mathcal{P}_{new}}} \cup \{(a', b') \in E \mid a' \in N_2(a) \text{ or } b' \in N(a)\}. \end{aligned}$$

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

$$\begin{aligned} |E_{(A\cup B)-V_{\mathcal{P}_{old}}}| &\leq |E_{(A\cup B)-V_{\mathcal{P}_{new}}} \cup \{(a', b') \in E \mid a' \in N_2(a) \text{ or } b' \in N(a)\}| \\ &\leq |E_{(A\cup B)-V_{\mathcal{P}_{new}}}| + |\{(a', b') \in E \mid a' \in N_2(a) \text{ or } b' \in N(a)\}| \\ &= |E_{(A\cup B)-V_{\mathcal{P}_{new}}}| + |\{(a', b') \in E \mid a' \in N_2(a)\}| \\ &= |E_{(A\cup B)-V_{\mathcal{P}_{new}}}| + |E(N_2(a))| \\ &= |E_{(A\cup B)-V_{\mathcal{P}_{new}}}| + h(a) \\ &\leq |E_{(A\cup B)-V_{\mathcal{P}_{new}}}| + h_{max}. \end{aligned}$$

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

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

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 4.

**Proof of Theorem 2**

*Proof.* Using Lemma 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 1, 2 and 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 2.

## 5 Sub-Exponential Time Algorithms for Smooth Projection Games

In this section, we prove Theorem 3 via Lemma 5 and Lemma 6 below.

### 5.1 Exact Algorithm for Graphs With Sufficiently Large Degrees

The idea of this algorithm is to randomly select $\Theta(\mu n_B)$ vertices from $B$ and try all possible assignments for them. When the assignment for the selected set is correct, we can determine the optimal assignment for every $a \in A$ such that more than $\mu d_a$ of its neighbors are in the selected set.

The next lemma shows that, provided that the degrees of the vertices in $A$ are not too small, the algorithm gives an assignment that satisfies all the edges with high probability.

**Lemma 5.** *For every constant $c \geq 1$, the following statement holds: given a satisfiable instance of projection games that satisfies the $\mu$-smoothness property and that $d_a \geq \frac{c \log n_A}{\mu}$ for all $a \in A$, one can find the optimal assignment for the game in time $exp(O(\mu n_B \log |\Sigma_B|)) \cdot poly(n_A, \Sigma_A)$ with probability $2/3$.*

*Proof.* Let $c_1$ be a constant greater than one.

The algorithm is as follows.

1. For each $b \in B$, randomly pick it with probability $c_1\mu$. Call the set of all picked vertices $B^*$.
2. Try every possible assignments for the vertices in $B^*$. For each assignment:
   (a) For each node $a \in A$, try every assignment $\sigma_a \in \Sigma_A$ for it. If there is exactly one assignment that satisfies all the edges that touch $a$, pick that assignment.
3. If encountered an assignment satisfying all edges, return that assignment.

Next, we will show that, with probability 2/3, the algorithm returns the optimal assignment with runtime $exp(O(\mu n_B \log|\Sigma_B|)) \cdot poly(n_A, |\Sigma_A|)$.

For each $b \in B$, let $X_b$ be an indicator variable for whether the vertex $b$ is picked, i.e. $b \in B^*$. From step 1, we have

$$E[X_b] = c_1\mu.$$

Let $X$ be a random variable representing the number of vertices that are selected in step 1, i.e. $X = |B^*|$. We have

$$E[X] = \sum_{b \in B} E[X_b] = n_B c_1 \mu.$$

For each $a \in A$, let $Y_a$ be a random variable representing the number of their neighbors that are picked, i.e. $Y_a = |B^* \cap N(a)|$. We have

$$E[Y_a] = \sum_{b \in N(a)} E[X_b] = d_a c_1 \mu.$$

Clearly, by iterating over all possible assignments for $B^*$, the algorithm runtime is $|\Sigma_B|^{O(|B^*|)} = exp(O(|B^*| \log|\Sigma_B|))$. Thus, if $X = |B^*| = O(\mu n_B)$, the runtime for the algorithm is $exp(O(\mu n_B \log|\Sigma_B|))$.

Let $c_2$ be a constant greater than one. If $X \leq c_2 c_1 \mu n_B$, then the runtime of the algorithm is $exp(O(\mu n_B \log|\Sigma_B|))$ as desired.

Since $\{X_b\}_{b \in B}$ are independent, using Chernoff bound, we have

$$\begin{aligned} Pr[X > c_2 c_1 \mu n_B] &= Pr[X > c_2 E[X]] \\ &< \left(\frac{e^{c_2 - 1}}{(c_2)^{c_2}}\right)^{n_B c_1 \mu} \\ &= e^{-n_B c_1 (c_2(\log c_2 - 1) + 1)\mu}. \end{aligned}$$

Now, consider each $a \in A$. By going through all possible combinations of assignments of vertices in $B^*$, we must go over the optimal assignment for $B^*$. In the optimal assignment, if more than $\mu$ fraction of $a$'s neighbors is in $B^*$ (i.e.,

$Y_a > \mu d_a$), then in step 3, we get the optimal assignment for $a$. Since $\{X_b\}_{b \in N(a)}$ are independent, using Chernoff bound, we can obtain the following inequality.

$$\begin{aligned} Pr[Y_a \leq d_a\mu] &= Pr[Y_a \leq \frac{1}{c_1}E[Y_a]] \\ &< \left( \frac{e^{\frac{1}{c_1}-1}}{\left(\frac{1}{c_1}\right)^{\frac{1}{c_1}}} \right)^{d_a c_1 \mu} \\ &= e^{-d_a c_1 (1 - \frac{1}{c_1} - \frac{1}{c_1}\log c_1)\mu}. \end{aligned}$$

Hence, we can conclude that the probability that this algorithm returns an optimal solution within $exp(O(\mu n_B \log |\Sigma_B|))$-time is at least

$$\begin{aligned} Pr\left[\left(\bigwedge_{a \in A}(Y_a > d_a\mu)\right) \wedge (X \leq c_2 c_1 \mu n_B)\right] &= 1 - Pr\left[\left(\bigvee_{a \in A}(Y_a \leq d_a\mu)\right) \vee (X > c_2 c_1 \mu n_B)\right] \\ &\geq 1 - \left(\sum_{a \in A} Pr[Y_a > d_a\mu]\right) - Pr[X > c_2 c_1 \mu n_B] \\ &> 1 - \left(\sum_{a \in A} e^{-d_a c_1 (1 - \frac{1}{c_1} - \frac{1}{c_1}\log c_1)\mu}\right) - e^{-n_B c_1 (c_2(\log c_2 - 1) + 1)\mu} \end{aligned}$$

Since $c_1(1 - \frac{1}{c_1} - \frac{1}{c_1}\log c_1)$ and $c_1(c_2(\log c_2 - 1) + 1)$ are constant, we can define constants $c_3 = c_1(1 - \frac{1}{c_1} - \frac{1}{c_1}\log c_1)$ and $c_4 = c_1(c_2(\log c_2 - 1) + 1)$. The probability that the algorithm returns an optimal solution can be written as

$$1 - e^{-n_B \mu c_4} - \sum_{a \in A} e^{-d_a \mu c_3}.$$

Moreover, since we assume that $d_a \geq \frac{c \log n_A}{\mu}$ for all $a \in A$, we can conclude that the probability above is at least

$$\begin{aligned} 1 - e^{-n_B \mu c_4} - \sum_{a \in A} e^{-cc_3 \log n_A} &= 1 - e^{-n_B \mu c_4} - n_A e^{-cc_3 \log n_A} \\ &= 1 - e^{-n_B \mu c_4} - e^{-(cc_3 - 1)\log n_A} \end{aligned}$$

Note that for any constants $c_3^*, c_4^*$, we can choose constants $c_1, c_2$ so that $c_3 = c_1(1 - \frac{1}{c_1} - \frac{1}{c_1}\log c_1) \geq c_3^*$ and $c_4 = c_1(c_2(\log c_2 - 1) + 1) \geq c_4^*$. This means that we can select $c_1$ and $c_2$ so that $c_3 \geq \frac{1}{c} + \frac{2}{c \log n_A} \in O(1)$ and $c_4 \geq \frac{2}{n_B \mu} \in O(1)$. Note also that here we can assume that $\log n_A > 0$ since an instance is trivial when $n_A = 1$. Plugging $c_3$ and $c_4$ into the lower bound above, we can conclude that, for this $c_1$ and $c_2$, the algorithm gives the optimal solution in the desired runtime with probability more than $2/3$.

### 5.2 Deterministic Approximation Algorithm For General Degrees

A deterministic version of the above algorithm is shown below. In this algorithm, we are able to achieve a $O(1)$ approximation ratio within asymptotically the same runtime as the algorithm above. In contrast to the previous algorithm, this algorithm works even when the degrees of the vertices are small.

The idea of the algorithm is that, instead of randomly picking a subset $B^*$ of $B$, we will deterministically pick $B^*$. We say that a vertex $a \in A$ is *saturated* if more than $\mu$ fraction of its neighbors are in $B^*$, i.e. $|N(a) \cap B^*| > \mu d_a$. In each step, we pick a vertex in $B$ that neighbors the most unsaturated vertices, and add it to $B^*$. We do this until a constant fraction of the edges are satisfied.

**Lemma 6.** *There exists an $exp(O(\mu n_B \log |\Sigma_B|)) \cdot poly(n_A, |\Sigma_A|)$-time $O(1)$-approximation algorithm for satisfiable $\mu$-smooth projection game instances.*

*Proof.* Without loss of generality, $\mu \geq 1/d_a$ for all $a \in A$. Otherwise, $\mu$-smoothness is the same as uniqueness, and one can find the optimal assignment for all the $a$'s with $\mu < 1/d_a$ in polynomial time (similarly to solving fully satisfiable instances of unique games in polynomial time).

Let $c_1$ be a real number between 0 and 1.

The algorithm can be described in the following steps.

1. Set $B^* \leftarrow \emptyset$.
2. Let $S$ be the set of all saturated vertices, i.e. $S = \{a \in A \mid |N(a) \cap B^*| > \mu d_a\}$. As long as $|\sum_{a \in S} d_a| < c_1|E|$:
   (a) Pick a vertex $b^* \in B - B^*$ with maximal $|N(b) - S|$. Set $B^* \leftarrow B^* \cup \{b^*\}$.
3. Iterate over all possible assignments to the vertices in $B^*$:
   (a) For each saturated vertex $a \in S$, search for an assignment that satisfies all edges in $\{a\} \times (N(a) \cap B^*)$. If, for any saturated vertex $a \in S$, this assignment cannot be found, skip the next part and go to the next assignment for $B^*$.
   (b) Assign each vertex in $B$ an assignment in $\Sigma_B$ that satisfies the maximum number of edges that touch it.
   (c) Assign arbitrary elements from $\Sigma_A$ to the vertices in $A$ that are not yet assigned.
4. Output the assignment that satisfies the maximum number of edges.

We will divide the proof into two steps. First, we will show that the number of edges satisfied by the output assignment is at least $c_1|E|$. Second, we will show that the runtime for this algorithm is $exp(O(\mu n_B \log |\Sigma_B|)) \cdot poly(n_A, |\Sigma_A|)$.

Observe that since we are going through all the possible assignments of $B^*$, we must go through the optimal assignment. Focus on this assignment. From the smoothness property, for each saturated vertex $a \in S$, there is exactly one assignment that satisfies all the edges to $B^*$; this assignment is the optimal assignment. Since we have the optimal assignments for all $a \in S$, we can satisfy all the edges with one endpoint in $S$; the output assignment satisfies at least

$\sum_{a\in S} d_a$ edges. Moreover, when the algorithm terminates, the condition in step 2 must be false. Thus, we can conclude that $\sum_{a\in S} d_a \geq c_1|E|$. As a result, the algorithm gives an approximation ratio of $\frac{1}{c_1} = O(1)$.

Since we go through all possible assignments for $B^*$, the runtime of this algorithm is $|\Sigma_B|^{O(|B^*|)} \cdot poly(n_A, |\Sigma_A|)$. In order to prove that the runtime is $exp(O(\mu n \log |\Sigma_B|)) \cdot poly(n_A, |\Sigma_A|)$, we only need to show that $|B^*| = O(\mu n_B)$.

When the algorithm picks a vertex $b^* \in B$ to $B^*$ we say that it *hits* all its neighbors that are unsaturated. Consider the total number of hits to all the vertices in $A$. Since saturated vertices do not get hit any more, we can conclude that each vertex $a \in A$ gets at most $\mu d_a + 1$ hits. As a result, the total number of hits to all vertices $a \in A$ is at most

$$\sum_{a\in A}(\mu d_a + 1) = \mu|E| + n_A.$$

Next, consider the set $B^*$. Let $B^* = \{b_1, \ldots, b_m\}$ where $b_1, \cdots, b_m$ are sorted by the time, from the earliest to the latest, they get added into $B^*$. Let $v(b_i)$ equals to the number of hits $b_i$ makes. Since the total number of hits by $B^*$ equals the total number of hits to $A$, from the bound we established above, we have

$$\sum_{i=1}^{m} v(b_i) \leq \mu|E| + n_A.$$

Now, consider the adding of $b_i$ to $B^*$. Let $B_i^*$ be $\{b_1, \ldots, b_{i-1}\}$, the set $B^*$ at the time right before $b_i$ is added to $B^*$, and let $S_i$ be $\{a \in A \mid |N(a) \cap B_i^*| > \mu d_a\}$, the set of saturated vertices at the time right before $b_i$ is added to $B^*$. Since we are picking $b_i$ from $B - B_i^*$ with the maximum number of hits, the number of hits from $b_i$ is at least the average number of possible hits over all vertices in $B - B_i^*$. That is

$$\begin{aligned} v(b_i) &= |N(b_i) - S_i| \\ &\geq \frac{1}{|B - B_i^*|}\left(\sum_{b\in B - B_i^*} |N(b) - S_i|\right) \\ &\geq \frac{1}{n_B}\left(\left(\sum_{b\in B} |N(b) - S_i|\right) - \left(\sum_{b\in B_i^*} |N(b) - S_i|\right)\right). \end{aligned}$$

We can also derive the following inequality.

$$\begin{aligned}
\sum_{b\in B}|N(b)-S_i| &= \sum_{b\in B}\sum_{a\in N(b)-S_i} 1 \\
&= \sum_{b\in B}\sum_{a\in A-S_i} \mathbf{1}_{(a,b)\in E} \\
&= \sum_{a\in A-S_i}\sum_{b\in B} \mathbf{1}_{(a,b)\in E} \\
&= \sum_{a\in A-S_i} d_a \\
&= |E| - \left(\sum_{a\in S_i} d_a\right) \\
&> (1-c_1)|E|.
\end{aligned}$$

Note that the last inequality comes from the condition in step 2 of the algorithm.

Moreover, we have

$$\begin{aligned}
\sum_{b\in B_i^*}|N(b)-S_i| &= \sum_{j=1}^{i-1}|N(b_j)-S_i| \\
(\text{Since } S_j\subseteq S_i) &\le \sum_{j=1}^{i-1}|N(b_j)-S_j| \\
&= \sum_{j=1}^{i-1} v(b_j) \\
&\le \sum_{j=1}^{m} v(b_j) \\
&\le \mu|E| + n_A.
\end{aligned}$$

Putting them together, we have

$$v(b_i) > \frac{1}{n_B}((1-c_1)|E| - \mu|E| - n_A)$$

for all $i = 1, \ldots, m$

From this and from $\sum_{i=1}^{m} v(b_i) \le \mu|E| + n_A$, we can conclude that

$$\begin{aligned}
m &< \frac{\mu|E| + n_A}{\frac{1}{n_B}((1-c_1)|E| - \mu|E| - n_A)} \\
&= n_B\mu\left(\frac{1+\frac{n_A}{|E|\mu}}{(1-c_1)-\mu-\frac{n_A}{|E|}}\right).
\end{aligned}$$

Consider the term $\frac{1+\frac{n_A}{|E|\mu}}{(1-c_1)-\mu-\frac{n_A}{|E|}}$. Since $\mu = O(1)$ and $\frac{n_A}{|E|} = O(1)$, we can conclude that the denominator is $\Theta(1)$.

Consider $\frac{n_A}{|E|\mu}$. We have

$$|E|\mu = \sum_{a \in A} \mu d_a.$$

Since we assume that $d_a \geq 1/\mu$ for all $a \in A$, we have $n_A = O(|E|\mu)$. In other words, $\frac{n_A}{|E|\mu} = O(1)$. Hence, we can conclude that the numerator is $\Theta(1)$.

As a result, we can deduce that $m = O(\mu n_B)$. Thus, the running time for the algorithm is $O(\mu n_B \log |\Sigma_B|)$, which concludes our proof.

## 6 PTAS for Projection Games on Planar Graphs

The NP-hardness of projection games on planar graphs is proved by reduction from 3-coloring on planar graphs. The latter was proven to be NP-hard by Garey, Johnson and Stockmeyer [10].

**Theorem 5.** Label Cover *on planar graphs is NP-hard.*

*Proof.* We will prove this by reducing from 3-colorability problem on planar graph, which was proven by Garey, Johnson and Stockmeyer to be NP-hard [10]. The problem can be formally stated as following.

Planar Graph 3-Colorability: Given a planar graph $\check{G} = (\check{V}, \check{E})$, decide whether it is possible to assign each node a color from $\{red, blue, green\}$ such that, for each edge, its endpoints are of different colors.

Note that even though $\check{G}$ is an undirected graph, we will represent each edge as a tuple $(u, v) \in \check{E}$ where $u, v \in \check{V}$. We will never swap the order of the two endpoints within this proof.

We will create an instance of projection game $(A, B, E, \Sigma_A, \Sigma_B, \{\pi_e\}_{e \in E})$ as follows.

- Let $A = \check{E}$ and $B = \check{V}$.
- $E = \{(a, b) \in A \times B \mid b \text{ is an endpoint of } a \text{ with respect to } \check{G}\}$.
- $\Sigma_A = \{(red, blue), (red, green), (blue, red), (blue, green), (green, red), (green, blue)\}$ and $\Sigma_B = \{red, blue, green\}$.
- For each $e = (u, v) \in \check{E} = A$, let $\pi_{(e,u)} : (c_1, c_2) \to c_1$ and $\pi_{(e,v)} : (c_1, c_2) \to c_2$, i.e., $\pi_{(e,u)}$ and $\pi_{(e,v)}$ are projections to the first and the second element of the tuple respectively.

It is obvious that $G = (A, B, E)$ is a planar graph since $A, B$ are $\check{E}, \check{V}$ respectively and there exists an edge between $a \in A$ and $b \in B$ if and only if node corresponding to $b$ in $\check{V}$ is an endpoint of an edge corresponding to $a$ in $\check{E}$. This means that we can use the same planar embedding from the original graph $\check{G}$ except that each node represent a node from $B$ and at each edge, we put in

a node from $A$ corresponding to that edge. It is also clear that the size of the projection game is polynomial of the size of $\check{G}$.

The only thing left to show is to prove that $(A, B, E, \Sigma_A, \Sigma_B, \{\pi_e\}_{e\in E})$ is satisfiable if and only if $\check{G}$ is 3-colorable.

($\Rightarrow$) Suppose that $(A, B, E, \Sigma_A, \Sigma_B, \{\pi_e\}_{e\in E})$ is satisfiable. Let $\sigma_u$ be the assignment for each vertex $u \in A \cup B$ that satisfies all the edges in the projection game. We will show that by assigning $\sigma_v$ to $v$ for all $v \in \check{V} = B$, we are able to color $\check{G}$ with 3 colors such that, for each edge, its endpoints are of different color.

Since $\check{V} = B$, $\sigma_v \in \{red, blue, green\}$ for all $v \in \check{V}$. Thus, this is a valid coloring. To see that no two endpoints of any edge are of the same color, consider an edge $e = (u, v) \in \check{E} = A$. From definition of $E$, we have $(e, u) \in E$ and $(e, v) \in E$. Moreover, from definition of $\pi_{(e,u)}$ and $\pi_{(e,v)}$, we can conclude that $\sigma_e = (\sigma_u, \sigma_v)$. Since $\sigma_e \in \Sigma_A$, we can conclude that $\sigma_u \neq \sigma_v$ as desired.

Thus, $\check{G}$ is 3-colorable.

($\Leftarrow$) Suppose that $\check{G}$ is 3-colorable. In a valid coloring scheme, let $c_v$ be a color of node $v$ for each $v \in \check{V} = B$. Define the assignment of the projection game $\varphi_A, \varphi_B$ as follows

$$\varphi_A(a) = (c_u, c_v) \text{ for all } a = (u, v) \in A = \check{E},$$
$$\varphi_B(b) = c_b \text{ for all } b \in B = \check{V}.$$

Since $c_u \neq c_v$ for all $(u, v) \in \check{E}$, we can conclude that the range of $\varphi_A$ is a subset of $\Sigma_A$. Moreover, it is clear that the range of $\varphi_B$ is a subset of $\Sigma_B$. As a result, the assignment defined above is valid. Moreover, it is obvious that $\pi_e(\varphi_A(a)) = \varphi_B(b)$ for all $e = (a, b) \in E$. Hence, the projection game $(A, B, E, \Sigma_A, \Sigma_B, \{\pi_e\}_{e\in E})$ is satisfiable.

As a result, we can conclude that Label Cover on planar graph is NP-hard.

Next, we will describe PTAS for projection game instances on planar graph and prove Theorem 4. We use the framework presented by Klein for finding PTAS for problems on planar graphs [15] to one for satisfiable instances of the projection games problem. The algorithm consists of the following two steps:

1. **Thinning Step:** Delete edges from the original graph to obtain a graph with bounded treewidth.
2. **Dynamic Programming Step:** Use dynamic programming to solve the problem in the bounded treewidth graph.

### 6.1 Tree Decomposition

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.

### 6.2 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. Since the set of edges that get deleted is small, if we are able to solve the modified graph, then we get a good approximate answer for the original graph.

Klein has proved the following lemma in his paper [15].

**Lemma 7.** *For any planar graph $G = (V, E)$ and integer $k$, there is a linear-time algorithm returns an edge-set $S$ such that $|S| \leq \frac{1}{k}|E|$, a planar graph $H$, such that $H - S = G - S$, and a tree decomposition of $H$ having width at most $3k$.*

By selecting $k = 1 + \frac{1}{\epsilon}$, we can conclude that the number of edges in $H - S = G - S$ is at least $\left(1 - \frac{1}{k}\right)|E| = \frac{1}{1+\epsilon}|E|$.

Moreover, since a tree decomposition of a graph is also a tree decomposition of its subgraph, we can conclude that the linear-time algorithm in the lemma gives tree decomposition for $G - S = H - S$ which is a subgraph of $H$ with width at most $3k = 3(1 + \frac{1}{\epsilon})$.

### 6.3 Dynamic Programming

In this section, we will present a dynamic programming algorithm that solves the projection game problem in a bounded treewidth bipartite graph $G' = (A', B', E')$ and projections $\pi_e : \Sigma_A \to \Sigma_B$ for each $e \in E'$, 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 $\phi$ 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 $\phi$ 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 $B_j$.

Let $a$ be a two-dimensional array such that $a[B_i][\phi]$ is true if an assignment $\phi$ is possible considered only a subtree starting from $B_i$ and $a[B_i][\phi]$ is false otherwise. The pseudo-code 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      possible ← True
 5      for each edge (u, v) ∈ E'
 6          if u, v ∈ B_i and π_(u,v)(φ(u)) ≠ φ(v)
 7              possible ← 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 φ(x) = φ'(x) for all x ∈ B_i ∩ B_j
12                  if a[B_j][φ'] is True
13                      agree ← True
14              if agree is False
15                  possible ← False
16
17      a[B_i][φ] ← possible
```

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 $\phi$, 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 $O(|B_i| + |B_j|) \leq O(w + 1) = O(w)$ 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 $\phi$ that satisfies the whole graph. The pseudo-code for doing this is shown below. $\phi$ is first initiated to be null. Then the procedure ANSWER is run on the root of $T$. After the program finishes, $\phi$ 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              Break
11  for each children B_j of B_i in T
12      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.

### 6.4 Summary

By using the dynamic programming algorithm presented above to solve a graph $G' = G - S$ got from the thinning step, since the instance is satisfiable, all the edges in $G - S$ can be satisfied. Thus, it gives an assignment that satisfies at least $\frac{1}{1+\epsilon}$ fraction of the edges in the original graph.

Moreover, if we treat $\epsilon$ as a constant, the width $w$ in the tree decomposition is at most $3\left(1+\frac{1}{\epsilon}\right)$ which is a constant. Thus, the dynamic programming algorithm runs in polynomial-time.

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

## References


1. ARORA, S., BARAK, B., AND STEURER, D. Subexponential algorithms for unique games and related problems. In *Proc. 51st IEEE Symp. on Foundations of Computer Science* (2010).
2. ARORA, S., LUND, C., MOTWANI, R., SUDAN, M., AND SZEGEDY, M. Proof verification and the hardness of approximation problems. *Journal of the ACM 45*, 3 (1998), 501–555.
3. ARORA, S., AND SAFRA, S. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM 45*, 1 (1998), 70–122.
4. BABAI, L., FORTNOW, L., LEVIN, L. A., AND SZEGEDY, M. Checking computations in polylogarithmic time. In *Proc. 23rd ACM Symp. on Theory of Computing* (1991), pp. 21–32.
5. BABAI, L., FORTNOW, L., AND LUND, C. Nondeterministic exponential time has two-prover interactive protocols. *Computational Complexity 1* (1991), 3–40.

6. Bellare, M., Goldreich, O., and Sudan, M. Free bits, PCPs, and nonapproximability—towards tight results. *SIAM J. Comput. 27*, 3 (1998), 804–915.
7. Charikar, M., Hajiaghayi, M., and Karloff, H. Improved approximation algorithms for label cover problems. In *In ESA* (2009), Springer, pp. 23–34.
8. Dinur, I., and Steurer, D. Analytical approach to parallel repetition. Tech. Rep. 1305.1979, arXiv, 2013.
9. Feige, U. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM 45*, 4 (1998), 634–652.
10. Garey, M. R., Johnson, D. S., and Stockmeyer, L. Some simplified np-complete problems. In *Proceedings of the sixth annual ACM symposium on Theory of computing* (New York, NY, USA, 1974), STOC '74, ACM, pp. 47–63.
11. Håstad, J. Some optimal inapproximability results. *Journal of the ACM 48*, 4 (2001), 798–859.
12. Holmerin, J., and Khot, S. A new PCP outer verifier with applications to homogeneous linear equations and max-bisection. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing* (New York, NY, USA, 2004), STOC '04, ACM, pp. 11–20.
13. Khot, S. Hardness results for coloring 3-colorable 3-uniform hypergraphs. In *Proc. 43rd IEEE Symp. on Foundations of Computer Science* (2002), pp. 23–32.
14. Khot, S. On the power of unique 2-prover 1-round games. In *Proc. 34th ACM Symp. on Theory of Computing* (2002), pp. 767–775.
15. Klein, P. N. A linear-time approximation scheme for TSP for planar weighted graphs. In *In Proceedings, 46th IEEE Symposium on Foundations of Computer Science* (2005), pp. 146–155.
16. Moshkovitz, D. 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* (2012), vol. 7408, pp. 276–287.
17. Moshkovitz, D., and Raz, R. Two query PCP with sub-constant error. *Journal of the ACM 57*, 5 (2010).
18. Peleg, D. Approximation algorithms for the label-cover max and red-blue set cover problems. *J. of Discrete Algorithms 5*, 1 (Mar. 2007), 55–64.
19. Raz, R. A parallel repetition theorem. In *SIAM J. Comput.* (1998), vol. 27, pp. 763–803.

# Appendix

## A 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 $|N_2(a)|$ 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.

| | |
|---|---|
| $\sigma_b^{max}$ | for each $b \in B$, denotes $\sigma_b \in \Sigma_b$ that maximizes the value of $\sum_{a \in N(b)} \lvert\pi_{(a,b)}^{-1}(\sigma_b)\rvert$. |
| $p_e^{max}$ | for each edge $e = (a, b)$, denotes $\left\lvert\pi_e^{-1}(\sigma_b^{max})\right\rvert$, the size of the preimage of $e$ if $b$ is assigned $\sigma_b^{max}$. |
| $\overline{p}^{max}$ | denotes the average of $p_e^{max}$ over all $e \in E$, i.e. $\frac{1}{\lvert E\rvert}\sum_{e \in E} p_e^{max}$. 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 $N(a)$ for all $a \in A$, i.e., $max_{a \in A}\{\lvert E(N(a))\rvert\}$. |
| $E'$ | 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. $E'_S = \{(a, b) \in E' \mid a \in S \text{ and } b \in S\}$. |
| $N'(u)$ | for each vertex $u$, denotes the set of vertices that are neighbors of $u$ in the graph $G'$. |
| $N'(U)$ | for each set of vertices $U$, denotes the set of vertices that are neighbors of at least one vertex in $U$ in the graph $G'$. |
| $N'_2(u)$ | for each vertex $u$, denotes $N'(N'(u))$, the set of neighbors of neighbors of $u$ in $G'$. |

$\Sigma_A^*(a)$ for each $a \in A$, denotes the set of all assignments $\sigma_a$ to $a$ that, for every $b \in B$, there exists an assignment $\sigma_b$ such that, if $a$ is assigned $\sigma_a$, $b$ is assigned $\sigma_b$ and all $a$'s neighbors are assigned according to $a$, then there are still possible assignments left for all vertices in $N_2(a) \cap N(b)$, i.e., $\{\sigma_a \in \Sigma_A \mid$ for each $b \in B,$ there is $\sigma_b \in \Sigma_B$ such that, for all $a' \in N_2(a) \cap N(b),$ $\left(\bigcap_{b' \in N(a') \cap N(a)} \pi^{-1}_{(a',b')}(\pi_{(a,b')}(\sigma_a))\right) \cap \pi^{-1}_{(a',b)}(\sigma_b) \neq \emptyset\}$. Note that $\sigma_a^{OPT} \in \Sigma_A^*(a)$. In other words, if we replace $\Sigma_A$ with $\Sigma_A^*(a)$ for each $a \in A$, then the resulting instance is still satisfiable.

$N^*(a, \sigma_a)$ for each $a \in A$ and $\sigma_a \in \Sigma_A^*(a)$, denotes $\{b \in N(a) \mid |\pi^{-1}_{(a',b)}(\pi_{(a,b)}(\sigma_a))| \leq 2\overline{p}^{max}$ for some $a' \in N(b)\}$. Provided that we assign $\sigma_a$ 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 $a$.

$N_2^*(a, \sigma_a)$ 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 $N(a)$ when $a$ is assigned $\sigma_a$, i.e., $\bigcup_{b \in N^*(a,\sigma_a)} \{a' \in N(b) \mid |\pi^{-1}_{(a',b)}(\pi_{(a,b)}(\sigma_a))| \leq 2\overline{p}^{max}\}$.

$h^*(a, \sigma_a)$ for each $a \in A$ and $\sigma_a \in \Sigma_A^*(a)$, denotes $|E(N_2^*(a, \sigma_a))|$. In other words, $h^*(a, \sigma_a)$ represents how well $N_2^*(a, \sigma_a)$ spans the graph $G$.

$E^*(a, \sigma_a)$ for each $a \in A$ and $\sigma_a \in \Sigma_A^*(a)$, denotes $\{(a', b) \in E \mid b \in N^*(a, \sigma_a)$, $a' \in N_2^*(a, \sigma_a)$ and $|\pi^{-1}_{(a',b)}(\pi_{(a,b)}(\sigma_a))| \leq 2\overline{p}^{max}\}$. When $a$ is assigned $\sigma_a$, this is the set of all good edges with one endpoint in $N(a)$.

$h^*_{max}$ 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 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

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

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

**Observation 2.** *If* $\sigma_a = \sigma_a^{OPT}$*, then* $N^*(a, \sigma_a) = N'(a)$*,* $N_2^*(a, \sigma_a) = N_2'(a)$ *and* $E^*(a, \sigma_a) = E'(N'(a))$*.*

This observation is obvious since, when pluging in $\sigma_a^{OPT}$, each pair of definitions of $N^*(a, \sigma_a)$ and $N'(a)$, $N_2^*(a, \sigma_a)$ and $N_2'(a)$, and $E^*(a, \sigma_a)$ and $E'(N'(a))$ 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, $\sigma_b^{max}, p_e^{max}, \overline{p}^{max}, E_N^{max}, \Sigma_A^*(a)$, $N^*(a, \sigma_a), N_2^*(a, \sigma_a)$, $h^*(a, \sigma_a)$ 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 algorithms. 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 $\sum_{a \in N(b)} |\pi_{(a,b)}^{-1}(\sigma_b)|$. Each vertex in $A$ is then assigned so that it satisfies as many edges as possible. This algorithm works well when $\Sigma_B$ 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 $N(a_0)$ accordingly. Then, for each node in $N_2(a_0)$, we find one assignment that satisfies all the edges between it and vertices in $N(a_0)$.
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 $N(a_0)$ accordingly. Then, for each node in $N_2(a_0)$, we keep only the assignments that satisfy all the edges between it and vertices in $N(a_0)$.
   When $a_0$ is assigned the optimal assignment, the number of choices for each node in $N_2^*(a_0)$ 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 $N_2^*(a_0)$. This satisfies many edges when there exists $a_0 \in A$ such that $N_2^*(a_0)$ spans many edges.
5. **Divide and Conquer** – $O(n_A n_B(h_{max}^* + E_N^{max})/|E|^2)$**-approximation.** For every $a \in A$, we can fully satisfy $N^*(a) \cup N_2^*(a)$ efficiently, and give up on satisfying other edges that touch this subset. Repeating this process, we can satisfy $\Omega(|E|^2/(n_A n_B(h_{max}^* + E_N^{max})))$ 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.

**Satisfy One Neighbor Algorithm.** The algorithm is exactly the same as that of the uniform case.

**Lemma 8.** *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 1.

**Greedy Assignment Algorithm.** The algorithm is exactly the same as that of the uniform case.

**Lemma 9.** *There exists a polynomial-time $\frac{|\Sigma_A|}{p^{max}}$-approximation algorithm for satisfiable instances of projection games.*

*Proof.* The proof of this lemma differs only slightly from the proof of Lemma 2. The algorithm works as follows:

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

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

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

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 $\Sigma_A$. This can be written as follows.

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

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

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

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

**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 10.** *For each $a_0 \in A$, there exists a polynomial time $\frac{|E|}{|E(N(a_0))|}$-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 $b \in N(a_0)$.
3. For each $a \in N_2(a_0)$, find the set of plausible assignments to $a$, i.e., $S_a = \{\sigma_a \in \Sigma_A \mid \forall b \in N(a) \cap N(a_0), \pi_{(a,b)}(\sigma_a) = \sigma_b\}$. Pick one $\sigma_a^*$ 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 $E(N(a_0))$ get statisfied. This yields $\frac{|E|}{|E(N(a_0))|}$ approximation ratio as desired.

**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 $N_2^*(a_0, \sigma_{a_0})$ instead of $N_2(a_0)$ 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 11.** *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 11, 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 $b \in N(a_0)$.
2. For each $a \in A$, find the set of plausible assignments to $a$, i.e., $S_a = \{\sigma_a \in \Sigma_A \mid \forall b \in N(a) \cap N(a_0), \pi_{(a,b)}(\sigma_a) = \sigma_b\}$. Note that $S_a \neq \emptyset$ from the definition of $\Sigma_A^*(a_0)$.
3. For all $b \in B$, pick an assignment $\sigma_b^*$ for $b$ that maximizes the average number of satisfied edges over all assignments in $S_a$ to vertices $a$ in $N(b) \cap N_2^*(a_0)$, i.e., maximizes $\sum_{a \in N(b) \cap N_2^*(a_0)} |\pi_{(a,b)}^{-1}(\sigma_b) \cap S_a|$.
4. For each vertex $a \in A$, pick an assignment $\sigma_a^* \in S_a$ that maximizes the number of satisfied edges, $|\{b \in N(a) \mid \pi_{(a,b)}(\sigma_a) = \sigma_b^*\}|$ 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

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

Since for each $a \in A$, the assignment $\sigma_a^*$ is chosen to maximize the number of edges satisfied, we can conclude that the number of edges satisfied by selecting $\sigma_a^*$ 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

$$\begin{aligned}
e^* &\geq \sum_{a \in A} \frac{\sum_{\sigma_a \in S_a} |\{b \in N(a) \mid \pi_{(a,b)}(\sigma_a) = \sigma_b^*\}|}{|S_a|} \\
&= \sum_{a \in A} \frac{\sum_{\sigma_a \in S_a} \sum_{b \in N(a)} 1_{\pi_{(a,b)}(\sigma_a) = \sigma_b^*}}{|S_a|} \\
&= \sum_{a \in A} \frac{\sum_{b \in N(a)} \sum_{\sigma_a \in S_a} 1_{\pi_{(a,b)}(\sigma_a) = \sigma_b^*}}{|S_a|} \\
&= \sum_{a \in A} \frac{\sum_{b \in N(a)} |\pi_{(a,b)}^{-1}(\sigma_b^*) \cap S_a|}{|S_a|} \\
&= \sum_{b \in B} \sum_{a \in N(b)} \frac{|\pi_{(a,b)}^{-1}(\sigma_b^*) \cap S_a|}{|S_a|} \\
&\geq \sum_{b \in B} \sum_{a \in N(b) \cap N_2^*(a_0,\sigma_{a_0})} \frac{|\pi_{(a,b)}^{-1}(\sigma_b^*) \cap S_a|}{|S_a|}
\end{aligned}$$

From the definition of $N_2^*(a_0, \sigma_{a_0})$, we can conclude that, for each $a \in N_2^*(a_0, \sigma_{a_0})$, there exists $b' \in N^*(a_0) \cap N(a)$ such that $|\pi^{-1}_{(a,b')}(\sigma_{b'})| \leq 2\overline{p}^{max}$. Moreover, from the definition of $S_a$, we have $S_a \subseteq \pi^{-1}_{(a,b')}(\sigma_{b'})$. As a result, we can arrive at the following inequalities.

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

This implies that

$$e^* \geq \frac{1}{2\overline{p}^{max}} \sum_{b \in B} \sum_{a \in N(b) \cap N_2^*(a_0, \sigma_{a_0})} |\pi^{-1}_{(a,b)}(\sigma_b^*) \cap S_a|.$$

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 $a \in N_2(a_0) \cap N(b)$. Since $N_2^*(a_0, \sigma_{a_0}) \subseteq N_2(a_0)$, we can conclude that $|\pi^{-1}_{(a,b)}(\sigma_b) \cap S_a| \geq 1$ for all $a \in N_2^*(a_0, \sigma_{a_0}) \cap N(b)$.

Since we pick the assignment $\sigma_b^*$ that maximizes $\sum_{a \in N(b) \cap N_2^*(a_0)} |\pi^{-1}_{(a,b)}(\sigma_b^*) \cap S_a|$ for each $b \in B$, we can conclude that

$$\begin{aligned} e^* &\geq \frac{1}{2\overline{p}^{max}} \sum_{b \in B} \sum_{a \in N(b) \cap N_2^*(a_0, \sigma_{a_0})} |\pi^{-1}_{(a,b)}(\sigma_b^*) \cap S_a| \\ &\geq \frac{1}{2\overline{p}^{max}} \sum_{b \in B} \sum_{a \in N(b) \cap N_2^*(a_0, \sigma_{a_0})} |\pi^{-1}_{(a,b)}(\sigma_b) \cap S_a| \\ &\geq \frac{1}{2\overline{p}^{max}} \sum_{b \in B} \sum_{a \in N(b) \cap N_2^*(a_0, \sigma_{a_0})} 1. \end{aligned}$$

The last term can be rewritten as

$$\begin{aligned} \frac{1}{2\overline{p}^{max}} \sum_{b \in B} \sum_{a \in N(b) \cap N_2^*(a_0, \sigma_{a_0})} 1 &= \frac{1}{2\overline{p}^{max}} \sum_{a \in N_2^*(a_0, \sigma_{a_0})} \sum_{b \in N(a)} 1 \\ &= \frac{1}{2\overline{p}^{max}} \sum_{a \in N_2^*(a_0, \sigma_{a_0})} d_a \\ &= \frac{h^*(a_0, \sigma_{a_0})}{2\overline{p}^{max}}. \end{aligned}$$

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.

**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 12.** *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 $\mathcal{P} \leftarrow \mathcal{P} \cup \{(N_2^*(a, \sigma_a) \cup N^*(a, \sigma_a)) - V_{\mathcal{P}}\}$.
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 $\sigma_a$ 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 2, we have $E^*(a, \sigma^{OPT}) = E'(N'(a))$. As a result, we have

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

for all $a \in A$.

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

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

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

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

Let $N^p(v) = N'(v) - V_{\mathcal{P}}$ and $N_2^p(v) = N'_2(v) - V_{\mathcal{P}}$. 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:

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

From Jensen's inequality, we have

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

Since $\frac{|E|^2}{16n_B} \geq \sum_{a\in A} |E_{(N'(a)\cup N_2'(a))-V_{\mathcal{P}}}|$ and $\sum_{a\in A} |E_{(N'(a)\cup N_2'(a))-V_{\mathcal{P}}}| \geq \frac{1}{n_B}\left|E'_{(A\cup B)-V_{\mathcal{P}}}\right|^2$, 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

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

From Observation 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 $\Omega\left(\frac{|E|^3}{n_A n_B (h^*_{max}+E_N^{max})}\right)$ 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 $|E_{\mathcal{P}}| \geq \frac{|E|^2}{16n_A n_B (h^*_{max}+E_N^{max})}\left(|E| - \left|E_{(A\cup B)-V_{\mathcal{P}}}\right|\right)$. 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, $|E_{\mathcal{P}}| \geq \frac{|E|^2}{16n_A n_B (h^*_{max}+E_N^{max})}\left(|E| - \left|E_{(A\cup B)-V_{\mathcal{P}}}\right|\right)$.

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

Since $|E_{\mathcal{P}_{new}}| = |E_{\mathcal{P}_{old}}| + |E_{(N^*(a,\sigma_a)\cup N_2^*(a,\sigma_a))-V_{\mathcal{P}_{old}}}|$, we have

$$\begin{aligned} |E_{\mathcal{P}_{new}}| &= |E_{\mathcal{P}_{old}}| + |E_{(N^*(a,\sigma_a)\cup N_2^*(a,\sigma_a))-V_{\mathcal{P}_{old}}}| \\ &= |E_{\mathcal{P}_{old}}| + |E_{(N^*(a,\sigma_a)\cup N_2^*(a,\sigma_a))} \cap E_{(A\cup B)-V_{\mathcal{P}_{old}}}|. \end{aligned}$$

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, $E_{(N^*(a,\sigma_a)\cup N_2^*(a,\sigma_a))} \supseteq E^*(a,\sigma_a)$ holds. As a result, we have

$$\begin{aligned} |E_{\mathcal{P}_{new}}| &= |E_{\mathcal{P}_{old}}| + |E_{(N^*(a,\sigma_a)\cup N_2^*(a,\sigma_a))} \cap E_{A\cup B-V_{\mathcal{P}_{old}}}| \\ &\geq |E_{\mathcal{P}_{old}}| + |E^*(a,\sigma_a) \cap E_{(A\cup B)-V_{\mathcal{P}_{old}}}| \\ &\geq |E_{\mathcal{P}_{old}}| + \frac{1}{16}\frac{|E|^2}{n_A n_B}. \end{aligned}$$

Now, consider $\left(|E| - |E_{(A\cup B)-V_{\mathcal{P}_{new}}}|\right) - \left(|E| - |E_{(A\cup B)-V_{\mathcal{P}_{old}}}|\right)$. 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 $V_{\mathcal{P}_{new}} = V_{\mathcal{P}_{old}} \cup (N_2^*(a,\sigma_a) \cup N^*(a,\sigma_a))$, we can conclude that

$$((A\cup B) - V_{\mathcal{P}_{old}}) \subseteq ((A\cup B) - V_{\mathcal{P}_{new}}) \cup (N_2^*(a,\sigma_a) \cup N^*(a,\sigma_a)).$$

Thus, we can also derive

$$\begin{aligned} E_{(A\cup B)-V_{\mathcal{P}_{old}}} &\subseteq E_{((A\cup B)-V_{\mathcal{P}_{new}})\cup\left(N_2^*(a,\sigma_a)\cup N^*(a,\sigma_a)\right)} \\ &= E_{(A\cup B)-V_{\mathcal{P}_{new}}} \cup \{(a',b') \in E \mid a' \in N_2^*(a,\sigma_a) \text{ or } b' \in N^*(a,\sigma_a)\}. \end{aligned}$$

Moreover, we can write $\{(a',b') \in E \mid a' \in N_2^*(a,\sigma_a) \text{ or } b' \in N^*(a,\sigma_a)\}$ as $\{(a',b') \in E \mid a' \in N_2^*(a,\sigma_a)\} \cup \{(a',b') \in E \mid b' \in N^*(a,\sigma_a)\}$. Since $N^*(a,\sigma_a) \subseteq N(a)$, we can conclude that

$$\begin{aligned} \{(a',b') \in E \mid a' \in N_2^*(a,\sigma_a) \text{ or } b' \in N^*(a,\sigma_a)\} \subseteq &\{(a',b') \in E \mid a' \in N_2^*(a,\sigma_a)\} \\ &\cup \{(a',b') \in E \mid b' \in N(a)\}. \end{aligned}$$

Thus, we can conclude that

$$\begin{aligned} |\{(a',b') \in E \mid a' \in N_2^*(a,\sigma_a) \text{ or } b' \in N^*(a,\sigma_a)\}| &\leq |\{(a',b') \in E \mid a' \in N_2^*(a,\sigma_a)\}| \\ &\quad + |\{(a',b') \in E \mid b' \in N(a)\}| \\ &= h^*(a,\sigma_a) + |E(N(a))|. \end{aligned}$$

Hence, we can conclude that

$$\begin{aligned} \left|E_{(A\cup B)-V_{\mathcal{P}_{old}}}\right| &\leq \left|E_{(A\cup B)-V_{\mathcal{P}_{new}}} \cup \{(a',b') \in E \mid a' \in N_2(a) \text{ or } b' \in N(a)\}\right| \\ &\leq \left|E_{(A\cup B)-V_{\mathcal{P}_{new}}}\right| + |\{(a',b') \in E \mid a' \in N_2(a) \text{ or } b' \in N(a)\}| \\ &\leq \left|E_{(A\cup B)-V_{\mathcal{P}_{new}}}\right| + h^*(a,\sigma_a) + |E(N(a))| \\ &\leq \left|E_{(A\cup B)-V_{\mathcal{P}_{new}}}\right| + h^*_{max} + E_N^{max}. \end{aligned}$$

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

Hence, since $\left(|E| - \left|E_{(A\cup B)-V_{\mathcal{P}}}\right|\right)$ increases by at most $h^*_{max} + E_N^{max}$ and $|E_{\mathcal{P}}|$ increases 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 12.

**Proof of Theorem 2**

*Proof.* Using Lemma 11 with $a_0$ and $\sigma_{a_0}$ 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 10 with $a_0$ that maximizes the value of $E(N(a_0))$, i.e., $|E(N(a_0))| = E_N^{max}$, there exists a polynomial-time $\frac{|E|}{E_N^{max}}$-approximation algorithm for satisfiable instances of projection games.

Moreover, from Lemmas 8, 9 and 12, 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 $h^*_{max} \geq E_N^{max}$, we have $O(n_A n_B (h^*_{max}+E_N^{max})/|E|^2) = O(n_A n_B h^*_{max}/|E|^2)$. 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 $E_N^{max} > h^*_{max}$, we have $O(n_A n_B (h^*_{max}+E_N^{max})/|E|^2) = O(n_A n_B E_N^{max}/|E|^2)$. 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.,

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

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 2 for the nonuniform preimage sizes case.