List Decoding with Double Samplers

October 13, 2019

Contents

1 Samplers and Direct Product Codes

1.1 Samplers

We will give a general overview of the algorithm of Dinur, Harsha, Kaufman, Navon, and Ta-Shma for list decoding direct product codes constructed using double samplers [2]. Before we can formulate the decoding problem and state their result, we need to define the notion of a double sampler, and there's no better way to start discussing double samplers than with the definition of plain old samplers.

Definition 1.1 (Sampler)**.** *Let $G = (V_1, V_0, E)$ be a bipartite graph with a distribution Π_1 on V_1. Define Π_0 as the distribution on V_0 obtained by sampling $v_1 \in V_1$ according to Π_1, then taking $v_0 \in V_0$ to be a random neighbor of v_1. We say that G is an (α, δ)-sampler if for any function $f : V_0 \to [0, 1]$,*

$$\Pr_{v \sim \Pi_1}\left[\left|\mathop{\mathbb{E}}_{u \sim v}[f(u)] - \mathop{\mathbb{E}}_{u \sim \Pi_0}[f(u)]\right| \geq \alpha\right] \leq \delta$$

Put another way, for all but δ fraction of vertices $v \in V_1$, the local expectation of f over the neighborhood of v differs from the global expectation of f over all of V_0 by at most α. We will generally think of samplers as inclusion graphs, where the vertices in V_1 are identified with their neighborhoods and the edges represent containment.

1.2 Codes on Samplers

Samplers can be used to amplify the minimum distance of an error-correcting code. Let G be a bipartite inclusion graph with $V_0 = [n]$ and $V_1 \subseteq \binom{[n]}{m_1}$, and let $C \subseteq \{0,1\}^n$ be a code. For every word $x \in C$, we can create a new word by placing the restriction $x|_S \in \{0,1\}^S$ on each vertex $S \in V_1$ and defining $Enc_G(x) = (x|_S)_{S \in V_1}$. Note that $Enc_G(x)$ is a string over the larger alphabet $\{0,1\}^{m_1}$ of length $|V_1|$. Performing this process on every $x \in C$, we obtain the direct product code $Enc_G(C) = \{Enc_G(x) \mid x \in C\}$. It is not difficult to see from the sampler definition above that if G is an (α, δ) sampler and C has minimum distance at least α, the minimum distance of the new code $Enc_G(C)$ is $1 - \delta$.

The direct product encoding $Enc_G(C)$ on a sampler lends itself to a simple decoding algorithm. Given a received word $(f_S)_{S \in V_1}$ on the V_1 layer, we want to find the word $x \in C$ that minimizes the distance between $Enc_G(x)$ and (f_S). We can do so by letting each bit y_i be determined by the majority vote of the sets containing the vertex $i \in [n]$, taking $y_i = \mathrm{Maj}_{S \ni i}(f_S(i))$. Then $x \in C$ is obtained by running the unique decoding algorithm for C on y. If G is a good sampler and C has a sufficiently large unique decoding radius, this process will recover x from error rates of almost $1/2$.

If we try to extend the majority decoding algorithm for $Enc_G(C)$ into the list decoding regime of error rates beyond $1/2$, we will very quickly run into issues. Once the error rate is greater than $1/2$, more than half of the sets $S \in V_1$ containing a typical vertex $i \in [n]$ will have errors, meaning there is no longer a guarantee that the majority will be correct most of the time. In order to break past the $1/2$ barrier, we need to employ graphs with much more structure than mere samplers.

2 Double Samplers and High-Dimensional Expanders

2.1 Double Samplers

To develop a list decoding algorithm for the direct product code, we extend the sampler by adding an additional level $V_2 \subseteq \binom{[n]}{m_2}$ to the graph with $m_2 > m_1$. Note that the original code still corresponds to the V_0 level, and encoding is still done on the V_1 level; the new V_2 level will only be used to assist with decoding. We require the tripartite inclusion graph $X = (V_2, V_1, V_0)$ to be a *double sampler* consisting of two layers of samplers plus a bonus locality property:

Definition 2.1 (Double sampler)**.** *Let $X = (V_2, V_1, V_0)$ be an inclusion graph with a distribution W on V_2. Define a joint distribution Π on $(v_2, v_1, v_0) \in V_2 \times V_1 \times V_0$ by sampling v_2 according to W, then letting v_1 be a random neighbor of v_2 and v_0 be a random neighbor of v_1. Let Π_i be the marginal distribution of Π on V_i for $i \in \{0,1,2\}$.*

The pair (X, W) is an $((\alpha_{2,1}, \delta_{2,1}), (\alpha_{1,0}, \delta_{1,0}), (\alpha_{local}, \delta_{local}))$ double sampler if

- *The bipartite graph between V_2 and V_1 with distribution Π_2 on V_2 is an $(\alpha_{2,1}, \delta_{2,1})$ sampler.*
- *The bipartite graph between V_1 and V_0 with distribution Π_1 on V_1 is an $(\alpha_{1,0}, \delta_{1,0})$ sampler.*

- *For each $T \in V_2$, let $X|_T$ be the inclusion graph between sets in V_1 contained in T and elements of V_0 contained in T. Each graph X_T with a uniform distribution on $\{S \in V_1 \mid S \subseteq T\}$ must be an $(\alpha_{local}, \delta_{local})$ sampler.*

For a code C on $\{0,1\}^{V_0}$, $Enc_X(C)$ will denote the direct sum encoding of C on the V_1 level of X. In broad terms, the advantage of using a double sampler to decode $Enc_X(C)$ is that it will allow us to first list decode the direct product code locally on all of the samplers $X|_T$, then stitch all of the local views together to obtain a list of words in $\{0,1\}^{V_0}$.

2.2 Constructing Double Samplers

Samplers with $|V_1| = O(n)$ are not too hard to come by—taking the elements of V_1 to be random subsets of $[n]$ of size m will result in a good sampler with high probability. To get explicit samplers, spectral expansion is sufficient to guarantee sampling properties.

Claim 2.2. *A bipartite weighted graph with second eigenvalue λ is an $(\alpha, \frac{\lambda^2}{\alpha^2})$ sampler.*

Proof. This essentially follows from a variation of the expander mixing lemma; see [2] for details. □

Finding a double sampler where V_1 and V_2 have linear size is a much taller order. Random graph models will no longer cut it; the only known construction of a double sampler is via high-dimensional expanders. Dinur and Kaufman [3] define the notion of a γ-HDX and describe its spectral properties. We can extract a double sampler from a γ-HDX by restricting to faces of three different dimensions, taking $V_2 = X(m_2 - 1)$, $V_1 = X(m_1 - 1)$, and $V_0 = X(0)$, each with the corresponding distributions from the high-dimensional expander. By choosing γ, m_1, and m_2 appropriately, we can obtain double samplers with arbitrarily small parameters.

Theorem 2.3. *For every $\alpha_{2,1}, \delta_{2,1}, \alpha_{1,0}, \delta_{1,0}, \alpha_{local}, \delta_{local} > 0$, there exist $D, m_1, m_2 \in \mathbb{N}$ and a family of explicit double samplers (X_n, W_n) for infinitely many n such that*

- *X_n is an inclusion graph (V_2, V_1, V_0) with $|V_0| = n$, $V_1 \subseteq \binom{V_0}{m_1}$, and $V_2 \subseteq \binom{V_0}{m_2}$, with distribution W_n over V_2.*
- *X_n is an $((\alpha_{2,1}, \delta_{2,1}), (\alpha_{1,0}, \delta_{1,0}), (\alpha_{local}, \delta_{local}))$ double sampler.*
- *$|V_1|, |V_2| \leq Dn$*

Proof. Let $V_2 = X(m_2 - 1)$, $V_1 = X(m_1 - 1)$, and $V_0 = X(0)$, where X is a γ-HDX of dimension $d \geq m_2 - 1$ and the parameters D, m_1, and m_2 will be determined later. The distribution W_n will be the distribution on V_2 from the high-dimensional expander. Dinur and Kaufman [3] offer the following bounds:

- The graph $G_{1,0}$ between V_1 and V_0 has $\lambda(G_{1,0})^2 \leq \frac{1}{m_1} + O(m_1\gamma)$.
- The graph $G_{2,1}$ between V_2 and V_1 has $\lambda(G_{2,1})^2 \leq \frac{m_1}{m_2} + O(m_1 m_2 \gamma)$.

From Claim 2.2, we know that a second eigenvalue of $\lambda < \alpha\sqrt{\delta}$ is enough to get an (α, δ) sampler. Combining this fact with the bounds above, we choose $m_1 > 2/\alpha_{1,0}^2\delta_{1,0}$, $m_2 = m_1/\alpha_{2,1}^2\delta_{2,1}$, and $\gamma < 1/m_1^2 m_2^2$ to ensure that $\lambda(G_{1,0})^2 < \alpha_{1,0}^2\delta_{1,0}$ and $\lambda(G_{2,1}) < \alpha_{2,1}^2\delta_{2,1}$. The local sampling property comes from the downward closure of the high-dimensional expander, which makes $X|_T$ a complete bipartite graph for all $T \in V_2$. (This also requires us to enforce $m_1 > 2/\alpha_{local}^2\delta_{local}$.) Finally, the bound on the size of V_1 and V_2 comes from the construction of the γ-HDX, with $D \leq \exp(m_2^{O(1)})$. □

The double sampler X comes equipped with distributions Π_0 and Π_1 on the V_0 and V_1 layers. The codes C and $Enc_X(C)$ that we place on these two layers of X have no weights on the entries of their codewords; we would need Π_0 and Π_1 to be uniform in order to have the vertices of X correspond to positions in these two codes. Fortunately, the distributions on the double sampler taken from a γ-HDX in Theorem 2.3 have some very helpful properties. It can be shown that X is *regular*, meaning Π_0 is uniform on V_0 and is uniform on each T when we condition on $\Pi_2 = T$. The distribution Π_1, while not uniform, has the property of being D-*flat*, meaning the probability it assigns to each vertex of V_1 can only be one of the D possible values $\frac{1}{R}, \frac{2}{R}, \ldots, \frac{D}{R}$. Thus we can make Π_1 uniform by duplicating each vertex in V_1 the appropriate number of times (at most D).

3 Main Theorem Overview

With double samplers in hand, we can now state the main theorem describing the decoding algorithm for the direct product construction.

Theorem 3.1. *Let $\varepsilon_0, \varepsilon > 0$. Let X be a double sampler with parameters $\alpha_{2,1} = \varepsilon_0^2 10^{-5}$, $\delta_{2,1} = \frac{\varepsilon_0}{200} c^{-8/\varepsilon}$, $\alpha_{local} = \frac{\varepsilon_0}{20} 10^{-8/\varepsilon}$, and $\delta_{local} = \frac{\varepsilon_0}{200} c^{-8/\varepsilon}$, where c is an absolute constant. Suppose $C \subseteq \{0,1\}^{|V_0|}$ is a code that can be efficiently decoded from up to ε_0 fraction of errors. Then the direct product encoding $Enc_X(C) \subseteq \Sigma^{|V_1|}$ with alphabet $\Sigma = \{0,1\}^{m_1}$ has a $\mathrm{poly}(|V_0|)$ time algorithm that returns a list of size $\leq \frac{8}{\varepsilon}$ of all codewords at distance $1 - \varepsilon$ from a received word in $\Sigma^{|V_1|}$.*

We will examine the choice of parameters more closely in Section 7. For now, we only point out that Theorem 2.3 can be used to construct a double sampler X with the stated parameters and $|V_1|, |V_2| \leq D|V_0|$ for some D which is doubly exponential in $1/\varepsilon$.

The list decoding problem can be stated as follows. The input is a received word $(f_S)_{S \in V_1} \in \Sigma^{V_1}$, and we have a guarantee that at least one function $g : V_0 \to \{0,1\}$ in the base code exists such that $\Pr_{S \in V_1}[f_S = g|_S] \geq \varepsilon$. Our task is to find all such functions g that agree with f on at least ε fraction of the sets in V_1.

The list decoding algorithm will begin by creating a local list $L_T \subseteq \{0,1\}^T$ for each vertex $T \in V_2$ containing every assignment with at least $\frac{\varepsilon}{2}$ agreement with f on the sets $\{S \in V_1 \mid S \subseteq T\}$. In Section 4, we will describe how to prune these down to lists of bounded size where every assignment with at least ε agreement on subsets of T is at least very close to something on the list.

The next task is to combine the local assignments for different T into a list of global assignments $g : V_0 \to \{0,1\}$. This will be done by constructing a new graph on V_2 where two sets T_1, T_2 are adjacent if they have an intersection large enough to contain at least one set $S \in V_1$. The local lists will be used to define a unique games instance on this graph where elements $\sigma \in L_{T_1}$ and $\sigma' \in L_{T_2}$ are matched in a constraint if they have a strong agreement on some $S \subseteq T_1 \cap T_2$. We will find a large subgraph of this graph which is an expander (Section 5) and then run an algorithm which outputs a list of essentially all approximate solutions to this unique games instance (Section 6). Finally, we obtain one word g for each unique games solution by choosing a random $T \in V_2$ containing each $i \in V_0$, then letting the bit $g(i)$ be determined by the local assignment $\sigma \in L_T$ chosen by the unique games solution. The unique decoding algorithm of C is then run on g to fix any accumulated errors.

4 Local List Decoding

Each vertex $T \in V_2$ induces a local view of the code $Enc_X(C)$ consisting of all $S \in V_1$ that are subsets of T. The nice thing about the local views is that each T has a constant size, so we can obtain a list of all assignments $\sigma \in \{0,1\}^T$ that exhibit some agreement with $(f_S)_{S \subseteq T}$ by brute force.

However, these raw lists of local assignments will not be sufficient for merging into global solutions. It is possible that a list will contain several local assignments that are very close to each other corresponding to the same global solution, and this will cause problems down the line when we try to match up solutions between different T to define a unique games instance. To get around this issue, we prune the lists to ensure that the entries have some distance between them.

Lemma 4.1. *Let $\rho = \frac{\varepsilon_0}{2} 10^{-8/\varepsilon}$. For every $T \in V_2$, there is a radius $r_T \in \{\rho 10^i \mid i \in \{0, \ldots, \lfloor \frac{8}{\varepsilon} \rfloor\}\}$ such that we can construct a list $L_T \subseteq \{0,1\}^T$ satisfying the following properties:*

- *(Small list size) $|L_T| \leq \frac{8}{\varepsilon}$*
- *(Covering) If $g : T \to \{0,1\}$ agrees with f on at least $\frac{\varepsilon}{2}$ fraction of sets $S \subseteq T$, then there exists a $\sigma \in L_T$ with $\Delta_T(\sigma, g) \leq \frac{r_T}{9}$.*
- *(Separation) For all $\sigma, \sigma' \in L_T$, $\Delta_T(\sigma, \sigma') \geq r_T$.*

where Δ_T denotes the fractional Hamming distance on T: $\Delta_T(\sigma, \sigma') = \frac{1}{|T|}|\{j \mid \sigma(j) \neq \sigma'(j)\}|$.

The pruning algorithm is as follows. We start by assigning $i = 0$, $r_0 = \rho$, and $L_0 = \{\sigma \in \{0,1\}^T \mid \Pr_{S \subseteq T}[f_S = \sigma|_S] \geq \frac{\varepsilon}{2}\}$. Then repeat:

1. Find a maximum set $L_{i+1} \subseteq L_i$ with a distance of at least r_i between all elements.
2. If $L_{i+1} = L_i$, terminate the algorithm and output $L_T = L_i$, $r_T = r_i$.
3. Let $r_{i+1} = 10 r_i$, $i = i + 1$ and loop.

The separation property of the list returned from this algorithm is obvious. It takes a little bit more work to prove the other two properties in Lemma 4.1.

Lemma 4.2. *The list returned from the pruning algorithm has size $|L_T| \leq \frac{8}{\varepsilon}$.*

Proof. Let $\ell_1 = |L_1|$. We will show that $\ell_1 \leq \frac{8}{\varepsilon}$. Fix $\sigma \neq \sigma' \in L_1$ and define $B = \{j \in T \mid \sigma_j \neq \sigma'_j\}$. After the first pruning step, we have

$$\Pr_{i \in T}[i \in B] = \Delta_T(\sigma, \sigma') \geq \rho = \frac{\varepsilon_0}{2} 10^{-8/\varepsilon} > \alpha_{local}$$

Let $A = \{S \in V_1 \mid S \subseteq T, f_S = \sigma|_S = \sigma'|_S\}$. Every $S \in A$ is disjoint from B by definition. The sampling properties of $X|_T$ ensure that

$$\Pr_{S \subseteq T}[f_S = \sigma|_S = \sigma'|_S] = \Pr_{S \subseteq T}[S \in A] \leq \delta_{local}$$

For every $\widehat{L_1} \subseteq L_1$ of size $\ell \leq \ell_1$, we can compute

$$\begin{aligned}
1 &\geq \Pr_{S \subseteq T}[\exists \sigma \in \widehat{L_1} \text{ such that } f_S = \sigma|_S] \\
&\geq \sum_{\sigma \in \widehat{L_1}} \Pr_{S \subseteq T}[f_S = \sigma|_S] - \sum_{\sigma \neq \sigma' \in \widehat{L_1}} \Pr_{S \subseteq T}[f_S = \sigma|_S = \sigma'|_S] \\
&\geq \ell \frac{\varepsilon}{2} - \ell^2 \frac{\delta_{local}}{2}
\end{aligned}$$

Rearranging, $\ell^2 \frac{\delta_{local}}{2} - \ell \frac{\varepsilon}{2} + 1 \geq 0$ for all $0 \leq \ell \leq \ell_1$. The polynomial $x^2 \frac{\delta_{local}}{2} - x \frac{\varepsilon}{2} + 1$ has two real roots, and since it takes positive values for $x \leq \ell_1$, both solutions must be greater than ℓ_1. Thus ℓ_1 is less than the smaller root of this polynomial:

$$\ell_1 \leq \frac{1}{\delta_{local}} \left(\frac{\varepsilon}{2} - \sqrt{\frac{\varepsilon^2}{4} - 2\delta_{local}} \right) = \frac{1}{\delta_{local}} \left(\frac{\varepsilon}{2} \right) \left(1 - \sqrt{1 - \frac{8}{\varepsilon^2} \delta_{local}} \right) \leq \frac{8}{\varepsilon}$$

□

Bounding the list size also ensures that the pruning algorithm will terminate after no more than $8/\varepsilon$ iterations. As the final step for the proof of Lemma 4.1, we verify the covering property.

Lemma 4.3. *If $g : T \to \{0,1\}$ agrees with f on at least $\frac{\varepsilon}{2}$ fraction of sets $S \subseteq T$, then there exists a $\sigma \in L_T$ with $\Delta_T(\sigma, g) \leq \frac{r_T}{9}$.*

Proof. Fix such a g, which is in the initial list L_0 by definition. Let $\sigma_0 = g$ and σ_i be the element of L_i closest to σ_{i-1}. Suppose the algorithm runs for K iterations, so that σ_K is in the final list L_T. At each step, σ_{i-1} is either equal to σ_i or is removed due to being too close to some other σ_i; either way, we have $\Delta_T(\sigma_{i-1}, \sigma_i) < r_{i-1}$. Hence

$$\Delta_T(\sigma_K, g) = \Delta_T(\sigma_0, \sigma_K) \leq \sum_{i=1}^{K} \Delta_T(\sigma_{i-1}, \sigma_i) \leq \sum_{i=1}^{K} r_{i-1} = r_K \sum_{i=1}^{K} 10^{i-K-1} \leq \frac{r_K}{9} = \frac{r_T}{9}$$

□

5 Constructing the Constraint Graph

Now that we have processed the local lists appropriately, the next order of business is to identify local assignments on different lists corresponding to the same global assignment. For each $T \in V_2$, we have a list L_T of assignments satisfying all three properties in Lemma 4.1 for some radius r_T. Pad each list to exactly $\ell = \lfloor 8/\varepsilon \rfloor$ entries by adding dummy strings if necessary.

We will match up solutions between lists by constructing a unique games instance on a new graph G_C with vertex set V_2. Place an edge between $T_1, T_2 \in V_2$ if there is an $S \in V_1$ with $S \subseteq T_1 \cap T_2$, and assign edge weights according to the distribution obtained by taking a random $S \sim \Pi_1$, then independently sampling T_1 and T_2 uniformly from the set of elements of V_2 containing S. The resulting graph G_C is known as the *two-step walk graph* for the sampler between V_1 and V_2.

The set of labels at each vertex T is L_T, the idea being that a solution to the unique games instance will give us a mostly consistent choice of one assignment for each T. The constraint π at an edge $\{T_1, T_2\}$ will be constructed as follows:

1. Let $\mathcal{S}, \mathcal{T}_1, \mathcal{T}_2$ be the random variables defined by sampling $\mathcal{S} \sim \Pi_1$, then letting $\mathcal{T}_1$, $\mathcal{T}_2$ be random elements of V_2 containing $\mathcal{S}$. Choose $S \subseteq T_1 \cap T_2$ according to the distribution $(S \mid \mathcal{T}_1 = T_1, \mathcal{T}_2 = T_2)$.
2. For each $\sigma \in L_{T_1}$, set $\pi(\sigma) = \sigma'$ if there is an unmatched $\sigma' \in L_{T_2}$ with $\Delta_S(\sigma, \sigma') \leq \frac{r_{T_1}}{2}$.
3. Match all remaining $\sigma \in L_{T_1}$ arbitrarily.

Ideally, labels matched by the constraints will accurately reflect the idea that these two local assignments came from the same global assignment. This is formalized in the definition of a constraint being "correct" with respect to a global assignment. For the rest of this section, fix such an assignment $g : V_0 \to \{0, 1\}$ with $\Pr_S[f_S = g|_S] \geq \varepsilon$.

Definition 5.1 (Correct constraint)**.** *For $T \in V_2$, let $C(T)$ indicate the entry of L_T closest to $g|_T$ (or an arbitrary one of these entries in the case of a tie). The constraint π is* correct *with respect to g on the edge $\{T_1, T_2\}$ if $\Delta_{T_1}(C(T_1), g) \leq \frac{r_{T_1}}{9}$, $\Delta_{T_2}(C(T_2), g) \leq \frac{r_{T_2}}{9}$, and $\pi(C(T_1)) = T_2$.*

We seek a large subgraph of G_C where almost all of the edges have correct constraints, so that solving the unique games instance on this subgraph actually will give us a mostly consistent choice of local assignments.

Lemma 5.2. *For $i \in [\ell]$, let $V_C^{(i)} = \{T \in V_2 \mid r_T = \rho 10^i\}$ and $G_C^{(i)}$ be the subgraph of G_C induced by $V_C^{(i)}$. With high probability over the choices of sets S used to create the constraints, there exists an $i \in [\ell]$ such that $\Pr_{T \sim \Pi_2}[T \in V_C^{(i)}] \geq \frac{\varepsilon}{16} = \frac{1}{2\ell}$ and almost all edges of $G_C^{(i)}$ have correct constraints:*

$$\Pr_{T_1, T_2}[\pi_{\{T_1,T_2\}} \text{ is correct} \mid \{T_1, T_2\} \in E(G_C^{(i)})] \geq 1 - 10\delta_{1,2} - 10\ell\delta_{local}$$

In order to prove Lemma 5.2, we'll need a few more definitions.

Definition 5.3. *A vertex $T \in V_2$ is* good *if there is a $\sigma \in L_T$ with $\Delta_T(\sigma, g) \leq \frac{r_T}{9}$.*

Definition 5.4. *A pair (S, T) with $S \in V_1$, $T \in V_2$, and $S \subseteq T$ is* good *if T is a good vertex, $\Delta_S(C(T), g) \leq \frac{r_T}{4}$, and $\Delta_S(\sigma, g) \geq \frac{3r_T}{4}$ for all $\sigma \in L_T$ other than $C(T)$.*

Good pairs (S, T_1) and (S, T_2) will ensure that the our process of choosing the constraints will produce a correct constraint on the edge between T_1 and T_2.

Lemma 5.5. *Let $T_1, T_2 \in V_2$ with $r_{T_1} = r_{T_2}$, and suppose $S \subseteq T_1 \cap T_2$ is the subset used to create the constraint π on the edge $\{T_1, T_2\}$. If (S, T_1) and (S, T_2) are both good, then π is correct.*

Proof. Set $r = r_{T_1} = r_{T_2}$, $\sigma_1 = C(T_1)$, and $\sigma_2 = C(T_2)$. We know that T_1 and T_2 are both good, so $\Delta_{T_1}(\sigma_1, g) \leq \frac{r}{9}$ and $\Delta_{T_2}(\sigma_2, g) \leq \frac{r}{9}$. To finish showing that π is correct, we need only show that the algorithm used to create π will match σ_1 with σ_2. The pairs (S, T_1) and (S, T_2) being good tells us that $\Delta_S(\sigma_1, g)$ and $\Delta_S(\sigma_2, g)$ are both less than or equal to $\frac{r}{4}$. By the triangle inequality,

$$\Delta_S(\sigma_1, \sigma_2) \leq \Delta_S(\sigma_1, g) + \Delta_S(\sigma_2, g) \leq \frac{r}{4} + \frac{r}{4} = \frac{r}{2}$$

so σ_1 will be matched with σ_2 if it is available. For any $\sigma' \neq \sigma_1$ on the list L_{T_1}, we have $\Delta_S(\sigma', g) > \frac{3r}{4}$ from (S, T_1) being a good pair. Then

$$\Delta_S(\sigma', \sigma_2) \geq \Delta_S(\sigma', g) - \Delta_S(\sigma_2, g) > \frac{3r}{4} - \frac{r}{4} = \frac{r}{2}$$

so no other $\sigma' \in L_{T_1}$ will be matched with σ_2. A symmetric argument shows that σ_1 can't be matched with any other element of L_{T_2}, so we do indeed end up with $\pi(\sigma_1) = \sigma_2$. □

The task of obtaining the large subgraph of G_C with correct constraints in Lemma 5.2 is thus reduced to showing that most pairs (S, T) are good, which is where we will next turn our attention. This will come from the sampling properties of X.

Claim 5.6. $\Pr_{T\sim\Pi_2}[T \text{ is good}] \geq 1 - \delta_{1,2}$

Proof. Let $B = \{S \in V_1 \mid f_S = g|_S\}$ and $A = \{T \in V_2 \mid \Pr_{S\sim\Pi_1|S\subseteq T}[S \in B] < \frac{\varepsilon}{2}\}$. By definition of g we have $\Pr_{S\sim\Pi_1}[S \in B] \geq \varepsilon$. The parameters of X are chosen so that the graph between the V_1 and V_2 layers is an $(\frac{\varepsilon}{2}, \delta_{1,2})$ sampler, which ensures that $\Pr_{T\sim\Pi_2}[T \in A] \leq \delta_{1,2}$. If $T \notin A$, then there exists a $\sigma \in L_T$ such that $\Delta_T(\sigma, g) \leq \frac{r_T}{9}$ from the covering property of the list L_T (Lemma 4.1), which is the condition for T to be good. □

Claim 5.7. $\Pr_{(S,T)\sim(\Pi_1,\Pi_2)}[(S, T) \text{ is good}] \geq 1 - \delta_{1,2} - \ell\delta_{local}$

Proof. Fix a good vertex $T \in V_2$ and let $\sigma = C(T)$. We will be using the restriction $X|_T$, which is an $(\alpha_{local}, \delta_{local})$ sampler. Let $U = \{S \in V_1 \mid S \subseteq T\}$. Define the sets $B = \{i \in T \mid \sigma_i \neq g_i\}$ and $A = \{S \in U \mid \Pr_{i\in S}[i \in B] > \frac{r_T}{4}\}$. As T is good, $\Pr_{i\in T}[i \in B] \leq \frac{r_T}{9}$ (recalling that the distribution of i is uniform over T thanks to the regularity of X). Since $\alpha_{local} \leq \frac{\rho}{9} \leq \frac{r_T}{9}$, we have

$$\Pr_{S\subseteq T}\left[\Delta_S(\sigma, g) > \frac{r_T}{4}\right] = \Pr_{S\subseteq T}[S \in A] \leq \delta_{local}$$

which is the second condition for (S, T) to be good.

For the third condition, let $\sigma' \neq \sigma$ be some other element of L_T and define $B_{\sigma'} = \{i \in T \mid \sigma'_i \neq g(i)\}$ and $A_{\sigma'} = \{S \in U \mid \Pr_{i\in S}[i \in B_{\sigma'}] \leq \frac{3r_T}{4}\}$. By the separation property of L_T from Lemma 4.1, $\Delta_T(\sigma, \sigma') \geq r_T$, and hence

$$\Pr_{i\in T}[i \in B_{\sigma'}] = \Delta_T(\sigma', g) \geq \Delta_T(\sigma, \sigma') - \Delta_T(\sigma, g) \geq \frac{8r_T}{9}$$

Using the $(\frac{r_T}{9}, \delta_{local})$ sampling of $X|_T$ once again,

$$\Pr_{S\subseteq T}\left[\Delta_S(\sigma', g) \leq \frac{3r_T}{4}\right] = \Pr_{S\subseteq T}[S \in A_{\sigma'}] \leq \delta_{local}$$

Overall, for (S, T) to be a good pair, we require that T be a good vertex, $S \notin A$, and $S \notin A_{\sigma'}$ for all $\sigma' \neq \sigma$. These events occur together with probability at least $1 - \delta_{1,2} - \ell\delta_{local}$ by Claim 5.6 and the union bound. □

We finally have all of the ingredients necessary to prove Lemma 5.2. Recall that $V_C^{(i)}$ is the set of vertices of G_C with list radius $r_T = \rho 10^i$.

Proof of Lemma 5.2. For each $i \in [\ell]$, let

$$p^{(i)} = \Pr_{(S,T)\sim(\Pi_1,\Pi_2)}[(S, T) \text{ is good} \mid T \in V_C^{(i)}]$$

Let $\eta = \delta_{1,2} + \ell\delta_{local}$. Claim 5.7 tells us

$$1 - \eta \leq \Pr_{(S,T)\sim(\Pi_1,\Pi_2)}[(S,T) \text{ is good}] = \sum_{i=1}^{\ell} \Pr_{T\sim\Pi_2}[T \in V_C^{(i)}]p^{(i)}$$

Splitting the sum between "high-weight" sets $V_C^{(i)}$ with weight at least $1/2\ell$ according to Π_2 and "low-weight" ones with weight less than $1/2\ell$,

$$1 - \eta \leq \sum_{i:\mu(V_C^{(i)})\geq 1/2\ell} \mu(V_C^{(i)})p^{(i)} + \sum_{i:\mu(V_C^{(i)})<1/2\ell} \mu(V_C^{(i)})p^{(i)}$$

The total weight of all low-weight $V_C^{(i)}$ is at most $\ell(1/2\ell) = 1/2$. In order for the (weighted) average value of $p^{(i)}$ over all i to be greater than $1-\eta$, the average value of $p^{(i)}$ over high-weight sets must be at least $1-2\eta$. Thus there is at least one i with $\mu(V_C^{(i)}) \geq 1/2\ell$ and $p^{(i)} \geq 1-2\eta$. Restricting to the grpah induced by $V_C^{(i)}$ for such an i, we have

$$\Pr_{(T_1,T_2,S)\sim(\mathcal{T}_1,\mathcal{T}_2,\mathcal{S})}[(S,T_1) \text{ and } (S,T_2) \text{ are both good} \mid T_1, T_2 \in V_C^{(i)}] \geq 1 - 4\eta$$

which is the same as the probability of π being a good constraint by Lemma 5.5. To pass from each constraint having a high probability of being correct to guaranteeing a large proportion of correct constraints, we can use the Hoeffding inequality to show that at least $1-8\eta$ constraints will be satisfied with probability at least $1 - e^{c'n}$ for some constant c'. □

In order to efficiently solve the unique games instance on the restriction $G_C^{(i)}$ of G_C to the set $V_C^{(i)}$, we need it to be an expander. Unfortunately this won't necessarily be the case, but we can use the following theorem to find a subgraph of $G_C^{(i)}$ that exhibits some expansion.

Theorem 5.8. *Let $\alpha, \beta, \delta \in (0,1)$ satisfy $\alpha, \delta < \frac{\beta^2}{100}$. Suppose G is an (α,δ) sampler with vertex sets V_1 and V_2, and let G' be the two-step walk graph of G. Let $A \subseteq V_2$ be a subset with $\mu_G(A) \geq \beta$. Then there exists a $B \subseteq A$ with $\mu_G(B) \geq \frac{\beta}{4}$ such that the induced graph of G' on B has second eigenvalue (in absolute value) $\lambda(G') \leq \frac{99}{100}$.*

The proof of this theorem can be found in Appendix A of [2]. For our purposes, we have $\beta = \frac{\varepsilon}{16}$, so we require $\sqrt{\alpha_{2,1}}$ and $\sqrt{\delta_{2,1}}$ to be less than $\frac{\varepsilon}{160}$. We obtain a subgraph G_i of $G_C^{(i)}$ with $\Pr_{T\sim\Pi_2}[T \in V(G_i)] \geq \frac{\varepsilon}{64}$ and $\lambda(G_i) \leq \frac{99}{100}$. This is the graph on which we will solve the unique games instance.

6 Solving the Unique Games Instance

The first technique for solving unique games on an expander in polynomial time was demonstrated by Arora et. al. in [1]. We will be using a modified version of the approximation algorithm of Makarychev and Makarychev [4], which given a unique games instance on a regular graph G with $1-\delta$ fraction of constraints satisfiable for $\delta \leq c'\lambda_G$, returns a solution with $1 - C\frac{\delta}{h_G}$ constraints satisfied, where λ_G is the second-smallest eigenvalue of the Laplacian of G and h_G is the edge expansion of G. The algorithm will be extended in two different ways: generalizing it to weighted, non-regular graphs and outputting a list representing all of the high-value assignments instead of just one.

Lemma 6.1. *Let $G = (V, E)$ be a graph with edge weights $W = \{w_e\}_{e \in E}$ and $\lambda(G) \leq 0.99$. Let $\{\pi_e\}_{e \in E}$ be the set of unique constraints over the edges, with ℓ labels. Then there exists a constant c and a polynomial time approximation algorithm that outputs a list of assignments $\{a^{(1)}, \ldots, a^{(t)}\}$, $a^{(i)} : V \to [\ell]$, such that for every assignment $a : V \to [\ell]$ satisfying at least $1 - \eta$ of the constraints, there exists an $a^{(i)}$ on the list with $\Pr_{v \in V}[a(v) = a^{(i)}(v)] \geq 1 - \eta c^{\ell}$.*

Extending the algorithm of [4] to weighted graphs takes quite a bit of work; see Appendix B of [2] for details. We will be focusing on how to generate the list of assignments, which will be accomplished with the following algorithm.

Start with $i = 1$ and $\pi_e^{(1)} = \pi_e$ for every edge $e \in E$. Then repeat:

1. Run the unique games algorithm from [4] on the graph G with constraints $\{\pi_e^{(i)}\}_{e \in E}$. Note that the algorithm will find an approximate solution if at least $1 - c'\lambda_G \geq 1 - 100c'$ of the constraints are satisfiable.
2. If this algorithm didn't find an approximate solution, terminate.
3. Otherwise, add the output $a^{(i)} : V \to [\ell - i + 1]$ to the list of solutions.
4. Update the constraints on $j = \ell - i + 1$ labels to $\{\pi_e^{(i+1)}\}_{e \in E}$ on $j - 1$ labels by removing the solution $a^{(i)}$. This is accomplished by reordering the labels at every vertex v so that $a(v) = j$. If π is a satisfied constraint ($\pi(j) = j$), the new constraint π' at that edge will be π restricted to $[j - 1]$. If π is unsatisfied, then $\pi(k_1) = j$ and $\pi(j) = k_2$ for some $k_1, k_2 \neq j$. Define the new constraint π' by making it identical to π except for $\pi'(k_1) = k_2$.
5. Increment i by 1 and loop.

The key to this algorithm's success is the observation that any two approximate solutions to the unique games instance must either be very similar to each other or very different.

Claim 6.2. *Let $a, a' : V \to [\ell]$ be two assignments satisfying $1 - \eta$ and $1 - \eta'$ of the constraints on the edges of G, respectively. Then either $\Pr_{v \in V}[a(v) = a'(v)] \geq 1 - 50(\eta + \eta')$ or $\Pr_{v \in V}[a(v) = a'(v)] \leq 50(\eta + \eta')$.*

Proof. Let $D = \{v \in V \mid a(v) \neq a'(v)\}$ be the set of vertices on which a and a' differ. Since the constraints on the edges are unique, if a and a' both satisfy an edge and agree at one endpoint of that edge, they must also agree on the other endpoint. Thus any edges between D and $V \setminus D$ must not be satisfied by a or a', so $\mu(E(D, V \setminus D)) \leq \eta + \eta'$. The second eigenvalue of G is at most $\frac{99}{100}$, so the Cheeger inequality implies the edge expansion of G is at least $\frac{1}{50}$. Then

$$\min\{\mu(D), \mu(V \setminus D)\} \leq 50\mu(E(D, V \setminus D)) \leq 50(\eta + \eta')$$

from which the claim follows. □

Let $a : V \to [\ell]$ be an assignment satisfying at least $1 - \eta$ of the constraints. To prove Lemma 6.1, we need to find an i such that $a^{(i)}$ is very close to a. Let

$$\eta_i = \Pr_{(u,v) \sim W}[a(u) \neq \pi_{uv}^{(i)}(a(v))]$$

be the fraction of constraints that are not satisfied by a during iteration i of the algorithm. Since solutions that aren't close have very little in common, we can show that removing every label chosen by a solution $a^{(i)}$ in step 4 of the algorithm won't affect a too much.

Claim 6.3. *There exists a constant $b > 1$ such that if $\eta < b^{-\ell}$ and $\Pr_{u\in V}[a(u) = a^{(i)}(u)] \le \frac{1}{2}$ for all $i \le j$, then $\eta_{j+1} \le \eta b^{j+1}$.*

Proof. Induct on i. For the base case, a satisfies $1-\eta$ of the original constraints $\{\pi_e^{(1)}\}_{e\in E}$ by definition. Assuming the claim holds for $j-1$, the unique games instance with constraints $\pi^{(j)}$ will have a solution satisfying $1-\eta_j$ of the constraints, where $\eta_j < \eta b^j$. Thus the algorithm will output an assignment $a^{(j)} : V \to [\ell - j + 1]$ satisfying at least $1 - 50C\eta_j$ of the constraints.

Since $\Pr_{u\in V}[a(u) = a^{(j)}(u)] \le \frac{1}{2}$, we know from Claim 6.2 that this probability must be bounded above by $50(\eta_j + 50C\eta_j)$. Observe that when the algorithm removes the labels selected by $a^{(j)}$, the constraints are not changed except for the parts involving $a^{(j)}(u)$ for a vertex $u \in V$. This means that any constraint π_{uv} satisfied by a will only be affected by this removal if $a(u) = a^{(j)}(u)$ or $a(v) = a^{(j)}(v)$. We can compute

$$\begin{aligned}
\eta_{j+1} &= \Pr_{(u,v)\sim W}[a(u) \neq \pi_{uv}^{(j+1)}(v)] \\
&\le \Pr_{(u,v)\sim W}[a \text{ does not satisfy } \pi_{uv}^{(j)}] + \Pr_{(u,v)\sim W}[a \text{ satisfies } \pi_u v^{(j)}, \text{but not } \pi_{uv}^{(j+1)}] \\
&\le \Pr_{(u,v)\sim W}[a(u) \neq \pi_{uv}^{(j)}(v)] + \Pr_{(u,v)\sim W}[a(u) = a^{(j)}(u) \text{ or } a(v) = a^{(j)}(v)] \\
&\le \eta_j + 50(\eta_j + 50C\eta_j) \\
&\le 3000C\eta b^j
\end{aligned}$$

so the claim holds for $b = \max\{3000C, \frac{1}{100c'}\}$. □

Proof of Lemma 6.1. Let $c = 2b$. The claim is trivial if $\eta \ge c^{-\ell}$, so we may assume $\eta < c^{-\ell}$. Assume for the sake of contradiction that $\Pr_{u\in V}[a(u) = a^{(i)}(u)] \le \frac{1}{2}$ for all $i \in [t]$, where t is the size of the list of solutions obtained from the algorithm.

If $t < \ell$, we have $\eta_{t+1} \le \eta b^t$ by Claim 6.3. The fraction of constraints $\{\pi_e^{t+1}\}_{e\in E}$ satisfied by a at iteration $t+1$ of the algorithm is

$$1 - \eta_{t+1} \ge 1 - \eta b^t \ge 1 - b^{t-\ell} \ge 1 - \frac{1}{b} \ge 1 - 100c'$$

so the unique games algorithm should have output a solution instead of terminating.

If $t = \ell$, every label at each vertex is used in exactly one assignment on the list; for every $u \in V$, there is an $i \in [\ell]$ with $a^{(i)}(u) = a(u)$. By Claim 6.3, we have $\eta_i \le \eta b^i$ for every $i \in [\ell]$. Since $\Pr_{u\in V}[a(u) = a^{(i)}(u)] \le \frac{1}{2}$, this probability is less than or equal to $50(\eta_i + 50C\eta_i) \le b\eta_i$ by Claim 6.2. Hence

$$\Pr_{u\in V}[a^{(i)}(u) = a(u) \text{ for some } i \in [\ell]] \le \sum_{i=1}^{\ell} b\eta_i \le \eta \sum_{i=1}^{\ell} b^{i+1} \le 2\eta b^\ell \le \eta c^\ell < 1$$

which contradicts all of the labels being used.

As we obtained a contradiction in both cases, there must be a $j \in [\ell]$ such that $\Pr_{u\in V}[a(u) = a^{(j)}(u)] > \frac{1}{2}$. By Claim 6.2,

$$\Pr_{u\in V}[a(u) = a^{(j)}(u)] \ge 1 - 50(\eta_j + 50C\eta_j) \ge 1 - b\eta_j \ge 1 - \eta c^\ell$$

as desired. □

7 Final Codeword Extraction

Applying the unique games algorithm of Lemma 6.1 to the expanding subgraph G_i and recalling that the labels correspond to entries in the lists L_T, we will receive a list of assignments $a : V(G_i) \to \{0,1\}^{m_2}$. As $\mu_{G_C}(V(G_i)) \geq \frac{1}{4}\mu_{G_C}(V_C^{(i)})$ by Theorem 5.8, the fraction of incorrect edges in G_i increases by at most a factor of 4 from $G_C^{(i)}$. Taking $\eta = 40(\delta_{1,2} + \ell\delta_{local})$ (the maximum fraction of incorrect edges) in Lemma 6.1 and using the definition of a correct edge, for any high-agreement global assignment $g : V_0 \to \{0,1\}$ there is an assignment a returned by the unique games algorithm satisfying

$$\Pr_{T\sim\Pi_2}[\Delta_T(a(T), g) \leq \frac{r_T}{9}] \leq 1 - c^\ell 40(\delta_{1,2} + \ell\delta_{local})$$

For each assignment a on the list of unique games solutions, determine the jth bit of $w \in \{0,1\}^{V_0}$ by choosing a random $T \sim \Pi_2$ conditioned on containing j and appearing as a vertex in G_i, then letting $w_j = a(T)_j$. For the global assignment g corresponding to a, our choice of parameters guarantees that

$$\Pr_{j\sim\Pi_0, T\sim\Pi_2|T\ni j}[w_j \neq g(j)] \leq \frac{r_T}{9} + c^\ell 40(\delta_{1,2} + \ell\delta_{local}) \leq \frac{\varepsilon_0}{2}$$

Therefore $\Delta(w, g) \leq \varepsilon_0$ with high probability, so running the unique decoding algorithm of the base code C on w will produce g. Repeating the entire process multiple times, the probability of success approaches 1. Decoding every entry on the list of unique games solutions in this way and taking the direct product encoding of each of the results to get back up to the V_1 level will produce the list promised in Theorem 3.1.

References

[1] Sanjeev Arora, Subhash A. Khot, Alexandra Kolla, David Steurer, Madhur Tulsiani, and Nisheeth K. Vishnoi. Unique games on expanding constraint graphs are easy: Extended abstract. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 21–28, New York, NY, USA, 2008. ACM.

[2] Irit Dinur, Prahladh Harsha, Tali Kaufman, Inbal Livni Navon, and Amnon Ta Shma. List decoding with double samplers. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, pages 2134–2153, Philadelphia, PA, USA, 2019. Society for Industrial and Applied Mathematics.

[3] Irit Dinur and Tali Kaufman. High dimensional expanders imply agreement expanders. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*, pages 974–985, 2017.

[4] Konstantin Makarychev and Yury Makarychev. How to play unique games on expanders. In *Proceedings of the 8th International Conference on Approximation and Online Algorithms*, WAOA'10, pages 190–200, Berlin, Heidelberg, 2011. Springer-Verlag.