Dense locally testable codes cannot have constant rate and distance

Irit Dinur and Tali Kaufman

[1] Weizmann Institute, Rehovot, ISRAEL.
irit.dinur@weizmann.ac.il[⋆]
[2] Bar Ilan University, Ramat Gan and Weizmann Institute, Rehovot, ISRAEL.
kaufmant@mit.edu[⋆⋆]

Abstract. A q-query locally testable code (LTC) is an error correcting code that can be tested by a randomized algorithm that reads at most q symbols from the given word. An important question is whether there exist LTCs that have the c^3 property: constant rate, constant relative distance, and that can be tested with a constant number of queries. Such LTCs are sometimes referred to as "asymptotically good".
We show that *dense* LTCs cannot be c^3. The *density* of a tester is roughly the average number of distinct local views in which a coordinate participates. An LTC is *dense* if it has a tester with density $\omega(1)$.
More precisely, we show that a 3-query locally testable code with a tester of density $\omega(1)$ cannot be c^3. Furthermore, we show that a q-locally testable code ($q > 3$) with a tester of density $\omega(1)n^{q-2}$ cannot be c^3. Our results hold when the tester has the following two properties:
- (no weights:) Every q-tuple of queries occurs with the same probability.
- ('last-one-fixed':) In every q-query 'test' of the tester, the value to any $q-1$ of the symbols determines the value of the last symbol. (Linear codes have constraints of this type).

We also show that several natural ways to quantitatively improve our results would already resolve the general c^3 question, i.e. also for non-dense LTCs.

1 Introduction

An error correcting code is a set $\mathcal{C} \subset \Sigma^n$. The rate of the code is $\log|\mathcal{C}|/n$ and its (relative) distance is the minimal Hamming distance between two different codewords $x, y \in \mathcal{C}$, divided by n. We only consider codes with distance $\Omega(1)$.

A code is called *locally testable* with q queries if it has a *tester*, which is a randomized algorithm with oracle access to the received word x. The tester reads at most q symbols from x and based on this local view decides if $x \in \mathcal{C}$ or not. It should accept codewords with probability one, and reject words that are far (in

⋆ Work supported by ISF grant 1179/09, BSF grant 2008293, and ERC starting grant 239985.
⋆⋆ Work supported by the Alon fellowship.

Hamming distance) from the code with noticeable probability. The tester has parameters (τ, ε) if

$$\forall x \in \Sigma^n,\ \operatorname{dist}(x, \mathcal{C}) \geq \tau \qquad \Longrightarrow \qquad \Pr[\text{Tester rejects } x] \geq \varepsilon$$

Locally Testable Codes (henceforth, LTCs) have been studied extensively in recent years. A priori, even the existence of LTCs is not trivial. The Hadamard code is a celebrated example of an LTC, yet it is highly "inefficient" in the sense of having very low rate ($\log n/n$). Starting with the work of Goldreich and Sudan [GS06], several other efficient constructions of LTCs have been given. The best known rate for LTCs is $1/\log^{O(1)} n$, and these codes have 3-query testers [BS05,Din07,Mei09]. The failure to construct constant rate, constant distance LTCs testable with constant number of queries leads to one of the main open questions in the area: are there LTCs that are c^3, i.e. $\underline{c}$onstant rate $\underline{c}$onstant distance and testable with a $\underline{c}$onstant number of queries (such LTCs are sometimes called in the literature "asymptotically good"). The case of two queries with LOF tests has been studied in [BSGS03]. However, the case of $q \geq 3$ is much more interesting and still quite open.

Dense testers. In this work we make progress on a variant of the c^3 question. We show that LTCs with so-called dense testers, cannot be c^3.

The density of a tester is roughly the average number, per-coordinate, of distinct local views that involve that coordinate. More formally, every tester gives rise to a constraint-hypergraph $H = ([n], E)$ whose vertices are the n coordinates of the codeword, and whose hyperedges correspond to all possible local views of the tester. Each hyperedge $h \in E$ is also associated with a constraint, i.e. with a Boolean function $f_h : \Sigma^q \to \{0, 1\}$ that determines whether the tester accepts or rejects on that local view. For a given string $x \in \Sigma^n$, we denote by x_h the substring obtained by restricting x to the coordinates in the hyperedge h. The value of $f_h(x_h)$ determines if the string x falsifies the constraint or not.

Definition 1 (The test-hypergraph, density). *Let $C \subseteq \Sigma^n$ be a code, let $q \in \mathbb{N}$ and $\varepsilon > 0$.*

Let H be a constraint hypergraph with hyperedges of size at most q. H is an (τ, ϵ)-test-hypergraph for $\mathcal{C}$ if

- *For every $x \in \mathcal{C}$ and every $h \in E$, $f_h(x_h) = 1$*
- *For every $x \in \Sigma^n$,*

$$\operatorname{dist}(x, \mathcal{C}) \geq \tau \quad \Rightarrow \quad \Pr_{h \in E}[f_h(x_h) = 0] \geq \epsilon$$

where dist(x, y) *denotes relative Hamming distance, i.e., the fraction of coordinates on which x differs from y.*

Finally, the density *of H is simply the average degree, $|E|/n$.*

The hypergraph describes a tester that selects one of the hyperedges uniformly at random. Observe that we disallow weights on the hyperedges. This will be discussed further below.

Goldreich and Sudan [GS06] proved that every tester with density $\Omega(1)$ can be made into a "sparse" tester with density $O(1)$ by randomly eliminating each hyper-edge with suitable probability. This means that a code can have both dense and sparse testers at the same time. Hence, we define a code to have *density* $\geq d$ if it has a tester with density d. We stress, that *density* in this work is a property of the code rather than of the tester. In this work we show that the existence of certain dense testers restricts the rate of the code.

We say that an LTC is *sparse* if it has no tester whose density is $\omega(1)$. We do not know of any LTC that is sparse. Thus, our work here provides some explanation for the bounded rate that known LTCs achieve.

In fact, one wonders whether density is an inherent property of LTCs. The intuition for such a claim is that in order to be locally testable the code seems to require a certain redundancy among the local tests, a redundancy which might be translated into density. If one were to prove that every LTC is dense, then it would rule out, by combination with our work, the existence of c^3-LTCs[3].

In support of this direction we point to the work of the second author with co-authors (Ben-Sasson et al [BSGK$^+$10]) where it is shown that every linear LTC (even with bounded rate) must have some non-trivial density: They show that no linear LTC can be tested only with tests that form a basis to the dual code. Moreover, any tester must have a significantly larger number of tests. Namely some constant density is required in every tester of such an LTC.

1.1 Our results

We bound the rate of LTCs with dense testers. We only consider testers whose constraints have the "last-one-fixed" (LOF) property, i.e. the value to any $q-1$ symbols determine the value of the last symbol. Note for instance that any linear constraint has this property.

We present different bounds for the case $q = 3$ and the case $q > 3$ where q denotes the number of queries.

Theorem 1. *Let $C \subseteq \{0,1\}^n$ be a 3-query LTC with distance δ, and let H be an $(\delta/3, \varepsilon)$-test-hypergraph with density d and LOF constraints. Then, the rate of C is at most $O(1/d^{1/2})$ (where the O-notation hides dependencies on δ, ε).*

For the case of $q > 3$ queries we have the following result whose proof, which is similar to the $q = 3$ case, is omitted due to space limitations.

Theorem 2. *Let $C \subseteq \{0,1\}^n$ be a q-query LTC with distance δ, and let H be an $(\delta/2, \varepsilon)$-test-hypergraph with density Δ, where $\Delta = dn^{q-2}$, and LOF constraints. Then, the rate of C is at most $O(1/d)$.*

[3] This is slightly imprecise, since our results only hold for uniform and LOF testers. So for this approach to work one would need to remove these extra conditions from our results.

Extensions. In this preliminary version we assume that the alphabet is Boolean, but the results easily extend to any finite alphabet Σ. It may also be possible to get rid of the "last-one-fixed" restriction on the constraints, but this remains to be worked out.

Improvements. We show that several natural ways of improving our results will already resolve the 'bigger' question of ruling out c^3-LTCs.

- In this work we only handle *non-weighted* testers, i.e., where the hypergraph has no weights. In general a tester can put different weights on different hyperedges. This is sometimes natural when combining two or more "types" of tests each with certain probability. This limitation cannot be eliminated altogether, but may possibly be addressed via a more refined definition of density. See further discussion Section 3.3.
- In Theorem 1 we prove that $\rho \leq O(1/d^{0.5})$. We show that any improvement of the 0.5 exponent (say to $0.5 + \varepsilon$) would again rule out the existence of c^3-LTCs, see Lemma 4
- In Theorem 2 we bound the rate only when the density is very high, namely, $\omega(n^{q-2})$. We show, in Lemma 5, that any bound for density $O(n^{q-3})$ would once more rule out the existence of c^3-LTCs. It seems that our upper bound of $\omega(n^{q-2})$ can be improved to $\omega(n^{q-3})$, possibly by arguments similar to those in the proof of Theorem 1.

Related work. In the course of writing our result we learned that Eli Ben-Sasson and Michael Viderman have also been studying the connection between density and rate and have obtained related results, through seemingly different methods.

2 Moderately dense 3-query LTCs cannot be c^3

In this section we prove Theorem 1 which we now recall:

Theorem 1. *Let $C \subseteq \{0,1\}^n$ be a 3-query LTC with distance δ, and let H be an $(\delta/3, \varepsilon)$-test-hypergraph with density d and LOF constraints. Then, the rate of C is at most $O(1/d^{1/2})$.*

In order to prove the main theorem, we consider the hypergraph $H = (V, E(H))$ whose vertices are the coordinates of the code, and whose hyper-edges correspond to the different tests of the tester. By assumption, H has dn distinct hyper-edges. We describe an algorithm in Figure 1 for assigning values to coordinates of a codeword, and show that a codeword is determined using $k = O(\frac{n}{d^{1/2}})$ bits.

We need the following definition. For a partition (A, B) of the vertices V of H, we define the graph $G_B = (A, E)$ where

$$E = \{\{a_1, a_2\} \subset A \mid \ \exists b \in B,\ \{a_1, a_2, b\} \in E(H)\}.$$

A single edge $\{a_1, a_2\} \in E(G_B)$ may have more than one "preimage", i.e., there may be two (or more) distinct vertices $b, b' \in B$ such that both hyper-edges $\{a_1, a_2, b\}, \{a_1, a_2, b'\}$ are in H. For simplicity we consider the case where the constraints are linear[4] which implies that for every codeword $w \in C$: $w_b = w_{b'}$. This is a source of some complication for our algorithm, which requires the following definition.

Definition 2. *Two vertices v, v' are* equivalent *if*

$$\forall w \in C, \qquad w_v = w_{v'}.$$

Clearly this is an equivalence relation. A vertex has multiplicity *m if there are exactly m vertices in its equivalence class. For the first read, the reader may assume that all multiplicities are* 1*.*

Denote by V^* the set of vertices whose multiplicity is at most $\beta d^{1/2}$ for $\beta = \alpha/16$ where $\alpha = 3\varepsilon/\delta$.

```
0. Init: Let $\alpha = 3\varepsilon/\delta$ and fix $\beta = \alpha/16$. Let $B$ contain all vertices with multiplicity at least $\beta d^{1/2}$. Let $F$ contain a representative from each of these multiplicity classes. Let $B$ also contain all vertices whose value is the same for all codewords.
1. Clean: Repeat the following until $B$ remains fixed:
   (a) Add to $B$ any vertex that occurs in a hyper-edge that has two endpoints in $B$.
   (b) Add to $B$ all vertices in a connected component of $G_B$ whose size is at least $\beta d^{1/2}$, and add an arbitrary element in that connected component into $F$.
   (c) Add to $B$ any vertex that has an equivalent vertex in $B$.
2. S-step: Each vertex outside $B$ tosses a biased coin and goes into $S$ with probability $1/d^{1/2}$. Let $B \leftarrow B \cup S$ and set $F \leftarrow F \cup S$.
3. If there are at least two distinct $x, y \in C$ such that $x_B = y_B$ goto step 1, otherwise halt.
```

Fig. 1. The Algorithm

The following lemma is easy.

Lemma 1. *If the algorithm halted, the code has at most $2^{|F|}$ words.*

Proof. This follows since at each step setting the values to vertices in F already fully determines the value of all vertices in B (in any valid codeword). Once the algorithm halts, the values of a codeword on B determines the entire codeword. Thus, there can be at most $2^{|F|}$ codewords.

Let B_t denote the set B at the end of the t-th Clean step (i.e. we refer to the main loop over steps 1a,1b,1b). In order to analyze the expected size of F when the algorithm halts, we analyze the probability that vertices not yet in B

[4] More generally, when the constraints are LOF the set of all such b's can be partitioned into all those equal to w_b and all those equal to $1 - w_b$.

will go into B on the next iteration. For a vertex v, this is determined by its neighborhood structure. Let

$$E_v = \{\{u, u'\} \mid \; u, u' \in V^*, \text{ and } \{u, u', v\} \in E(H)\}$$

be a set of edges. Denote by A the vertices v with large $|E_v|$,

$$A = \{v \mid \; |E_v| \geq \alpha d\}\,.$$

The following lemma (whose proof appears in the next section) says that if v has sufficiently large E_v then it is likely to enter B in the next round:

Lemma 2. *For $t \geq 2$, if $v \in A$ then*

$$\Pr_S[v \in B_t] \geq \frac{1}{2}.$$

Next, consider a vertex $v \notin A$ that is adjacent, in the graph $G_{B_{t-1}}$, to a vertex $v' \in A$. This means that there is a hyper-edge $h = \{v, v', b\}$ where $b \in B_{t-1}$. If it so happens that $v' \in B_t$ (and the above lemma guarantees that this happens with probability $\geq \frac{1}{2}$), then the hyper-edge h would cause v to go into B_t as well. In fact, one can easily see that if v goes into B_t then all of the vertices in its connected component in $G_{B_{t-1}}$ will go into B_t as well (via step 1a). Let A_t be the set of vertices outside B_t that are in A or are connected by a path in G_{B_t} to some vertex in A. We have proved

Corollary 1. *For $t \geq 2$, let $v \in A_{t-1}$ then*

$$\Pr_S[v \in B_t] \geq \frac{1}{2}.$$

□

Lemma 3. *If the algorithm hasn't halted before the t-th step and $|A_{t-1}| < \frac{\delta}{2} n$ then the algorithm will halt at the end of the t-th step.*

Before proving the two lemmas, let us see how they imply the theorem.

Proof. (of theorem 1) For each $t \geq 2$, Corollary 1 implies that for each $v \in A_{t-1}$ half of the S's put it in B_t. We can ignore the sets S whose size is above $2 \cdot n/d^{1/2}$, as their fraction is negligible. By linearity of expectation, we expect at least half of A_{t-1} to enter B_t. In particular, fix some S_{t-1} to be an S that attains (or exceeds) the expectation. As long as $|A_{t-1}| \geq \delta n/2$ we get

$$|B_t| \geq |B_{t-1}| + |A_{t-1}|/2 \geq |B_{t-1}| + \delta n/4.$$

Since $|B_t| \leq n$ after $\ell \leq 4/\delta$ iterations when the algorithm runs with $S_1, \ldots, S_\ell$ we must have $|A_\ell| < \delta n/2$. This means that the conditions of Lemma 3 hold, and the algorithm halts.

How large is the set F? In each S-step the set F grew by $|S| \leq 2n/d^{1/2}$ (recall we neglected S's that were larger than that). The total number of vertices that were added to F in S-steps is thus $O(\ell \cdot n/d^{1/2})$.

Other vertices are added into F in the init step and in step 1b. In both of these steps one vertex is added to F for every $\beta d^{1/2}$ vertices outside B that are added into B. Since vertices never exit B, the total number of this type of F-vertices is $n/(\beta d^{1/2})$.

Altogether, with non-zero probability, the final set F has size $O(\frac{1}{d^{1/2}}) \cdot n$. Together with Lemma 1 this gives the desired bound on the number of codewords and we are done.

We now prove the two lemmas.

2.1 Proof of Lemma 2

We fix some $v \in A$. If $v \in B_{t-1}$ then we are done since $B_t \supseteq B_{t-1}$. So assume $v \notin B_{t-1}$ and let us analyze the probability of v entering B_t over the random choice of the set S at iteration $t-1$. This is dictated by the graph structure induced by the edges of E_v. Let us call this graph $G = (U, E_v)$, where U contains only the vertices that touch at least one edge of E_v. We do not know how many vertices participate in U, but we know that $|E_v| \geq \alpha d$.

We begin by observing that all of the neighbors of $u \in U$ must be in the same equivalence class[5]. Indeed each of the edges $\{v, u, u_i\}$ is a hyper-edge in H and the value of u_i is determined by the values of v and u. Therefore, the degree in G of any vertex $u \in U$ is at most $\beta d^{1/2}$, since vertices with higher multiplicity are not in V^* and therefore do not participate in edges of E_v.

For each $u \in U$ let I_u be an indicator variable that takes the value 1 iff there is a neighbor of u that goes into S. If this happens then either

– $u \in S$: this means that v has a hyperedge whose two other endpoints are in B_t and will itself go into B_t (in step 1a).
– $u \notin S$: this means that the graph G_{B_t} will have an edge $\{v, u\}$.

If the first case occurs for any $u \in U$ we are done, since v goes into B_t in step 1a. Otherwise, the random variable $\sum_{u \in U} I_u$ counts how many distinct edges $\{v, u\}$ will occur in G_{B_t}. If this number is above $\beta d^{1/2}$ then v will go into B_t (in step 1b) and we will again be done. It is easy to compute the expected value of I. First, observe that

$$\mathbb{E}[I_u] = 1 - (1 - 1/d^{1/2})^{deg(u)}$$

where $deg(u)$ denotes the degree of u in G and since the degree of u is at most $\beta d^{1/2}$, this value is between $deg(u)/2d^{1/2}$ and $deg(u)/d^{1/2}$. By linearity of expectation

$$\mathbb{E}[I] = \sum_u \mathbb{E}[I_u] \geq \sum_u deg(u)/2d^{1/2} = |E_v|\, d^{-1/2} \geq \alpha d^{1/2}.$$

[5] Or, more generally for LOF constraints, in one of a constant number of equivalence classes.

We will show that I has good probability of attaining a value near the expectation (and in particular at least $\alpha d^{1/2}/2 \geq \beta d^{1/2}$), and this will put v in B_t at step 1b. The variables I_u are not mutually independent, but we will be able to show sufficient concentration by bounding the variance of I, and applying Chebychev's inequality.

The random variables I_u and $I_{u'}$ are dependent exactly when u, u' have a common neighbor (the value of I_u depends on whether the neighbors of u go into S). We already know that having a common neighbor implies that u, u' are in the same multiplicity class. Since $U \subset V^*$, this multiplicity class can have size at most $\beta d^{1/2}$. This means that we can partition the vertices in U according to their multiplicity class, such that I_u and $I_{u'}$ are fully independent when u, u' are from distinct multiplicity classes. Let $u_1, \ldots, u_t$ be representatives of the multiplicity classes, and let $d_i \leq \beta d^{1/2}$ denote the size of the ith multiplicity class. Also, write $u \sim u'$ if they are from the same multiplicity class.

$$\begin{aligned}
Var[I] = \mathbb{E}[I^2] - (\mathbb{E}[I])^2 &= \mathbb{E}\sum_{u,u'} I_u I_{u'} - \sum_{u,u'} \mathbb{E}I_u \mathbb{E}I_{u'} \\
&= \sum_{u\sim u'} \mathbb{E}[I_u I_{u'}] + \sum_{u\not\sim u'} \mathbb{E}I_u \mathbb{E}I_{u'} - \sum_{u,u'} \mathbb{E}I_u \mathbb{E}I_{u'} \\
&\leq \sum_i \sum_{u\sim u_i} \sum_{u'\sim u_i} \mathbb{E}I_u I_{u'} \\
&\leq \sum_i \sum_{u\sim u_i} \sum_{u'\sim u_i} \mathbb{E}I_u \cdot 1 \\
&\leq \sum_i \sum_{u\sim u_i} \mathbb{E}I_u \cdot d_i \leq \sum_i \sum_{u\sim u_i} \frac{deg(u)}{d^{1/2}} \cdot \beta d^{1/2} \\
&= \beta \sum_u deg(u) = 2\beta\,|E_v|
\end{aligned}$$

By Chebychev's inequality,

$$\Pr[|I - \mathbb{E}[I]| \geq a] \leq Var[I]/a^2$$

Plugging in $a = \mathbb{E}[I]/2$ we get

$$\Pr\left[|I - \mathbb{E}[I]| \geq \frac{\mathbb{E}[I]}{2}\right] \leq \frac{Var[I]}{(\mathbb{E}[I]/2)^2} \leq (2\beta\,|E_v|)\cdot((\frac{1}{2}\,|E_v|\,d^{-1/2})^2)^{-1} \leq 8\beta d/\,|E_v| \leq 8\beta/\alpha.$$

and so by choosing $\beta = \alpha/16$ this probability is at most a half. Thus, the probability that $I \geq \mathbb{E}I/2 \geq \alpha d^{1/2}/2$ is at least a half. As we said before, whenever $I \geq \beta d^{1/2}$ we are guaranteed that v will enter B_t in the next Clean step 1b and we are done. □

2.2 Proof of Lemma 3

We shall prove that if the algorithm hasn't halted before the t-th step and $|A_{t-1}| < \frac{\delta}{2}n$ then $|B_t| > (1-\delta)n$. This immediately implies that the algorithm must halt because after fixing values to more than $1-\delta$ fraction of the coordinates of a codeword, there is a unique way to complete it.

Recall that A is the set of all vertices v for which $|E_v| \geq \alpha d$. The set B_{t-1} is the set B in the algorithm after the $t-1$-th Clean step. The set A_{t-1} is the set of vertices outside B_{t-1} that are connected by a path in $G_{B_{t-1}}$ to some vertex in A. Finally, denote $G = G_{B_{t-1}}$.

Assume for contradiction that $|B_t| \leq (1-\delta)n$ and $|A_{t-1}| < \delta n/2$. This means that $Z = V \setminus (A_{t-1} \cup B_t)$ contains more than $\delta n/2$ vertices. Since $Z \cap A_{t-1} = \phi$, every vertex $v \in Z$ has $|E_v| < \alpha d$. Our contradiction will come by finding a vertex in Z with large E_v. If the algorithm doesn't yet halt, there must be two distinct codewords $x, y \in C$ that agree on B_t. Let $U_{x\neq y} = \{u \in V \mid \ x_u \neq y_u\}$. This is a set of size at least δn that is disjoint from B_t. Since $|A_t| \leq \delta n/2$ there must be at least $\delta n/2$ vertices in $Z \cap U_{x\neq y}$. Suppose $u \in Z \cap U_{x\neq y}$ and suppose u' is adjacent to u in G. First, by definition of Z, $u \in Z$ implies $u' \in Z$. Next, we claim that $u \in U_{x\neq y}$ implies $u' \in U_{x\neq y}$. Otherwise there would be an edge $\{u, u', b\} \in E(H)$ such that $b \in B_t$, and such that $x_u \neq y_u$ but both $x_{u'} = y_{u'}$ and $x_b = y_b$. This means that either x or y must violate this edge, contradicting the fact that all hyper-edges should accept a legal codeword. We conclude that the set $Z \cap U_{x\neq y}$ is a union of connected components of G. Since each connected component has size at most $\beta d^{1/2}$ (otherwise it would have gone into B in a previous Clean step) we can find a set $D \subset Z \cap U_{x\neq y}$ of size s, for

$$\frac{\delta}{3}n \leq \frac{\delta}{2}n - \beta d^{1/2} \leq s \leq \frac{\delta}{2}n,$$

that is a union of connected components, i.e. such that no G-edge crosses the cut between D and $V \setminus D$. Now define the hybrid word

$$w = x_D y_{V\setminus D}$$

that equals x on D and y outside D. We claim that $dist(w, C) = dist(w, y) = |D|/n \geq \delta/3$. Otherwise there would be a word $z \in C$ whose distance to w is strictly less than $|D|/n \leq \delta/2$ which, by the triangle inequality, would mean it is less than δn away from y thereby contradicting the minimal distance δn of the code.

Finally, we use the fact that C is an LTC,

$$\mathrm{dist}(w, C) \geq \delta/3 \qquad \Longrightarrow \qquad Prob_{h\sim E(H)}[h \text{ rejects } w] \geq \varepsilon.$$

Clearly to reject w a hyperedge must touch D. Furthermore, such a hyperedge cannot intersect B on 2 vertices because then the third non-B_t vertex also belongs to B_t. It cannot intersect B_t on 1 vertex because this means that either the two other endpoints are both in D, which is imopssible since such a hyperedge would reject the legal codeword x as well; or this hyperedge induces an edge in G that crosses the cut between D and $V \setminus D$. Thus, rejecting hyper-edges must not intersect B_t at all.

Altogether we have εdn rejecting hyperedges spanned on $V \setminus B_t$ such that each one intersects D. This means that there must be some vertex $v \in D$ that touches at least $\varepsilon dn/(\delta n/3) = \alpha d$ rejecting hyperedges. Recall that $D \subset Z$ is disjoint from A, which means that $|E_v| < \alpha d$. On the other hand, each rejecting hyperedge touching v must add a distinct edge to E_v. Indeed recall that E_v contains an edge $\{u, u'\}$ for each hyperedge $\{u, u', v\}$ such that $u, u' \in V^*$ and where V^* is the set of vertices with multiplicity at most $\beta d^{1/2}$. The claim follows since obviously all of the αd rejecting hyperedges are of this form (they do not contain a vertex of high multiplicity as these vertices are in B). $\square$

3 Exploring possible improvements

3.1 Tradeoff between rate and density

Any improvement over our bound of $\rho < 1/d^{1/2}$, say to a bound of the form $\rho < 1/d^{0.501}$ would already be strong enough to rule out c^3-LTCs (with a non-weighted tester) regardless of their density. The reason for this is the following reduction by Oded Goldreich.

Lemma 4. *Suppose for some $q \geq 3$ and some $\epsilon, \delta > 0$ the following were true.*

> *For any family $\{C_n\}$ of q-query LTCs with rate $\leq \rho$ distance $\delta > 0$ and such that each C_n has a tester with density at least d, then $\rho \leq 1/d^{\frac{1}{q-1}+\epsilon}$.*

Then, there is no family of q-query LTCs with constant rate, distance $\delta > 0$, and any density, such that the tester is non-weighted.

Proof. Let $\beta = \frac{1}{q-1} + \epsilon$, and let $t \in \mathbb{N}$. Let $\{C_i\}$ be an infinite family of q-query LTCs with density $d = O(1)$, relative rate $\rho = \Omega(1)$, and distance $\delta > 0$. Then there is another infinite family $\left\{\tilde{C_i}\right\}$ of q-query LTCs with density $d \cdot t^{q-1}$, same distance, and relative rate ρ/t. $\tilde{C_i}$ is constructed from C_i by duplicating each coordinate t times and replacing each test hyper-edge by t^q hyperedges. Clearly the density and the rate are as claimed. The testability can also be shown. Plugging in the values $\tilde{\rho} = \rho/t$ and $\tilde{d} = dt^{q-1}$ into the assumption we get

$$\rho/t = \tilde{\rho} \leq 1/\tilde{d}^{\beta} = 1/(dt^{q-1})^{\beta}$$

In other words $\rho d^{\beta} \leq t^{1-(q-1)\beta}$. Since t is unbounded this can hold only if the exponent of t is positive, i.e., $\beta \leq 1/(q-1)$, a contradiction.

3.2 For $q > 3$ density must be high

Lemma 5. *Let C be a q-query LTC with rate ρ, and density d. Then for every $q' > 0$ there is a $(q+q')$-query LTC C' with density $d \cdot \binom{n}{q'}$ such that C' has rate $\rho/2$, distance $\delta/2$.*

Corollary 2. *If there is a 3-query LTC with constant rate and density, then there are LTCs with $q > 3$-queries, constant rate, and density $\Omega(n^{q-3})$.*

The corollary shows that our upper bounds from Theorem 2 are roughly correct in their dependence on n, but there is still a gap in the exponent.

Proof. (of lemma 5) Imagine adding another n coordinates to the code C such that they are always all zero. Clearly the distance and the rate are as claimed. For the testability, we replace each q-hyper-edge e of the hypergraph of C with $\binom{n}{q'}$ new hyperedges that consist of the vertices of e plus any q' of the new vertices. The test associated with this hyperedge will accept iff the old test would have accepted, and the new vertices are assigned 0. It is easy to see that the new hypergraph has average degree $d \cdot \binom{n}{q'}$. Testability can be shown as well.

3.3 Allowing weighted hypergraph-tests

In this section we claim that when considering hypergraph tests *with weights*, the density should not be defined as the ratio between the number of edges and the number of vertices. Perhaps a definition that takes the min-entropy of the graph into consideration would be better-suited, but this seems elusive, and we leave it for future work.

We next show that if one defines the density like before (ignoring the weights) then every LTC can be modified into one that has a maximally-dense tester. This implies that bounding the rate as a function of the density is the same as simply bounding the rate.

Lemma 6. *Let C be a q-query LTC with $q \geq 3$, rate ρ, distance δ, and any density. Then there is another q-query LTC C' with a* weighted*-tester of maximal density $\Omega(n^{q-1})$ such that C' has rate $\rho/2$, distance $\delta/2$.*

Corollary 3. *Let $f : \mathbb{N} \to \mathbb{N}$ be any non-decreasing non-constant function. Any bound of the form $\rho \leq 1/f(d)$ for weighted testers implies $\rho \leq 1/f(n^{q-1})$, and in particular $\rho \to 0$.* □

Proof. (of lemma 6:) One can artificially increase the density of an LTC tester hypergraph H by adding n new coordinates to the code that are always zero, and adding all possible q-hyperedges over those coordinates (checking that the values are all-zero). All of the new hyper-edges will be normalized to have total weight one half, and the old hyperedges will also be re-normalized to have total weight one half. Clearly the rate and distance have been halved, and the testability is maintained (with a different rejection ratio). However, the number of hyperedges has increased to n^q so the density is as claimed.

Acknowledgement

We would like to thank Oded Goldreich for very interesting discussions, and for pointing out the reduction in Section 3.1.

References

[BS05] Eli Ben-Sasson and Madhu Sudan. Simple PCPs with poly-log rate and query complexity. In *Proc. 37th ACM Symp. on Theory of Computing*, pages 266–275, 2005.

[BSGK$^+$10] Eli Ben-Sasson, Venkatesan Guruswami, Tali Kaufman, Madhu Sudan, and Michael Viderman. Locally testable codes require redundant testers. *SIAM J. Comput.*, 39(7):3230–3247, 2010.

[BSGS03] Eli Ben-Sasson, Oded Goldreich, and Madhu Sudan. Bounds on 2-query codeword testing. In *RANDOM-APPROX*, pages 216–227, 2003.

[Din07] Irit Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3), 2007.

[GS06] Oded Goldreich and Madhu Sudan. Locally testable codes and PCPs of almost-linear length. *J. ACM*, 53(4):558–655, 2006.

[Mei09] Or Meir. Combinatorial construction of locally testable codes. *SIAM J. Comput.*, 39(2):491–544, 2009.