Electronic Colloquium on Computational Complexity, Report No. 128 (2009)

Locally Testable Codes Analogues to the Unique Games Conjecture Do Not Exist[†]

Gillat Kol Ran Raz

Abstract

The *Unique Games Conjecture* (*UGC*) is possibly the most important open problem in the research of PCPs and hardness of approximation. The conjecture is a strengthening of the PCP Theorem, predicting the existence of a special type of PCP verifiers: 2-query verifiers that only make *unique tests*. Moreover, the UGC predicts that such PCP verifiers can have almost-perfect completeness and low-soundness.

The computational complexity notion of a PCP is closely related to the combinatorial notion of a *Locally Testable Code* (*LTC*). LTCs are error-correcting codes with codeword testers that only make a *constant* number of queries to the tested word. All known PCP constructions use LTCs as building blocks. Furthermore, to obtain PCPs with certain properties, one uses LTCs with corresponding properties.

In light of the strong connection between PCPs and LTCs, one may conjecture the existence of LTCs with properties similar to the ones required by the UGC. In this work we show that such LTCs do not exist. That is, we consider 2-query LTCs with codeword testers that only make unique tests. Roughly speaking, we show that any such LTC with relative distance close to 1, almost-perfect completeness and low-soundness, is of constant size.

The result intends to point out the limitations of the current techniques for constructing PCPs. It suggests that proving the UGC may require a new approach.

[†]Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Israel. Research supported by Binational Science Foundation (BSF) and Israel Science Foundation (ISF).

ISSN 1433-8092

1 Introduction

1.1 The Unique Games Conjecture

The discovery of the PCP theorem [4, 10, 2, 1] in 1992 has led to breakthrough results in the field of hardness of approximation. Yet, for many fundamental problems, optimal hardness results are still not known. To deal with such problems, a strengthening of the PCP Theorem, called the *Unique Games Conjecture* (UGC) was proposed by Subhash Khot in 2002 [13].

The PCP Theorem states that any statement of the form "$\varphi \in SAT$" has a proof that can be checked probabilistically by reading only a *constant* number of locations in the proof. Moreover, the check can be performed by an *efficient* probabilistic algorithm called a *verifier.*

We say that a verifier has *completeness* $1-\epsilon$, if for every $\varphi \in SAT$, there exists a proof that causes the verifier to accept with probability at least $1-\epsilon$. A verifier has *soundness* (*error*) s if for any $\varphi \notin SAT$, no matter which proof is provided, the verifier accepts with probability at most s. The PCP Theorem shows the existence of a verifier achieving completeness 1 ("perfect completeness") and constant soundness $s > 0$.

The UGC considers a special type of PCP verifiers: verifiers that read at most *two* locations in the proof, and only make *unique tests.* That is, given the value read from the first location, there is a unique value for the second location that makes the verifier accept, and vice versa. The conjecture states that for any arbitrarily small constants $\epsilon, s > 0$, there exists a constant size alphabet set Σ (whose size may depend on ϵ and s), satisfying the following: There exists a verifier with unique tests that checks proof strings over the alphabet Σ, and has completeness $1-\epsilon$ and soundness s. We mention that if the unique tests requirement is waived, the desired verifier is known to exist [22].

The UGC has been shown to imply optimal hardness results for Vertex Cover [15] and Max-Cut [14, 20]. In addition, the conjecture, or variants of it, has been used to obtain improved hardness results for other problems [13, 16, 7, 9]. Furthermore, for every constraint satisfaction problem, there exists an approximation algorithm (based on Semi-Definite Programming) that is known to be optimal under the UGC [21].

1.2 Locally Testable Codes

The computational complexity notion of a PCP is closely related to the combinatorial notion of a *Locally Testable Code* (*LTC*). LTCs are error-correcting codes that have local codeword testers. A *local codeword tester* is an efficient algorithm that probabilistically checks whether a given word belongs to the code, by only reading a *constant* number of locations in the word. We say that a tester has *completeness* $1-\epsilon$, if it accepts every codeword with probability at least $1-\epsilon$. The tester has *soundness* (*error*) s, if for any word that agrees with every codeword on at most a $\delta = \frac{2}{3}$-fraction of the coordinates, the tester accepts with probability at most s.

All known PCP constructions use LTCs as building blocks. For example, Reed-Muller, Hadamard, and the Long Code are codes that have been extensively used in PCP constructions. Furthermore, to obtain PCPs with certain properties, one uses LTCs with corresponding properties. For instance, low-error PCP constructions (i.e., PCPs with sub-constant soundness in the size of φ), use low-error LTCs [23, 3, 8, 17, 18, 19].

In the opposite direction, some PCP constructions were shown to imply LTCs [11]. Moreover, in the high-error (constant soundness) region, the existence of a special type of PCP, called *PCP of Proximity* (*PCPP*), is known to imply LTCs [5]. For further discussion of the relations between PCPs and LTCs, see [11].

1.2.1 Low-Error Locally Testable Codes

In light of the strong connection between PCPs and LTCs, in this work we study the UGC by examining LTCs with corresponding properties. Since proving the UGC requires the construction of PCPs with arbitrarily small constant soundness (error), we will be interested in LTCs with arbitrarily small constant soundness (error). Recall that the soundness requirement of an LTC (see Subsection 1.2) was that for any word that agrees with every codeword on at most a δ-fraction of the coordinates, the tester accepts with probability at most s. Note that if δ is a fixed constant (say, $\delta = \frac{2}{3}$), and the number of queries is a fixed constant, we cannot get s to be lower than some fixed constant. Hence, we will consider an arbitrarily small δ, and take s to be a function of δ.

We assume that the dependence of the acceptance probability s, on the agreement with the code δ, is given by an arbitrary weakly monotone increasing function $s : (0,1) \to [0,1]$. We say that the tester has a *soundness function* $s(\delta)$, if for any word that agrees with every codeword on at most a δ-fraction of the coordinates, the tester accepts with probability at most $s(\delta)$. We mention that, typically, low-error PCP constructions use codes with soundness function $s(\delta) = c_1\delta^{c_2}$ where c_1 is a large constant (say, 100), and c_2 is a small constant (say, $\frac{1}{100}$) (for non-negligible δ). See for example [23, 3, 8, 17, 18, 19].

We now give the formal definition of an LTC. We use the following notations. For a natural number $t \in \mathbb{N}$, denote $[t] \doteq \{1, ..., t\}$. Let Σ be a finite alphabet set, and let $n \in \mathbb{N}$ be a natural number. The agreement set of two words $u, w \in \Sigma^n$ is defined as $agree(u, w) \doteq \{i \in [n] \mid u_i = w_i\}$, and the distance between u and w is $\Delta(u, w) \doteq 1 - \frac{1}{n}|agree(u, w)|$. A subset $C \subseteq \Sigma^n$ is called a code. The relative distance of the code C is $\min_{u \neq w \in C}\{\Delta(u, w)\}$, and the distance of a word $v \in \Sigma^n$ from the code C is $\Delta(v, C) \doteq \min_{u \in C}\{\Delta(v, u)\}$.

Definition 1 (**Local Tester**, **LTC**, Generalized Version)**.** Let Σ be a finite alphabet set, $n \in \mathbb{N}$ be a natural number, and $C \subseteq \Sigma^n$ be a code. Let $\epsilon \in [0, 1)$ be a real number, and let $s : (0, 1) \to [0, 1]$ be a weakly monotone increasing function. Assume that T is a probabilistic, non-adaptive, oracle machine with access to a string $v \in \Sigma^n$. In addition, assume that T makes a constant number of queries to v, and outputs either `accept` or `reject`. Then, T is an (ϵ, s)-local tester for C if it satisfies the following conditions:

- **Completeness:** If $v \in C$, then $\Pr[T^v = \texttt{accept}] \geq 1 - \epsilon$.

- **Soundness:** For every $\delta \in (0, 1)$ the following holds. If $\Delta(v, C) > 1 - \delta$, then $\Pr[T^v = \texttt{accept}] \leq s(\delta)$. In other words, if $\Pr[T^v = \texttt{accept}] > s(\delta)$, then there exists a codeword u such that $|agree(v, u)| \geq \delta n$.

A code C is an (α, ϵ, s)-LTC if it has relative distance at least $1-\alpha$, and has an (ϵ, s)-local tester.

In known PCP constructions, the error of the verifier originates from the relative distance of the LTC used, as well as its completeness and soundness. Therefore, in order to get PCPs with error close to 0, we study (α, ϵ, s)-LTCs for which there exists $\delta \in (0, 1)$ such that α, ϵ, $s(\delta)$ are simultaneously close to 0.

1.3 Unique Locally Testable Codes

To match the requirements of the UGC, we define the notion of *unique LTCs*. Unique LTCs are LTCs with local testers that read at most *two* locations in the tested word, and only make *unique tests.* That is, given the value read from the first location, there is a unique value for the second location that makes the tester accept, and vice versa. Formally, an (α, ϵ, s)-unique LTC is an (α, ϵ, s)-LTC that only makes unique tests.

Definition 2 (Unique Local Tester, Unique LTC). Let Σ be a finite alphabet set, $n \in \mathbb{N}$ be a natural number, and $C \subseteq \Sigma^n$ be a code. Let $\epsilon \in [0,1)$ be a real number, and let $s : (0,1) \to [0,1]$ be a weakly monotone increasing function. Assume that T is a probabilistic, non-adaptive, oracle machine with access to a string $v \in \Sigma^n$. In addition, assume that T makes at most two queries to v, and outputs either `accept` or `reject`. Then, T is an (ϵ, s)-unique local tester for C if it satisfies the following conditions:

- **Uniqueness:** For every pair of indices $(i, j) \in [n]^2$ that may be queried by T in a single execution, there exists a permutation $\pi_{ij} : \Sigma \to \Sigma$ satisfying the following. After querying v_i and v_j, T outputs `accept` if and only if $v_j = \pi_{ij}(v_i)$.
- **Completeness:** If $v \in C$, then $\Pr\left[T^v = \texttt{accept}\right] \geq 1 - \epsilon$.
- **Soundness:** For every $\delta \in (0,1)$ the following holds. If $\Delta(v, C) > 1 - \delta$, then $\Pr\left[T^v = \texttt{accept}\right] \leq s(\delta)$. In other words, if $\Pr\left[T^v = \texttt{accept}\right] > s(\delta)$, then there exists a codeword u such that $|agree(v, u)| \geq \delta n$.

A code C is an (α, ϵ, s)-unique LTC if it has relative distance at least $1 - \alpha$, and has an (ϵ, s)- unique local tester.

1.4 Our Result

As discussed above, in known PCP constructions, the error of the verifier originates from the relative distance of the LTC used, as well as its completeness and soundness. Recall that the UGC predicts the existence of a PCP verifier with unique tests and arbitrarily small constant error. Therefore, in terms of codes, one may expect the existence of (α, ϵ, s)-unique LTCs for arbitrarily small α and ϵ, and s that attains arbitrarily small values. However, the following theorem shows that if s attains a value smaller than 10^{-5}, then there exists a constant δ' (that depends on s), such that the following holds. Every (α, ϵ, s)-unique LTC with both α and ϵ smaller than δ', is of constant size (where the constant size depends on $|\Sigma|$ and δ'). Specifically, $\delta' = 10^{-10}\delta^2$, where δ is as in the theorem below.

Theorem 3 (**Main**)**.** *Let $s : (0,1) \to [0,1]$ be a weakly monotone increasing function, attaining a value no larger than 10^{-5}. Let $\delta \in (0,1)$ be such that $s(\delta) \leq 10^{-5}$. Then, there exist sufficiently small constants $\alpha, \epsilon \in (0,1)$ (specifically, $\alpha = \epsilon \doteq 10^{-10}\delta^2$) such that for every finite alphabet set Σ, and every natural number $n \in \mathbb{N}$, the following holds. Every (α, ϵ, s)-unique locally testable code $C \subseteq \Sigma^n$ satisfies*

$$|C| \leq 10^3 \frac{|\Sigma|}{\sqrt{\alpha}} = 10^8 \frac{|\Sigma|}{\delta}.$$

The result intends to point out the limitations of the current techniques for constructing PCPs. It suggests that proving the UGC may require a new approach.

1.5 Previous Works

In [6] it is shown that *linear* 2-query LTCs with constant distance and alphabet size, must be of constant size (where linear means that the code is a linear subspace of the vector space $\mathbb{F}^n$, for some finite field $\mathbb{F}$). The same result is obtained for general (non-linear) LTCs, when *perfect completeness* and *binary* alphabet are assumed [6].

In [12], *binary* LTCs with almost-perfect completeness (say $1-\epsilon$, for ϵ approaching 0) are investigated. It is shown that such codes must either have a constant size, or have soundness s of at least $1 - O(\epsilon)$.

The results described above are incomparable to ours. On the one hand, to match the parameters of the UGC, our result considers the more general case of LTCs over *arbitrarily large alphabet set*s, with *almost-perfect completeness.* (We mention that the UGC with a small alphabet set, or with perfect completeness, is known to be false). On the other hand, we only consider LTCs with unique tests.

1.6 Discussion

1.6.1 Can the UGC Still be Proven Using Unique LTCs?

For a soundness function $s(\delta)$, we denote $\delta_s \doteq \sup_\delta \{\delta \in (0,1) \mid s(\delta) \leq 10^{-5}\}$. Our result shows that for any $n \in \mathbb{N}$, and any soundness function s, there is no (α, ϵ, s)-unique LTC $C \subseteq \Sigma^n$ with both α and ϵ smaller than $10^{-10}\delta_s^2$ (except if C is of constant size). However, it does not rule out the possibility of having, for every $n \in \mathbb{N}$, a sequence of soundness functions $(s_{k,n})_{k \in \mathbb{N}}$ with $\lim_{k\to\infty} \lim_{n\to\infty} \delta_{s_{k,n}} = 0$, for which the following holds. For every $n \in \mathbb{N}$, there exists a sequence $(C_{k,n})_{k\in\mathbb{N}}$, $C_{k,n} \subseteq \Sigma^n$ of $(\alpha_{k,n}, \epsilon_{k,n}, s_{k,n})$-

unique LTCs with $\lim_{k\to\infty}\lim_{n\to\infty}\alpha_{k,n} = \lim_{k\to\infty}\lim_{n\to\infty}\epsilon_{k,n} = 0$. Our result does not exclude the possibility of proving the UGC using such a sequence of LTCs.

Note, however, that such a sequence of LTCs is weaker and hence it seems harder to use it in PCP constructions. Note also, that known constructions of low-error PCPs use a single soundness function s (typically of the form $s(\delta) = c_1\delta^{c_2}$, for non-negligible δ). Furthermore, the errors of the resulting PCPs depend polynomially on $\frac{1}{\delta_s}$, as this is the number of codewords obtained from the list decoding procedure that is typically used in low-error PCP constructions. Therefore, when bounding the errors of such PCPs, the term $\frac{1}{\delta_s}$ should also be taken into account.

1.6.2 Unique LTCs with Affine Tests Do Not Exist (a Follow-Up Work)

In a follow-up work, we consider the special case of unique LTCs where the tests are *affine*. Formally, we assume that the alphabet set is a finite field $\Sigma = \mathbb{F}$, and consider testers that perform tests of the form $av_i - bv_j = c$, where $v \in \mathbb{F}^n$ is the tested word, $i, j \in [n]$ and $a, b, c \in \mathbb{F}$. We note that many of the LTCs used in PCP constructions are affine, e.g. variants of Reed-Muller, Hadamard, and the Long Code. Moreover, the UGC with affine tests was shown to be equivalent to the UGC [14].

In the current work we show that low-error unique LTCs are of constant size. The follow-up paper shows that in the affine case, even unique LTCs with a large constant error, are of constant size. Furthermore, the paper gives a better upper bound on the size of such LTCs (the bound depends only on the size of the alphabet set). We mention that the follow-up work uses a different (and simpler) technique.

2 Definitions

Let S_Σ be the set of all permutations over Σ. A unique constraint graph (UCG) is a triple $G = (H = (V, E), \Sigma, \Pi)$, where H is an undirected multigraph (we allow parallel edges), Σ is a finite label set, and $\Pi : E \to S_\Sigma$ is a function mapping edges to permutations. We denote $E(G) \doteq E$ and $V(G) \doteq V$.

Let $G = (H = (V, E), \Sigma, \Pi)$ and $G' = (H' = (V', E'), \Sigma, \Pi')$ be a pair of UCGs. We say that G' is a subgraph of G if $V' \subseteq V$, $E' \subseteq E \cap (V')^2$, and Π' is the restriction of Π to E'. The UCG G' is an edges subgraph of G if it is a subgraph of G, and also $V' = V$. When the label set and permutation mapping of a UCG G are known, we sometimes refer to G simply as a graph. Furthermore, if G' is a subgraph of a known UCG G, we only describe the vertex and edge sets of G'. Finally, if G' is an edges subgraph of G, we only describe the edge set of G'.

Let $G = (H = (V, E), \Sigma, \Pi)$ be a UCG. Let $e = (i, j) \in E$ be an edge of G, let $\pi = \Pi(e)$ be the permutation mapped to e, and let $v \in \Sigma^n$ be a word. We say that v satisfies e if $v_j = \pi(v_i)$. Otherwise, we say that v violates e. We denote by $G(v)$ the edges subgraph of G containing the edges satisfied by v.

Let T be a unique local tester for a code $C \subseteq \Sigma^n$. Let Q be the set of all the pairs of indices (i, j) that may be queried by T in a single execution. Assume that all the pairs in Q are queried by T with an equal probability. Then, T gives rise to the following natural UCG, denoted G_T. The vertex set of G_T is $[n]$, the edge set is Q, and the label set is Σ. An edge $e = (i, j)$ of G_T is mapped to the permutation π_{ij} promised by the uniqueness property of T. If T queries different pairs in Q with different probabilities, then the number of edges between a pair of vertices in Q is proportional to the probability of this pair being queried.

Note that G_T fully characterizes the behavior of T in the event that T makes exactly two queries: Assume that T tests a word v, and decides to make two queries. Then, T can be thought of as operating as follows: It first selects a random edge $e = (i, j) \in E(G_T)$, and queries v_i and v_j. It accepts if v satisfies e , and otherwise rejects.

3 Proof of Main Theorem

In this section we prove the main result, Theorem 3, using two key lemmas. We state the lemmas, then prove the theorem using the lemmas.

The theorem is proved by way of contradiction. We assume to be given a function s and a value δ, as described by the theorem. We denote $\alpha = \epsilon \doteq 10^{-10}\delta^2$. We let Σ be a finite alphabet set, and let $n \in \mathbb{N}$ be a natural number. We then assume for contradiction the existence of a code $C \subseteq \Sigma^n$, satisfying the following properties:

- The relative distance of C is at least $1 - \alpha$.
- There exists an (ϵ, s)-unique local tester T for C.
- $|C| > 10^3 \frac{|\Sigma|}{\sqrt{\alpha}}$.

We next state the key lemmas. Both lemmas hold under the above assumptions. The first lemma constructs a "special" edges subgraph G^* of G_T. The graph G^* is special since it contains a constant fraction of G_T's edges, but has no large connected components. The second lemma uses the graph G^* to create a "special" word $v^* \in \Sigma^n$. The word v^* is special since it is accepted by T with constant probability, but is very far from the code C. Specifically, as the code's parameters keep improving (ϵ and α tend to 0), the distance of v^* from the code becomes larger (tends to 1). Below are the formal statements of the lemmas.

We denote by Q_2 the event that T makes exactly two oracle calls, and by $\overline{Q_2}$ the event that T makes at most one call. For simplicity of notation, we write $E_T \doteq E(G_T)$ and $e_T \doteq |E_T|$.

Lemma 4 (**Graph Decomposition**). *Assume that* $\Pr[Q_2] \geq \frac{1}{2}$ *(the probability is taken over* T*'s randomness). Then, there exists an edges subgraph* G^* *of* G_T *satisfying both of the following:*

- $|E(G^*)| > 2 \cdot 10^{-4} e_T$.
- *Every connected component of* G^* *is of size at most* αn.

Lemma 5 (**Existence of a Special Word**). *There exists a word* $v^* \in \Sigma^n$ *satisfying both of the following:*

- $\Pr\left[T^{v^*} = \texttt{accept}\right] > 10^{-5}$.
- Denote $\beta \doteq \sqrt{\alpha} = \sqrt{\epsilon}$. Then, $\Delta(v^*, C) > 1 - 3\beta$.

The proof of lemma 4 requires most of the effort, and can be found in Sections 5 and 6. The proof of Lemma 5 is presented in Section 7. Overviews of both proofs can be found in Section 4.

Theorem 3 is an easy corollary of Lemma 5.

Proof of Theorem 3. Let v^* be the special word guaranteed by Lemma 5. The lemma shows

$$\Pr\left[T^{v^*} = \texttt{accept}\right] > 10^{-5} \geq s(\delta).$$

Due to T's soundness, it must hold that $\Delta(v^*, C) \leq 1 - \delta$. However, it follows from Lemma 5 that

$$\Delta(v^*, C) > 1 - 3\beta = 1 - 3\sqrt{\alpha} = 1 - 3\left(10^{-5}\delta\right) > 1 - 3\left(\frac{1}{3}\delta\right) = 1 - \delta.$$

We have reached a contradiction. □

4 Overview of Key Lemmas

This section is devoted to informal sketches of the proofs of the two key lemmas. The sketches assume for simplicity that G_T is d-regular, and that T always makes exactly two oracle calls.

4.1 Overview of Lemma 4

The proof suggests an algorithm called decompose (Figure 2). The algorithm proceeds in iterations. The goal of each iteration is to disconnect a new, non-empty, set of vertices A from the rest of G_T's vertices. The set A should be of size at most αn, and should be disconnected by the removal of less than $1 - 2 \cdot 10^{-4}$ fraction of the edges touching it in G_T. We next describe and analyze the first iteration. The following iterations are similar, and work on the part of G_T induced by the vertices that are not yet disconnected. Since the sets disconnected in different iterations are disjoint, the result follows.

The decompose Algorithm

First Attempt. As a warm up, consider the following algorithm for the first iteration of decompose:

1. Randomly select a pair of different codewords u and w:
   (a) Let $A = agree\left(u, w\right)$.
   (b) Let E be the set of edges of G_T that do not belong to at least one of $G_T\left(u\right)$ and $G_T\left(w\right)$ (i.e., violated by at least one of u and w).
2. Let G_T^1 be the graph obtained from G_T by removing every edge in E that has exactly one endpoint in A.

In the case that A is empty, we repeat the algorithm. Recall that $|C| > |\Sigma|$, therefore a non-empty A will eventually be found. In addition, since the relative distance of the code C is at least $1 - \alpha$, it holds that $|A| \leq \alpha n$. We next claim that A is disconnected from the rest of G_T^1. Let $e = (i, j)$ be an edge of G_T^1, and assume for contradiction that $i \in A$ (i.e., $u_i = w_i$), but $j \notin A$ (i.e., $u_j \neq w_j$). Since e was not removed, it must be satisfied by both u and w. Therefore, $u_j = \pi_{ij}\left(u_i\right) = \pi_{ij}\left(w_i\right) = w_j$, a contradiction.

Problem. The key problem is that the algorithm may remove too many edges. According to T's completeness, a codeword may violate up to ϵ fraction of G_T's edges. Note that the algorithm may remove all the edges violated by either u or w. Therefore, it may remove up to $2\epsilon e_T$ edges. Consider a random code for example. Since u and w are random words, A's expected size is $\frac{n}{|\Sigma|}$. For A of size $\frac{n}{|\Sigma|}$, on average over $i \in A$, the algorithm may remove up to $\frac{2\epsilon e_T}{|A|} = \epsilon |\Sigma| d$ of the edges touching i. Since $|\Sigma|$ can be larger than $\frac{1}{\epsilon}$, all the edges touching i may be removed.

We are therefore interested in disconnecting a "cheap" subset of A. The subset should have the following property. On average over i in the subset, less than $1-4\cdot 10^{-4}$ fraction of the edges touching i are removed while disconnecting the subset.

Solution. Let v be a word, $i \in [n]$ be an index, and G be a subgraph of G_T. We say that i is expensive with respect to G, if G contains less than 0.1 fraction of the edges touching i in G_T. Specifically, i is expensive with respect to $G_T(v)$ if v violates more than 0.9 fraction of edges touching i in G_T. Let G_v be the graph obtained from $G_T(v)$ by the execution of the following $\mathsf{remove} - \mathsf{expensive}$ procedure (Figure 1): While there exists an expensive vertex i with respect to $G_T(v)$, remove i and all the edges touching i from $G_T(v)$. We say that i is iteratively expensive with respect to v if it is eventually removed by the above procedure.

To obtain a "cheap" subset, we remove from A all iteratively expensive indices with respect to either u or w, for the following reason. Let v be either u or w. If A contains an expensive index i with respect to $G_T(v)$, then G_T^1 may remove more than 0.9 fraction of the edges touching i in G_T. Thus, we remove from A all the indices that are expensive with respect to $G_T(v)$. However, some of the vertices in the updated A may be connected to expensive vertices just removed from A. To make sure that the updated A is still disconnected, we delete all the edges touching expensive vertices. Note that the removal of more edges may cause some of the vertices that were originally cheap to become expensive. We therefore need to iterate this process, removing the new expensive vertices from A, and removing edges touching expensive vertices.

The following is a corrected version of the previously suggested first iteration of decompose:

1. For a pair of codewords u' and w':

   (a) Let $A_{u',w'}$ be the set obtained from $agree(u', w')$ by removing all the indices that are iteratively expensive with respect to either u' or w'.

(b) Let $E_{u',w'}$ be the set of edges of G_T that do not belong to at least one of $G_{u'}$ and $G_{w'}$.

2. Let u and w be a pair of different codewords for which $A \doteq A_{u,w} \neq \phi$, and $E \doteq E_{u,w}$ contains less than $1 - 2 \cdot 10^{-4}$ fraction of edges touching A in G_T. (We will next see that such u and w exist).

3. Let G_T^1 be the graph obtained from G_T by removing every edge in E that has exactly one endpoint in A.

Proof Overview

In order to prove that decompose performs properly, we show the followings: (i) There exist codewords u and w satisfying the requirements of Step 2 of the algorithm (Lemma 6). (ii) $|A| \leq \alpha n$. (iii) A is disconnected from the rest of G_T^1.

As before, $|A| \leq |agree\,(u,w)| \leq \alpha n$. In addition, A is disconnected from the rest of G_T^1 for the following reason. Let $e = (i,j)$ be an edge of G_T^1, and assume for contradiction that $i \in A$, but $j \notin A$. Since e was not removed, it belongs to both G_u and G_w. In particular, j is not iteratively expensive with respect to either u or w. As before, since $i \in A \subseteq agree\,(u,w)$, it holds that $u_j = \pi_{ij}\,(u_i) = \pi_{ij}\,(w_i) = w_j$, and $j \in agree\,(u,w)$. We conclude that $j \in A$ and reach a contradiction.

The main challenge is proving Lemma 6. In order to do so, we show the following claim: *There exists a pair of different codewords u and w satisfying the following. On average over $i \in A \doteq A_{u,w}$, the set $E \doteq E_{u,w}$ contains strictly less than $1 - 4 \cdot 10^{-4}$ fraction of the edges touching i in G_T.* Note that since we demand a strict inequality, the claim also implies that $A \neq \phi$. We remark that the claim does not follow directly from the fact that A contains no iteratively expensive indices with respect to either u or w. The latter implies that u satisfies at least 0.1 fraction of the edges touching i, and so does w. However, it may be the case that there is not even a single edge that they both satisfy.

We next describe the steps taken towards proving Lemma 6. We loosely state the needed lemmas and sketch their proofs. We use the following notations. Let $i \in [n]$ be an index, and let $\sigma \in \Sigma$ be a symbol. Denote by L_i the set of codewords for which i is not iteratively expensive with respect to. In addition, denote $L_{i,\sigma} \doteq \{v \in L_i \mid v_i = \sigma\}$.

Step 1 (Lemma 10). *Let v be a codeword. Then,* remove − expensive *removes at most $\frac{\epsilon}{4} e_T$ of the edges of $G_T\,(v)$.* We represent a subgraph G of G_T as a linked list. Each node represents a vertex i of G, and stores the names of i's neighbors in a d-element array. If i has less than d neighbors, the unused array elements are called *free slots*.

To begin with, $G_T(v)$ contains at most $2\epsilon e_T$ free slots, as $G_T(v)$ is obtained by deleting at most ϵ fraction of G_T's edges. Now, assume that iteration t removes vertex i, as well as e of the edges touching it. Due to the edge removal, an additional e slots may now be free. However, since i is expensive, more than $9e$ of its slots are free. These slots are lost because i is removed from the list. Therefore, after $\frac{\epsilon}{4}e_T$ edges are removed, the number of free slots drops to less than $2\epsilon e_T - (9-1)\frac{\epsilon}{4}e_T = 0$.

Step 2 (Lemma 11). *Let $i \in [n]$ be a random index. The expected size of L_i is at least $0.2|C|$.* Fix a codeword v. Using Lemma 10, $\mathsf{remove-expensive}$ removes at most $\frac{\epsilon}{4}e_T$ of the edges of $G_T(v)$. Therefore, G_v contains at least $(1-\epsilon)e_T - \frac{\epsilon}{4}e_T \geq 0.2e_T$ edges. Since we assume that G_T is regular, G_v contains at least 0.2 fraction of G_T's vertices. These vertices are not iteratively expensive with respect to v. We conclude that $v \in L_i$ with probability at least 0.2.

Step 3 (Lemma 13). *Fix $i \in [n]$ and $\sigma \in \Sigma$, and assume that $|L_{i,\sigma}| > 100$. Let u' and w' be a pair of different codewords selected at random from $L_{i,\sigma}$. Then, in expectation, more than $0.9 \cdot (0.1)^2 = 0.009$ fraction of the edges touching i in G_T are contained in both $G_{u'}$ and $G_{w'}$.* For $v \in L_{i,\sigma}$, let S_v be the set of edges touching i in G_v. Note that every S_v contains at least 0.1 fraction of the edges touching i in G_T. If $|L_{i,\sigma}| = 10$, for example, then the 10 sets $(S_v)_{v\in L_{i,\sigma}}$ may be disjoint, causing the desired expectation to be 0. But, for $|L_{i,\sigma}| > 10$ disjointness is impossible, thus the expectation is positive. A combinatorial argument shows that as $|L_{i,\sigma}|$ gets larger, the sets $(S_v)_{v\in L_{i,\sigma}}$ behave like random sets, and the desired expectation gets closer to $(0.1)^2$.

Step 4 (Proof of Lemma 6). Randomly select a quadruple (u, w, i, σ) that satisfies $u \neq w \in L_{i,\sigma}$. Observe that the quadruple may be viewed as being selected as follows. First randomly select a triple (u, w, i) that satisfies $u \neq w \in C$ and $i \in A_{u,w}$. Then, set $\sigma \doteq u_i = w_i$. In order to prove the claim it suffices to show that, in expectation, the following holds. The fraction of edges touching i that are removed by the algorithm is less than $1 - 4\cdot 10^{-4}$.

For simplicity, assume that all the sets $(L_{i,\sigma})_{i\in[n],\sigma\in\Sigma}$ are of the same size. Using Lemma 11 and the assumption that $|C| > \frac{10^3|\Sigma|}{\sqrt{\epsilon}}$, every such set is of size at least $\frac{0.2|C|}{|\Sigma|} > 100$. Note that u and w are a pair of different codewords randomly selected from $L_{i,\sigma}$. Therefore, using Lemma 13, at least 0.009 fraction of the edges touching i in G_T are in both G_u and G_w. These edges are not removed.

4.2 Overview of Lemma 5

The following algorithm, called $\mathsf{create} - \mathsf{special} - \mathsf{word}$ (Figure 3), creates v^*:

1. Set $k \doteq \lfloor \beta^{-1} \rfloor$, and arbitrarily select k *different* codewords $v_1, ..., v_k$.
2. Let G^* be the graph promised by Lemma 4, and let G' be a subgraph of G^* containing only edges that are satisfied by all $v_1, ..., v_k$.
3. Partition $[n]$ to at most k subsets $S_1, ..., S_k$, such that every set S_t satisfies:
   (a) S_t is a union of connected components of G'.
   (b) $\frac{n}{k} \leq |S_t| < \frac{n}{k} + \alpha n$.
4. Let v^* be the word that agrees with the codeword v_t on S_t for every t.

First note that Step 1 is always possible, as $k \leq \beta^{-1} = \epsilon^{-\frac{1}{2}} < |C|$. In addition, Step 3 is always possible, as all the connected components of G^* (and thus also of G') are of size at most αn.

We next claim that T accepts v^* with probability greater than 10^{-5}. The graph G^* contains more than $2 \cdot 10^{-4}$ fraction of G_T's edges. In addition, each of $v_1, ..., v_k$ violates at most ϵ fraction of G_T's edges. Since k was chosen to be much smaller than $\frac{1}{\epsilon}$, a calculation shows that $2 \cdot 10^{-4} - k\epsilon > 10^{-5}$. Therefore, G' contains more than 10^{-5} fraction of G_T's edges. Since v^* satisfies G', the claim follows.

Our last claim is that for every codeword v it holds that $\Delta(v^*, v) > 1 - 3\beta$. Recall that v^* is a hybrid of $v_1, ..., v_k$. If $v \notin \{v_1, ..., v_k\}$ then $agree(v^*, v) \subseteq \bigcup_{t \in [k]} agree(v_t, v)$. Otherwise, if $v = v_r$ then $agree(v^*, v) \subseteq \left(\bigcup_{t \in [k] \setminus \{r\}} agree(v_t, v)\right) \cup S_r$. The set S_t is small for every t. In addition, the set $agree(v_t, v)$ is small for $v \neq v_t$. Therefore, in both cases the set $agree(v^*, v)$ is small.

5 Graph Decomposition (Proof of Lemma 4)

Let $G = (H = (V, E), \Sigma, \Pi)$ be a UCG, and let $i \in V$ be a vertex of G. The degree of the vertex i in G is defined as $deg_G(i) \doteq |\{j \in V \mid (i,j) \in E\}|$. The degree of a set of vertices $U \subseteq V$ in G is $deg_G(U) \doteq \sum_{i \in U} deg_G(i)$. The degree of the graph G is $deg_G \doteq deg_G(V)$. Let G_1 and G_2 be a pair of subgraphs of G. We denote by $G_1 \cap G_2$ the subgraph of G whose vertex set is $V(G_1) \cap V(G_2)$, and edge set is $E(G_1) \cap E(G_2)$.

```
remove-expensive$(G_T, G)$
1. While there exists a vertex $i$ in $G$ satisfying $deg_G(i) < 0.1 deg_{G_T}(i)$
    (a) Remove vertex $i$, and all the edges touching $i$, from $G$.
```

Figure 1: $\mathsf{remove}-\mathsf{expensive}$ algorithm

Consider the $\mathsf{remove}-\mathsf{expensive}$ algorithm described in Figure 1, and the $\mathsf{decompose}$ algorithm described in Figure 2. $\mathsf{decompose}$ receives the graph G_T as an input. It either returns `failure`, or else returns an edges subgraph G^* of G_T. The main claim of this section is that the graph G^* satisfies the requirements of Lemma 4.

The following three lemmas prove Lemma 4.

Lemma 6. *Assume that* $\Pr[Q_2] \geq \frac{1}{2}$. *Then,* $\mathsf{decompose}(G_T)$ *always returns a graph* G^*. *That is, the execution of* $\mathsf{decompose}(G_T)$ *always terminates in finite time, and never returns* `failure`.

Lemma 7. *Every connected component of* G^* *is of size at most* αn.

Lemma 8. $|E(G^*)| > 2 \cdot 10^{-4} e_T$.

We begin by proving Lemmas 7 and 8. The proof of Lemma 6 is more involved, and is deferred to Section 6. The proofs use the following notations. Assume that the execution of $\mathsf{decompose}(G_T)$ consists of s iterations of Step 2. Let $t \in [s]$ be an iteration, and let v be a codeword. Denote by u_t and w_t the pair of codewords selected in Step 2(b)i of iteration t. Denote by I_t, E_t, $G_{t,v}$, $I_{t,v}$, $G_t^{\cap}$ and A_t the values of the sets I, E, G_v, I_v, $G_{u_t,w_t}^{\cap}$ and A_{u_t,w_t} (respectively), at the end of iteration t. Set $I_0 = A_0 = \phi$. Let G_t be the edges subgraph obtained from G_T by removing the edges in E_t.

We begin by showing the following claim.

Claim 9. For every $t \in \{0, ..., s\}$, the set A_t is a union of connected components of G_t (and therefore also of G^*). In other words, no edge of G_t crosses the cut $(A_t, [n] \setminus A_t)$.

```
**decompose**$(G_T)$

1. **Initialization:**
    (a) Set $I = \phi$. ($I$ is the set of vertices already disconnected)
    (b) Set $E = \phi$. ($E$ is the set of edges to be removed from $G_T$ to obtain $G^*$)

2. While $deg_{G_T}(I) < 0.2 deg_{G_T}$
    (a) **Remove Expensive Vertices:** For every codeword $v$
        i. Let $G$ be the edges subgraph obtained from $G_T$ after removing the
           edges in $E$. Set $G_v = G(v)$.
        ii. Update $G_v$ by running $\mathsf{remove} - \mathsf{expensive}(G_T, G_v)$.
            Let $I'_v$ be the set vertices removed by the execution, and set $I_v = I'_v \backslash I$.
    (b) **Disconnect a Cheap Vertex Set:**
        For a pair of codewords $w$ and $u$ denote $G^{\cap}_{u,w} \doteq G_u \cap G_w$,
        and $A_{u,w} \doteq agree(u, w) \setminus (I \cup I_u \cup I_w)$.
        i. Find a pair of different codewords $w$ and $u$ satisfying
               $$deg_{G^{\cap}_{u,w}}(A_{u,w}) > 10^{-3} deg_{G_T}(A_{u,w}).$$
           If no such pair exists, return failure.
        ii. Add to $E$ every edge in $E_T \backslash E\left(G^{\cap}_{u,w}\right)$ with exactly one endpoint in $A_{u,w}$.
        iii. Add to $I$ all the vertices in $A_{u,w}$.

3. **Create Singletons:** Add to $E$ every edge in $E_T \backslash E$ with two endpoints in $[n] \setminus I$.

4. **Create $G^*$:** Return the graph $G^*$ obtained by removing the edges in $E$
   from $G_T$.
```

Figure 2: decompose algorithm

Proof. We prove the claim by induction on t. For $t = 0$, the claim holds trivially. Let $r \in [s]$ and assume that the claim holds for $t \leq r - 1$. We prove the claim for $t = r$. Assume for contradiction that there exits an edge $e = (i, j)$ in G_r such that $i \in A_r$, but $j \notin A_r$.

We first show that $j \notin I_{r-1}$. Using the induction hypothesis and the fact that $I_{r-1} = \bigcup_{t \in \{0,..,r-1\}} A_t$, it holds that I_{r-1} is a union of connected components of G_r.

Since we assume that $i \in A_r \subseteq [n] \setminus I_{r-1}$, it must hold that $j \notin I_{r-1}$.

We next show that $j \notin I_{r,u_r} \cup I_{r,w_r}$. Observe that e has exactly one endpoint in A_r, but it was not added to E by iteration r. Therefore, according to Step 2(b)ii of decompose, it holds that $e \in E\left(G_r^{\cap}\right)$. In particular, $j \in V\left(G_{r,u_r}\right) \subseteq [n] \setminus I_{r,u_r}$ and $j \in V\left(G_{r,w_r}\right) \subseteq [n] \setminus I_{r,w_r}$.

Finally, we show that $j \in agree\left(u_r, w_r\right)$. Let π be the permutation mapped to e in G_r. Since $e \in E\left(G_r^{\cap}\right)$, e is satisfied by both u_r and w_r. In addition, the assumption that $i \in A_r$ implies $\left(u_r\right)_i = \left(w_r\right)_i$. Thus, $\left(u_r\right)_j = \pi\left(\left(u_r\right)_i\right) = \pi\left(\left(w_r\right)_i\right) = \left(w_r\right)_j$.

We conclude that $j \in agree\left(u_r, w_r\right) \setminus \left(I_{r-1} \cup I_{r,u_r} \cup I_{r,w_r}\right) = A_r$, and reach a contradiction. □

Proof of Lemma 7. We first claim that I_s is a union of connected components of G^*, each of size at most αn. Recall that $I_s = \bigcup_{t \in [s]} A_t$. Claim 9 shows that for every $t \in [s]$, the set A_t is a union of connected components of G^*. In addition, since u_t and w_t are different codewords, it holds that $\left|A_t\right| \leq \left|agree\left(u_t, w_t\right)\right| \leq \alpha n$.

Step 3 makes sure that every $i \in [n] \setminus I_s$ is a singleton (connected component of size 1) of G^*. □

Proof of Lemma 8. Fix an iteration $t \in [s]$. We first claim that the edges touching A_t in $G_t^{\cap}$ are never added to E, and conclude $deg_{G^*}\left(A_t\right) = deg_{G_t^{\cap}}\left(A_t\right)$. Clearly, no edges of $G_t^{\cap}$ are added to E during iteration t. Consider an iteration $r > t \in [s]$. Claim 9 shows that A_t is a union of connected components of G_t, and therefore also of G_{r-1}. Since $A_t \subseteq I_t \subseteq I_{r-1}$ and $A_r \subseteq [n] \setminus I_{r-1}$, it holds that $A_t \cap A_r = \phi$. Therefore, since iteration r only adds to E edges touching A_r in G_{r-1}, it does not add edges touching A_t.

Recall that u_t and w_t satisfy the condition in Step 2(b)i, and conclude

$$deg_{G^*}\left(A_t\right) \geq deg_{G_t^{\cap}}\left(A_t\right) > 10^{-3} deg_{G_T}\left(A_t\right).$$

Since I_s can be written as a disjoint union $I_s = \bigcup_{t \in [s]} A_t$, it holds that

$$deg_{G^*} \;\; \geq \;\; deg_{G^*}\left(I_s\right) = \sum_{t \in [s]} deg_{G^*}\left(A_t\right) > 10^{-3} \sum_{t \in [s]} deg_{G_T}\left(A_t\right) = 10^{-3} deg_{G_T}\left(I_s\right).$$

We assume that Step 2 is executed exactly s times. Therefore, it must hold that $deg_{G_T}\left(I_s\right) \geq 0.2 deg_{G_T}$. Conclude,

$$\left|E\left(G^*\right)\right| = 0.5 deg_{G^*} > 0.5 \cdot 10^{-3} deg_{G_T}\left(I_s\right) \geq 0.5 \cdot 10^{-3} \cdot 0.2 deg_{G_T} = 2 \cdot 10^{-4} e_T.$$

□

6 Finding a Cheap Vertex Set (Proof of Lemma 6)

In this section we prove Lemma 6. As described in the sketch (see Subsection 4.1), the lemma in proved in four steps.

6.1 Step 1

The following lemma bounds the number of edges removed by $\mathsf{remove}-\mathsf{expensive}$.

Lemma 10. *Let G be an edges subgraph of G_T, satisfying $|E(G)| \geq \lambda e_T$. Let E_{rem} be the set of edges removed from G by the execution of $\mathsf{remove}-\mathsf{expensive}(G_T, G)$. Then,*

$$|E_{rem}| \leq \frac{1-\lambda}{4} e_T.$$

Recall that the sketch assumed that T always makes two oracle calls. Therefore, for a codeword v, T's completeness implies $|E(G_T(v))| > (1-\epsilon) e_T$. In this case, Lemma 10 shows that the execution of $\mathsf{remove}-\mathsf{expensive}(G_T, G_T(v))$ removes at most $\frac{\epsilon}{4} e_T$ edges, as claimed in the sketch.

Proof. We represent a subgraph F of G_T as a linked list of $|V(F)|$ nodes. Each node represents a vertex j of F, and stores the names of j's neighbors in a $deg_{G_T}(j)$-element array. By "node j", we refer to the node that corresponds to vertex j. If j has less than $deg_{G_T}(j)$ neighbors in F, then the unused array elements are called *free slots*. Formally, the node j contains $deg_{G_T}(j) - deg_F(j)$ free slots. The number of free slots in F is the total number of free slots in all the $|V(F)|$ nodes in the representation of F.

To begin with, G contains at most $2(1-\lambda) e_T$ free slots, as G is obtained by deleting at most $1-\lambda$ fraction of G_T's edges. Now, assume that during iteration t of the execution vertex i is removed, as well as e of the edges touching it. Observe that each of the e edges removed has exactly one endpoint in $V(G) \setminus \{i\}$. Therefore, the total number of free slots in all the nodes excluding i increases by e.

After iteration $t-1$, it holds that $deg_G(i) = e$. Since i was selected for removal by iteration t, it is the case that $e < 0.1 deg_{G_T}(i)$, i.e, $deg_{G_T}(i) > 10e$. Therefore, after iteration $t-1$, the number of free slots in node i is $deg_{G_T}(i) - deg_G(i) > 9e$. Since iteration t removes node i, the free slots in node i are lost.

We conclude that iteration t causes the number of free slots in G to drop by more than $9e - e = 8e$. Therefore, if more than $\frac{1-\lambda}{4}e_T$ edges are removed by the execution, then the number of free slots in G drops to less than $2(1-\lambda)e_T - 8\left(\frac{1-\lambda}{4}e_T\right) = 0$. But this is impossible as the number of free slots is non-negative for every subgraph of G_T. □

6.2 Step 2

For the rest of Section 6, we fix t and assume that the execution of $\mathsf{decompose}(G_T)$ has just reached the t^{th} iteration of Step 2. The sets I,E,I_v and graph G are updated by $\mathsf{decompose}$ throughout its execution. When any of these variables is used in the following lemmas, we refer to its value at the end of Step 2(a)ii of iteration t.

For an index $i \in [n]$, we denote

$$L_i \doteq \{v \in C \mid i \in [n] \setminus (I \cup I_v)\}.$$

Let $\ell_i \doteq |L_i|$. The main result of this subsection is the following lemma.

Lemma 11. *Assume that* $\Pr[Q_2] \geq \frac{1}{2}$. *Then,*

$$\sum_{i\in[n]} \ell_i deg_{G_T}(i) > 0.2\,|C|\,deg_{G_T}.$$

Recall that the sketch assumed that G_T is d-regular. In this case, the above lemma suggests

$$\mathop{\mathbb{E}}_{i\in[n]}[\ell_i] = \frac{1}{n}\sum_{i\in[n]} \ell_i = \frac{1}{dn}\sum_{i\in[n]} \ell_i deg_{G_T}(i) > \frac{1}{dn}\cdot 0.2\,|C|\,deg_{G_T} = 0.2\,|C|,$$

as claimed in the sketch.

The proof of Lemma 11 uses the following claim. The claim shows that if T makes two queries with high probability, then every codeword satisfies almost all of G_T's edges.

Claim 12. Let v be a codeword, and assume that $\Pr[Q_2] \geq \frac{1}{2}$. Then,

$$|E(G_T(v))| \geq (1-2\epsilon)\,e_T.$$

Proof. Due to T's completeness, it holds that $\Pr[T^v = \texttt{reject}] \leq \epsilon$. Using Bayes' formula, and the assumption that $\Pr[Q_2] \geq \frac{1}{2}$, it holds that

$$\Pr\left[T^v = \texttt{reject}\ |Q_2\right] = \frac{\Pr\left[T^v = \texttt{reject} \wedge Q_2\right]}{\Pr\left[Q_2\right]} \leq \frac{\epsilon}{\Pr\left[Q_2\right]} \leq 2\epsilon.$$

Conclude that $\Pr\left[T^v = \texttt{accept}\ |Q_2\right] \geq 1 - 2\epsilon$, and the claim follows. □

Proof of Lemma 11. Denote

$$R \doteq \sum_{i\in[n]} \ell_i deg_{G_T}\left(i\right).$$

It holds that

$$R \;=\; \sum_{i\in[n]}\sum_{v\in L_i} deg_{G_T}\left(i\right) = \sum_{v\in C}\sum_{i\in[n]\backslash(I\cup I_v)} deg_{G_T}\left(i\right) = \sum_{v\in C}\left(deg_{G_T} - deg_{G_T}\left(I\right) - deg_{G_T}\left(I_v\right)\right).$$

We assume that decompose has entered the t^{th} iteration of Step 2. Therefore, the following condition holds

$$deg_{G_T}\left(I\right) < 0.2 deg_{G_T}.$$

Let v be a codeword, and let E_{I_v} be the set of edges in G_T with at least one endpoint in I_v. We turn to bound $deg_{G_T}\left(I_v\right)$, by bounding the size of E_{I_v}. We define the following three sets E_1, E_2 and E_3, that satisfy $E_{I_v} \subseteq E_1 \cup E_2 \cup E_3$. The first two sets are $E_1 \doteq E_T \backslash E\left(G\right)$ and $E_2 \doteq E\left(G\right)\backslash E\left(G\left(v\right)\right)$. The third set, E_3, contains the edges in $E\left(G\left(v\right)\right)$ with at least one endpoint in I_v. We next bound the sizes of E_1, E_2 and E_3 separately.

Consider E_1. Note that $E_1 = E$, as G was obtained in Step 2(a)i by the removal of the edges in E from G_T. In addition, $|E| \leq deg_{G_T}\left(I\right)$, as only edges touching vertices in I were added to E. Thus,

$$|E_1| = |E| \leq deg_{G_T}\left(I\right) < 0.2 deg_{G_T}.$$

Consider E_2. Claim 12 shows that v violates at most $2\epsilon e_T$ of G_T's edges. Therefore, v violates at most $2\epsilon e_T$ of G's edges. Recall that $\epsilon < 10^{-10}$ and get

$$|E_2| \leq 2\epsilon e_T < 0.01 deg_{G_T}.$$

Consider E_3. Observe that E_3 is contained in the set of edges removed by the execution of $\mathsf{remove} - \mathsf{expensive}(G_T, G_v)$ in Step 2(a)ii. We use Lemma 10 to bound the size of E_3. In order to do so, we first upper bound the size of $E(G(v))$. It holds that

$$|E(G(v))| = e_T - (|E_1| + |E_2|) > (1 - 2(0.2 + 0.01))\, e_T = 0.58 e_T.$$

Now, using Lemma 10 with $\lambda = 0.58$, we get

$$|E_3| \leq \frac{(1-\lambda)}{4} e_T < 0.06 deg_{G_T}.$$

Using the bounds obtained for E_1, E_2 and E_3, we conclude,

$$deg_{G_T}(I_v) \leq 2\,|E_{I_v}| \leq 2\,(|E_1| + |E_2| + |E_3|) < 0.6 deg_{G_T}.$$

We return to lower bounding R,

$$R > \sum_{v \in C} (deg_{G_T} - 0.2 deg_{G_T} - 0.6 deg_{G_T}) = 0.2\,|C|\, deg_{G_T},$$

as claimed. □

6.3 Step 3

For an index $i \in [n]$ and a symbol $\sigma \in \Sigma$, we denote

$$L_{i,\sigma} \doteq \{v \in L_i \mid v_i = \sigma\}.$$

Let $\ell_{i,\sigma} \doteq |L_{i,\sigma}|$. Let $pairs(x) : \mathbb{R}^+ \cup \{0\} \to \mathbb{R}$ be the function $pairs(x) \doteq x(x-1)$. For $x \in \mathbb{N} \cup \{0\}$, the value $pairs(x)$ is the number of possible ways of selecting an ordered pair of elements out of an x-element set. Note that $pairs(x) < 0$ for $x \in (0,1)$. The main result of this subsection is the following lemma.

Lemma 13. *Let $i \in [n]$ and $\sigma \in \Sigma$. Then,*

$$\sum_{u \neq w \in L_{i,\sigma}} deg_{G^{\cap}_{u,w}}(i) \geq deg_{G_T}(i) \cdot pairs(0.1 \ell_{i,\sigma}).$$

Recall that the sketch assumed that G_T is d-regular. In addition, observe that if $\ell_{i,\sigma}$ is large, it follows that $pairs(0.1\ell_{i,\sigma}) \approx (0.1)^2\, pairs(\ell_{i,\sigma})$. In particular, a calculation shows that if $\ell_{i,\sigma} > 100$ then $pairs\left(0.1\ell_{i,\sigma}\right) > 0.009 pairs\left(\ell_{i,\sigma}\right)$. Conclude,

$$\mathop{\mathbb{E}}_{u \neq w \in L_{i,\sigma}} \left[deg_{G^{\cap}_{u,w}}(i)\right] = \frac{1}{pairs\left(\ell_{i,\sigma}\right)} \sum_{u \neq w \in L_{i,\sigma}} deg_{G^{\cap}_{u,w}}(i) \geq \frac{pairs\left(0.1\ell_{i,\sigma}\right)}{pairs\left(\ell_{i,\sigma}\right)} deg_{G_T}(i) > 0.009d,$$

as claimed in the sketch.

The proof of Lemma 13 uses the following combinatorial claim.

Claim 14. Let $\ell, m \in \mathbb{N}$ be natural numbers, and let $S_1, ..., S_\ell$ be subsets of $[m]$. Assume that for every $t \in [\ell]$ it holds that $|S_t| \geq 0.1m$. Then,

$$\sum_{t \neq r \in [\ell]} |S_t \cap S_r| \geq m \cdot pairs\left(0.1\ell\right).$$

Note that the bound suggested by the claim is negative for $0 < \ell < 10$.

Proof. Denote

$$R \doteq \sum_{t \neq r \in [\ell]} |S_t \cap S_r|.$$

For $i \in [m]$, denote by x_i the number of sets containing the element i. That is,

$$x_i = \left|\left\{S_t \mid t \in [\ell], i \in S_t\right\}\right|.$$

It holds that

$$\sum_{i \in [m]} x_i = \sum_{t \in [\ell]} |S_t| \geq 0.1\ell m.$$

In addition, note that

$$R = \sum_{i \in [m]} pairs\left(x_i\right).$$

Therefore, in order to lower bound R, we need to minimize the function $\sum_{i \in [m]} pairs\left(x_i\right)$ under the constraint $\sum_{i \in [m]} x_i \geq 0.1\ell m$. Since *pairs* is a convex function, the minimum is attained for $x_i = \frac{0.1\ell m}{m} = 0.1\ell$. Conclude,

$$R \geq \sum_{i \in [m]} pairs\left(0.1\ell\right) = m \cdot pairs\left(0.1\ell\right),$$

as claimed. □

Proof of Lemma 13. Denote $\ell \doteq \ell_{i,\sigma}$ and $m \doteq deg_{G_T}(i)$. For a codeword $v \in L_{i,\sigma}$, denote by S_v the set of neighbors of vertex i in G_v. This defines ℓ sets. Each such set is a subset of the m-element set containing all the neighbors of i in G_T. In addition, for $v \in L_{i,\sigma}$ it holds that $i \in [n] \setminus (I \cup I_v)$. Therefore, $deg_{G_v}(i) \geq 0.1 deg_{G_T}(i)$. Conclude,

$$|S_v| = deg_{G_v}(i) \geq 0.1 deg_{G_T}(i) = 0.1m.$$

For a pair of codewords $u, w \in L_{i,\sigma}$ it holds that

$$deg_{G^{\cap}_{u,w}}(i) = |S_u \cap S_w|.$$

We use Claim 14, and get

$$\sum_{u \neq w \in L_{i,\sigma}} deg_{G^{\cap}_{u,w}}(i) = \sum_{u \neq w \in L_{i,\sigma}} |S_u \cap S_w| \geq m \cdot pairs\left(0.1\ell\right) = deg_{G_T}(i) \cdot pairs\left(0.1\ell_{i,\sigma}\right),$$

as claimed. □

6.4 Step 4 (Proof of Lemma 6)

Proof of Lemma 6. In order to show that decompose never returns `failure`, we need to show that every time it reaches Step 2(b)i, there exists two different codewords u and w satisfying

$$deg_{G^{\cap}_{u,w}}\left(A_{u,w}\right) > 10^{-3} deg_{G_T}\left(A_{u,w}\right).$$

Note that since we require a strict inequality, the above condition also implies $A_{u,w} \neq \phi$. Therefore, decompose always terminates in finite time.

Denote

$$\begin{aligned} R &\doteq \sum_{u \neq w \in C} deg_{G^{\cap}_{u,w}}(A_{u,w}) - 10^{-3} deg_{G_T}(A_{u,w}) \\ &= \sum_{u \neq w \in C} \sum_{i \in A_{u,w}} \sum_{\sigma = u_i = w_i} \left(deg_{G^{\cap}_{u,w}}(i) - 10^{-3} deg_{G_T}(i)\right). \end{aligned}$$

(Note that the sum over σ sums over one element). We aim to show that $R > 0$, and the result follows.

Currently, R sums over the set

$$B_1 \doteq \left\{(u, w, i, \sigma) \in C^2 \times [n] \times \Sigma \mid u \neq w,\, i \in A_{u,w},\, \sigma = u_i = w_i\right\}.$$

We change the summation to be over the set

$$B_2 \doteq \left\{(u,w,i,\sigma) \in C^2 \times [n] \times \Sigma \mid u \neq w \in L_{i,\sigma}\right\}.$$

We next show that $B_1 = B_2$, and therefore the value of the sum does not change. Recall that $A_{u,w} = agree\,(u,w) \setminus (I \cup I_u \cup I_w)$.

- $B_1 \supseteq B_2$: Let $(u,w,i,\sigma) \in B_2$. Since $w,u \in L_i$ it holds that $i \in [n] \setminus (I \cup I_u)$ and $i \in [n] \setminus (I \cup I_w)$. In addition, it holds that $w_i = u_i = \sigma$, and in particular $i \in agree\,(u,w)$. Conclude $i \in A_{u,w}$.
- $B_1 \subseteq B_2$: Let $(u,w,i,\sigma) \in B_1$. Since $i \in A_{u,w}$ it holds that $i \in [n] \setminus (I \cup I_u \cup I_w)$. Therefore, $u,w \in L_i$. Since $\sigma = u_i = w_i$ it holds that $u,w \in L_{i,\sigma}$.

After the change,

$$\begin{aligned} R &= \sum_{\substack{i\in[n]\\ \sigma\in\Sigma}} \sum_{u\neq w\in L_{i,\sigma}} \left(deg_{G^{\cap}_{u,w}}(i) - 10^{-3} deg_{G_T}(i)\right) \\ &= \sum_{\substack{i\in[n]\\ \sigma\in\Sigma}} \left(\sum_{u\neq w\in L_{i,\sigma}} deg_{G^{\cap}_{u,w}}(i) - 10^{-3} \sum_{u\neq w\in L_{i,\sigma}} deg_{G_T}(i) \right) \end{aligned}$$

Using Lemma 13,

$$R \geq \sum_{\substack{i\in[n]\\ \sigma\in\Sigma}} \left(pairs\,(0.1\ell_{i,\sigma}) - 10^{-3} \cdot pairs\,(\ell_{i,\sigma})\right) deg_{G_T}(i).$$

By expending the *pairs* function,

$$R \geq \sum_{i\in[n]} \left(\sum_{\sigma\in\Sigma} \left(0.009\ell_{i,\sigma}^2 - 0.099\ell_{i,\sigma}\right) \right) deg_{G_T}(i).$$

Note that for every $x \in \mathbb{R}$ it holds that

$$0.009x^2 - 0.099x > 0.01x - 1.$$

Hence,

$$\begin{aligned} R &> \sum_{i\in[n]}\left(\sum_{\sigma\in\Sigma}(0.01\ell_{i,\sigma}-1)\right)deg_{G_T}(i) \\ &= 0.01\sum_{i\in[n]}\left(\sum_{\sigma\in\Sigma}\ell_{i,\sigma}\right)deg_{G_T}(i)-\sum_{i\in[n]}\left(\sum_{\sigma\in\Sigma}1\right)deg_{G_T}(i) \\ &= 0.01\left(\sum_{i\in[n]}\ell_i deg_{G_T}(i)\right)-|\Sigma|\,deg_{G_T}. \end{aligned}$$

Recall that we assume $\Pr[Q_2]\geq\frac{1}{2}$. Therefore, we may use Lemma 11 and get

$$R \;>\; (0.002\,|C|-|\Sigma|)\,deg_{G_T}.$$

The assumption $\Pr[Q_2]\geq\frac{1}{2}$ suggests that $deg_{G_T}>0$. In addition, the assumption $|C|>\frac{10^3|\Sigma|}{\sqrt{\alpha}}$ implies $0.002\,|C|-|\Sigma|>0$. We conclude $R>0$, as claimed. □

7 Existence of a Special Word (Proof of Lemma 5)

Consider the $\mathsf{create}-\mathsf{special}-\mathsf{word}$ algorithm described in Figure 3. The algorithm receives the graph G_T as an input, and returns a word $v^*\in\Sigma^n$. The main claim of this section is that the word v^* satisfies the requirements of Lemma 5.

The following three lemmas prove Lemma 5.

Lemma 15. $\mathsf{create}-\mathsf{special}-\mathsf{word}\,(G_T)$ *never returns* `failure`.

Lemma 16. $\Pr\left[T^{v^*}=\texttt{accept}\right]>10^{-5}$.

Lemma 17. $\Delta(v^*,C)>1-3\beta$.

Proof of Lemma 15. Assume that $\Pr[Q_2]\geq\frac{1}{2}$. We next construct k sets $S_1,...,S_k$ that satisfy the requirements of Step 2b. Using Lemma 4, every connected component of G^* (and therefore also of G') is of size at most αn. Hence, we may construct the sets $S_1,...,S_k$ one-by-one as follows. To construct S_t: Start with an empty set. As long as $|S_t|<\frac{n}{k}$ and there exits a vertex i that does not belong to any of $S_1,...,S_{t-1}$, add to S_t all the vertices in the connected component of i in G'. □

```
create-special-word($G_T$)
1. Let $k \doteq \lfloor \beta^{-1} \rfloor$. Note that $k \leq \beta^{-1} = \epsilon^{-\frac{1}{2}} < |C|$.
   Let $v_1, \ldots, v_k$ be $k$ different codewords, arbitrarily selected.
2. Partition $[n]$ to $k$ subsets $S_1, ..., S_k$ as follows:
   (a) If $\Pr\left[\overline{Q_2}\right] > \frac{1}{2}$:
       i. For every $t \in [k]$, $|S_t| \leq \left\lceil \frac{n}{k} \right\rceil$ (the sets are of almost equal sizes).
   (b) If $\Pr\left[Q_2\right] \geq \frac{1}{2}$:
       i. Let $G^*$ be the graph returned by decompose $(G_T)$. Let $G'$ be the edges
          subgraph of $G^*$ containing only edges satisfied by all $v_1, ..., v_k$.
       ii. For every $t \in [k]$, the set $S_t$ should either be empty or satisfy both
           of the followings:
           A. $S_t$ is a union of connected components of $G'$.
           B. $|S_t| < \frac{n}{k} + \alpha n$.
       If no such partition exists, return failure.
3. Return the word $v^* \in \Sigma^n$ created as follows: For every $t \in [k]$ and $i \in S_t$,
   set $v^*_i = (v_t)_i$.
```

Figure 3: create − special − word algorithm

Proof of Lemma 16. We first bound the term $k\epsilon$. Recall that $\epsilon < 10^{-10}$, and get

$$k\epsilon = \left\lfloor \beta^{-1} \right\rfloor \epsilon \leq \left(\sqrt{\epsilon}\right)^{-1} \epsilon = \sqrt{\epsilon} < 10^{-5}.$$

Denote by Acc the event that T accepts v^*, and by $\overline{Acc}$ the event that T rejects v^*. We consider the following two cases, and show that in both cases $\Pr\left[Acc\right] > 10^{-5}$.

Case 1: $\Pr\left[\overline{Q_2}\right] > \frac{1}{2}$ (Partition according to Step 2a).

Due to T's completeness, and since v^* is a hybrid of the k codewords $v_1, ..., v_k$, it holds that

$$\Pr\left[\overline{Q_2} \wedge \overline{Acc}\right] \leq \sum_{t \in [k]} \Pr\left[T^{v_t} = \texttt{reject}\right] \leq k\epsilon < 10^{-5}.$$

Therefore,

$$\Pr\left[Acc\right] \geq \Pr\left[\overline{Q_2} \wedge Acc\right] = \Pr\left[\overline{Q_2}\right] - \Pr\left[\overline{Q_2} \wedge \overline{Acc}\right] > \frac{1}{2} - 10^{-5} > 10^{-5}.$$

Case 2: $\Pr\left[Q_2\right] \geq \frac{1}{2}$ (Partition according to Step 2b).

The conditions of Lemma 4 and Claim 12 are satisfied. Lemma 4 suggests that G^* contains more than $2 \cdot 10^{-4}$ fraction of G_T's edges. Using Claim 12, for every $t \in [k]$, the codeword v_t violates at most 2ϵ fraction of G_T's edges. Therefore,

$$\left|E\left(G'\right)\right| > \left(2 \cdot 10^{-4} - k \cdot 2\epsilon\right) e_T > \left(2 \cdot 10^{-4} - 2 \cdot 10^{-5}\right) e_T > 10^{-4} e_T.$$

That is,

$$\Pr\left[Acc \mid Q_2\right] > 10^{-4}.$$

Conclude,

$$\Pr\left[Acc\right] \geq \Pr\left[Q_2 \wedge Acc\right] = \Pr\left[Acc \mid Q_2\right] \cdot \Pr\left[Q_2\right] > 10^{-4} \cdot \frac{1}{2} > 10^{-5}.$$

□

Proof of Lemma 17. Let v be a codeword. We consider the following two cases, and show that in both cases $\left|agree\left(v^*, v\right)\right| < 3\beta n$.

Case 1: For every $t \in [k]$, it holds that $v \neq v_t$.

Since v^* is a hybrid of the codewords $v_1, ..., v_k$, it follows that

$$\left|agree\left(v^*, v\right)\right| \leq \sum_{t=1}^{k} \left|agree\left(v_t, v\right)\right|.$$

The code C has relative distance at least $1 - \alpha$. Therefore, since v and v_t are different codewords for every $t \in [k]$, it holds that

$$\left|agree\left(v^*, v\right)\right| \leq k\alpha n \leq \left(\beta^{-1}\right) \beta^2 n \leq \beta n.$$

Case 2: There exists $r \in [k]$ such that $v = v_r$.

We claim that $|S_r| < \frac{n}{k} + \alpha n$. The claim holds for the case that $\Pr\left[Q_2\right] \geq \frac{1}{2}$ by definition. Consider the case that $\Pr\left[\overline{Q_2}\right] > \frac{1}{2}$. Since $|C| > |\Sigma|$ there exists two different codewords that agree on at least one coordinate. Therefore, $\alpha \geq \frac{1}{n}$. Conclude, $|S_r| \leq \lceil \frac{n}{k} \rceil < \frac{n}{k} + 1 \leq \frac{n}{k} + \alpha n$.

As before, since v^* is a hybrid of the codewords $(v_t)_{t\in[k]\setminus\{r\}}$ and v, it follows that

$$\begin{aligned} |agree\,(v^*, v)| &\leq \left(\sum_{t\in[k]\setminus\{r\}} |agree\,(v_t, v)| \right) + |S_r| \\ &< (k-1)\,\alpha n + \left(\frac{n}{k} + \alpha n \right) \\ &= k\alpha n + \frac{n}{k} \\ &\leq \beta n + \frac{n}{\lfloor \beta^{-1} \rfloor}. \end{aligned}$$

Recall that $\beta < 10^{-5}$. Therefore, $\lfloor \beta^{-1} \rfloor > \beta^{-1} - 1 > \frac{1}{2}\beta^{-1}$. Conclude,

$$|agree\,(v^*, v)| < \beta n + \frac{n}{\frac{1}{2}\beta^{-1}} = 3\beta n.$$

□

References

[1] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

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

[3] Sanjeev Arora and Madhu Sudan. Improved low-degree testing and its applications. *Combinatorica*, 23(3):365–426, 2003.

[4] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[5] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of proximity, shorter PCPs, and applications to coding. *SIAM Journal on Computing*, 36(4):889–974, 2006.

[6] Eli Ben-Sasson, Oded Goldreich, and Madhu Sudan. Bounds on 2-query codeword testing. *RANDOM-APPROX*, pages 216–227, 2003.

[7] Shuchi Chawla, Robert Krauthgamer, Ravi Kumar, Yuval Rabani, and D. Sivakumar. On the hardness of approximating multicut and sparsest-cut. *Computational Complexity*, 15(2):94–114, 2006.

[8] Irit Dinur, Eldar Fischer, Guy Kindler, Ran Raz, and Shmuel Safra. PCP characterizations of NP: towards a polynomially-small error-probability. *STOC*, pages 29–40, 1999.

[9] Irit Dinur, Elchanan Mossel, and Oded Regev. Conditional hardness for approximate coloring. *STOC*, pages 344–353, 2006.

[10] Uriel Feige, Shafi Goldwasser, Laszlo Lovász, Shmuel Safra, and Mario Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268–292, 1996.

[11] Oded Goldreich and Madhu Sudan. Locally testable codes and PCPs of almost-linear length. *Journal of the ACM*, 53(4):558–655, 2006.

[12] Venkatesan Guruswami. On 2-query codeword testing with near-perfect completeness. *ISAAC*, pages 267–276, 2006.

[13] Subhash Khot. On the power of unique 2-prover 1-round games. *STOC*, pages 767–775, 2002.

[14] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? *Journal of the ACM*, 37(1):319–357, 2007.

[15] Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2-ϵ. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.

[16] Subhash Khot and Nisheeth K. Vishnoi. The unique games conjecture, integrality gap for cut problems and embeddability of negative type metrics into ℓ_1. *FOCS*, pages 53–62, 2005.

[17] Dana Moshkovitz and Ran Raz. Sub-constant error probabilistically checkable proof of almost linear size. *ECCC Report TR07-026*, 2007.

[18] Dana Moshkovitz and Ran Raz. Sub-constant error low degree test of almost-linear size. *SIAM Journal on Computing*, 38(1):140–180, 2008.

[19] Dana Moshkovitz and Ran Raz. Two query PCP with sub-constant error. *FOCS*, pages 314–323, 2008.

[20] Elchanan Mossel, Ryan O'Donnell, and Krzysztof Oleszkiewicz. Noise stability of functions with low influences: invariance and optimality. *FOCS*, pages 21–30, 2005.

[21] Prasad Raghavendra. Optimal algorithms and inapproximability results for every CSP? *STOC*, pages 245–254, 2008.

[22] Ran Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, 1998.

[23] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. *STOC*, pages 475–484, 1997.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de