Approximation Algorithms for Label Cover and The Log-Density Threshold

Eden Chlamtáč * Pasin Manurangsi † Dana Moshkovitz ‡

Aravindan Vijayaraghavan §

Abstract

Many known optimal NP-hardness of approximation results are reductions from a problem called Label-Cover. The input is a bipartite graph $G = (L, R, E)$ and each edge $e = (x, y) \in E$ carries a projection π_e that maps labels to x to labels to y. The objective is to find a labeling of the vertices that satisfies as many of the projections as possible. It is believed that the best approximation ratio efficiently achievable for Label-Cover is of the form N^{-c} where $N = nk$, n is the number of vertices, k is the number of labels, and $0 < c < 1$ is *some* constant.

Inspired by a framework originally developed for Densest k-Subgraph, we propose a "log density threshold" for the approximability of Label-Cover. Specifically, we suggest the possibility that the Label-Cover approximation problem undergoes a computational phase transition at the same threshold at which local algorithms for its random counterpart fail. This threshold is $N^{3-2\sqrt{2}} \approx N^{-0.17}$. We then design, for any $\varepsilon > 0$, a polynomial-time approximation algorithm for *semi-random* Label-Cover whose approximation ratio is $N^{3-2\sqrt{2}+\varepsilon}$. In our semi-random model, the input graph is random (or even just expanding), and the projections on the edges are arbitrary.

For *worst-case* Label-Cover we show a polynomial-time algorithm whose approximation ratio is roughly $N^{-0.233}$. The previous best efficient approximation ratio was $N^{-0.25}$. We present some evidence towards an N^{-c} threshold by constructing integrality gaps for $N^{\Omega(1)}$ rounds of the Sum-of-squares/Lasserre hierarchy of the natural relaxation of Label Cover. For general 2CSP the "log density threshold" is $N^{-0.25}$, and we give a polynomial-time algorithm in the semi-random model whose approximation ratio is $N^{-0.25+\varepsilon}$ for any $\varepsilon > 0$.

1 Introduction

1.1 Label Cover In the past couple of decades researchers have succeeded in basing many known optimal NP-hardness of approximation results on the hardness of a combinatorial optimization problem called Label-Cover. The decision version of this problem relevant to hardness of approximation is parameterized by an approximation ratio $\delta : \mathbb{N} \times \mathbb{N} \to [0, 1]$.

> Label-Cover:
> Input: A bipartite graph $G = (L, R, E)$, where the L vertices have degree Δ_L, and the R vertices have degree Δ_R; finite alphabets Σ_L, Σ_R; for each one of the edges $e \in E$, a function $\pi_e : \Sigma_L \to \Sigma_R \cup \{\perp\}$.
> Goal: Distinguish between the following two cases:
>
> - *Completeness/accept:* There are labels $\phi_L : L \to \Sigma_L$ and $\phi_R : R \to \Sigma_R$ such that $\pi_e(\phi_L(x)) = \phi_R(y)$ for all $e = (x, y) \in E$.
> - *Soundness/reject:* For all labels $\phi_L : L \to \Sigma_L$ and $\phi_R : R \to \Sigma_R$, for at most $\delta = \delta(|L|, |\Sigma_L|)$ fraction of $e = (x, y) \in E$ we have $\pi_e(\phi_L(x)) = \phi_R(y)$.

In the approximation algorithms literature, as opposed to the hardness of approximation literature, the term

*chlamtac@cs.bgu.ac.il. Department of Computer Science, Ben-Gurion University. Partially supported by ISF grant 1002/14.

†pasin@berkeley.edu. Department of Electrical Engineering and Computer Science, UC Berkeley. This material is based upon work supported by the National Science Foundation under grants number CCF 1540685 and CCF 1655215.

‡danama@cs.utexas.edu. Department of Computer Science, UT Austin. This material is based upon work supported by the National Science Foundation under grants number 1218547 and 1648712.

§aravindv@northwestern.edu. Department of Electrical Engineering and Computer Science, Northwestern University. Partially supported by NSF grant CCF-1637585.

Label-Cover is used when the constraints on the edges are general predicates, and not only when they are functions ("projections"). In this work we will stick to the terminology of the vast hardness of approximation literature. We will use the term 2CSP to refer to the problem with general constraints.

We denote the number of vertices by n and the alphabets size by k. We denote the input size by $N = nk$. The condition $\pi_e(\phi_L(x)) = \phi_R(y)$ is called a "projection test". If the test holds we say that the labels ϕ_L, ϕ_R *satisfy* the edge $e = (x, y)$. Label-Cover is also referred to as a "projection game" (the famous "unique games" [Kho02] were defined analogously). The maximum fraction of edges that can be satisfied simultaneously is called the *value* of the game.

A different perspective on Label-Cover is as a *robust* constraint satisfaction problem (CSP). In a CSP the input is a collection of constraints over a large number of variables, where each constraint depends on Δ variables. The variables may assume labels over an alphabet Σ and each constraint specifies a subset of Σ^Δ. A constraint π is satisfied if its Δ variables are assigned labels in its set. Let I_π indicate whether π is satisfied. The goal is to label the variables as to maximize $\sum_\pi I_\pi$. In a *robust* CSP the input is the same as in CSP, but the goal is different. Rather than having a binary I_π that indicates whether π is satisfied or not, we define a real $\tilde{I}_\pi$ that measures π's proximity to being satisfied. Formally, the agreement between two strings in Σ^Δ is the fraction of the Δ indices on which the two strings are the same, and $\tilde{I}_\pi$ is the agreement of π's Δ variables with a satisfying assignment. The goal is to maximize $\sum_\pi \tilde{I}_\pi$. Label Cover is equivalent to a robust CSP where L corresponds to the constraints, $\Delta = \Delta_L$, and R corresponds to the variables, $\Sigma = \Sigma_R$. The labels to the L vertices correspond to satisfying assignments to the constraints.

The Projection Games Conjecture [BGLR93, Mos15] states that Label-Cover is NP-hard for some $\delta = \Theta(1/n^c)$ for a constant $0 < c < 1$ ("polynomially small"). If it were proved, one could prove polynomial hardness for the Closest-Vector-Problem [Kho10] in lattices, for Directed-Sparsest-Cut [CK09] and many other problems [Mos15]. Currently, the best NP-hardness result known for Label-Cover [MR10] is for poly-logarithmically small δ. Under the stronger assumption that NP does not have quasi-polynomial time algorithms, one can prove that there are no polynomial time algorithms for $\delta = 2^{-\Omega(\sqrt{\log N})}$. In contrast, the (previously) best polynomial-time approximation algorithm works for $\delta = \Theta(N^{-1/4})$ [MM13].

In this work we study random, semi-random and worst-case Label-Cover. In random Label-Cover both the graph G and the projections $\{\pi_e\}$ are random. In semi-random Label-Cover the graph G is random but the projections are arbitrary. In worst-case Label-Cover both the graph and the projections are arbitrary. Random CSP is extensively studied due to its importance to average-case complexity, probability and statistical physics [MZK+99], cryptography [ABW10], hardness of approximation [Fei02] and other fields. It is also a prototypical example of a hard CSP [Gri01, Sch08]. Random Label-Cover, i.e., random *approximate* CSP, is therefore a natural object for study. Moreover, Label-Cover on random (or pseudorandom) graphs is extremely useful for hardness of approximation, and there are several known reductions that take advantage of it, e.g., [Fei02, AAM+11, HK04]. Indeed, Label-Cover is known to be NP-hard in certain regimes even when the underlying graph obeys random-like expansion properties (see, e.g., [HK04]).

1.2 Main Theorems In this work we show approximation algorithms for Label-Cover that achieve better approximation ratio than existing algorithms, as well as Lasserre integrality gaps. Some of the algorithms approach the "log density threshold" for Label-Cover, a natural threshold explained in the next section, which *might* correspond to the true computational threshold for Label-Cover. Quantitatively, the threshold is $N^{-\beta(1-\beta)/(2-\beta)}$ where $|L| = N^\beta$. For the worst β the log density threshold is $N^{-(3-2\sqrt{2})} \approx N^{-0.17}$.

We show a polynomial-time algorithm that works for random graphs and worst-case projections, and whose approximation ratio comes arbitrarily close to the log density threshold (see Section 5).

Theorem 1.1. (Semi-random Label-Cover) *For every $\varepsilon > 0$, there is a approximation algorithm running in time $N^{O(1/\varepsilon^2)}$ for* Label-Cover *with $\delta = N^{-\beta(1-\beta)/(2-\beta)-\varepsilon}$ on random graphs $G = (L, R, E)$ with alphabet size k, where $N = |L|k$ and $|L| = N^\beta$. The algorithm works with high probability over the choice of G.*

We will provide more details about the "log density method" this algorithm is based on in the next section.

The algorithm in Theorem 1.1 works even if G is only "weakly expanding" as opposed to random (for exact definitions, see the appendix), and even if the completeness case has $\tilde{\Theta}(1)$ fraction of satisfied edges as opposed to all edges. For Label-Cover where all edges are satisfied

in the completeness case there was a polynomial-time algorithm with $\Theta(N^{1/4})$ approximation [MM13]. We improve on this algorithm by showing a polynomial-time algorithm for $\delta \approx N^{-0.233}$ (see Section 6).

THEOREM 1.2. (WORST-CASE LABEL-COVER) *For any constant $\varepsilon > 0$, there exists a polynomial-time approximation algorithm for* LABEL-COVER *with $\delta = O\left(N^{-\frac{1}{6}(5-\sqrt{13})-\varepsilon}\right)$.*[1]

Finally, we put forward the *possibility* that the log density threshold gives the correct exponent for the Projection Games Conjecture:

CONJECTURE 1.1. *For any $\varepsilon > 0$,* LABEL-COVER *on alphabet size $k = N^{1-\beta}$ is NP-hard for $\delta = N^{-\beta(1-\beta)/(2-\beta)+\varepsilon}$. Furthermore, there are no efficient algorithms for the problem even when the underlying graph is random with average degree $N^{\beta(1-\beta)/(2-\beta)}$.*

1.3 Additional Results In addition to the above theorems, we now list some further results whose proofs we will omit due to space limitations. These will appear in the full version of the paper.

First, we consider 2CSP. Here, the log density threshold is $N^{-\beta(1-\beta)}$, and we show a matching approximation algorithm for the semi-random case.

THEOREM 1.3. (SEMI-RANDOM ALGORITHM, 2CSP) *For every $\varepsilon > 0$, there is an approximation algorithm for 2CSP running in time $N^{O(1/\varepsilon^2)}$ with $\delta = N^{-\beta(1-\beta)-\varepsilon}$ on random graphs $G = (L, R, E)$ with alphabet size k, where $N = nk$ and $|L| = N^{\beta}$. The algorithm works with high probability over the choice of G.*

This gives a approximation ratio of $N^{1/4+\varepsilon}$ for semirandom instances. For worst case 2CSP the best efficient algorithm known gives only $\Theta(N^{1/3})$-approximation [CHK09], but works in the more general case where the fraction of satisfied edges in the completeness case is arbitrary.

We prove limitations on approximation algorithms as well. We show an $N^{1/8-\varepsilon}$ integrality gap for the $N^{\Omega(\varepsilon)}$-level Lasserre relaxation of LABEL-COVER, as stated below. We remark that the gap instance is in fact semi-random, as the graph is a random left-regular bipartite graph, and thus can be $N^{3-2\sqrt{2}-\varepsilon}$-approximated in polynomial time by our algorithm.

THEOREM 1.4. *For every constant $0 < \varepsilon < 1/8$ and sufficiently large N, the integrality gap of the $N^{\Omega(\varepsilon)}$-level Lasserre SDP relaxation of Label Cover of size N is at least $N^{1/8-\varepsilon}$.*

We show that, even after $N^{1-\varepsilon}$ rounds of the Lasserre SDP hierarchy, there is still a polynomial lower bound of $N^{\Omega(\varepsilon)}$ on the integrality gap. Note that this gap matches, up to a multiplicative constant in $\Omega(\varepsilon)$, with the naive algorithm that tries every assignment to $N^{1-\varepsilon}$ vertices (see Theorem 6.1 in [Man15]).

THEOREM 1.5. *For every constant $0 < \varepsilon < 1$ and sufficiently large N, the integrality gap of the $N^{1-\varepsilon}$-level Lasserre SDP relaxation of Label Cover of size N is at least $N^{\Omega(\varepsilon)}$.*

Since the r-level Lasserre relaxation takes $N^{O(r)}$ time to solve, the above integrality gaps imply that the Lasserre hierarchy cannot refute the Projection Game Conjecture. Given how powerful semidefinite programs and Lasserre hierarchy are in approximating CSPs on small alphabets [Rag08, LRS15], our result is an indication that the Projection Games Conjecture may indeed be true. Moreover, since all our algorithms can be described in terms of rounding linear programs, Theorem 1.4 presents a barrier for such approaches, i.e., it implies that no such algorithm can achieve $N^{1/8-\varepsilon}$-approximation in polynomial time. Unfortunately, this is still not a tight lower bound and we leave it to future work to improve this bound to match our $N^{3-2\sqrt{2}+\varepsilon}$-approximation algorithm.

To prove the Lasserre integrality gaps, we reduce from an integrality gap of random MAX K-CSP from [Tul09]. The reduction and proof follow closely from those in [BCV+12], in which similar integrality gaps for DENSEST k-SUBGRAPH were shown. In fact, the only main difference is that we prove a stronger soundness result, which ultimately leads to a larger gap in Theorem 1.4. Our analysis also yields the following integrality gap for DENSEST k-SUBGRAPH, improving upon the $N^{\Omega(\varepsilon)}$-level $N^{2/53-\varepsilon}$-gap of [BCV+12].

THEOREM 1.6. *For every $0 < \varepsilon < 1/14$ and sufficiently large N, the integrality gap of the $N^{\Omega(\varepsilon)}$-level Lasserre SDP relaxation of* DENSEST k-SUBGRAPH *on a graph of N vertices is at least $N^{1/14-\varepsilon}$.*

2 Notation

Before we explain the intuition behind our algorithms, let us define additional conventions to be used in the paper. A LABEL-COVER instance $\mathcal{G} = (L, R, E, \Sigma_L, \Sigma_R, \{\pi_e\}_{e\in E})$ is said to be *satisfiable* or *fully satisfiable* if its value is one. The graph $G = (L, R, E)$ is

[1] $\frac{1}{6}(5-\sqrt{13}) = 0.23240812075600178448\ldots$.

called the *supergraph* of $\mathcal{G}$. The *label extended graph* associated with the Label-Cover instance, denoted G_{ext} has vertices $L \times \Sigma_L$ and $R \times \Sigma_R$. An edge in G_{ext} connects (x, σ_x) to (y, σ_y) if $(x, y) \in E$ and $\pi_{(x,y)}(\sigma_x) = \sigma_y$.

An important procedure in our algorithms is label sets reduction, in which we discard some labels that we are certain are not the satisfying labels. We are then left with candidate labels for each vertex. When we focus on a set $S \subseteq L \cup R$, we use a function $\Sigma_S : S \to \mathcal{P}(\Sigma_L \cup \Sigma_R)$ where $\mathcal{P}(\Sigma_L \cup \Sigma_R)$ is the power set of $\Sigma_L \cup \Sigma_R$ to represent the reduced label sets of all vertices in S, i.e., $\Sigma_S(u) \subseteq \Sigma_L \cup \Sigma_R$ is the reduced label set of u.

We use $\Gamma^G(u)$ to denote the set of neighbors of a vertex u in a graph G and $\Gamma^G(S)$ to denote the $\bigcup_{u \in S} \Gamma^G(S)$ for $S \subseteq L \cup R$. For $i \geq 2$, we write $\Gamma^G_i(S)$ to denote $\Gamma^G(\Gamma^G_{i-1}(S))$ and $\Gamma^G_1(S) = \Gamma^G(S$. $deg^G(u)$ is defined as the degree of u in the graph G. In addition, for $S, S' \subseteq L \cup R$, we write $E^G(S, S')$ to denote the set of all edges whose one end point is in S and the other is in S'; as a shorthand, we write $E^G(S)$ to denote $E^G(S, L \cup R)$, the set of edges whose at least one endpoint is in S. When it is clear from the context which graph we are refering to, we may drop G altogether.

We say that a bipartite graph (L, R, E) is *biregular* if the degrees of all vertices in each side is equal. Similarly, (L, R, E) is said to be λ-*nearly biregular* if, on each side, the degrees of any two vertices are at most λ times each other, i.e., for every $u, u' \in L$ and $v, v' \in R$, we have $deg(u) \leq \lambda \cdot deg(u'), deg(v) \leq \lambda \cdot deg(v')$. We say that a bipartite graph is *nearly-biregular* as a shorthand for $O(\log^C(n))$-*nearly biregular* where C is some constant and n is the number of vertices in the graph.

Throughout the paper, we use polylog$(f(n))$ to denote $\log^C(f(n))$ for some constant C. In addition, from this point on, we will abuse the notations $\tilde{O}$ and $\tilde{\Omega}$, and use them to hide a factor of polylog$(n_L n_R k_L k_R)$; this is in contrast to the standard convention, in which $\tilde{O}(f(n))$ and $\tilde{\Omega}(f(n))$ represent $O(f(n)\text{polylog}(f(n))$ and $\Omega(f(n)\text{polylog}(f(n)))$ respectively.

3 The Log Density Method

Some of our algorithms use a method first introduced in [BCC+10] to study Densest k-Subgraph, and which has since been used successfully for the related problems of Smallest k-Edge Subgraph [CDK12] and Small Set Bipartite Vertex Expansion [CDM17]. The method consists of the following steps, which follow in the rest of the section.

1. *Random Problems:* Study random problems, where the goal is to distinguish a random input from a random input with a planted solution.
2. *Witnesses:* Restrict attention to constant-sized witnesses, such that the witness occurs and is satisfied in an instance with a planted solution, whereas the witness occurs but is not satisfied with high probability in a random (unsatisfiable) instance.
3. *Log density threshold:* There is typically a threshold (depending on the size of the graph, the degree, etc) at which such witnesses appear and this threshold is relatively simple to compute. This threshold is called the "log density threshold" for reasons that will become clear shortly.
4. *Algorithms:* One can often design efficient algorithms inspired by the random model, whose approximation ratio approaches the log density threshold *for more general cases than just the random case.* This often involves a subtle argument, which must show that either the input is sufficiently random looking (in order to emulate algorithms for random models), or take advantage of non-random behavior in the input in order to produce a good solution.

Interestingly, even though the log density method uses a simplified model (the random model) and an incredibly simple algorithmic technique (constant-sized witnesses), for Densest k-Subgraph it captures existing algorithmic ideas, including much more sophisticated techniques: spectral and semidefinite programming based algorithms do not improve over that threshold [BCC+10] and there is an SA+ integrality gap with the same ratio [BCV+12]. It is believed that the log density threshold is the computational threshold for Densest k-Subgraph, namely, where the approximation problem changes from tractable to intractable. In fact, this log-density threshold has been used as an average-case hardness assumption in [ABBG10, ACLR15].

In this section we define the random problem for Label-Cover and derive the log density threshold for it. The difficult, elusive, step of the log density method is to design algorithms whose approximation ratio approaches the log density threshold, and much of the rest of the paper is devoted to this challenging task.

Random Label Cover: Given a Label-Cover instance of one of the following forms, determine which of the two distributions it was drawn from.

$\mathcal{D}_{NO}$: $G(L, R, E)$ is an Erdős-Rényi random bipartite graph $\mathcal{G}(n/2, n/2, p = \Delta/n)$, and each projection π_e between $e = (u, v)$ is given by a random right-regular

bipartite graph of degree $d = k^\gamma$.

$\mathcal{D}_{YES}$: $G(L, R, E)$ is an Erdős-Rényi random bipartite graph $\mathcal{G}(n/2, n/2, p = \Delta/n)$, and the projections $\{\pi_e\}_{e\in E}$ are chosen arbitrarily in such a way that it satisfies at least one assignment.

For simplicity, in this section we consider $|L| = |R| = n/2$ and $\Delta_L = \Delta_R \doteq \Delta$. Furthermore, we consider projections which are d to 1 functions only, so $|\Sigma_R| = k/d$ where $|\Sigma_L| = k$. Intuitively, this "regular" case is the hardest, since an algorithm cannot make progress by exploiting local irregularities.

We consider a family of algorithms based on counting local witness templates. Any such algorithm will use a constant sized graph $\mathcal{W}$, and look for occurences[2] $H \subseteq G$ of $\mathcal{W}$ in the graph G. If the instance is satisfiable, then H can be satisfied by a suitable assignment of the variables. Since $\mathcal{W}$ only has constant size, this can be checked in polynomial time. The hope is to find $\mathcal{W}$ such that its occurrences are not satisfiable with high probability in the purely random case, allowing us to distinguish. Any witness template $\mathcal{W}$ needs to satisfy two main properties to refute instances generated in the soundness case:

1. The template $\mathcal{W}$ should occur in G (a random graph of degree Δ),
2. For random instances, every assignment to an occurrence H of $\mathcal{W}$ should fail with high probability.

Let the witness template $\mathcal{W}$ have a vertices (with a_L of them on the left, and a_R of them on the right) and b edges respectively. In the calculations that follow, we will ignore constant factors for convenience. To satisfy the first property in expectation, we require the following condition on a and b to hold:

(3.1)
$$\mathbf{E}\left[\text{number of occurrences of } \mathcal{W}\right] = \left(\frac{n}{2}\right)^a \left(\frac{\Delta}{n}\right)^b > 1$$

This upper bound on the edge density corresponds to the well-studied threshold phenomenon in random graphs. Consider an occurrence $H \subseteq G$ of $\mathcal{W}$. For $(u, v) \in E$, the vertices $u \in L, v \in R$ have k labels and k/d labels respectively, and a fixed label assignment to u and v satisfies π_e with probability d/k. Hence, to satisfy the second property, we need

(3.2) $\mathbf{E}\left[\text{number of satisfying assignments for } H\right]$

$$= k^{a_L} \left(\tfrac{k}{d}\right)^{a_R} \left(\tfrac{d}{k}\right)^b = k^a \left(\frac{d^{b-a_R}}{k^b}\right) < 1. \tag{3.3}$$

Thus it would seem, based on the above, that we could distinguish whenever the parameters n, Δ, k, d are such that there exists such a witness $\mathcal{W}$ as above satisfying

$$\frac{b-a}{b} < \frac{\log \Delta}{\log n} \qquad \text{and} \qquad \frac{b-a}{b-a_R} > \frac{\log d}{\log k}. \tag{3.4}$$

It would seem advantageous to design the witness $\mathcal{W}$ so that as many of its a nodes as possible are on the right, in order to make the upper bound on the right as unrestrictive as possible. However, this can only be done within some limit: while we have been using expectation to imply the existence of the witness $\mathcal{W}$, it turns out that this only corresponds to a high-probability event when the degrees in $\mathcal{W}$ are all at least 2, and thus we require in particular that $a_r \leq b/2$, meaning that in the best possible scenario, we would need the existence of a witness $\mathcal{W}$ with parameters a, b such that

$$\frac{b-a}{b} < \frac{\log \Delta}{\log n} \qquad \text{and} \qquad \frac{b-a}{b/2} > \frac{\log d}{\log k}.$$

In other words, a necessary condition for distinguishing is that

$$\frac{\log(\Delta^2)}{\log n} > \frac{\log d}{\log k}. \tag{3.5}$$

It turns out that in fact this is also a sufficient condition, and that in fact, when the above inequality holds with a constant additive gap, we *can* find a witness $\mathcal{W}$ satisfying (3.4).

Motivated by the above calculation we define log density as follows.

Definition 3.1. *In a* Label Cover *instance as above, the log density of the supergraph is* $\log(\Delta^2)/\log|V|$ *(or more generally,* $\log(\Delta_L\Delta_R)/\log|L|$*), while the log density of the projections is* $\log d/\log k$*. We will use the term* log density gap *to refer to the excess (difference) of the log density of the supergraph over the log density of the projections.*

Further, the above calculation shows that such *local algorithms cannot work* if $\log_n(\Delta^2) < \log_k d$. This log-density condition points to a barrier for local algorithms to approximate Label Cover, much like the log-density barrier for Densest k-subgraph. This is formalized in Conjecture 1.1. To see where our approximation guarantee comes from, we need to examine simpler algorithms which we must resort to when there is no log density gap.

[2] An occurrence of $\mathcal{W}$ is just a subgraph $H \subseteq G$ that is isomorphic to $\mathcal{W}$.

Simple algorithms and the derivation of the threshold. For a Label-Cover instance with degree Δ one can find a labeling that satisfies $1/\Delta$ fraction of the edges by picking a perfect matching in the graph and satisfying only those edges. For a Label-Cover instance whose projections are d to 1 and whose alphabet is of size k, one can find a labeling that satisfies d/k fraction of the edges by picking the labels of the R vertices randomly and picking the labels of the L vertices in a greedy fashion. If we have a gap in the log-density, then the above local witness distinguishes a random label cover instance from a satisfiable instance. If there is no log-density gap then the simple approximations give a $\min\{\Delta, k/d\}$ approximation. If $d = k^\gamma$, by balancing the guarantees of these different algorithms, the worst setting of parameter γ is when $\gamma = \frac{2-2\beta}{2-\beta}$. Substituting, we see that that log-density threshold is N^{-c} for $c = \beta(1-\beta)/(2-\beta)$. Maximizing over $\beta \in [0,1]$ gives an $c = 3 - 2\sqrt{2}$.

4 Algorithms for Random Label Cover based on Counting Local Structures

We now describe counting-based algorithms for Random Label Cover which we will later build on in when designing an algorithm for semi-random models. Consider the distinguishing problem described in the previous section, in which the algorithm must distinguish between a random instance of Label Cover and a fully satisfiable instance.

As noted earlier, a necessary condition for a certain class of distinguishing algorithms to succeed is that we have a positive log-density gap, i.e., that inequality (3.5) holds with a constant additive gap. In what follows, let us assume that this is indeed the case. Specifically, let us assume that $d \leq k^\gamma$ and $\Delta \geq n^{\alpha/2}$ for some constants $\gamma < \alpha$. Our algorithms will be parametrized by a rational approximation to the log-density. Thus, let r, s be integers and $\varepsilon_1, \varepsilon_2 > 0$ be constants such that $\gamma + \varepsilon_1 \leq r/s \leq \alpha - \varepsilon_2$. When this condition holds, we will show that we can successfully solve the distinguishing problem, as well as find a better-than-random assignment when a satisfying assignment is planted in a random instance. Note that, if we do not have a positive log-density gap, we can still run the simple algorithms described earlier and satisfy at least an $\Omega(N^{-(3-2\sqrt{2})})$ fraction of all edges.

4.1 Distinguishing algorithm and Witness Templates.

Our distinguishing algorithm consists of a family of algorithms Alg-Dist$_{r,s}$ given by different local witness templates $\mathcal{W}_{r,s}$ that are parameterized by two integers r, s. Our algorithm will differ slightly from the general recipe of a local witness-based algorithm described earlier. We will use a local witness that will involve special constant-size trees, which we call *witness templates*. In a witness template based on a tree $\mathcal{W}$, we fix a small set of vertices U in G, and consider the subgraphs of G that are isomorphic to $\mathcal{W}$ whose set of leaves is exactly U. These witness templates are inspired by the templates for Densest k-subgraph [BCC+10]. In addition to satisfying conditions (3.1), (3.3) and (3.4), these witnesses will occur not just in random graphs, but in any supergraph G with sufficient average degree [3] Δ.

The template witness structure $\mathcal{W}_{r,s}$ parameterized by r, s will correspond to a tree with $(r+1)$ leaves of "fixed vertices", all of which can only be occupied by vertices in L. This structure will also have $(2s-r)$ internal vertices or "free vertices" in total, with s of the internal vertices to be occupied by vertices in R, and $(s-r)$ internal vertices to be occupied by vertices in L. Hence, in the notation of the previous section, $a_R = s$ and $a_L = s - r$ denote the number of internal vertices on the left and right respectively.

In our algorithm, we will fix the leaves to be a small set of vertices $u_0, u_1, \ldots, u_r \in L$ and consider all subgraphs $H \subseteq G$ that are isomorphic to $\mathcal{W}$ with the $(r+1)$ leaves being $U = (u_0, u_1, \ldots, u_r)$. Each such subgraph H is called an *occurence* of $\mathcal{W}$ supported on leaves U.

We now formally describe the structure of the witness, by first describing a caterpillar graph $W(r,s)$ with $(s+1)$ vertices in total. This caterpillar exactly corresponds to the caterpillar templates used in Densest k-subgraph [BCC+10]. Our final witness template $\mathcal{W}_{r,s}$ will have all the $(s+1)$ nodes in $W(r,s)$ (both leaves and internal nodes) corresponding to vertices from L, and we will just replace each of the s edges of $W(r,s)$ with a path of length 2, with its own distinct node on the right. Hence $\mathcal{W}_{r,s}$ will have $(r+1)$ leaves and $(s-r)$ internal vertices on the left, and s internal vertices on the right. The case of $r \geq s$ is a degenerate case, where $\mathcal{W}_{r,s}$ is just a path of length 2 with both leaves on the left, and one internal node on the right.

Definition 4.1. *[Witness Template $\mathcal{W}_{r,s}$ and caterpillar $W(r,s)$] For $r < s$, the caterpillar structure $W(r,s)$ is a tree constructed inductively. It has a backbone consisting of the $(s-r)$ free vertices, and $(r+1)$ leaves fixed vertices or leaves.*

[3] For instances $\mathcal{W}$ with triangles may not occur if G is bipartite, even if it satisfies various expansion properties.

1. *Begin with two vertices: one being a leaf, and another being a free vertex z_1 that is connected by an edge*[4]*. This is the first step.*
2. *For steps $i = 2, 3, \ldots, (s-1)$, do the following:*
   - *At step i, if the interval $[\frac{ir}{s}, \frac{(i+1)r}{s}]$ contains an integer, add a leaf to the (current) rightmost free vertex. This is called a* hair step.
   - *Otherwise, add another free vertex to the right, with an edge to the previous free vertex (increasing the backbone length by 1). This is called a* backbone step.

For $r \geq s$, then $\mathcal{W}_{r,s}$ is just a path of length 2 with both leaves on the left, and one internal node on the right.

Illustrations of a witness template and a caterpillar can be found in Figure 1.

Let $B = \{z_1, z_2, \ldots z_{s-r}\}$ denote the vertices on the backbone. The number of hair steps encountered in the first t steps is $\lfloor \frac{(t)r}{s} \rfloor$. Further, the final step in the construction adds the $(r+1)$th leaf of the structure.

The distinguishing algorithm. Let $d \leq k^\gamma$, $\Delta \geq n^{\alpha/2}$. We will pick integers r, s such that $\gamma + \varepsilon_1 \leq r/s \leq \alpha - \varepsilon_2$, where $\varepsilon_1, \varepsilon_2 \geq \varepsilon > 0$ are constants.

Algorithm Alg-Dist$_{r,s}$:

1. For every guess of $r+1$ vertices $u_0, u_1, u_2, \ldots, u_r \in L$ along with an assignment of labels $\sigma_0, \sigma_1, \ldots, \sigma_r$ to each of them
   (a) Given the *fixed vertices or leaves* of $\mathcal{W}_{r,s}$ being $U = (u_0, u_1, u_2, \ldots, u_r)$, check if there are at least $\log^{2s} n$ occurences of $\mathcal{W}_{r,s}$ in G, supported on U.
   (b) For each of the first $O(\log n)$ occurences $H \subseteq G$ of $\mathcal{W}_{r,s}$ supported on U, check if there is a satisfying assignment for the vertices of H that is (individually) consistent with the assignment $\sigma_0, \sigma_1, \ldots, \sigma_r$ for U.
   (c) If TRUE (i.e. there are such consistent occurences), output $\mathcal{D}_{YES}$ and return.
2. Output $\mathcal{D}_{NO}$.

We note that we can check for every occurence H of $\mathcal{W}_{r,s}$ if there is a satisfying assignment consistent with $\sigma_0, \ldots, \sigma_r$ in time $k^r \cdot \text{poly}(k, n)$ i.e., polynomial time. Hence, the running time of the algorithm is at most $N^{O(s+r)}$.

[4]There is a small difference c.f. [BCC+10] where the initial vertex is a backbone vertex – this is more of a notational thing.

The analysis of the distinguishing algorithm is straightforward from the following proposition which shows that the distinguishing algorithm works if the log-density condition is satisfied.

Proposition 4.1. *Consider the problem of distinguishing between projection games drawn from $\mathcal{D}_{YES}$ vs $\mathcal{D}_{NO}$ with parameters $\Delta = n^{\alpha/2}$, $d = k^\gamma$. If r, s are integers satisfying $\gamma + \varepsilon_1 \leq r/s \leq \alpha - \varepsilon_2$ for some small constants $\varepsilon_1, \varepsilon_2 > 0$, then* Alg-Dist$_{r,s}$ *will run in time $n^{O(r+s)}$ and distinguish between $\mathcal{D}_{YES}$ and $\mathcal{D}_{NO}$ with high probability.*

We only sketch the proof here, since this is subsumed by the algorithm in Section 5.

Proof Sketch. The proof consists of three main steps:

1. In any random graph G of average degree $\Delta > n^{\frac{r}{2s}+\varepsilon_2}$ for some $\varepsilon_2 > 0$, w.h.p. for any fixing U of the $r+1$ leaves, the number of occurrences of $\mathcal{W}_{r,s}$ in G with mutually disjoint internal vertices is at least $n^{\varepsilon_2 s}$. This can be shown using an inductive argument similar to Densest k-subgraph [BCC+10, Vij12]. Getting high probability bounds is not completely straightforward : it is done by a careful induction that follows the construction of the template, and along with a trick called color coding to decouple some dependencies.
2. Given an instance from $\mathcal{D}_{YES}$, it is satisfiable (let us call this assignment ϕ^*). Hence, for every fixing of leaves U in G, the assignment ϕ^* satisfies every occurence H supported on U, as long as the fixed assignment to the leaves $\sigma_0, \sigma_1, \ldots, \sigma_r$ is consistent with ϕ^*.
3. For an instance drawn from $\mathcal{D}_{NO}$ i.e., a random projection game with $d \leq k^{r/s-\varepsilon_1}$ let $H \subseteq G$ be an occurence of $\mathcal{W}_{r,s}$ supported on U. For a fixed assignment to the leaves $\sigma_0, \sigma_1, \ldots, \sigma_r$, the probability that there is an assignment to the internal vertices that satisfies H is at most $k^{-\varepsilon_1 r}$.
4. Since each witness template $\mathcal{W}_{r,s}$ has many occurences in G, for a fixed assignment $\sigma_0, \sigma_1, \ldots, \sigma_r$ to U, the probability that there is an extension of this assignment that satisfies at least $\Omega(\log n)$ many occurences is at most $k^{-\varepsilon_1 r \log n}$. By taking a union bound over all the k^{r+1} assignments to the leaves U we get the required result.

□

The above algorithms require that $r < s$: this corresponds to the regime when $\Delta \leq n^{1/2}$. On the other hand, if $\Delta > n^{1/2} \log N$, then it is easy to see that w.h.p.

from every vertex $u \in L$, there are $O(\log N)$ paths of length 2 from u to every other vertex v in L. This will allow us to use similar arguments to show that for every fixing of a label for u, for every vertex $v \in L$, there is no label that satisfies $O(\log N)$ of the paths from u to v.

4.2 Planted Random model The distinguishing algorithm can refute the existence of a satisfying assignment for $\mathcal{D}_{NO}$, but it does not find a satisfying assignment for instances from $\mathcal{D}_{YES}$. We now consider the corresponding search version in planted random instances.

DEFINITION 4.2. (PLANTED RANDOM MODEL $\mathcal{D}_{\text{PL}}$) *The distribution of instances $\mathcal{D}_{PL}$ is parameterized by a planted assignment ϕ^*, and parameters n, Δ, k, d. A projection game drawn from this model $\mathcal{D}_{PL}$ has a supergraph $G(L, R, E)$ that is drawn from an Erdős-Rényi bipartite random graph with parameters $\mathcal{G}(n/2, n/2, p = \Delta/n)$. For each edge $e \in E$, the projection π_e is chosen from a random right-regular bipartite graph of degree d that is consistent with ϕ^*.*

The instance has a satisfying assignment ϕ^*, but otherwise the projections are *random*. Here the algorithmic goal is to maximize the number of constraints satisfied. In this section we will design an algorithm ALG-PLANTED that given an instance drawn from $\mathcal{D}_{\text{PL}}$ finds an assignment satisfying at least $N^{-(3-2\sqrt{2}+\varepsilon)}$ fraction of all constraints, for any constant $\varepsilon > 0$.

The algorithm proceeds in iterations: there may be $O(1/\varepsilon)$ of them. In each iteration, we will run two steps, one of which will succeed:

Simple Algorithms. Run two simple algorithms as described in Section 3 in order to satisfy at least $\max\{\frac{d}{k}, \frac{1}{\Delta}\}$ fraction of the edges (see Lemma 5.3). This approach will be used when we do *not* have a significant log density gap.

Label Reduction. We will reduce the label set size (k initially) for each vertex in L by a factor of $N^{\Omega(\varepsilon)}$, while guaranteeing that the label-reduced instance remains satisfiable. We will then recurse into this instance.

In fact, the Label Reduction step will produce a polynomial number (say q of them) of "left label reductions" i.e., subinstances in which the label sets $\Sigma'_i : L \to \mathcal{P}([k])$ for nodes in L have been reduced by a factor of at least N^{ε} for a constant $\varepsilon > 0$, with the guarantee that at least one of these subinstances is still satisfiable (in general, it will be $\tilde{\Omega}(1)$-satisfiable). This label reduction step gives a new notion of progress which does not correspond to any aspect of similar algorithms for Densest k-Subgraph which inspired our algorithm.

If $\Delta > \sqrt{n}$, then a simple algorithm based on considering paths of length 2 can be used to get either a good approximation of $N^{-O(\varepsilon)}$ or we can get the required label reduction (see Lemma 5.4). The main component of our algorithm performs well when there is a significant log density gap i.e. if $\Delta \geq n^{\frac{\alpha}{2}} N^{\varepsilon}$ and $d \leq k^{\alpha}/N^{\varepsilon}$ for some $\varepsilon > 0$. In this case, our algorithm will reduce the label sets in L by a factor of $N^{\Omega(\varepsilon)}$. It may seem strange that this can always be achieved, but note that for a significant log density gap (as required below) we must have $k > N^{\varepsilon}$, so the label reduction will not continue indefinitely (one of the two simple algorithms may work at this point). In fact, it may not occur at all, if our instance does not have a log density gap to begin with. The guarantee of the main component of our algorithm is as follows:

LEMMA 4.1. *In a Planted Random Label Cover instance as above, if for some $\alpha > 0$ and $\varepsilon > 0$ we have $d \leq k^{\alpha}/N^{\varepsilon}$ and $\Delta^2 \geq n^{\alpha} N^{\varepsilon}$, then we can find $k^{O(1/\varepsilon)}$ left label reductions $\{\Sigma'_i : L \to \mathcal{P}([k]) \mid i \in [q]\}$ with alphabet size $|\Sigma'_i(u)| \leq k/N^{\varepsilon/2}$ $\forall i \in [q]$ and $u \in L$, such that for some $i \in [q]$ the instance defined by G and Σ'_i is satisfiable.*

Before we describe the Label Reduction algorithm and analysis, we will first see why this implies the stated approximation guarantee.

Approximation Ratio. Suppose $n = N^{\beta}$. Our goal is to obtain an $N^{\frac{\beta(1-\beta)}{2-\beta}+O(\varepsilon)}$ approximation. We can assume that $\Delta \geq n^{\delta/2+\varepsilon}$ and $d \leq k^{\delta-\varepsilon}$: otherwise we get a $\min\{\Delta, k/d\} = N^{\frac{\beta(1-\beta)}{2-\beta}+O(\varepsilon)} \leq N^{3-2\sqrt{2}+O(\varepsilon)}$-approximation.

Otherwise, we have a log-density gap. Hence, the label-reduction procedure can be used to bring down the label size by N^{ε} and we recurse. This recursion bottoms out when we have at most $\tilde{O}(1)$ labels per left vertex, or we do not have a log-density gap, in which case the simple algorithms get a $N^{\frac{\beta(1-\beta)}{2-\beta}+O(\varepsilon)}$ approximation. The final labeling that the algorithm outputs is the best of all the labelings output by the two simple algorithms, and the labeling output after the end of label reduction. As we shall see, a single iteration of label-reduction will involve choosing $O(1/\varepsilon)$ leaf vertices and guessing the correct labels for them. Considering the $O(1/\varepsilon)$ iterations, we will need to choose $O(1/\varepsilon^2)$ leaf vertices and guess their labels — hence, the total run time is $k^{O(1/\varepsilon^2)} N^{O(1)}$ i.e.,

polynomial time for any constant $\varepsilon > 0$.

Label Reduction from Log-density Gap For any $\alpha \in (0,1)$ as in Lemma 4.1, we can choose natural numbers $r, s > 0$ so that $|\alpha - \frac{r}{s}| < \varepsilon/2$ and $s \leq \lceil 1/\varepsilon \rceil$. In the algorithm, we will guess the assignment for a constant number of vertices on the left bipartition L of the supergraph G, and try to infer (or narrow down) the potential labels for the rest of the vertices on the left. The algorithm consists of a family of algorithms ALG-PLANTED$_{r,s}$ given by the witness templates $\mathcal{W}_{r,s}$ introduced earlier, parameterized by two integers r, s.

A single iteration of algorithm ALG-PLANTED$_{r,s}$ corresponding to structure $\mathcal{W}_{r,s}$ will fix all but the last leaf of $\mathcal{W}_{r,s}$ to be a set of vertices $u_0, u_1, \ldots, u_{r-1}$ (on the left) and then consider the candidates for the "free vertices" of $\mathcal{W}_{r,s}$ and possible labels for each of these vertices. In fact, we will focus on the $(s-r)$ free vertices from the left and their labels. Specifically, at every step of the iteration, we will maintain some set of nodes S, and some set of *feasible labels* for each node in the set. Initially, this set will simply be the first leaf which we fix to be $S = \{u_0\}$, with a label which we will guess (in the final algorithm, we will try all possible labelings for the constant number of fixed nodes). The rest of the iteration proceeds by following along the inductive construction of $\mathcal{W}_{r,s}$ as defined in Definition 4.1:

Backbone Step. If the current step is a "backbone step", or the last step in the construction, then we let our new set be $\Gamma(\Gamma(S))$, and we restrict the label sets as follows. For every vertex $v \in \Gamma(S)$, restrict its label set only to labels that are consistent with some label in all of v's neighbors in S. Similarly, for every vertex $u \in \Gamma(\Gamma(S))$, restrict its label set only to labels that are consistent with some label for each neighbor of u in $\Gamma(S)$. Please refer to Figure 1 for an illustration of a backbone step.

Note that the expansion of G (a random graph) guarantees that the new set will have cardinality $|\Gamma(\Gamma(S))| = \min\{|L|, \Omega(\Delta^2|S|)\}$. Moreover, since these the constraints are d-to-1 projections, the label set sizes in the new set will be at most a d-factor larger than the label sets we consider for nodes in S, and will still contain the correct labels if this was the case for S. In fact, with some minor modifications, a similar approach will also work in the semi-random model, since these bounds do not require the projections to be random. Thus, the effect of a backbone step can be summarized as follows:

CLAIM 4.1. *In the random planted model, performing a backbone step on a set $S \subseteq L$ with label sets of size at most k' yields a set $S' \subseteq L$ such that $|S| = \min\{n, \Omega(|S|\Delta^2)\}$ with label sets of size at most $k'd$. If the label sets of S contain the planted labeling, then so do the label sets of S'.*

Hair Step. If the current step in the construction is a "hair step" (except for the last step), then we guess a label for the next leaf u_i, and let our new set be the intersection[5] $S \cap \Gamma_2(u_i)$. We also restrict the set of possible labels for each of these nodes to be labels that match any labels of neighbors in $\Gamma(u_i)$ induced by the labeling of u_i.

Again, since G is a random graph, we are guarantee (w.h.p.), that the new set will have cardinality $|S \cap \Gamma_2(u_i)| = \Omega(|S|\Delta^2/n)$. Moreover, the randomness of the projections guarantees that the label sets for nodes in the new set will be a factor d/k smaller. This cannot be guaranteed in the semi-random model, and will need to be handled differently. In the random planted setting, the effect of a hair step can be summarized as follows:

CLAIM 4.2. *In the random planted model, performing a hair step on a set $S \subseteq L$ of size $\tilde{\omega}(n/\Delta^2)$ with label sets of size at most k' yields a set $S' \subseteq L$ such that $|S| = \Theta(|S|\Delta^2/n)$ with label sets of size at most $\max\{k'd/k, \tilde{O}(1)\}$. If we correctly guess the planted label for u_i and the label sets of S contain the planted labeling, then so do the label sets of S'.*

The purpose of a single iteration is to narrow down the label sets of all vertices in L. Once this is done, we can fix a new set of leaves and labels and repeat the process until every vertex on the left has $\tilde{O}(1)$ labels (assuming all our guesses for the labels of fixed vertices were correct, otherwise the label sets will be empty), or we no longer have a log density gap (in which case the simple algorithms apply). We will show that for the right choice of parameters, we only need a constant number of iterations (and thus a constant number of leaves for which we have to guess all possible labelings) until we have discovered the right labels. The analysis of a single iteration is given in the following proposition:

PROPOSITION 4.2. *Given an instance of the planted random model with parameters $\Delta \geq n^{\frac{r}{2s}+\varepsilon}$, with the projections having $d \leq k^{\frac{r}{s}-\varepsilon}$, a single iteration of Algorithm ALG-PLANTED$_{r,s}$ which guesses the correct labels for the fixed vertices returns label sets for all*

[5] Note that for any $S \subseteq L$, $\Gamma_2(S) \subseteq L$ is the set of all nodes in L that have a path of length 2 to S.

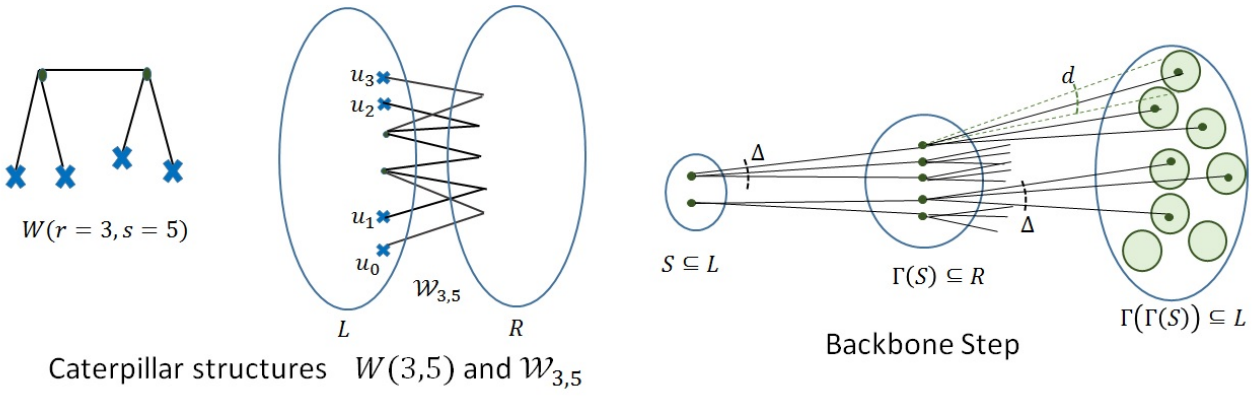

Figure 1: The figure on the left shows the caterpillar structures $W(r,s)$ and $\mathcal{W}_{r,s}$ for $r=3, s=5$. The structure has 4 leaves, and 7 internal vertices in total. The figure on the right shows a backbone step. S is the set of candidate vertices at the start of the step, and the solid circles denote the set of feasible labels of each vertex in S.

vertices in L of size $k^{1-\varepsilon}$ which contain the right labels for those vertices.

Proof. Consider the first leaf (from the left). Suppose we fix the leaves to be vertices $u_0, u_1, \ldots, u_{r-1}$ (picked arbitrarily) from the left partition of the supergraph G. By step $t \in [s]$ in the inductive construction of caterpillar $W(r,s)$, $h_t = \lfloor \frac{tr}{s} \rfloor$ of them are leaf steps and the rest $b_t = t - h_t$ steps are backbone steps. Let $M(t)$ denote the number of candidates of the set S after t steps, and $K(t)$ represented the largest label set size after t steps. Then the following claim follows by induction on t using Claim 4.2 and Claim 4.1 for the corresponding steps.

Claim 4.3. *Conditioned on the correct assignment to the first leaf (u_0), the number of candidates $M(t)$ for the rightmost backbone vertex after t steps of the caterpillar $C_{a_L,b}$ satisfies w.h.p.*
$M(t) \geq (n^{r/s})^{b_t}(n^{-(s-r)/s})^{h_t}\log^t N$, *and the average number of valid labels for each of these vertices is* $K(t) \leq (k^{r/s})^{b_t}(k^{-(s-r)/s})^{h_t}\log^t N$.

It is fairly straightforward to check the bounds $M(t)$ and $K(t)$ hold in expectation. For the exact bounds, we will get w.h.p. estimates (up to a loss of $polylog(N)$ factors) since $M(t), K(t) \in [N^{\varepsilon}, N^{1-\varepsilon}]$ for some small constant $\varepsilon > 0$ (this follows from the properties of the structure). We do not prove the claim here.

Consider the final step, i.e., for $t = s$. After step $t-1$, the number of candidates for the rightmost backbone vertex is at least $(n^{r/s})^{s-r}(n^{-(s-r)/s})^{r-1} = n^{1-r/s}$ (by the choice of parameters). Similarly, $K(t-1) \leq (k^{r/s})^{s-r}(k^{-r/s})^{r-1} = k^{1-r/s}$.

To find a valid labeling, we do a backbone step at the last step $t = s$ (instead of the hair step in the distinguishing algorithm). Since w.h.p. the graph on L (induced by paths of length 2) has expansion $\Delta^2 \geq n^{r/s+\varepsilon}$, and in the second-to-last set we have more than n/Δ^2 candidates, this will reach all vertices in L. On the other hand, the number of valid labels for these vertices (in our algorithm) is at most $d \cdot K(t-1) \leq k^{1-\varepsilon}$. Furthermore, it is easy to see that if all guesses of labels were correct, than these label sets still include the (correct, satisfiable) planted assignment. □

5 Approximating Label-Cover in the Semi-random model

5.1 Expansion Properties and Random graphs

Our algorithm in the semi-random model is designed for instances in which the supergraph is a random bipartite graph. However, the algorithm prunes the graph in non-random ways in several iterations, so that we need to prove correctness not only for random graphs, but more generally for graphs which retain some of the expansion properties of random graphs. It turns out that for our analysis, simple edge-expansion or vertex-expansion is not enough, since these properties will not survive the pruning.

Definition 5.1. *We call a bipartite graph on vertices $L \times R$ such that $|L| = n_L$ and $|R| = n_R$ a* strong (Δ_L, Δ_R)-expander *if the following conditions hold:*

- $\Delta_L n_L = \Delta_R n_R$. *Let us denote* $p = \Delta_R/n_L = \Delta_L/n_R$, *and* $n = n_L + n_R$.
- *Every $u \in L$ has degree $\Delta_L \geq \deg(u) = \tilde{\Omega}(\Delta_L)$.*
- *Every $u \in R$ has degree $\Delta_R \geq \deg(v) = \tilde{\Omega}(\Delta_R)$.*
- *For every vertex set $S \subseteq L$, the vertices $\{v \in R \mid |\Gamma(v) \cap S| \geq \max\{2p|S|, \log n\}\}$ contribute at most $\tilde{O}(|S|)$ edges to S.*

- *For every vertex set $T \subseteq R$, the vertices $\{u \in L \mid |\Gamma(v) \cap T| \geq \max\{2p|T|, \log n\}\}$ contribute at most $\tilde{O}(|T|)$ edges to T.*

Note that a random bipartite graph with edge probability $(1 - o(1))p$ is a strong (pn_R, pn_L)-expander w.h.p. We may also consider a somewhat weaker property:

DEFINITION 5.2. *We call a bipartite graph on vertices $L \times R$ such that $|L| = n_L$ and $|R| = n_R$ a* weak (Δ_L, Δ_R)-expander *if the following conditions hold:*

- $\Delta_L n_L = \Delta_R n_R$. *Let us denote $p = \Delta_R/n_L = \Delta_L/n_R$, and $n = n_L + n_R$.*
- *Every $u \in L$ has degree $\Delta_L \geq \deg(u) = \tilde{\Omega}(\Delta_L)$.*
- *Every $u \in R$ has degree $\Delta_R \geq \deg(v) = \tilde{\Omega}(\Delta_R)$.*
- *For every vertex set $S \subseteq L$, the vertices $\{v \in R \mid |\Gamma(v) \cap S| \geq \max\{2p|S|, \log n\}\}$ contribute at most $\tilde{o}(|S|\Delta_L)$ edges to S.*
- *For every vertex set $T \subseteq R$, the vertices $\{u \in L \mid |\Gamma(v) \cap T| \geq \max\{2p|T|, \log n\}\}$ contribute at most $\tilde{o}(|T|\Delta_R)$ edges to T.*

Note that both strong (Δ_L, Δ_R) expansion and weak (Δ_L, Δ_R) expansion are hereditary properties in the following sense. Any subgraph of a strong/weak (Δ_L, Δ_R)-expander $G = (L, R, E)$ which contains at least $\tilde{\Omega}(|E|)$ edges and is nearly-biregular is also a strong/weak (Δ_L, Δ_R)-expander, since the last two properties are clearly hereditary.

Finally, note that a (weak) (Δ_L, Δ_R)-expander is, in particular, a good vertex expander. Indeed, for any vertex set $S \subseteq L$, for example (the argument for subsets of R is identical), by the degree bounds, there are $\tilde{\Omega}(\Delta_L|S|)$ edges incident to vertices in S, and most of them have endpoints in R which have at most $O(\max\{p|S|, \log n\})$ neighbors in S (by the second-to-last property). Thus, the number of such vertices that S expands to as at least $\tilde{\Omega}(\Delta_L|S|/(\max\{p|S|, 1\})) = \tilde{\Omega}(\min\{n_R, \Delta_L|S|\})$ vertices.

5.2 From planted random to semi-random models The algorithms for the *semi-random model* are directly inspired by the *planted random* model; however, there are significant technical issues that arise because of the adversarial nature of the projections. For simplicity, we will assume the same regularity properties as in the previous section. The preprocessing in Section 5.3 ensures that by bucketing and pruning, these conditions are approximately true up to polylogarithmic factors. However, this is at the loss of full satisfiability (or perfect completeness): in the remaining instance, we have an assignment that satisfies a $1/\text{poly}(\log N)$ fraction of the edges. This introduces some technical complications, which are handled by modifying each step of the algorithm (and analysis) appropriately.

The primary challenge is in each of the *hair steps* in the caterpillar based procedure. The aim in a *hair step* is to reduce the label size of a current working subset $S \subseteq L$ of left vertices. This is achieved by intersecting S with the two-step neighborhood of a carefully chosen vertex $v \in L$. If we fix the correct label for v, the label set of these vertices $S \cap \Gamma_2(v)$ will be significantly smaller when the projections are random. However, this need not happen when the projections are arbitrary. To handle this, we show that if for every leaf $v \in L$, the correct label for v does not give the required label size reduction, then these labels stand out. We show that there are significantly fewer than k labels for every vertex $v \in L$ that satisfy these properties. Hence, if the *hair step* doesn't work this can be used to obtain obtain a significant label size reduction for every vertex $v \in L$. Since we cannot know which kind of label is more prevalent among the correct labels for nodes in L, our algorithm will fork by essentially trying both possibilities.

As in Section 4.2, the algorithm proceeds in $O(1/\varepsilon)$ iterations. In each iteration, we will run two steps, one of which will succeed: *Simple algorithms* or *Label Reduction*. The iteration proceeds by following the inductive construction of $\mathcal{W}_{r,s}$, as in the algorithm for the Planted Random model (section 4.2). The backbone step proceeds in a similar identical fashion to section 4.2, though we also need to handle the instance not being fully satisfiable. The hair step differs significantly from section 4.2.

Hair Step. If the current step in the construction is a "hair step", let k' be the current bound on the size of label sets in S. We will try two approaches. Fix the next leaf $u_i \in L$ and a label σ_{u_i} for u_i (the algorithm will try performing the hair step for all possible choices for u_i and σ_{u_i}). Let $S' = S \cap \Gamma(\Gamma(u_i))$. For every vertex in $u \in S'$, restrict its current label set to labels that match the label of at least one neighbor in $\Gamma(u) \cap \Gamma(u_i)$ induced by the labeling of u_i. That is, if the current label set for u is $\Sigma(u)$, let

$$\Sigma'(u) = \Sigma(u) \cap \bigcup_{v \in \Gamma(u_i) \cap \Gamma(u)} \pi^{-1}_{(u,v)}(\pi_{(u_i,v)}(\sigma_{u_i})).$$

In the planted random case, the label sets for S' will contract to size $k'' \leq k'/k^{1-r/s}$ as required.

However, in the semirandom case, the projections are no long random, and this contraction of label sets may not occur. In this case, we do the following. For every vertex

$v \in \Gamma(S)$, consider every possible label σ_v of v. If for every neighbor $u \in \Gamma(v) \cap S$ we have $|\pi^{-1}_{(u,v)}(\sigma_v)| > k''$, then call σ_v a bad label for v (in this step). For every $u \in L$, call a label σ_u bad for u if for every neighbor $v \in \Gamma(u) \cap \Gamma(S)$ the matching label $\pi_{(u,v)}(\sigma_u)$ is bad for v. Add to the current list of label reductions the reduction which assigns to every left node $u \in L$ only its bad labels. These will sometimes informally be called *abort steps*.

Formally, our algorithm proceeds by a sequence of backbone and hair steps in the order corresponding to $\mathcal{W}_{r,s}$, enumerating over every possible choice of leaves for $\mathcal{W}_{r,s}$ and labels for those leaves (this will include choices consistent with ϕ^*). When these latter choices behave well (like in the random planted model), the final step is guaranteed to give reduced label sets, such that they retain the correct labels, and thus all such label reductions are added to the list (for every choice in our enumeration). However, when a leaf node in G behaves badly (its labeling according to ϕ^* does not bring about the desired label reduction), that actually allows us to identify a reduced label set for this leaf, since, as we will show, few labels can have the combinatorial properties associated with a bad label. Thus, at every hair step we also add to our list of label reductions the label reduction corresponding to possible bad labels.

Our main algorithmic tool of Label Reduction is described in the following lemma:

LEMMA 5.1. *For every constant $\varepsilon > 0$, there is a polynomial time algorithm which, given an $\tilde{\Omega}(1)$-satisfiable projection games on a weakly (Δ_L, Δ_R)-expanding supergraph $G = (L, R, E)$ with alphabet size k, returns one of the following:*

- *A labeling which satisfies at least a $1/N^{\beta(1-\beta)/(2-\beta)+\varepsilon}$ fraction of constraints, where $N = k|L|$ and $|L| = N^\beta$.*
- *A polynomial number of left label sets $\{\Sigma_i : L \to \mathcal{P}([k]) \mid i \in [t]\}$ such that for every index $i \in [t]$ and left node $u \in L$, we have $|\Sigma_i(u)| \leq k/N^{\varepsilon/2}$, and for at least one index $i \in [t]$, the subinstance defined by label sets Σ is $\tilde{\Omega}(1)$-satisfiable.*

5.3 Pruning highly satisfiable projection games

As in the previous section, we only consider nearly-biregular bipartite graphs. That is, bipartite graphs $G = (L, R, E)$ such that $|L| = n_L$, $|R| = n_R$, every vertex $u \in L$ has degree $\Delta_L \geq \deg(u) = \tilde{\Omega}(\Delta_L)$, and every vertex $v \in R$ has degree $\Delta_R \geq \deg(v) = \tilde{\Omega}(\Delta_R)$, for some Δ_L, Δ_R such that $n_L\Delta_L = n_R\Delta_R$. We show that any projection game which is at least $\tilde{\Omega}(1)$-satisfiable can be decomposed into a logarithmic number of subinstances with additional regularity properties which will help in the design and analysis of our algorithm.

LEMMA 5.2. *Let $\mathcal{G} = (L, R, E, [k], [k], \{\pi_e\}_{e\in E})$ be a projection game with supergraph $G = (L, R, E)$, alphabet size k, and projection $\{\pi_e\}_{e\in E}$, such that G is nearly-biregular, and such that there exists a labeling which simultaneously satisfies at least $\tilde{\Omega}(|E|)$ edges. Then we can find a set of at most $\log k$ subinstances $\{\pi'_e\}_{e\in E'}$ on $\mathcal{G}$ such that at least one has the following properties:*

- *The projections are d-to-1 for some $d > 0$: For every superedge $(u, v) \in E$, every label σ in the image of $\pi'_{(u,v)}$ has a preimage of size $|\pi'^{-1}_{(u,v)}(\sigma)| \leq d$.*
- *For the same d the label set of every right vertex $v \in R$ has size $|\bigcup_{u\in\Gamma(v)} \pi'_{(u,v)}([k])| = \tilde{O}(k/d)$.*[6]
- *There is a nearly-biregular subgraph of G with $\tilde{\Omega}(|E|)$ edges which is fully satisfiable (satisfying projections $\{\pi'_e\}_{e\in E}$).*

Proof. For every $j \in [\log k]$, construct a subinstance as follows: For every edge $(u, v) \in E$, consider only labels σ for v such $|\pi^{-1}_{(u,v)}(\sigma)| \in [2^{j-1}, 2^j]$, and only the preimages of these labels for u. Note that different edges incident in u (or v) can now associate a different label set with that node. There are $\log k$ such subinstances, therefore, at least one of them retains a correct labeling for a $1/\log k$ fraction of edges which were originally simultaneously satisfiable. For the index j corresponding to such a subinstance, let $d = 2^j$.

Since in this subinstance, we can still satisfy $\tilde{\Omega}(|E|)$ edges, it is easy to see that there exists a nearly-biregular subgraph H of G which is fully satisfiable: Indeed, as a thought-experiment, consider a fixed labeling which satisfies $m = \tilde{\Omega}(|E|)$ edges. Then for some sufficiently large constant $C > 0$, suppose we removed all vertices $u \in L$ with at most $\Delta_L/\log^C(n)$ satisfied incident edges, and vertices $v \in R$ with at most $\Delta_R/\log^C(n)$ satisfied incident edges. Then the total number of edges removed is at most $o(m)$, and all remaining vertices in L, R have satisfiable-degree $\tilde{\Theta}(\Delta_L), \tilde{\Theta}(\Delta_R)$, respectively. Removing unsatisfied edges would give the desired subgraph.

Note that in the subgraph H above, for every right vertex, the correct label participates in at least $\Delta_R/\log^C(n)$ satisfied edges. Therefore, we can further prune our instance by discarding, for every vertex $v \in R$ (in the original graph G), all labels that currently participate

[6] We think of the projections as being defined by a label-extended graph, so for certain edges, some labels on the left may not necessarily have a corresponding label on the right.

in fewer than $\Delta_R/\log^C(n)$ edges. This does not reduce the number of satisfied edges in H.

Finally, let $\{\pi'_e\}_{e\in E}$ be the projections in the current subinstance. Note that for any edge $(u,v) \in E$, the image $\Sigma^R_{(u,v)} := \pi'_{(u,v)}([k])$ has cardinality at most $2k/d$. This is because every label $\sigma \in \Sigma^R_{(u,v)}$ has a preimage $\pi'^{-1}_{(u,v)}(\sigma)$ of cardinality at least $d/2$, and these preimages are disjoint (since $\pi'_{(u,v)}$ is a projection). Moreover, for any vertex $v \in R$, every remaining label of v participates in at least a $1/\log^C(n)$ fraction of these images $\Sigma^R_{(u,v)}$, therefore, the union of all these label sets has size at most $\frac{\log^C(n)}{\deg(v)} \sum_{u\in\Gamma(v)} |\Sigma^R_{(u,v)}| = \tilde{O}(k/d)$. □

5.4 Labeling and label reduction As opposed to previous algorithms, our algorithm may not immediately choose a labeling. Rather, we produce a polynomial number of subinstances in which the label sets nodes in L have been reduced by a factor of at least N^ε for a constant $\varepsilon > 0$, with the guarantee that at least one of these subinstances is still $\tilde{\Omega}(1)$-satisfiable even when restricted to the label-reduced nodes on the left. This is a novel feature which does not correspond to any aspect of similar algorithms for Densest k-Subgraph which inspired our algorithm.

Lemma 5.1 immediately gives our approximation guarantee for Projection Games on weak (Δ_L, Δ_R)-expanders, and in particular for semi-random instances.

THEOREM 5.1. *For every $\varepsilon > 0$, there is a polynomial time $N^{\beta(1-\beta)/(2-\beta)+\varepsilon}$-approximation algorithm for an $\tilde{\Omega}(1)$-satisfiable projection games on weakly (Δ_L, Δ_R)-expanding supergraphs $G = (L, R, E)$ with alphabet size k, where $N = |L|k$ and $|L| = N^\beta$.*

Proof. Given a projection game on a graph $G_0 = (L_0, R_0, E_0)$ as above with alphabet size k_0, and a constant $\varepsilon_0 > 0$, we let $N_0 = |L_0|k_0$, and run the algorithm from Lemma 5.1 recursively. That is, we run the algorithm on the original instance. If we get a good labeling, we output this labeling. Otherwise, we recurse on all the subinstances induced defined by label set Σ_i for $i = [t]$. Formally, the algorithm is as follows:

- Given a projection game on graph $G = (L, R, E)$ and alphabet size k, let $N = |L|k$, and $\varepsilon \geq \varepsilon_0$ such that $N^\varepsilon = {N_0}^{\varepsilon_0}$. Run the algorithm from Lemma 5.1 with parameter ε.
- If the algorithm returns a labeling, return this labeling.
- Otherwise, for each Σ_i, run the algorithm recursively on the subinstance defined by these label sets.
- If for at least one Σ_i, the algorithm finds a good labeling, return this labeling.

The correctness of the algorithm can be shown for all $\tilde{\Omega}(1)$ instances, by induction on k. If $k < N^{\varepsilon/2}$, then by Lemma 5.1 we must get a good labeling, since the label sets in L cannot be reduced to size at most $k/N^{\varepsilon/2}$.

Otherwise, we may still get a good labeling, in which case we are done. Otherwise, suppose we get a set of reduced label sets $\{\Sigma_i \mid i \in [t]\}$. The lemma guarantees that for at least one i, the subinstance defined by label sets Σ_i is $\tilde{\Omega}(1)$ satisfiable. By our induction hypothesis, running the algorithm recursively on this instance will yield a labeling. Let us analyze the fraction of edges satisfied by this labeling. The new label set size is $k' = k/N^{\varepsilon/2}$, giving us total size $N' = k'|L| = N^{1-\varepsilon/2}$, and therefore $|L| = (N')^{\beta/(1-\varepsilon/2)}$. Let $\beta' = \beta/(1-\varepsilon/2)$ and $\varepsilon' = \varepsilon/(1-\varepsilon/2)$ (so $(N')^{\varepsilon'} = N^\varepsilon = N_0^{\varepsilon_0}$). Then by the induction hypothesis, the fraction of edges satisfied by the labeling (up to a polylogarithmic factor) will be

$$(N')^{-\frac{\beta'(1-\beta')}{(2-\beta')}-\varepsilon'} = N^{-\frac{\beta(1-\beta')}{(2-\beta')}-\varepsilon} > N^{-\frac{\beta(1-\beta)}{(2-\beta)}},$$

where the inequality follows since $\beta' > \beta$.

Note that the recursion depth can be at most $2/\varepsilon$, so the number of subinstances produced is polynomial, since the fanout at every level of the recursion is polynomial, and the recursion depth is constant. □

5.5 Simple algorithms and parameterization The algorithm which proves Lemma 5.1 starts by applying the pruning procedure of Lemma 5.2. Thus, throughout this section and the next, we may assume that our instance is already of the form guaranteed by the lemma (and by the hereditary properties of weakly (Δ_L, Δ_R)-expanding graphs):

- The supergraph $G = (L, R, E)$ is weakly (Δ_L, Δ_R)-expanding, and nodes in L have alphabet size k.
- The projections are d-to-1 for some $d > 0$: For every superedge $(u, v) \in E$, every label σ in the image of $\pi_{(u,v)}$ has a preimage of size $|\pi^{-1}_{(u,v)}(\sigma)| \leq d$.
- For the same d the label set of every right vertex $v \in R$ has size $|\bigcup_{u\in\Gamma(v)} \pi_{(u,v)}([k])| = \tilde{O}(k/d)$.
- There is a subgraph H of G with $\tilde{\Omega}(|E|)$ edges which is also weakly (Δ_L, Δ_R)-expanding, and fully satisfiable (satisfying projections $\{\pi_e\}_{e\in E}$).

We formally state and give the guarantees for the simple algorithms that are described at the end of section 3.

LEMMA 5.3. *In a Label Cover instance as above, we can find a labeling which satisfies at least a* $1/\tilde{O}(\min\{\Delta_L, \Delta_R, k/d\})$*-fraction of constraints.*

Proof. To see that we can always satisfy an $\tilde{\Omega}(d/k)$-fraction of constraints, consider the following algorithm:

- For every right node, choose a label independent uniformly at random.
- For each $u \in L$, choose a label which maximizes the number of satisfied constraints involving u.

For every right node in H, there is an $\tilde{O}(d/k)$ probability that we will choose the correct label. Since H contains $\tilde{\Omega}(|R|)$ right nodes, in expectation an $\tilde{\Omega}(d/k)$ fraction of right nodes will be correctly labeled, and thus, by near-regularity, the greedy labeling on the left will satisfy at least an $\tilde{\Omega}(d/k)$-fraction of all constraints.

To see that we can satisfy a $1/\tilde{O}(\min\{\Delta_L, \Delta_R\})$-fraction of constraints, first consider the case when $|L| \geq |R|$. Note that in this case we have $\Delta_L = \tilde{O}(\Delta_R)$, so we only need to show that there is an $\tilde{\Omega}(1/\Delta_L)$-approximation. Indeed, consider the following algorithm:

- For every $u \in L$ choose a neighbour $v \in R$ of u uniformly at random, and let F be the set of edges chosen.
- For every $v \in R$, consider its neighbours $\Gamma^F(v)$ in F, and choose a label σ_v for v which maximizes the number of neighbours $u \in \Gamma^F(v)$ such that the preimage $\pi^{-1}_{(u,v)}(\sigma_v)$ is non-empty. For every $u \in L$ choose a label which satisfies $\pi_{(u,v)}$ if one is available.

It is easy to check that every right node in H will have at least $\tilde{\Omega}(|L|/|R|)$ F-neighbours in H with probability $\tilde{\Omega}(1)$. Thus, the greedy labeling will satisfy in expectation at least $\tilde{\Omega}(|L|)$ edges out of $|E| = \tilde{O}(|L|\Delta_L)$, as required. The case when $|L| < |R|$ is symmetric. □

Now recall that Lemma 5.1 does not necessarily require that we label any nodes. We may also reduce the size of label sets on the left. This allows us to deal with a special case:

LEMMA 5.4. *In the above notation, if* $\Delta_L \cdot \Delta_R \geq |L|$*, then for any* $\varepsilon > 0$*, we can either satisfy an* $\tilde{\Omega}(1/N^\varepsilon)$*-fraction of all constraints, or output a polynomial number of label set reductions* $\Sigma_1, \ldots, \Sigma_t : L \to [k]$ *as in Lemma 5.1 with alphabet size at most* k/N^ε*.*

Proof. First, if $d \geq k/N^\varepsilon$, then by Lemma 5.3, we can label an $\tilde{\Omega}(d/k) = \tilde{\Omega}(1/N^\varepsilon)$-fraction of all constraints.

Thus, suppose $d < k/N^\varepsilon$. Then for every node $u \in L$ and label $\sigma_u \in [k]$, let R_{u,σ_u} be the set of neighbours v of u such that the projection $\pi_{(u,v)}$ maps σ_u to some label of v. Now, let $L_{u,\sigma_u} = \Gamma(R_{u,\sigma_u})$. For every node $u' \in L_{u,\sigma_u}$, assign to this node the label set

$$\Sigma_{u,\sigma_u}(u') = \bigcap_{v \in \Gamma(u') \cap R_{u,\sigma_u}} \pi^{-1}_{(u',v)}(\pi_{(u,v)}(\sigma_u)).$$

That is, $\Sigma_{u,\sigma_u}(u')$ is the set of labels for that node whose projections agree completely with the projections of σ_u for u. Note that fixing any neighbour $v \in \Gamma(u') \cap R_{u,\sigma_u}$, the new label set for u' has size $|\pi^{-1}_{(u',v)}(\pi_{(u,v)}(\sigma_u))| \leq d < k/N^\varepsilon$, as required. For nodes $u' \in L \setminus L_{u,\sigma_u}$, let $\Sigma_{u,\sigma_u}(u')$ be any arbitrary set of k/N^ε labels.

We now output the list of all label reductions Σ_{u,σ_u} (for all $u \in L$ and $\sigma_u \in [k]$). To see that at least one of these label sets satisfies the requirements of the lemma, consider a left node u in H, and a correct label σ_u for u (in a labeling which completely satisfies H). Then every node in $v \in R_{u,\sigma_u}$ will be assigned the correct label $\pi_{(u,v)}(\sigma_u)$, and thus for every node $u' \in V(H) \cap L_{u,\sigma_u} = V(H) \cap \Gamma(R_{u,\sigma_u})$, the label set $\Sigma_{u,\sigma_u}(u')$ will also contain the correct label for u'. Since $\Delta_L \cdot \Delta_R \geq |L|$, by the expansion properties of H, we have that $|V(H) \cap L_{u,\sigma_u}| = \tilde{\Omega}(|L|)$, which, by near-regularity, makes the instance defined by label sets Σ_{u,σ_u} at least $\tilde{\Omega}(1)$-satisfiable. □

As in the random planted case, the main component of our algorithm performs well when there is a significant log density gap. In this case, our algorithm will reduce the label sets in L by a factor of N^ε. It may seem strange that this can always be achieved, but note that for a significant log density gap we must have $k > N^\varepsilon$, so the label reduction will not continue indefinitely. In fact, it may not occur at all, if we our instance does not have a log density gap to begin with. The guarantee of the main component of our algorithm is as follows:

LEMMA 5.5. *In a Label Cover instance as above, if for some* $\alpha > 0$ *and* $\varepsilon > 0$ *we have* $d \leq k^\alpha/N^\varepsilon$ *and* $\Delta_L\Delta_R \geq |L|^\alpha N^\varepsilon$*, then an algorithm that runs in time* $N^{O(1/\varepsilon)}$ *finds a polynomial number of left label reductions* $\{\Sigma_i : L \to [k] \mid i \in [t]\}$ *as in Lemma 5.1 with alphabet size at most* $k/N^{\varepsilon/2}$*.*

The next section is devoted entirely to the algorithm which proves this lemma. Let us now see how our main guarantee now follows.

Proof of Lemma 5.1. First, consider the case when $|R| \leq |L|$, and let $|R| = |L|^\gamma$ (for some $\gamma \leq 1$). Let $\alpha = \frac{2-2\beta}{2-\beta} \cdot \gamma + 1 - \gamma (= 1 - \frac{\beta}{2-\beta} \cdot \gamma)$. By Lemma 5.5, if we have $d \leq k^\alpha/N^\varepsilon$ and $\Delta_L \geq |L|^{(1-\beta)\gamma/(2-\beta)}N^\varepsilon$, then

we are done, since the latter inequality implies that

$$\Delta_L \cdot \Delta_R = \Delta_L \cdot (\Delta_L \cdot |L|/|R|) = \Delta_L^2 |L|^{1-\gamma} \geq |L|^{\alpha} N^{2\varepsilon}.$$

On the other hand, suppose one of the two bounds does not hold. If $d > k^{\alpha}/N^{\varepsilon}$, then by Lemma 5.3, we can find a labeling where the fraction of constraints satisfied is

$$\frac{d}{k} > \frac{k^{\alpha-1}}{N^{\varepsilon}} = \frac{k^{-\frac{\beta}{2-\beta}\cdot\gamma}}{N^{\varepsilon}} = N^{-\frac{\beta(1-\beta)}{2-\beta}\cdot\gamma-\varepsilon} \geq N^{-\frac{\beta(1-\beta)}{2-\beta}-\varepsilon}.$$

If $\Delta_L < |L|^{(1-\beta)\gamma/(2-\beta)}N^{\varepsilon}$, then again we can apply Lemma 5.3, and get a labeling in which the fraction of satisfied constraints is at least

$$\Delta_L^{-1} \geq 1/(|L|^{\frac{(1-\beta)}{(2-\beta)}\cdot\gamma}N^{\varepsilon}) = N^{-\frac{\beta(1-\beta)}{2-\beta}\cdot\gamma-\varepsilon} \geq N^{-\frac{\beta(1-\beta)}{2-\beta}-\varepsilon}.$$

Now consider the case when $|R| \geq |L|$, where $|R| = |L|^{\gamma}$ (now for some $\gamma \geq 1$). Let us first eliminate a simple case: If $\gamma \geq 2$, then note we must have $\Delta_R \geq 1$ (otherwise, by Lemma 5.3, we can satisfy an $\tilde{\Omega}(1)$-fraction of constraints). But this means that $\Delta_L = |R|\Delta_R/|L| = |L|^{\gamma-1}\Delta_R \geq |L|^{\gamma-1} \geq |L|$, and so $\Delta_L \cdot \Delta_R \geq |L|$, and then we are done by Lemma 5.4. Thus, we may assume $\gamma \in [1, 2)$. Now set $\alpha = 1 - \frac{\beta}{2-\beta} \cdot (2-\gamma)$. If $d \geq k^{\alpha}/N\varepsilon$ and $\Delta_L \geq |L|^{(\gamma-\beta)/(2-\beta)} \cdot N^{\varepsilon}$, then we are done by Lemma 5.5, since the latter inequality implies that

$$\Delta_L \cdot \Delta_R = \Delta_L^2 |L|^{1-\gamma} \geq |L|^{1-\gamma+\frac{2\gamma-2\beta}{2-\beta}} N^{2\varepsilon} = |L|^{\alpha} N^{2\varepsilon}.$$

On the other hand, suppose one of the two bounds does not hold. If $d > k^{\alpha}/N^{\varepsilon}$, then by Lemma 5.3, we get a labeling in which the fraction of satisfied constraints is at least

$$\frac{d}{k} > k^{-\frac{\beta}{2-\beta}(2-\gamma)}/N^{\varepsilon} = N^{\frac{-\beta(1-\beta)}{2-\beta}\cdot(2-\gamma)-\varepsilon} \geq N^{-\frac{\beta(1-\beta)}{2-\beta}-\varepsilon}.$$

If $\Delta_L < |L|^{(\gamma-\beta)/(2-\beta)}N^{\varepsilon}$, then again we can apply Lemma 5.3, and get a labeling in which the fraction of satisfied constraints is at least

$$\begin{aligned} 1/\Delta_R = |R|/(\Delta_L|L|) &\geq 1/(|L|^{\frac{\gamma-\beta}{2-\beta}+1-\gamma}N^{\varepsilon}) \\ &= N^{-\frac{\beta(1-\beta)}{2-\beta}\cdot(2-\gamma)-\varepsilon} \geq N^{-\frac{\beta(1-\beta)}{2-\beta}-\varepsilon}. \end{aligned}$$

□

5.6 Label Reduction from Log-density Gap In this section we will prove Lemma 5.5 and present the algorithm that achieves the corresponding alphabet size reduction, if there is a non-negligible gap in the log-density. The algorithm is very similar to the algorithm Alg-Planted$_{r,s}$ which we used for the random planted setting. We will sometimes use for convenience n to represent $|L|$. Recall that as in section 5.5, we have approximate regularity properties:

- The supergraph $G = (L, R, E)$ is weakly (Δ_L, Δ_R)-expanding, and nodes in L have alphabet size k.
- The projections are d-to-1 for some $d > 0$: For every superedge $(u, v) \in E$, every label σ in the image of $\pi_{(u,v)}$ has a preimage of size $|\pi^{-1}_{(u,v)}(\sigma)| \leq d$.
- For the same d the label set of every right vertex $v \in R$ has size $|\bigcup_{u\in\Gamma(v)} \pi_{(u,v)}([k])| = \tilde{O}(k/d)$.
- There is a subgraph H of G with $\tilde{\Omega}(|E|)$ edges which is also weakly (Δ_L, Δ_R)-expanding, and fully satisfiable (satisfying projections $\{\pi_e\}_{e\in E}$). Denote by $\phi_H : V(H) \to [k]$ a satisfying assignment to the vertices of H.

For any $\alpha \in (0, 1)$, we can choose natural numbers $r, s > 0$ so that $|\alpha - \frac{r}{s}| < \varepsilon/2$ and $s \leq \lceil 1/\varepsilon \rceil$. Note that if we assume (as in Lemma 5.5) that $\Delta_L\Delta_R \geq |L|^{\alpha}N^{\varepsilon}$ and $d \leq k^{\alpha}/N^{\varepsilon}$ then for $\alpha' = r/s$ we still have

$$\Delta_L\Delta_R \geq |L|^{\alpha'}N^{\varepsilon/2} \qquad \text{and} \qquad d \leq k^{\alpha'}/N^{\varepsilon/2},$$

so $\alpha' = r/s$ still lies within the log-density gap. We now describe the algorithm for Label reduction in semi-random instances in detail.

Algorithm Alg-Semirandom$_{r,s}$ for witness $\mathcal{W}_{r,s}$. The algorithm Alg-Semirandom$_{r,s}$ corresponding to structure $\mathcal{W}_{r,s}$ will try all possible choices to fix the leaves of $\mathcal{W}_{r,s}$ to be vertices $u_0, u_1, \ldots, u_{r-1}$ (on the left) and every possible labeling for these leaves, and then consider the candidates for the "free vertices" of $\mathcal{W}_{r,s}$ (in particular, the $(s-r)$ free vertices from the left) and possible labels for each of these vertices. Specifically, at every step of the iteration, we will maintain some set of nodes S, and some set of labels for each node in the set. Initially, this set will simply be the first leaf which we fix to be $S = \{u_0\}$, with a label which we will guess (in the final algorithm, we will try all possible labelings for the constant number of fixed nodes). At each step, we will also enforce some bound on the size of label sets of vertices in S (initially, this bound will be 1, meaning we will guess a singly label for u_0). The rest of the iteration proceeds by following along the inductive construction of $\mathcal{W}_{r,s}$ as defined in Definition 4.1:

Backbone Step: If the current step is a "backbone step", we let our new set be $\Gamma_2(S)$, and we restrict the label sets in the following way:

1. For every vertex $v \in \Gamma(S)$, restrict its label set only to labels that are consistent with some label in at least a $\tilde{\Omega}(1)$-fraction of v's neighbors in S (in particular, if it has a single neighbor in $\Gamma(S)$, simply take the labels corresponding to the remaining label set of that neighbor).

2. For every vertex $u \in \Gamma_2(S)$, restrict its label set only to labels that are consistent with some label in at least a $\tilde{\Omega}(1)$-fraction of u's neighbors in $\Gamma(S)$ (in particular, if it has a single neighbor in $\Gamma(S)$, simply take the labels corresponding to the remaining label set of that neighbor).

Hair Step: If the current step in the construction is a "hair step", let k' be the current bound on the size of label sets in S. We will try two approaches:

1. Fix the next leaf $u_i \in L$ and a label σ_{u_i} for u_i (the algorithm will try performing the hair step for all possible choices for u_i and σ_{u_i}). Let $S' = S \cap \Gamma_2(u_i)$. Our target bound for label sets in S' will be $k'' = k'/k^{1-r/s}$. For every vertex in $u \in S'$, restrict its current label set to labels that match the label of at least one neighbor in $\Gamma(u) \cap \Gamma(u_i)$ induced by the labeling of u_i. That is, if the current label set for u is $\Sigma(u)$, let
$$\Sigma'(u) = \Sigma(u) \cap \bigcup_{v \in \Gamma(u_i) \cap \Gamma(u)} \pi^{-1}_{(u,v)}(\pi_{(u_i,v)}(\sigma_{u_i})).$$
Discard from S' any vertices which still have more than $\tilde{O}(k'')$ labels, and let the new vertex set be the remaining vertices in S'.
2. For every vertex $v \in \Gamma(S)$, consider every possible label σ_v of v. If for at least an $\tilde{\Omega}(1)$-fraction of neighbors $u \in \Gamma(v) \cap S$ we have $|\pi^{-1}_{(u,v)}(\sigma_v)| > k''$, then call σ_v a bad label for v (in this step). For every $u \in L$, call a label σ_u bad for u if for at least an $\tilde{\Omega}(1)$ fraction of neighbors $v \in \Gamma(u) \cap \Gamma(S)$ the matching label $\pi_{(u,v)}(\sigma_u)$ is bad for v. Add to the current list of label reductions the reduction which assigns to every left node $u \in L$ only its bad labels.

The following two lemmas, whose proofs are deferred to the full version, allow us to maintain the required lower bound on the size of the node set S, and an upper bound on the size of its labels after the *backbone step* and *hair step*. Lemma 5.5 follows by combining these two lemmas, based on the inductive definition of $\mathcal{W}_{r,s}$.

LEMMA 5.6. (BACKBONE STEP) *Let $S \subseteq L$ be a node set with label sets $\Sigma_S : S \to \mathcal{P}([k])$ such that for at least an $\tilde{\Omega}(1)$ fraction of nodes $u \in S$ we have $u \in H$ and $\phi_H(u) \in \Sigma_S(u)$, and suppose $\max_{u \in S} |\Sigma_S(u)| \leq k'$. Then letting $S' = \Gamma_2(S)$ be the new set of nodes and $\Sigma_{S'} : S' \to \mathcal{P}([k])$ be the label sets for these nodes after applying a backbone step to (S, Σ_S), then we have*

(5.6)
$$|\{u' \in S' \cap V(H) \mid \phi_H(u') \in \Sigma_{S'}(u')\}| = \tilde{\Omega}(|S'|), \quad \text{and} \quad \max_{u' \in S'} |\Sigma_{S'}(u')| = \tilde{O}(dk').$$

LEMMA 5.7. (HAIR STEP) *Let $S \subseteq L$ be a node set of size $|S| \geq \tilde{\Omega}(|L|)/\Delta_L\Delta_R$, with label sets $\Sigma_S : S \to \mathcal{P}([k])$ such that for at least an $\tilde{\Omega}(1)$ fraction of nodes $u \in S$ we have $u \in H$ and $\phi_H(u) \in \Sigma_S(u)$, and suppose $\max_{u \in S} |\Sigma_S(u)| \leq k'$ for some $k' \geq k^{1-r/s}$. Let $\Sigma' : L \to \mathcal{P}([k])$ be the label reduction after applying the second part of a hair step to (S, Σ_S), and for every choice of leaf and label (u_i, σ_{u_i}), let $S'(u_i, \sigma_{u_i})$ be the new set of nodes and $\Sigma_{S'(u_i,\sigma_{u_i})} : S'(u_i, \sigma_{u_i}) \to \mathcal{P}([k])$ after applying the first part of the hair step. Then at least one of the following two statements hold:*

- *There exists some choice of leaf $u_i \in L$ and label σ_{u_i} for u_i such that*
$$\max_{u' \in S'(u_i,\sigma_{u_i})} |\Sigma_{S'(u_i,\sigma_{u_i})}(u')| = \tilde{O}(k'/k^{1-r/s}), \quad \text{and}$$
$$|\{u' \in S'(u_i, \sigma_{u_i}) \cap V(H) \mid \phi_H(u') \in \Sigma_{S'(u_i,\sigma_{u_i})}\}| = \tilde{\Omega}(|S'(u_i, \sigma_{u_i})|).$$
- *The label reduction Σ' restricts the alphabet size to at most $\tilde{O}(k/N^{\varepsilon/2})$, and moreover for an $\tilde{\Omega}(1)$ fraction of left nodes $u \in L \cap V(H)$ we retain the correct label, that is, $\phi_H(u) \in \Sigma'(u)$.*

Completing the Proof of Lemma 5.5. We follow the same approach as in the Planted Random model (Subsection 4.2) and combine the guarantees of Lemma 5.6, Lemma 5.7 as given by the structure $\mathcal{W}_{r,s}$. Recollect that by step $t \in [s]$ in the inductive construction of caterpillar $W(r,s)$, $h_t = \lfloor \frac{tr}{s} \rfloor$ of them are hair steps and the rest $b_t = t - h_t$ steps are backbone steps. We will abuse notation and use $S(t)$ to represent the number of candidates of the set S after t steps, and $K(t)$ represented the largest label set size after t steps. The following simple claim proof follows by induction:

CLAIM 5.1. *In the above notation, let after t steps of the algorithm, $S \subseteq L$ be the node set with label sets $\Sigma_S : S \to \mathcal{P}([k])$. Further, let $\{\Sigma'_i : L \to \mathcal{P}([k]) \mid i \in q_t\}$ be the reduced label sets obtained by from the hair steps (approach 2). Then we have at least one of the following:*

1. *For at least an $\tilde{\Omega}(1)$ fraction of nodes $u \in S$ we have $u \in H$ and $\phi_H(u) \in \Sigma_S(u)$, and suppose $K(t) = \max_{u \in S} |\Sigma_S(u)|$,*
$$K(t) \leq \tilde{O}(1) \cdot k^{b_t \cdot r/s} k^{h_t(r/s-1)} = \tilde{O}(1) \cdot \frac{k^{tr/s}}{k^{h_t}}.$$
2. *$q_t \leq N^t$, and for every $i \in [q_t]$ and $u \in L$, we have $|\Sigma'_i(u)| = \tilde{O}(k/N^{\varepsilon/2})$. Further there exists $i \in [q_t]$ such that for an $\tilde{\Omega}(1)$ fraction of the left nodes $u \in L \cap V(H)$ we retain the correct label i.e., $\phi_H(u) \in \Sigma'_i(u)$.*

The proof of the above claim follows by a simple induction. The base case is true because at $t = 0$, we start with a candidate set of size 1 and a guess of the correct label. The induction follows easily using Lemma 5.7 and Lemma 5.6 as follows. If case (2) holds after step t, then it also holds at step $t+1$. If case (1) holds after step t, the condition of Lemma 5.6 or Lemma 5.7 holds, depending on whether it is a backbone step or hair step (respectively). Then, by setting $k' = K(t)$, we get the required bound for $K(t+1)$. Further, the number of reduced label sets $q_t \leq N^t$ because every hair step (and guess of leaf and its label) gives rise to a reduced label set Σ'_i.

Consider the final step, i.e., for $t = s$. Unlike the distinguishing algorithm, where the last step step is a hair step, here the last step is a backbone step to find a labeling. If case (2) in the above Claim holds, then we are done. Otherwise, we will now show that the label set size is at most $\tilde{O}(k)/N^{\varepsilon/2}$. Further, since the last step is a backbone step, we have that $S(s-1) \geq \tilde{\Omega}(|L|)/(\Delta_L \Delta_R)$ (this also holds when restricted to H). From Lemma 5.6, the candidate set after the final step in both G and H is $\tilde{\Omega}(|L|)$. From the Claim after step $(s-1)$ i.e., before step s, the label set size $k' \leq \tilde{O}(1)k^{(s-1)r/s}/k^{r-1} = k^{1-r/s}$. Hence, using Lemma 5.6, we see that the size of final label set is at most

$$\tilde{O}(k'd) = \tilde{O}(1) \cdot k/N^{\varepsilon/2}.$$

This completes the proof of the Lemma 5.5.

6 Algorithm for Worst Case Label Cover

In this section, we describe an $O\left((nk)^{\frac{1}{6}(5-\sqrt{13})+\delta}\right)$-approximation algorithm for satisfiable Label-Cover for any constant $\delta > 0$, thereby proving Theorem 1.2. To construct the algorithm, our strategy is to take an efficient $\tilde{O}((n_L k_L)^{\alpha})$-approximation algorithm for some $\alpha > \frac{1}{6}(5-\sqrt{13})$ and boost its approximation ratio to $\tilde{O}((n_L k_L)^{\alpha'})$ for some $\alpha' < \alpha$. Specifically, we will prove the following lemma.

Lemma 6.1. *For any $\alpha \in (0,1]$, if there exists a polynomial-time $\tilde{O}\left((n_L k_L)^{\alpha}\right)$-approximation algorithm for satisfiable* Label-Cover*, then there also exists a polynomial-time $\tilde{O}\left((n_L k_L)^{\frac{1}{5-3\alpha}}\right)$-approximation algorithm for satisfiable* Label-Cover*.*

Theorem 1.2 can now be easily proved by starting with any algorithm and repeatedly applying Lemma 6.1.

Proof of Theorem 1.2. We start with the $O((n_L k_L)^{1/4})$-approximation algorithm from [MM13]. We then define the i-th algorithm by applying Lemma 6.1 to the $(i-1)$-th algorithm. The lemma implies that the i-th algorithm has approximation ratio $\tilde{O}((n_L k_L)^{\alpha_i})$ where $\alpha_1 = 0.25$ and $\alpha_i = 1/(5 - 3\alpha_{i-1})$ for all $i > 1$. Clearly, the sequence $\{\alpha_i\}_{i \in \mathbb{N}}$ converges to $\frac{1}{6}(5-\sqrt{13})$, completing the proof of the theorem. □

Due to space constraints, we will only give an overview of the proof of Lemma 6.1. The rest of this section is organized as follows. First, in Subsection 6.1, we discuss the main difference between the semi-random case and the worse case; with this in mind, we introduce a notation of *cutting algorithms* and relate them to approximation algorithms. In Subsection 6.2, we describe a simplified version of our algorithm, in the "uniform" setting. Finally, in Subsection 6.3, we outline how "non-uniformity" is handled in our algorithm.

6.1 Game Partitioning and Approximation Algorithms

The main difference between worst case instances and semi-random instances is that, in the worst case scenario, the supergraphs need not expand. This means that, if we try to use the semi-random algorithm for worst case instances, it is possible that, even after a certain number of the backbone steps, the set S still remains small. In this case, even though the backbone steps do not work well, we learn that the graph does not expand much, which allows us to partition the graph into smaller components in such a way that a considerable fraction of edges remain inside the components. If we can also choose each component so that it is easy for us to solve or approximate the subinstance induced on the component, then we can arrive at a good solution to the original instance by solving these subinstances separately and combining their solutions.

Before we specify how the supergraph is partitioned, let us first formalize the notion of *cutting algorithms* and how good they are as follows.

Definition 6.1. *A* cutting algorithm *for* Label-Cover *is an algorithm that takes in a* Label-Cover *instace $\mathcal{G} = (L, R, E, \Sigma_L, \Sigma_R, \{\pi_e\}_{e \in E})$, and outputs a non-empty set $T \subseteq L \cup R$ and a partial assignment $\phi^T : T \to \Sigma_L \cup \Sigma_R$.*

The cutting ratio *for such algorithm is defined as the ratio between edges in $E(T,T)$ satisfied by the assignment and the number of edges in $E(T)$, i.e.,*

$$\frac{|\{(a,b) \in E(T,T) \mid \pi_{(a,b)}(\phi^T(a)) = \phi^T(b)\}|}{|E(T)|}.$$

Note that the cutting ratio is the ratio of the number of satisfied edges over the number of edges that we lose when we cut T from the rest of the graph.

Cutting algorithms and approximation algorithms are closely related; if there is a cutting algorithm with a high cutting ratio, there is also an approximation algorithm with a low approximation ratio:

LEMMA 6.2. *For any non-decreasing function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{R}^+$, if there is a polynomial-time cutting algorithm with cutting ratio at least $1/f(n_L, k_L)$ for every satisfiable* LABEL-COVER *instance, then there is a polynomial-time $f(n_L, k_L)$-approximation algorithm for satisfiable* LABEL-COVER.

The proof of the lemma is rather simple: we use the cutting algorithm to find T and ϕ^T. We then remove T from the instance and repeat until it is empty. We do not elaborate further here since a generalized version of this lemma (Lemma 6.4) will appear later. With Lemma 6.2 in hand, to prove Lemma 6.1, we only need to find a sufficiently good cutting algorithm, i.e., it is enough to prove the following lemma.

LEMMA 6.3. *For any constant $\alpha \in (0, 1]$, if there exists a polynomial-time $\tilde{O}\left((n_L k_L)^\alpha\right)$-approximation algorithm for satisfiable* LABEL-COVER, *then there is a cutting algorithm with cutting ratio $\tilde{\Omega}\left(1/(n_L k_L)^{\frac{1}{5-3\alpha}}\right)$ for every satisfiable* LABEL-COVER *instance.*

6.2 Algorithm Overview for the Uniform Case

To gain intuition for our algorithm, we first describe a simplified algorithm under the assumptions that the input instance $\mathcal{G}$ is exactly d-to-1 (i.e. $|\pi_e^{-1}(\sigma)| = d$ for all $e \in E, \sigma \in \Sigma$) and its supergraph is biregular.

Our algorithm once again employs backbone steps as subroutines. Since $\mathcal{G}$ is satisfiable, we use the simpler version of the backbone step from Section 4.2. Recall that, for $S \subseteq L$ and restricted label sets, a backbone step restricts the label set of each $v' \in \Gamma(S)$ to only labels that are consistent with at least one label of each vertex in $\Gamma(v') \cap S$. Then, it similarly restricts the label set of each $u' \in \Gamma(\Gamma(S))$ only to labels that are consistent with some label of each vertex in $\Gamma(u) \cap \Gamma(S)$. The step ends by setting $S = \Gamma(\Gamma(S))$.

Clearly, if the starting label $\sigma_u \in \Sigma_L$ is the right label for the starting vertex $u \in L$, then the backbone step retains the right label for every vertex. Moreover, if $\mathcal{G}$ is d-to-1, after $\lfloor (i+1)/2 \rfloor$ backbone steps, the number of labels left in $\Gamma_i(u)$ is at most $d^{\lfloor i/2 \rfloor}$. With these observations, our algorithm can be described as follows. The algorithm tries five different cutting approaches and uses the best of them. More specifically, for a vertex $u \in L$, we can devise polynomial-time cutting algorithms with the following cutting ratios:

1. $\frac{d}{k_L}$: Assign an arbitrary label to all vertices in R and, for each vertex in L, pick a label that satisfies the maximum number of edges touching it.
2. $\frac{|E(\{u\})|}{|E(\Gamma(u))|}$: Pick $T = \{u\} \cup \Gamma(u)$, which can be fully satisfied in polynomial time.
3. $\frac{|E(\Gamma(u))|}{|E(\Gamma_2(u))|}$: Pick $T = \Gamma(u) \cup \Gamma_2(u)$. This again can be fully satisfied in polynomial time since a backbone step restricts the label sets of $\Gamma(u)$ to be of size one.
4. $\tilde{\Omega}\left(\frac{|E(\Gamma_2(u))|}{|E(\Gamma_3(u))|}\left(\frac{1}{|\Gamma_2(u)|d}\right)^\alpha\right)$: Pick $T = \Gamma_2(u) \cup \Gamma_3(u)$. A backbone step reduces the number of labels for each vertex in $\Gamma_2(u)$ to be at most d. Then use the $\tilde{O}\left((n_L k_L)^\alpha\right)$-approximation algorithm on the subgame induced on S with these reduced alphabets.
5. $\frac{|E(\Gamma_3(u))|}{|E(\Gamma_4(u))|}\left(\frac{1}{d}\right)$: Pick $T = \Gamma_3(u) \cup \Gamma_4(u)$. Two backbone steps reduce the number of labels of each vertex in $\Gamma_3(u)$ to at most d. A greedy algorithm can then satisfy $1/d$ fraction of the edges in $E(\Gamma_3(u))$.

By picking the best of the five, the resulting cutting ratio is the maximum of the five ratios, which is always at least their weighted geometric mean. Assign weight $(1-\alpha)$ to each of $\frac{|E(\{u\})|}{|E(\Gamma(u))|}$, $\frac{|E(\Gamma(u))|}{|E(\Gamma_2(u))|}$, $\frac{|E(\Gamma_3(u))|}{|E(\Gamma_4(u))|}\left(\frac{1}{d}\right)$ and assign weight 1 to each of $\frac{d}{k_L}$, $\frac{|E(\Gamma_2(u))|}{|E(\Gamma_3(u))|}\left(\frac{1}{|\Gamma_2(u)|d}\right)^\alpha$. The cutting ratio of the algorithm is at least

$$\tilde{\Omega}\left(\sqrt[5-3\alpha]{\frac{|E(\{u\})|^{1-\alpha}|E(\Gamma_2(u))|^{1-\alpha}}{k_L|E(\Gamma_3(u))|^\alpha|E(\Gamma_4(u))|^{1-\alpha}|\Gamma_2(u)|^{1-\alpha}}}\right) \geq \tilde{\Omega}\left(\sqrt[5-3\alpha]{\frac{1}{n_L k_L}}\right),$$

where the inequality comes from $|E(\Gamma_3(u))|, |E(\Gamma_4(u))| \leq |E|$ and the biregularity of the graph.

Note here that, for these uniform instances, the algorithm from [MM13], although phrased differently, is simply the best of four algorithms with cutting ratios $\frac{d}{k_L}$, $\frac{|E(\{u\})|}{|E(\Gamma(u))|}$, $\frac{|E(\Gamma(u))|}{|E(\Gamma_2(u))|}$ and $\frac{|E(\Gamma_2(u))|}{|E(\Gamma_3(u))|}\left(\frac{1}{d}\right)$. The first three were described above whereas the last is a simple greedy algorithm on $\Gamma_2(u) \cup \Gamma_3(u)$ after the label sets of $\Gamma_2(u)$ have been reduced to have size at most d. By picking the best of these four, the cutting ratio is at least $(nk)^{-1/4}$ and the equality occurs only when $\Gamma_3(u) = R$. However, in this case, the restricted label set of every vertex in R is of size at most p! Hence, a reasonable thing to do in this case is to apply the algorithm recursively on this restricted instance, which ultimately inspired the fourth algorithm described above.

6.3 Handling Non-Uniformity In this subsection, we outline the main ideas needed to turn the algorithm from the previous subsection to handle non-uniform instances. There are two non-uniformities we have to deal with: the degrees and the preimages.

6.3.1 Part I: The Degrees Non-uniformity of the degrees is easy to deal with. Specifically, we can prove the following lemma, which implies that, to prove Theorem 6.1, it suffices to find a cutting algorithm with cutting ratio $\tilde{\Omega}\left(1/(n_L k_L)^{\frac{1}{5-3\alpha}}\right)$ for $O(\log^4(n_L n_R))$-nearly biregular satisfiable games.

LEMMA 6.4. *For any non-decreasing function $f : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{R}^+$, if there is a polynomial-time cutting algorithm whose cutting ratio is at least $1/f(n_L, k_L, \lambda)$ for every satisfiable λ-nearly biregular* LABEL-COVER *instance, then there exists a polynomial-time $O(\log^4(n_L n_R) \cdot f(n_L, k_L, O(\log^4(n_L n_R))))$-approximation algorithm for satisfiable* LABEL-COVER.

While we do not provide a full proof of Lemma 6.4 here, the idea behind it is quite intuitive. Again, we would like to repeatedly apply the cutting algorithm to find T, ϕ^T, remove T from the instance, and continue doing so until the instance is empty. However, since the given cutting algorithm has guarantees only on nearly biregular instances, we need to make the graph nearly biregular before we appeal to the algorithm. Here we do the most natural thing before we call the cutting algorithm: as long as there is some vertex whose degree is too large or too small, remove it from the graph.

In particular, the approximation algorithm works on input $\mathcal{G} = (L, R, E, \Sigma_L, \Sigma_R, \{\pi_e\}_{e \in E})$ as follows:

1. Pick $\Delta_L \in [n_R], \Delta_R \in [n_L]$ that maximize the number of $(u, v) \in E$ with $\deg(u) \in [\Delta_L/2, \Delta_L]$ and $\deg(v) \in [\Delta_R/2, \Delta_R]$. Let $\lambda = 20 \log^4(n_L n_R)$ and let $G^{(1)} = (L^{(1)}, R^{(1)}, E^{(1)}) = (L, R, E)$.
2. While $E^{(i)} \neq \emptyset$, do the following:
   (a) As long as some $u \in L^{(i)}$ has degree in $G^{(i)}$ less than Δ_L/λ or more than Δ_L or some $v \in R^{(i)}$ has degree less than Δ_R/λ or more than Δ_R, remove one such u or v from $(L^{(i)}, R^{(i)}, E^{(i)})$.
   (b) Run the cutting algorithm on $\mathcal{G}^{(i)} = (L^{(i)}, R^{(i)}, E^{(i)}, \Sigma_L, \Sigma_R, \{\pi_e\}_{e \in E^{(i)}})$. Let us call the output of the algorithm T_i and ϕ^{T_i}.
   (c) Set $L^{(i+1)} \leftarrow L^{(i+1)} - T_i, R^{(i+1)} \leftarrow R^{(i)} - T_i$ and $E^{(i+1)} \leftarrow E^{(i)} - E^{(i)}(T_i)$ where $E^{(i)}(T_i)$ is the set of all edges with at least one endpoint in T_i with respect to $(L^{(i)}, R^{(i)}, E^{(i)})$.
3. For each $u \in T_i$, set $\phi(u) \leftarrow \phi^{T_i}(u)$. For any vertex u not assigned in any step, assign any label to it.

Clearly, when we invoke the cutting algorithm in Step 2b, $\mathcal{G}^{(i)}$ is λ-nearly biregular. Moreover, it is possible to argue that $\sum_i |E^{(i)}(T_i)| \geq \Omega(|E|/\lambda)$. This implies that the algorithm indeed provides the desired approximation for satisfiable LABEL-COVER.

6.3.2 Part II: The Preimages Alas, dealing with nonuniformity of preimages is harder. For instance, if we use the preprocessing lemma from the semi-random algorithm, the game becomes unsatisfiable. Since the graph is not random, a well-spread satisfiable subgraph need not exist, causing the backbone step to fail.

Fortunately, there is a rather straightforward approach to get around nonuniformity of preimages: in the backbone step, we will discard the vertices whose reduced alphabets are "too large". But how large is too large? The threshold is going to be based on what we get from the following greedy algorithm: for each $v \in R$, pick σ_v^{greedy} that maximizes the sum of its preimages size with respect to all edges touching v, i.e. $\sum_{u \in \Gamma(v)} |\pi^{-1}_{(u,v)}(\sigma_v)|$. Then, for each $u \in L$, simply pick its assignment uniformly at random from Σ_L. Define d_{greedy} to be the average number of preimages with respect to σ_v^{greedy} over all edges, i.e., $d_{\text{greedy}} = \frac{1}{|E|} \sum_{(u,v) \in E} |\pi^{-1}_{(u,v)}(\sigma_v^{\text{greedy}})|$. It is easy to see that this algorithm gives an approximation ratio of $\frac{d_{\text{greedy}}}{k_L}$ in expectation and that it can be derandomized:

LEMMA 6.5. *There exists a polynomial-time $\frac{d_{greedy}}{k_L}$-approximation algorithm for* LABEL-COVER.

Since Lemma 6.5 was also proved in [MM13], we do not prove it here. Let us turn our focus back to the rest of the algorithm. We use $2d_{\text{greedy}}$ as the threshold for considering the reduced label set for each vertex in $\Gamma_2(u)$ being "too large". For every $u \in L$ and $\sigma_u \in \Sigma_L$, let $S^2(u, \sigma_u)$ be the set of all vertices in $\Gamma_2(u)$ that, after a backbone step, their remaining label sets are of size at most $2d_{greedy}$. These are the set of "good" vertices that we use in place of $\Gamma_2(u)$. The algorithm remains almost the same as the algorithm for the uniform preimages case except that we use $S^2(u, \sigma_u)$ instead of $\Gamma_2(u)$. This translates to the following guarantees on the algorithm. For convenience, here we use S^2_u to denote $S^2(u, \sigma_u^{opt})$ where σ_u^{opt} is the satisfying assignment for u.

LEMMA 6.6. *If there is a polynomial-time $\tilde{O}\left((n_L k_L)^\alpha\right)$-approximation algorithm for satisfiable* LABEL-COVER*, then there are polynomial-time cutting algorithms that, for any satisfiable* LABEL-COVER *instance and any vertex $u \in L$, their cutting ratios are at least*

$\frac{|E(\{u\})|}{|E(\Gamma(u))|}, \frac{|E(\Gamma(u),S_u^2)|}{|E(S_u^2)|}, \tilde{\Omega}\left(\frac{|E(S_u^2)|}{|E(\Gamma(S_u^2))|}\left(\frac{1}{d_{greedy}|S_u^2|}\right)^{\alpha}\right)$ *and* $\frac{|E(\Gamma(S_u^2))|}{|E(\Gamma_2(S_u^2))|}\left(\frac{1}{2d_{greedy}}\right)$.

Since the algorithms in Lemma 6.6 are essentially the same as the algorithms from Subsection 6.2 except that $S^2(u,\sigma_u)$ is used instead of $\Gamma_2(u)$, we do not restate their descriptions here.

Once we have Lemma 6.5 and Lemma 6.6, we again use the best of the five algorithms from the two lemmas. When the supergraph is λ-nearly biregular, it is not hard to argue, similarly to the uniform case, that, with an appropriate choice of u, the maximum ratio is at least $\tilde{\Omega}\left(1/(n_L k_L)^{\frac{1}{5-3\alpha}}\right)/\lambda$. This, combined with Lemma 6.4, yields an $\tilde{\Omega}\left(1/(n_L k_L)^{\frac{1}{5-3\alpha}}\right)$-approximation algorithm for satisfiable Label-Cover, which concludes our proof overview for Lemma 6.1.

References

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

[ABBG10] S. Arora, B. Barak, M. Brunnermeier, and R. Ge. Computational complexity and information asymmetry in financial products. In *Proceedings of the First Symp. on Innovations in Computer Science (ICS)*, 2010.

[ABW10] B. Applebaum, B. Barak, and A. Wigderson. Public-key cryptography from different assumptions. In *Proc. 42nd ACM Symp. on Theory of Computing*, pages 171–180, 2010.

[ACLR15] P. Awasthi, M. Charikar, K. A. Lai, and A. Risteski. Label optimal regret bounds for online local learning. In *Proceedings of The 28th Conference on Learning Theory COLT*, pages 150–166, 2015.

[BCC+10] A. Bhaskara, M. Charikar, E. Chlamtáč, U. Feige, and A. Vijayaraghavan. Detecting high log-densities: an $O(n^{1/4})$ approximation for densest k-subgraph. In *Proc. 42nd ACM Symp. on Theory of Computing*, pages 201–210, 2010.

[BCV+12] A. Bhaskara, M. Charikar, A. Vijayaraghavan, V. Guruswami, and Y. Zhou. Polynomial integrality gaps for strong SDP relaxations of densest k-subgraph. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 388–405, 2012.

[BGLR93] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximations. In *Proc. 25th ACM Symp. on Theory of Computing*, pages 294–304, 1993.

[CDK12] E. Chlamtáč, M. Dinitz, and R. Krauthgamer. Everywhere-sparse spanners via dense subgraphs. In *Proc. 53rd IEEE Symp. on Foundations of Computer Science*, pages 758–767, 2012.

[CDM17] E. Chlamtáč, M. Dinitz, and Y. Makarychev. Minimizing the union: Tight approximations for small set bipartite vertex expansion. In *Proceedings of the 28th annual ACM-SIAM Symposium on Discrete Algorithms*, 2017.

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

[CK09] J. Chuzhoy and S. Khanna. Polynomial flow-cut gaps and hardness of directed cut problems. *Journal of the ACM*, 56(2), 2009.

[Fei02] U. Feige. Relations between average case complexity and approximation complexity. In *Proc. 34th ACM Symp. on Theory of Computing*, pages 534–543, 2002.

[Gri01] D. Grigoriev. Linear lower bound on degrees of Positivstellensatz calculus proofs for the parity. *Theoret. Comput. Sci.*, 259(1-2):613–622, 2001.

[HK04] J. Holmerin and S. Khot. A new PCP outer verifier with applications to homogeneous linear equations and max-bisection. In *Proc. 36th ACM Symp. on Theory of Computing*, pages 11–20, 2004.

[Kho02] S. Khot. On the power of unique 2-prover 1-round games. In *Proc. 34th ACM Symp. on Theory of Computing*, pages 767–775, 2002.

[Kho10] S. Khot. Inapproximability results for computational problems on lattices. In *The LLL Algorithm*, pages 453–473. 2010.

[LRS15] J. R. Lee, P. Raghavendra, and D. Steurer. Lower bounds on the size of semidefinite programming relaxations. In *Proc. 47th ACM Symp. on Theory of Computing*, pages 567–576, 2015.

[Man15] P. Manurangsi. On approximating projection games. Master's thesis, Massachusetts Institute of Technology, January 2015.

[MM13] P. Manurangsi and D. Moshkovitz. Improved approximation algorithms for projection games. In *Algorithms – ESA 2013: 21st Annual European Symposium Proceedings*, pages 683–694. Springer, 2013.

[Mos15] D. Moshkovitz. The projection games conjecture and the NP-hardness of $\ln n$-approximating set-cover. *Theory of Computing*, 11(7):221–235, 2015.

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

[MZK+99] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky. Determining computational complexity from characteristic "phase transitions". *Nature*, 400:133–137, 1999.

[Rag08] P. Raghavendra. Optimal algorithms and inapproximability results for every csp? In *Proc. 40th ACM Symp. on Theory of Computing*, pages 245–254, 2008.

[Sch08] G. Schoenebeck. Linear level lasserre lower bounds for certain k-CSPs. In *Proc. 49th IEEE Symp. on Foundations of Computer Science*, pages 593–602, 2008.

[Tul09] M. Tulsiani. CSP gaps and reductions in the lasserre hierarchy. In *Proc. 41st ACM Symp. on Theory of Computing*, pages 303–312, 2009.

[Vij12] A. Vijayaraghavan. *Beyond worst-case analysis in approximation algorithms.* PhD thesis, Ph. D. thesis, Princeton University, 2012.