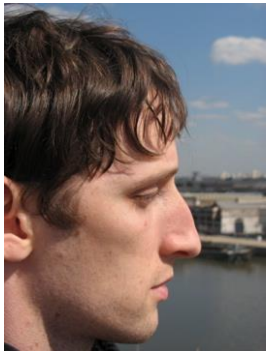
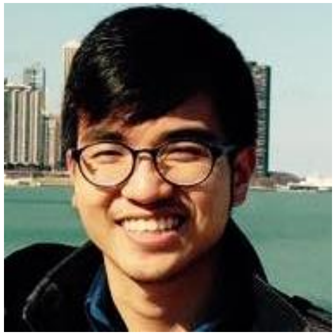
Approximation Algorithms for Label Cover and the Log-Density Threshold

Aravindan Vijayaraghavan

Northwestern University

Based on joint works with





Eden Chlamtac

Ben-Gurion University

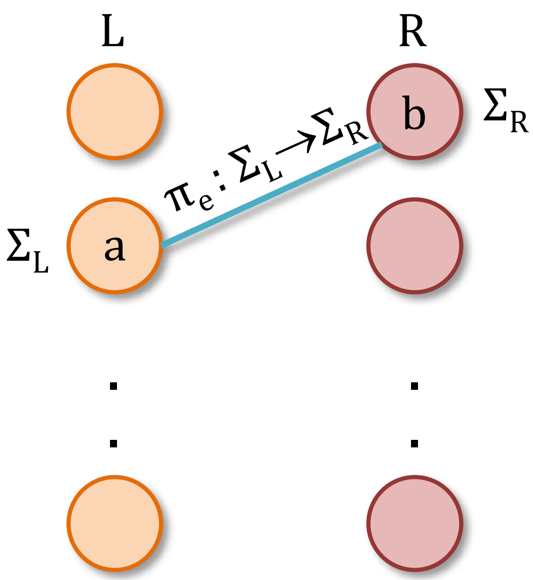
Pasin Manurangsi

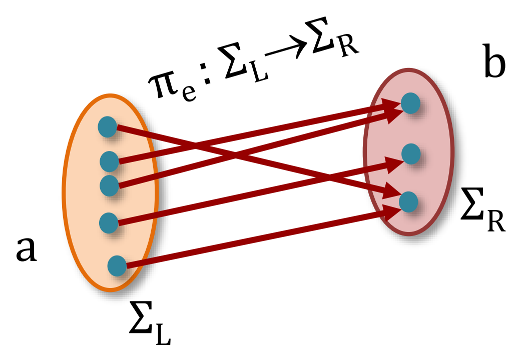
UC Berkeley

Dana Moshkovitz

UT Austin

Label Cover





Input

- A bipartite graph $G = (L, R, E)$ ***constraint graph***
- ***Alphabet sets*** Σ_L, Σ_R
- ***Projections*** $\pi_e : \Sigma_L \to \Sigma_R$ for each $e \in E$

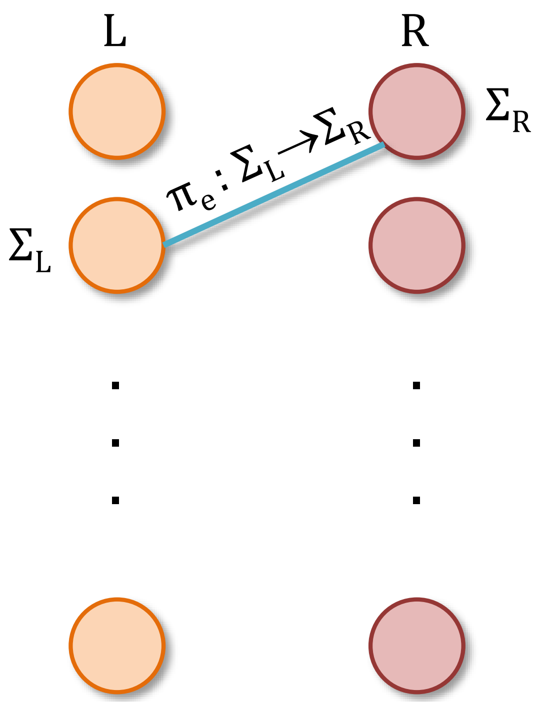
Goal

- Find ***assignments*** $\phi_L : L \to \Sigma_L$ and $\phi_R : R \to \Sigma_R$ maximizing the number of edges $e = (a, b)$ s.t.

$$\pi_e(\phi_L(a)) = \phi_R(b).$$

- ***Value*** – fraction of edges satisfied.

Some Notation



- **n** = number of vertices in the graph, i.e., **$n = |L| + |R|$**
- **k** = the size of left alphabet set, i.e., **$k = |\Sigma_L| \geq |\Sigma_R|$**
- **N** = the size of instance, i.e., **$N = nk$**
- **δ** = the approximation ratio

Why is Label Cover Important?

PCP Theorem (*[AS98][ALMSS98]…*)
Given a Label Cover instance, it is NP-Hard to distinguish between the following two cases:

- The instance is satisfiable, i.e., its value is 1.
- Its value is at most δ.

Max-3SAT is NP-Hard to approximate.
[Hastad97]
[BGS95]

Set Cover is NP-Hard to approximate.
[Lund-Yannakakis94]
…
[Feige98]
[Moshkovitz12]
[Moshkovitz-Raz08]+
[Dinur-Steurer13]

Closest Vector Problem is NP-Hard to approximate.
[Khot10]

Directed Sparsest Cut is NP-Hard to approximate.
[Chuzhoy-Khanna09]

…

Label Cover: what do we know?

Approximation factor δ : given a satisfiable instance of Label Cover, how many constraints does the algorithm satisfy?

Hardness Results

(Note: $\mathbf{N = nk}$)

[Arora-Safra98, ALMSS98]	NP-hard	Some constant $\mathbf{0 < \delta < 1}$
[Raz98]	NP-hard	Every constant $\mathbf{0 < \delta}$
[Moshkovitz-Raz08]	NP-hard	$\boldsymbol{\delta = 1/\log^c N}$ for some $\mathbf{c > 0}$
[Dinur-Steurer13]	NP-hard	$\boldsymbol{\delta = 1/\log^c N}$ for every $\mathbf{c > 0}$
[Dinur07] + [Raz98]	ETH-hard	$\boldsymbol{\delta = 1/N^{1/poly \log\log\log N}}$
Projection Games Conjecture *[BGLR93, Moskovitz15]*	NP-hard	$\boldsymbol{\delta = 1/N^c}$ for some $\mathbf{c > 0}$

Q: What's the right **c**?

Approximation Algorithms

Folklore	$\boldsymbol{\delta \geq 1/n, 1/k}$
[Peleg02]	$\boldsymbol{\delta \geq 1/N^{1/2}}$
[Charikar-Hajiaghayi-Karloff09]	$\boldsymbol{\delta \geq 1/N^{1/3}}$
[M-Moshkovitz13]	$\boldsymbol{\delta \geq 1/N^{1/4}}$

Better Inapproximability from LC?

PCP Theorem (*[AS98][ALMSS98]…*)
Given a Label Cover instance, it is NP-Hard to distinguish between the following two cases:

- The instance is satisfiable, i.e., its value is 1.
- Its value is at most δ.

$\delta = 1/N^c$

Max-3SAT is NP-Hard to approximate. *[Hastad97] [BGS95]*

Set Cover is NP-Hard to approximate. *[Lund-Yannakakis94]* … *[Feige98] [Moshkovitz12] [Moshkovitz-Raz08]+ [Dinur-Steurer13]*

Closest Vector Problem is NP-Hard to approximate. *[Khot10]*

Directed Sparsest Cut is NP-Hard to approximate. *[Chuzhoy-Khanna09]*

…

$\delta = 1/N^{c'}$

This Work

Conjecture $\mathbf{c = 3 - 2\sqrt{2} \approx 0.1716}$ (approximability threshold $\mathrm{N}^{-\mathrm{c}}$), using the Log-density method

Results:

- A matching $\boldsymbol{N^{-(3-2\sqrt{2})}}$–approximation algorithm for semi-random case where the constraint graph is random.
- An improved $\boldsymbol{N^{-0.2325}}$–approximation algorithm for worst case instances.
- A $\boldsymbol{N^{-1/8\,+O(\varepsilon)}}$ integrality gap for $\boldsymbol{N^{\varepsilon}}$–level Lasserre SDP (aka Sum-of-Square) relaxation of Label Cover.

Log-Density Method

- Introduced while studying Densest k-subgraph by Bhaskara-Charikar-Chlamtac-Feige-V [2010]

- Study how "simple local" algorithms perform on **average-case instances**

- Use the insight to come up with an algorithm and/or lower bounds for more general (worst-case) instances

- Results in state-of-the-art algorithms (and some lower bounds) for Densest-k-Subgraph[BCCFV'10], Degree-bounded Spanners[CDK'12], Small-Set Vertex Expansion [CDM'17]…

Log-Density Method: Outline

Random vs Planted	Consider a distinguishing problem between a “random” instance and one with a “planted” solution

Log-Density Method: Outline

Random vs Planted	Consider a distinguishing problem between a “random” instance and one with a “planted” solution
Counting Witnesses	Restrict ourselves to algorithms that only count a certain kind of “witnesses”

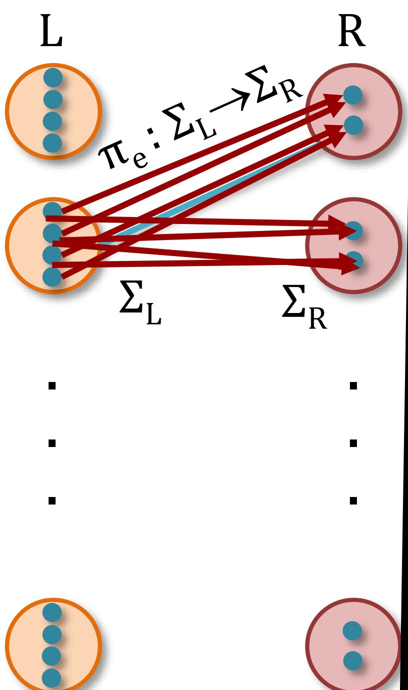
Log-Density Method: Outline

Random vs Planted	Consider a distinguishing problem between a "random" instance and one with a "planted" solution
Counting Witnesses	Restrict ourselves to algorithms that only count a certain kind of "witnesses"
Log Density Threshold	Compute the threshold at which witness counting algorithms stop/start working

Log-Density Method: Outline

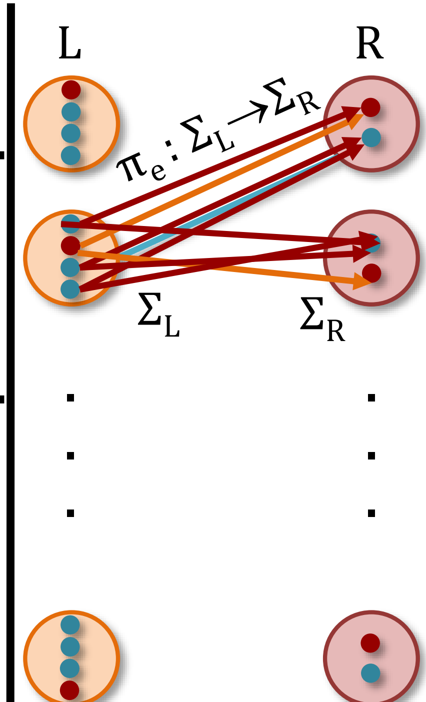
Random vs Planted	Consider a distinguishing problem between a “random” instance and one with a “planted” solution
Counting Witnesses	Restrict ourselves to algorithms that only count a certain kind of “witnesses”
Log Density Threshold	Compute the threshold at which witness counting algorithms stop/start working
Algorithm	Use the ideas from previous steps to come up with an algorithm

Distinguish Random vs Planted



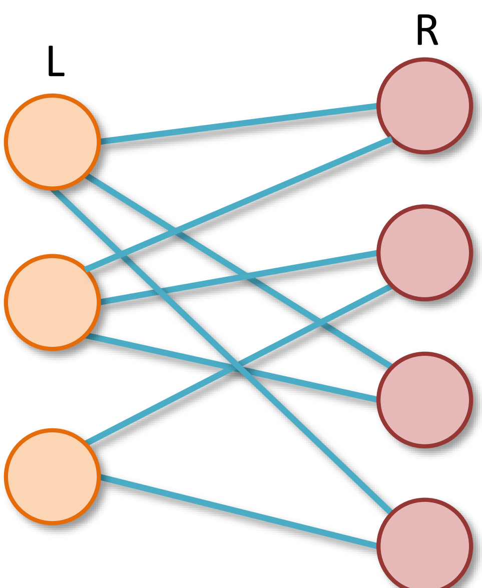
<table>
<tr><th>Random</th><th>Planted</th></tr>
<tr><td colspan="2">Constraint graph G = (L, R, E) is an Erdős-Rényi random bipartite graph G(n/2, n/2, p = Δ/n).</td></tr>
<tr><td>Each π_e is a random d-to-1 function</td><td>Fix a solution φ_L^*, φ_R^*

Each $\pi_{(a,b)}$ is a random d-to-1 function s.t. $\pi_{(a,b)}\left(\varphi_L^*(a)\right) = \varphi_R^*(b)$</td></tr>
</table>



Witness-Based Algorithm

Witness: a constant-size graph **W**

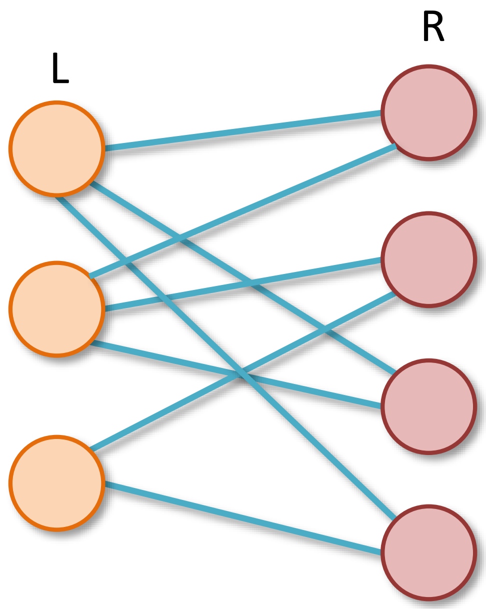


Distinguishing Algorithm

- Find each occurrence **H** of **W** in the constraint graph **G(L,R,E)**
  - Check whether there is a satisfying assignment of **H**
- If there are satisfying assignments for each **H**, output "**PLANTED**".
- Otherwise, output "**RANDOM**".

The Log-density Threshold

Witness: a constant-size graph **W**



When does the algorithm work?

1. **W** must appear (w.h.p.) in the constraint graph **G = (L, R, E)**
2. There must be no satisfying assignment for **W** (w.h.p.) for **RANDOM** instances (random projections)

The Log-density Threshold

Witness exists exactly when:

$$\boxed{2\underbrace{\log_n \Delta}_{\text{Log-density of the constraint graph}} > \overbrace{\log_k d}^{\text{Log-density of the projection}}}$$

Log-density of the **projection**

Log-density of the **constraint graph**

Δ: degree, **n**: # of vertices
k: alphabet size, **d**: # of preimages

When does the algorithm work?

1. **W** must appear (w.h.p.) in the constraint graph **G = (L, R, E)**
2. There must be no satisfying assignment for **W** (w.h.p.) for random instances

Towards an Approximation Algorithm

Random Planted Model

Step 1: Constraint graph **G = (L, R, E)** is random.
An Erdős-Rényi random bipartite graph **G(n/2, n/2, p = Δ/n)**.

Step 2: Fix a planted solution/ assignment φ_L , φ_R.
Each projection $\pi_{(a,b)}$ is a random d-to-1 function s.t.

$$\pi_{(a,b)}(\varphi_L(a)) = \varphi_R(b)$$

Goal: Find an assignment that satisfies as many edges as possible

Note: easier than semi-random model

Algorithm for Planted Model

<u>**Case 1 (Lower Log-density of G):**</u> $\mathbf{2 \log_n \Delta \leq \log_k d}$

Pick the best of the following:

- **d/k**-approx: pick a random assignment
- **1/Δ**-approx: satisfy a spanning tree

$N^{-(3-2\sqrt{2})}$– approximation algorithm

- Hard for local witnesses. Output something trivial.

- Better approx. guarantee would solve distinguishing problem in this regime as value of a random instance is at most **O(max{d/k, 1/Δ})**

Algorithm for Planted Model

Case 2 (Higher Log-density of G): $\mathbf{2\log_n \Delta > \log_k d}$

$\mathbf{2\log_n \Delta > \log_k d}$?

NO

Output the best of the following:
- **d/k**-approx: pick a random assignment
- **1/Δ**-approx: satisfy a single spanning

YES

ALPHABET-REDUCTION

If $\mathbf{2\log_n \Delta \geq \log_k d} + \epsilon$, return a reduced alphabet $\mathbf{\Sigma_u}$ for each $\mathbf{u \in L \cup R}$ of size much less than $\mathbf{\Sigma}$ $(\lesssim |\Sigma|^{1-\epsilon})$.

Alphabet Reduction Algorithm

Algorithm works in multiple steps and maintains

- a candidate set **S,** and
- a candidate set of labels $\boldsymbol{\Sigma_u}$ for each vertex $\boldsymbol{u \in S}$

First step:
Enumerate all possible choices of a left vertex $\mathbf{v}$ and a label $\boldsymbol{\sigma_v}$

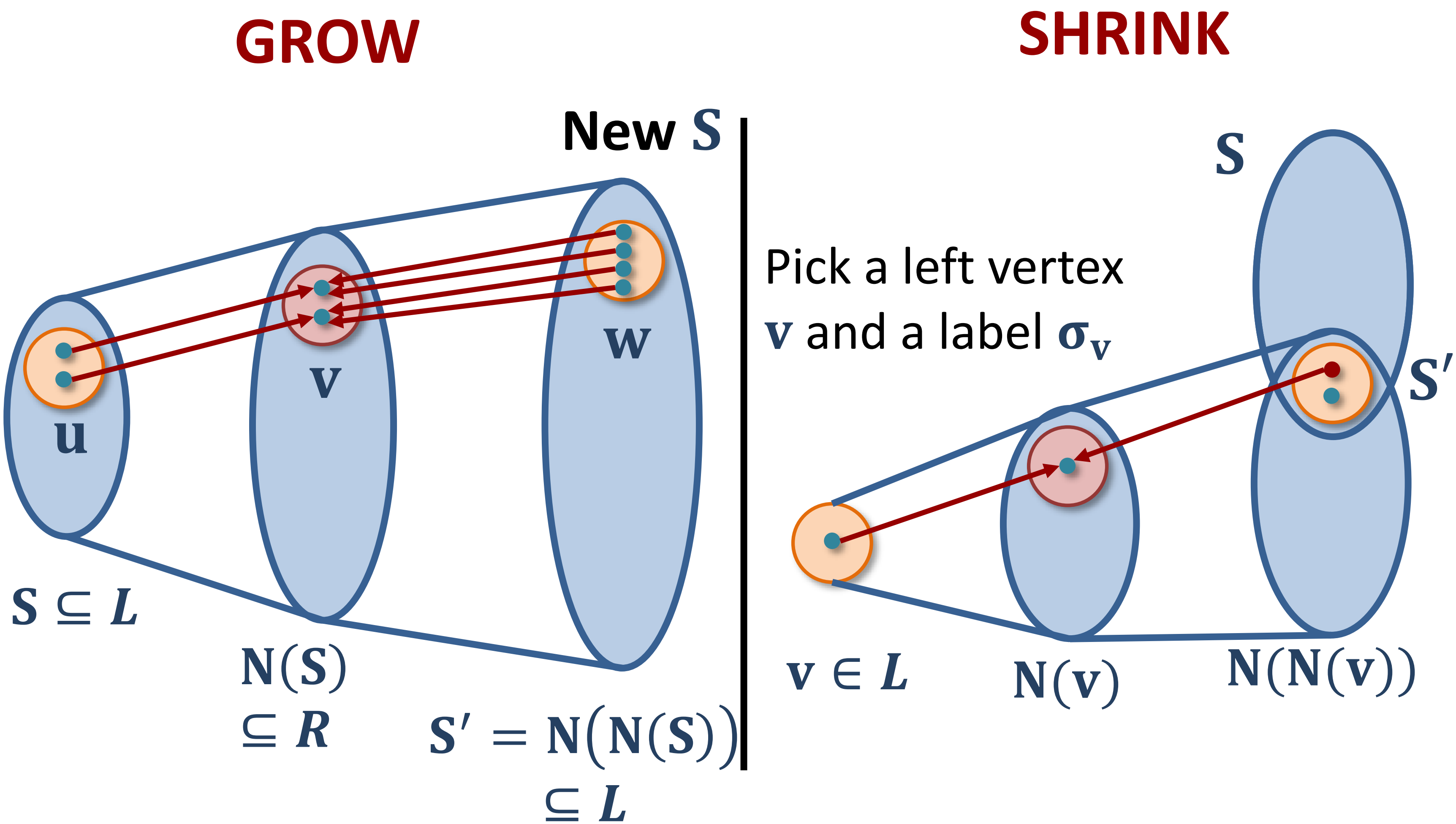
GROW
$\boldsymbol{S}$ larger
$\boldsymbol{\Sigma_u}$ larger

SHRINK
$\boldsymbol{S}$ smaller
$\boldsymbol{\Sigma_u}$ smaller

$$\begin{array}{|c|}\hline \mathbf{S} = \{\mathbf{v}\} \\ \boldsymbol{\Sigma_v} = \{\boldsymbol{\sigma_v}\} \\ \hline \end{array} \rightarrow \boxed{\textbf{GROW}} \rightarrow \boxed{\textbf{GROW}} \rightarrow \boxed{\textbf{SHRINK}} \ \dots \rightarrow \begin{array}{|c|}\hline \mathbf{S} \approx \mathbf{L} \\ |\boldsymbol{\Sigma_u}| \ll \boldsymbol{k} \\ \hline \end{array}$$

The sequence of Growth and Shrink steps given by the local witness

Alphabet Reduction Algorithm



Algorithm for Planted Model

$\mathbf{S} = \{\mathbf{v}\}$, $\boldsymbol{\Sigma_v} = \{\boldsymbol{\sigma_v}\}$ → **GROW** → **GROW** → **SHRINK** … → $\mathbf{S} \approx \mathbf{L}$, $|\boldsymbol{\Sigma}_{\boldsymbol{u}}| \ll \boldsymbol{k}$

ALPHABET-REDUCTION: If $\mathbf{2\log_n \Delta \geq \log_k d} + \epsilon$, we can find in polynomial time, an instance on G(V,E) with alphabets $\boldsymbol{\Sigma'}_{\mathbf{L}}$, $\boldsymbol{\Sigma'}_{\mathbf{R}}$ that is satisfiable and has $|\boldsymbol{\Sigma'}_{\mathbf{L}}| \lesssim |\Sigma_{\mathrm{L}}|^{1-\epsilon}$.

Implies a $\boldsymbol{N}^{-(\mathbf{3}-\mathbf{2}\sqrt{\mathbf{2}})}$–approximation algorithm for planted random instances.

Semi-random Models

Semi-Random Model 1 (easy)

Step 1: Constraint graph **G = (L, R, E)** is ~~random~~ *arbitrary*
An Erdős-Rényi random bipartite graph **G(n/2, n/2, p = Δ/n)**.

Step 2: Fix φ_L, φ_R. Each projection $\pi_{(a,b)}$ is a *random* d-to-1 function s.t. $\pi_{(a,b)}\big(\varphi_L(a)\big) = \varphi_R(b)$

Semi-Random Model 2 (more challenging)

Step 1: Constraint graph **G = (L, R, E)** is random
An Erdős-Rényi random bipartite graph **G(n/2, n/2, p = Δ/n)**.

Step 2: Fix φ_L, φ_R. Each projection $\pi_{(a,b)}$ is a ~~random~~ arbitrary d-to-1 function s.t. $\pi_{(a,b)}\big(\varphi_L(a)\big) = \varphi_R(b)$

Semi-random models

Semi-Random model 2:

Step 1: Constraint graph **G = (L, R, E)** is random
An Erdős-Rényi random bipartite graph **G(n/2, n/2, p = Δ/n)**.

Step 2: Fix φ_L, φ_R. Each projection $\pi_{(a,b)}$ is a ~~random~~ arbitrary d-to-1 function s.t. $\pi_{(a,b)}\big(\varphi_L(a)\big) = \varphi_R(b)$

- The projections are not **d-to-1 or random**
  Solution: Bucketing, and ensuring approximate regularity

Problem: The alphabet reduction algorithm may not reduce size of compatible alphabet; **SHRINK** fails!

Main Idea: If **SHRINK** fails, we can find a good assignment to the whole instance (need "robust" vertex expansion of instance)

Worst-Case Instances

Step 1: Constraint graph **G = (L, R, E)** is ~~random~~ *arbitrary*
An Erdős-Rényi random bipartite graph **G(n/2, n/2, p = Δ/n)**.

Step 2: Fix $\boldsymbol{\varphi_L}$, $\boldsymbol{\varphi_R}$. Each projection $\boldsymbol{\pi_{(a,b)}}$ is a ~~random~~ *arbitrary* d-to-1 function s.t. $\boldsymbol{\pi_{(a,b)}\big(\varphi_L(a)\big) = \varphi_R(b)}$

Problem: The constraint graph may not be expanding. **GROW** fails!

Partial Solution: Partitioning the instance into subinstances

Takeaways

- Via **log-density method**, identify a barrier for algorithms, and conjecture a threshold at which Label Cover becomes hard.

- An algorithm for **semi-random** case that matches threshold
- An improved (but **not matching**) algorithm for **worst** case

- A polynomial (**not matching**) **Sum-of-Squares** lower bound

- Similar results for Max-CSPs with approximability threshold at $N^{-1/4}$

Log-density of the **projection**

$$\boxed{\underbrace{2\log_n \Delta}_{\text{Log-density of the constraint graph}} > \overbrace{\log_k d}^{\text{Log-density of the projection}}}$$

Log-density of the **constraint graph**

Δ: degree, **n**: # of vertices
k: alphabet size, **d**: # of preimages

Open Questions

Conjecture $\mathbf{c = 3 - 2\sqrt{2} \approx 0.1716}$ (approximability threshold $\mathrm{N}^{-\mathrm{c}}$), using the Log-density method

- A **matching** algorithm for **worst** case?
- Is our conjectured threshold correct even in average-case?
- Evidence for hardness using **Sum-of-Squares** lower bounds?
- A useful average-case hardness assumption?

Thank you!

Questions?