Alcom-FT Technical Report Series
ALCOMFT-TR-01-133

Testing Hypergraph Coloring

(Extended abstract)

Artur Czumaj* and **Christian Sohler†**

Abstract

In this paper we initiate the study of testing properties of hypergraphs. The goal of property testing is to distinguish between the case whether a given object has a certain property or is "far away" from the property. We prove that the fundamental problem of ℓ-colorability of k-uniform hypergraphs can be tested in time independent of the size of the hypergraph. We present a testing algorithm that examines only $(k\,\ell/\epsilon)^{\mathcal{O}(k)}$ entries of the adjacency matrix of the input hypergraph, where ϵ is a distance parameter independent of the size of the hypergraph. Notice that this algorithm tests only a constant number of entries in the adjacency matrix provided that ℓ, k, and ϵ are constant.

1 Introduction

A classical problem in computer science is to verify if a given object possesses a certain property. For example, we want to determine if a boolean formula is satisfiable, or if a graph is connected. In its very classical formulation, the goal is to give an exact solution to the problem, that is, to provide an algorithm that always returns a correct answer. In many situation, however, this formulation is too restrictive, for example, because there is no fast (or just fast enough) algorithm that gives the exact solution. Recently, many researchers started studying a relaxation of the "exact decision task" and considered various forms of approximation algorithms for decision problems. In *property testing* (see, e.g., [1, 11, 13, 17, 16, 18, 19, 26, 29]), one considers the following class of problems:

> Let $\mathfrak{C}$ be a class of objects, $\mathfrak{O}$ be an unknown object from $\mathfrak{C}$, and $\mathcal{Q}$ be a fixed property of objects from $\mathfrak{C}$. The goal is to determine (possibly probabilistically) if $\mathfrak{O}$ has property $\mathcal{Q}$ or if it is *far* from any object in $\mathfrak{C}$ which has property $\mathcal{Q}$, where distance between two objects is measured with respect to some distribution $\mathcal{D}$ on $\mathfrak{C}$.

The motivation behind this notion of property testing is that while relaxing the exact decision task we expect the testing algorithm to be significantly more efficient than any exact decision algorithm, and in many cases, we achieve this goal by exploring only a small part of the input.

A notion of property testing was first explicitly formulated in [31] and then extended and further developed in many follow-up works (see, e.g., [1, 6, 7, 13, 14, 18, 19, 30]). Property testing arises naturally in the context of program verification, learning theory, and, in a more theoretical setting, in probabilistically checkable proofs. For example, in the context of program checking, one may first

*Department of Computer and Information Science, New Jersey Institute of Technology, Newark, NJ 07102, USA. Email: czumaj@cis.njit.edu.

†Heinz Nixdorf Institute and Department of Mathematics & Computer Science, University of Paderborn, D-33095 Paderborn, Germany. Email: csohler@uni-paderborn.de. Research supported in part by DFG Grant Me872/7-1 and by the IST Programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

choose to test whether the program's output satisfies certain properties before checking that it is as desired. This approach is a very common practice in software development, where it is (typically) infeasible to require to formally test that a program is correct, but by verifying whether the output satisfies certain properties one can gain a reasonable confidence about the quality of the program's output.

The study of property testing for *combinatorial objects*, and mainly for labeled graphs, was initiated by Goldreich et al. [18]. They investigated several interesting graph properties and showed, for example, that testing ℓ-colorability of graphs is testable in time independent of the input size.

We refer the reader to the excellent survey by Ron [29], where a very thorough exposition of this field is presented and applications of this model are discussed.

1.1 Our contribution

In this paper we extend the notion of property testing to hypergraphs, and study the problem of *testing colorability properties of hypergraphs*.

Hypergraphs. Recall that a *hypergraph* is a pair $\mathcal{H} = (V, E)$ such that E is a subset of the power set of V. The set V is the set of *vertices* and E is the set of *edges*. We consider only finite hypergraphs (i.e., V is finite) and such that $V \cap E = \emptyset$. If E contains only sets of size k then $\mathcal{H}$ is said to be k-*uniform*. A hypergraph is a well-known generalization of a graph; a 2-uniform hypergraph is a standard undirected graph.

An ℓ-*coloring* of a hypergraph $\mathcal{H} = (V, E)$ is a mapping $\chi : V \to \{1, \dots, \ell\}$. An ℓ-coloring χ of a hypergraph $\mathcal{H} = (V, E)$ is called *proper* if $\mathcal{H}$ contains no monochromatic edge (that is, for every $e \in E$, there exist $x, y \in e$ such that $\chi(x) \neq \chi(y)$). A hypergraph $\mathcal{H}$ is ℓ-colorable, if there exists a proper ℓ-coloring of $\mathcal{H}$.

In the case when we will discuss the 2-coloring problem, we shall frequently consider χ to be a function that assigns to every vertex either color $\mathtt{red}$ or $\mathtt{blue}$.

Testing colorability property of hypergraphs. In this paper we study the problem of testing the property that a given hypergraph is ℓ-colorable. We assume the hypergraph $\mathcal{H} = (V, E)$ with n vertices is k-uniform and it is represented by its *adjacency matrix* A of size n^k, that is, the entry $A[v_{i_1}, v_{i_2}, \dots, v_{i_k}]$ is equal to 1 if and only if $\{v_{i_1}, v_{i_2}, \dots, v_{i_k}\} \in E$; it is 0 otherwise.

In general case, we are using the following definition:

Definition 1.1 *Let $\mathcal{P}$ be any property of hypergraphs. Let ϵ be any real $0 \le \epsilon \le 1$. A k-uniform hypergraph $\mathcal{H} = (V, E)$ is* ϵ-far from property $\mathcal{P}$ *if it has Hamming distance at least ϵn^k from any hypergraph having property $\mathcal{P}$, that is, in order to construct from $\mathcal{H}$ a hypergraph having property $\mathcal{P}$ one has to delete or insert at least ϵn^k edges of $\mathcal{H}$.*

Using this notion of distance, we can formally define testing algorithms:

Definition 1.2 *Let $\mathcal{P}$ be any property of hypergraphs. Let ϵ be any real $0 \le \epsilon \le 1$. An* ϵ-tester *for property $\mathcal{P}$ of k-uniform hypergraphs is an algorithm that*

- accepts *every hypergraph having property $\mathcal{P}$, and*
- rejects *with probability at least $2/3$ any hypergraph that is ϵ-far from property $\mathcal{P}$.*

Observe that the behavior of an ϵ-*tester* may be arbitrary for hypergraphs that neither have property $\mathcal{P}$ nor are ϵ-far from property $\mathcal{P}$.

Specifically, given query access to an adjacency matrix A representing $\mathcal{H}$, and a distance parameter ϵ, we study the problem of determining with reasonably high probability whether $\mathcal{H}$ is ℓ-colorable, or whether more than an ϵ-fraction of entries of A should be modified so that the hypergraph defined by the modified adjacency matrix becomes ℓ-colorable. In the later case, we say $\mathcal{H}$ is ϵ-far from being ℓ-colorable.

There are two measures of the complexity of testing algorithms: the *query complexity* and the *running time complexity* of an ϵ-tester. The query complexity of a tester (in our context of hypergraphs) is measured only by the number of queries to the entries of the adjacency matrix of the input hypergraph, while the running time complexity counts also the time needed by the algorithm to perform other tasks (e.g., to verify if a given sub-hypergraphs is ℓ-colorable).

To exemplify the notion of ϵ-testers, let us compare the notion of standard approximation of 2-colorability with the notion of testing 2-colorability in 3-uniform hypergraphs (this is a slight modification of an example used in [7]):

> *A hypergraph $\mathcal{H}$ might be* nearly *2-colorable in the sense that there is a 2-colorable hypergraph $\mathcal{H}^*$ at small Hamming distance to $\mathcal{H}$, but* far *from 2-colorable in the sense that many colors are required to properly color $\mathcal{H}$. Similarly, a hypergraph $\mathcal{H}$ might be* nearly *2-colorable in the sense that it is 3-colorable, but far from 2-colorable in the sense that no hypergraphs having small Hamming distance to $\mathcal{H}$ are 2-colorable. Therefore, both these notions are natural and the preferred choice depends on the application at hand.*

Results. Our main theorem is an ϵ-tester for ℓ-colorability of k-uniform hypergraphs that has query complexity that is independent of the input hypergraph size.

Our ϵ-tester follows the standard approach in this area: it first samples at random a subset of vertices of the hypergraph $\mathcal{H}$, and then checks whether the sub-hypergraph of $\mathcal{H}$ induced by the vertices chosen is colorable:

Tester(s, ℓ)

Pick a subset $\mathcal{S} \subseteq V$ of size s uniformly at random.

Let $\mathcal{H}_{\mathcal{S}}$ be the hypergraph induced by $\mathcal{S}$ in $\mathcal{H}$.

If *$\mathcal{H}_{\mathcal{S}}$ is ℓ-colorable* **then** accept $\mathcal{H}$;

else reject $\mathcal{H}$.

We can prove the following result.

Theorem 1 Tester(s, ℓ) *with* $s = \tilde{\mathcal{O}}((k\,\ell/\epsilon)^2)$ *is an ϵ-tester for ℓ-coloring k-uniform hypergraphs.*[1]

This immediately implies the following.

Theorem 2 *There is an ϵ-tester for ℓ-colorability of k-uniform hypergraphs that has query complexity $\tilde{\mathcal{O}}((k\,\ell/\epsilon)^{2k})$ and the running time of* $\exp(\tilde{\mathcal{O}}(k\,\ell/\epsilon)^2)$.

[1] $\tilde{\mathcal{O}}$ is a standard asymptotic notation that "hides" polylogarithmic factors.

1.2 Context and related work

Hypergraph coloring. Hypergraph coloring is a well studied problem in the literature in discrete mathematics, combinatorics, and computer science. In contrast to graphs, where one can decide in linear time if a graph is 2-colorable (or equivalently, bipartite), testing if a given hypergraph is 2-colorable is $\mathcal{NP}$-hard even for 3-uniform hypergraphs [23]. In [22], it is shown that unless $\mathcal{NP} \subseteq \mathcal{ZPP}$, for any fixed $k \geq 3$, it is impossible to approximate in polynomial time the chromatic number of k-uniform hypergraphs within a factor $n^{1-\varepsilon}$ for any constant $\varepsilon > 0$. Very recently, Guruswami et al. [20] proved that for any constant c, it is $\mathcal{NP}$-hard to color a 2-colorable 4-uniform hypergraph using c colors. In [20] even a stronger inapproximability result is shown, that there exists a constant c_0 such that, unless $\mathcal{NP} \subseteq \mathrm{DTIME}(n^{\mathcal{O}(\log\log n)})$, there is no polynomial time algorithm that colors a 2-colorable 4-uniform hypergraph using $c_0 \log\log\log n$ colors.

The property of hypergraph 2-colorability (called also "Property B" by Erdős) has been extensively studied in the combinatorics literature (see, e.g., [5, 10, 12, 27]). In particular, the study of this problem led to the discovery of the celebrated Lovász Local Lemma [12]. In computer science the problems of coloring hypergraphs have been studied mostly due to its connection to important graph coloring and satisfiability problems (cf., e.g., [9, 24]). Extending the approximation results for graph coloring, several authors have provided approximation algorithms for coloring 2-colorable hypergraphs [2, 8, 21, 22]. For example, the very recent polynomial-time approximation algorithm from [21] colors any 3-uniform 2-colorable hypergraphs using $\tilde{\mathcal{O}}(n^{1/5})$ colors.

Testing colorability. We are not aware of any prior testing algorithms for colorability of hypergraphs. However, very recently we have heard that, independently to our work, Alon and Shapira (personal communication, 2001) developed a testing algorithm for some general version of satisfiability that includes and testing ℓ-colorability of uniform hypergraphs.

Goldreich et al. [18] were the first who studied the problem of testing ℓ-colorability in graphs (although implicitly this problem could be traced to [28]). In the most basic case of graph 2-coloring (that is, testing bipartitness), they designed an algorithm with $\tilde{\mathcal{O}}(1/\epsilon^3)$ query complexity (and running time). Their analysis was later improved by Alon and Krivelevich [3], who showed that the complexity of this algorithm is $\tilde{\mathcal{O}}(1/\epsilon^2)$. For the more general case of testing ℓ-colorability for arbitrary $\ell \geq 2$, Goldreich et al. [18] presented an algorithm with the query complexity of $\tilde{\mathcal{O}}(\ell^4/\epsilon^6)$ and the running-time complexity of $2^{\tilde{\mathcal{O}}(\ell^2/\epsilon^3)}$. Again, Alon and Krivelevich [3] improved the analysis of the algorithm and obtained a bound of $\tilde{\mathcal{O}}(\ell^2/\epsilon^4)$ on the query complexity and $2^{\tilde{\mathcal{O}}(\ell/\epsilon^2)}$ on the running time. Alon et al. [1] presented another "constant-time" (i.e., independent of the size of the input graph) property testing algorithm; their algorithm uses the Szemerédi Regularity Lemma, and therefore the bounds for the query complexity and the running time, though independent of the size of the graph, have huge dependency of ℓ and ϵ. Fischer [15] extended the methods from [1] and investigated more general graph colorability properties.

1.3 Organization of the paper

Because of space limitations, we concentrate our analysis mostly on testing 2-colorability of 3-uniform hypergraphs and only briefly discuss extensions to the general case. In the main part of the paper, in Section 2, we present a detailed analysis of $\mathsf{Tester}(s, 2)$ and prove Theorems 1 and 2 for 2-colorability of 3-uniform hypergraphs. Then, in Section 3, we briefly discuss extensions of this result to ℓ-colorability of k-uniform hypergraphs.

2 Testing 2-Colorability of 3-Uniform Hypergraphs

In this section we only consider 2-coloring of 3-uniform hypergraphs. Let $\mathcal{H} = (V, E)$ be a 3-uniform hypergraph. This section is devoted to the proof the following result.

Theorem 3 $\mathsf{Tester}(s, 2)$ *with* $s = \mathcal{O}((1/\epsilon)^2)$ *is an* ϵ*-tester for* 2*-coloring* 3*-uniform hypergraphs.*

Theorem 3 immediately implies the following.

Theorem 4 *There is an* ϵ*-tester for* 2*-coloring* 3*-uniform hypergraphs with query complexity of* $\Theta(1/\epsilon^6)$ *and the running time of* $\exp(\mathcal{O}(1/\epsilon^2))$. □

We choose $s = 4 \cdot 10^3 \cdot (1/\epsilon)^2$, though we did not try to optimize the constant and it is easy to improve over our constant $4 \cdot 10^3$ significantly, perhaps even to a one digit number.

In order to prove Theorem 3 we must show the following properties of $\mathsf{Tester}(s, 2)$:

1. if $\mathcal{H}$ is 2-colorable, then the algorithm accepts $\mathcal{H}$ (that is, $\mathcal{H}_{\mathcal{S}}$ is 2-colorable);
2. if $\mathcal{H}$ is ϵ-far from 2-colorable, then the algorithm rejects $\mathcal{H}$ (that is, $\mathcal{H}_{\mathcal{S}}$ is not 2-colorable) with probability at least $2/3$.

Since if a hypergraph is 2-colorable, so is any its sub-hypergraph (and in particular, $\mathcal{H}_{\mathcal{S}}$), property (1) trivially holds. Therefore we must only prove that property (2) holds as well. From now on, we shall assume $\mathcal{H}$ is ϵ-far from having 2-coloring.

2.1 Coloring game with the adversary

For the purpose of the analysis, we partition our sample set $\mathcal{S}$ into $100/\epsilon$ sets U_i, $1 \le i \le 100/\epsilon$, of size $40/\epsilon$ each.

We analyze the following *game* on $\mathcal{H}$:

We play $100/\epsilon$ rounds starting with an initially empty set V_{colored} of *colored* vertices. In the course of the game we are adding new vertices to V_{colored} and the *adversary* chooses a color for each of these vertices. The coloring procedure of the adversary may be arbitrary, but the partial coloring of $\mathcal{H}$ on the sub-hypergraph induced by V_{colored} must be always proper. If the adversary is unable to properly color the vertex chosen, then *we win*. If the adversary properly colors the vertices during all $100/\epsilon$ rounds, *he wins*.

Formally, round i of the game looks as follows:

- We choose a vertex v from set U_i and add it to V_{colored}.
- The adversary colors v either red or blue. He is not allowed to create monochromatic edges.

The following claim that plays the key role in our analysis explains the idea behind introducing the game.

Claim 2.1 *If for any* 3*-uniform hypergraph* $\mathcal{H}$ *that is* ϵ*-far from* 2*-colorable we win independently of the strategy of the adversary with probability at least* $2/3$*, then the hypergraph* $\mathcal{H}_{\mathcal{S}}$ *computed by* $\mathsf{Tester}(s, 2)$ *is not* 2*-colorable with probability at least* $2/3$*. Therefore, in particular,* $\mathsf{Tester}(s, 2)$ *is an* ϵ*-tester for* 2*-coloring* 3*-uniform hypergraphs.*

Proof : The proof is by contradiction. Let us assume that $\mathcal{H}_{\mathcal{S}}$ has a proper coloring $\chi_{\mathcal{H}_{\mathcal{S}}}$ with probability greater than $1/3$ (over the choice of $\mathcal{S}$). Then, the adversary may color each vertex $v \in \mathcal{S}$ according to $\chi_{\mathcal{H}_{\mathcal{S}}}(v)$. Since the adversary wins if $\chi_{\mathcal{H}_{\mathcal{S}}}$ is proper, he wins with probability greater than $1/3$, which is a contradiction.

By our discussion above, this implies that $\mathsf{Tester}(s, 2)$ is an ϵ-tester for 2-coloring 3-uniform hypergraphs. □ □

Therefore, our plan is to show that if $\mathcal{H}$ is ϵ-far from 2-colorable, then we win the game with probability at least $2/3$ independently of the strategy of the adversary. In order to prove this result, we first concentrate ourself on estimating the probability that we win against a single fixed strategy of the adversary, and then generalize this estimation to winning against all strategies of the adversary.

2.2 Our strategy

Informally, our strategy in round i is to choose an especially selected vertex v from U_i that either cannot be properly colored or that adds many new "constraints" to the colors of the vertices of the hypergraph no matter what color the adversary chooses to color v.

During the game, some of the vertices are already colored. This coloring defines *constraints* for the colors of the remaining, yet uncolored vertices. We model these constraints by five sets V_{colored}, V_{conflict}, V_{red}, V_{blue}, V_{free} that form a partition of the vertex set V, and by two graphs $G_{\mathrm{red}} = (V, E_{\mathrm{red}})$ and $G_{\mathrm{blue}} = (V, E_{\mathrm{blue}})$.

V_{colored}**:** contains all vertices that have been already colored by the adversary.

V_{conflict}**:** contains all yet uncolored vertices that are incident to both an edge with two blue vertices (in V_{colored}) and another edge with two red vertices (in V_{colored}); notice that these vertices cannot be properly colored by the adversary.

V_{red}**:** contains all yet uncolored vertices that are not in V_{conflict} and can be properly colored only in red (that is, these are vertices incident to an edge with two blue vertices in V_{colored}).

V_{blue}**:** contains all yet uncolored vertices that are not in V_{conflict} can be properly colored only in blue (that is, these are vertices incident to an edge with two red vertices in V_{colored}).

V_{free}**:** contains all remaining vertices (that is, yet uncolored vertices that can be properly colored both red and blue).

G_{red}**:** contains an edge between two vertices v and w in V, if and only if there is an edge $e = \{v, w, u\}$ with a red colored vertex $u \in V_{\mathrm{colored}}$ (thus, an edge in G_{red} means that coloring both its endpoints red creates a monochromatic edge).

G_{blue}**:** contains an edge between two vertices v and w in V, if and only if there is an edge $e = \{v, w, u\}$ with a blue colored vertex $u \in V_{\mathrm{colored}}$ (thus, an edge in G_{blue} means that coloring both its endpoints blue creates a monochromatic edge).

Now, in order to formalize our strategy we define *heavy vertices*.

Definition 2.1 *Let $\mathcal{H} = (V, E)$ be a 3-uniform hypergraph. Let V_{colored} be a subset of V that is properly 2-colored by $\chi : V_{\mathrm{colored}} \to \{\mathrm{red}, \mathrm{blue}\}$. A vertex $v \in V - V_{\mathrm{colored}}$ is called* heavy *for $(V_{\mathrm{colored}}, \chi)$ if at least one of the following two conditions is satisfied after extending χ by any proper coloring of v:*

- *there are at least $\epsilon\, n^2/10$ new edges between vertices either in G_{red} or in G_{blue}, or*
- *there are at least $\epsilon\, n/10$ new vertices in one of the sets V_{red}, V_{blue}, or V_{conflict}.*

Now, we state our main lemma about heavy vertices:

Lemma 2.2 *Let $\mathcal{H} = (V, E)$ be a 3-uniform hypergraph and let V_{colored} be an arbitrary subset of its vertices that is properly 2-colored by $\chi : V_{\mathrm{colored}} \to \{\mathrm{red}, \mathrm{blue}\}$. Then, one of the following conditions hold:*

- *$\mathcal{H}$ is* not *ϵ-far from 2-colorable,*
- *there are at least $\epsilon n/10$ heavy vertices for $(V_{\mathrm{colored}}, \chi)$,*
- $|V_{\mathrm{conflict}}| \geq \epsilon n/10$.

Proof : The proof is by contradiction. Suppose none of the three conditions above holds. Then, $\mathcal{H}$ is ϵ-far from 2-colorable. Using the negation of the other two conditions we will construct a 2-coloring of $\mathcal{H}$ that violates less than ϵn^3 edges. This implies that $\mathcal{H}$ is not ϵ-far from 2-colorable, which is a contradiction.

The algorithm below constructs a 2-colorable hypergraph $\mathcal{H}'$ by deleting less than ϵn^3 edges from $\mathcal{H}$.

At the beginning of the algorithm we fix the sets V_{red}, V_{blue}, V_{free}, and V_{conflict} as well as the graphs G_{red} and G_{blue}. Then the algorithm colors the vertices one after the other. Each time a vertex is colored its coloring may introduce new constraints, that is, new vertices in the sets V_{red}, V_{blue}, or V_{conflict} or new edges in the graphs G_{red} or G_{blue}. For each such new constraint there is a set of edges that is responsible for the new constraint. These edges are called the witnesses of the new constraint. E.g., if vertex v is colored red, then the edge $\{v, u, w\}$ is a witness for the edge (constraint) (u, w) in G_{red}. The algorithm deletes all witnesses for new constraints. Thus, it maintains the following invariant at the beginning of each **for each** loop:

> *The constraints for the colors of uncolored vertices are given by (a subset of) the constraints represented by the sets V_{red}, V_{blue}, V_{free}, V_{conflict}, and the graph G_{red} and G_{blue}. E.g., if a vertex is in the set V_{red} it can be colored red without creating monochromatic edges in the current hypergraph at any time in the algorithm.*

Below it is proven that we can maintain this invariant by removing less than ϵn^3 edges.

ConstructColoring($\mathcal{H}$)

for each $v \in V$ *that is either heavy or is in* V_{conflict} **do**
 $\chi(v) = \mathrm{red}$
 remove all edges incident to v
for each $v \in V_{\mathrm{red}}$ *that is not heavy* **do**
 $\chi(v) = \mathrm{red}$
 remove all edges that cause new constraints
for each $v \in V_{\mathrm{blue}}$ *that is not heavy* **do**
 $\chi(v) = \mathrm{blue}$
 remove all edges that cause new constraints
for each $v \in V_{\mathrm{free}}$ *that is not heavy* **do**
 if *coloring* v *red causes fewer new constraints than coloring* v *blue* **then**
  $\chi(v) = \mathrm{red}$
 else
  $\chi(v) = \mathrm{blue}$
 remove all edges that cause new constraints

In what follows we prove that the so obtained hypergraph $\mathcal{H}'$ is properly 2-colored by χ and that it is obtained from $\mathcal{H}$ by deleting less than ϵn^3 edges. It is easy to see that the algorithm maintains the invariant that the constraints for the colors of the remaining vertices do not change. Indeed, if coloring a certain vertex creates new constraints, then all edges that cause these constraints are deleted from the hypergraph. Thus at any time, coloring a vertex in V_{red} (V_{blue}) red (blue) does not create any monochromatic edges in the current hypergraph. Coloring heavy and conflict vertices obviously is correct because all incident edges are deleted. And finally, coloring a vertex in V_{free} either red or blue again does not create any monochromatic edges because of the invariant. Therefore, the obtained hypergraph $\mathcal{H}'$ is properly 2-colored by χ.

It remains to show that the number of deleted edges is less than ϵn^3.

We remove at most n^2 edges incident to any heavy vertex or a vertex in $V_{conflict}$. Since we know that there are less than $\epsilon n/10$ heavy vertices as well as less than $\epsilon n/10$ vertices in $V_{conflict}$, the loop over these two sets of vertices (that removes all incident edges) will delete less than $2\epsilon n^3/10$ edges.

All remaining vertices are not heavy. Thus, coloring any such a vertex will create less than $\epsilon n/10$ new constraints in V_{red}, V_{blue}, and $V_{conflict}$ and less than $\epsilon n^2/10$ new constraints in G_{red} and G_{blue} (cf. Definition 2.1). Each of the new constraints in V_{red}, V_{blue}, and $V_{conflict}$ can cause at most n edges to become new constraints. Since there are at most n vertices in $V_{red} \cup V_{blue} \cup V_{free}$, the last three loops delete at most $5\epsilon n^3/10$ edges from $\mathcal{H}$.

Thus, overall, the hypergraph $\mathcal{H}'$ is obtained from $\mathcal{H}$ by deleting less than $7\epsilon n^3/10$ edges. This yields a contradiction, because on one hand we have assumed that $\mathcal{H}$ is ϵ-far from 2-colorable, but on the other hand we have just shown that there is a 2-colorable hypergraph $\mathcal{H}'$ that is obtained from $\mathcal{H}$ by deletion of less than ϵn^3 edges. □ □

2.3 Proof of Theorem 3

Now we are ready to formulate our strategy in details and to complete the proof of Theorem 3. We consider only the case that $\mathcal{H}$ is ϵ-far from 2-colorable. We want to show that for any strategy of the adversary, we win with probability at least $2/3$. Then, Claim 2.1 would imply the proof of Theorem 3.

Observe that there are at most $2^{100/\epsilon}$ strategies of the adversary, each one corresponding to a binary string of length $100/\epsilon$ such that if the ith bit is 1 (or 0, respectively), then the adversary colors vertex $v \in U_i$ red (or blue, respectively). Let us fix any strategy of the adversary Υ. Then, in round i we may assume we know the current status of the game (the coloring of the vertices in P chosen prior to round i). We further may assume that the set U_i is chosen at random. Then we choose the next vertex $v \in U_i$ to be colored by the adversary as follows: If there is a vertex in U_i that belongs also to $V_{conflict}$ then we choose one such a vertex and win the game. If there is no vertex in $U_i \cap V_{conflict}$, then we choose a heavy vertex if one exists in U_i. If there is no heavy vertex in U_i, then we choose an arbitrary vertex from U_i.

Now, let us observe that since U_i is a randomly chosen set of vertices of size $40/\epsilon$, from Lemma 2.2 we may conclude that in round i

$$\mathbf{Pr}\left[v \text{ is neither heavy nor belongs to } V_{conflict} \mid \Upsilon\right] \le (1-\epsilon/10)^{40/\epsilon} \le e^{-4} \quad . \tag{1}$$

Now, let us recall that the coloring by the adversary of any heavy vertex either inserts at least $\epsilon n^2/10$ new edges to one of G_{red} or G_{blue}, or inserts at least $\epsilon n/10$ new vertices to one of the sets V_{red}, V_{blue}, or $V_{conflict}$. Furthermore, if a vertex v is chosen that is neither heavy nor belongs to $V_{conflict}$, then the number of constraints does not decreases. Therefore, since each of the sets V_{red}, V_{blue}, or $V_{conflict}$ may have at most n vertices, and each of the graphs G_{red} or G_{blue} may have at most n^2 edges, we can conclude that a heavy vertex may be chosen at most $50/\epsilon$ times.

For a given strategy of the adversary Υ and for a given round i, $1 \le i \le 100/\epsilon$, let $\mathcal{X}_i^\Upsilon$ be the indicator random variable of the event that for the strategy of the adversary Υ (1) we have neither won in round $j < i$, (2) nor the vertex v chosen in round i either is heavy or belongs to V_{conflict}. Let $\mathcal{X}^\Upsilon = \sum_{i=1}^{100/\epsilon} \mathcal{X}_i^\Upsilon$. Observe that by our arguments above, if $\mathcal{X}^\Upsilon < 50/\epsilon$, then we win! Therefore, our goal now is to estimate the probability that $\mathcal{X}^\Upsilon \ge 50/\epsilon$.

By (1), for every Υ and every i, we have $\mathbf{Pr}[\mathcal{X}_i^\Upsilon = 1 \mid \Upsilon] \le e^{-4}$. Therefore, we can conclude that for every Υ and every $t \in \mathbb{R}$ it holds that[2] $\mathbf{Pr}[\mathcal{X}^\Upsilon \ge t] \le \mathbf{Pr}[\mathbb{B}(100/\epsilon, e^{-4}) \ge t]$, where $\mathbb{B}(N,p)$ is a binomially distributed random variable with parameters N and p, that is, $\mathbf{Pr}[\mathbb{B}(N,p) = k] = \binom{N}{k} p^k (1-p)^{N-k}$ for every $0 \le k \le N$. Given this majorization result, we can use basic calculations to estimate the probability that $\mathcal{X}^\Upsilon \ge 50/\epsilon$. Let $N = 100/\epsilon$ and $p = e^{-4}$.

$$
\begin{aligned}
\mathbf{Pr}[\mathcal{X}^\Upsilon \ge 50/\epsilon] &\le \mathbf{Pr}[\mathbb{B}(N,p) \ge N/2] = \sum_{k=N/2}^{N} \binom{N}{k} \cdot p^k \cdot (1-p)^{N-k} \\
&\le \sum_{k=N/2}^{N} \left(\frac{eN}{k}\right)^k \cdot p^k = \sum_{k=N/2}^{N} \left(\frac{eNp}{k}\right)^k \le \sum_{k=N/2}^{N} (2ep)^k \\
&\le \sum_{k \le N/2} (2ep)^k = \frac{(2ep)^{N/2}}{1-2ep} = \frac{(2/e^3)^{50/\epsilon}}{1-2/e^3} \le \frac{1}{3} \cdot 2^{-100/\epsilon} \quad .
\end{aligned}
$$

Thus, we have shown that for a given strategy Υ the adversary wins with probability upper bounded by $(1/3) \cdot 2^{-100/\epsilon}$. Now, we can incorporate the union bound to obtain an upper bound for the probability that there is a strategy Υ for which the adversary wins:

$$
\mathbf{Pr}\left[\exists \Upsilon \ \mathcal{X}^\Upsilon \ge 50/\epsilon\right] \le \sum_{\Upsilon} \mathbf{Pr}[\mathcal{X}^\Upsilon \ge 50/\epsilon] \le 2^{100/\epsilon} \cdot ((1/3) \cdot 2^{-100/\epsilon}) \le 1/3 \quad .
$$

Hence, we have proven that we win for all strategies with probability greater than or equal to $2/3$. By Claim 2.1, this implies the proof of Theorem 3. □

3 Testing ℓ-Colorability of k-Uniform Hypergraphs

In this section we briefly describe how to generalize the result from Section 2 to ℓ-colorability of k-uniform hypergraphs and prove Theorem 1. Our analysis follows roughly the same approach as the proof of Theorem 3 and we will frequently refer to that proof for some details.

Let us fix $s = 1600\, k^2\, \ell^2 \ln \ell/\epsilon^2$ and consider $\mathsf{Tester}(s, \ell)$. Since it is easy to see that any ℓ-colorable hypergraph is accepted by the tester, it is sufficient to prove that any hypergraph that is ϵ-far from ℓ-colorable is rejected by $\mathsf{Tester}(s, \ell)$ with probability at least $2/3$.

Our goal is to show that we win the game against the adversary who is now allowed to use ℓ colors instead of 2 as in Section 2. We partition the sample set $\mathcal{S}$ into $20\, k^2\, \ell^2/\epsilon$ sets $\mathcal{U}_i$, $1 \le i \le 20\, k^2\, \ell^2/\epsilon$ of size $80 \ln \ell/\epsilon$ each.

We obtain the general result by adjusting our constraint modeling from Section 2 to ℓ-coloring of k-uniform hypergraphs. We model the constraints by a set of ℓ j-uniform hypergraphs $H_{i,j}$ for each $1 \le i \le \ell$ and $1 \le j \le k-1$. The $H_{i,2}$ are graphs and the $H_{i,1}$ are sets. Again, we also have the sets V_{colored}, and V_{conflict}.

[2] This is a standard fact on majorization in probability theory, see, e.g., [4, Lemma 3.1].

$H_{i,j}$ contains an edge between vertices $v_1, \ldots, v_j$, if and only if there is an edge $\{v_1, \ldots, v_j, v_{j+1}, \ldots, v_k\}$ in $\mathcal{H}$ such that $v_{j+1}, \ldots, v_k$ are colored with color i. Thus an edge $\{v_1, \ldots, v_j\}$ in the hypergraph $H_{i,j}$ means that coloring vertices $v_1, \ldots, v_j$ with color i will create a monochromatic edge. Also, note that the meaning of the sets $H_{i,1}$ is different from the meaning of V_{red} and V_{blue} in Section 2 in the sense that $H_{i,1}$ contains all vertices that may *not* be colored with color i.

Definition 3.1 *Let $\mathcal{H} = (V, E)$ be a k-uniform hypergraph. Let V_{colored} be a subset of V that is properly l-colored by $\chi : V_{\mathrm{colored}} \to \{1, \ldots, \ell\}$. A vertex $v \in V - V_{\mathrm{colored}}$ is called* heavy *for $(V_{\mathrm{colored}}, \chi)$ if at least one of the following two conditions is satisfied after extending χ by any proper coloring of v:*

- *there are at least $\epsilon\, n^j/(10\, k\, \ell)$ new edges between vertices in $H_{i,j}$ for some i, j*
- *there are at least $\epsilon\, n/10$ new vertices in the set V_{conflict}.*

Using similar arguments (though technically more involved) as those used in Section 2, we can prove the following main technical result.

Lemma 3.1 *Let $\mathcal{H} = (V, E)$ be a k-uniform hypergraph and let V_{colored} be a subset of its vertices that is properly ℓ-colored by $\chi : V_{\mathrm{colored}} \to \{1, \ldots, \ell\}$. Then, one of the following conditions hold:*

- *there are at least $\epsilon\, n/10$ heavy vertices for $(V_{\mathrm{colored}}, \chi)$,*
- $|V_{\mathrm{conflict}}| \geq \epsilon\, n/10$,
- *$\mathcal{H}$ is* not *ϵ-far from ℓ-colorable.* □

Once we have Lemma 3.1, we can proceed similarly as in Subsection 2.3 to prove that we win the game with probability greater than or equal to $2/3$ no matter which strategy is chosen by the adversary.

This implies the proof of Theorem 1. □

References

[1] N. Alon, E. Fischer, M. Krivelevich, and M. Szegedy. Efficient testing of large graphs. In *Proc. 40th FOCS*, pages 656–666, 1999.

[2] N. Alon, P. Kelsen, S. Mahajan, and H. Ramesh. Coloring 2-colorable hypergraphs with a sublinear number of colors. *Nordic Journal of Computing*, 3:425–439, 1996.

[3] N. Alon and M. Krivelevich. To appear in *SIAM Journal on Discrete Mathematics*.

[4] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal. Balanced allocations. *SIAM Journal on Computing*, 29(1):180–200, September 1999.

[5] J. Beck. An algorithmic approach to the Lovász local lemma. I. *Random Structures and Algorithms*, 2(4):343–365, 1991.

[6] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47(3):549–595, December 1993.

[7] M. A. Bender and D. Ron. Testing acyclity of directed graphs in sublinear time. In *Proc. 27th ICALP*, pages 809–820, 2000.

[8] H. Chen and A. Frieze. Coloring bipartite hypergraphs. In *Proc. 5th IPCO*, pages 345–358, 1996.

[9] A. Czumaj and C. Scheideler. An algorithmic approach to the general Lovász Local Lemma with applications to scheduling and satisfiability problems. In *Proc. 32nd STOC*, pages 38–47, 2000.

[10] A. Czumaj and C. Scheideler. Coloring non-uniform hypergraphs: A new algorithmic approach to the general Lovász Local Lemma. In *Proc. 11th SODA*, pages 30–39, 2000.

[11] A. Czumaj, C. Sohler, and M. Ziegler. Property testing in computational geometry. In *Proc. 8th ESA*, pages 155–166, 2000.

[12] P. Erdős and L. Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In A. Hajnal, R. Rado, and V. T. Sós, editors, *Infinite and Finite Sets (to Paul Erdős on his 60th birthday)*, volume II, pages 609–627. North-Holland, Amsterdam, 1975.

[13] F. Ergün, S. Kannan, S. R. Kumar, R. Rubinfeld, and M. Viswanathan. Spot-checkers. *Journal of Computer and System Sciences*, 60:717–751, 2000.

[14] F. Ergün, S. Ravi Kumar, and R. Rubinfeld. Approximate checking of polynomials and functional equations. In *Proc. 37th FOCS*, pages 592–601, 1996.

[15] E. Fischer. Testing graphs for colorability properties. In *Proc. 12th SODA*, pages 873–882, 2001.

[16] E. Fischer and I. Newman. Testing of matrix properties. To appear in *Proc. 33rd STOC*, 2001.

[17] A. Frieze and R. Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19:175–220, 1999.

[18] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, July 1998.

[19] O. Goldreich and D. Ron. A sublinear bipartiteness tester for bounded degree graphs. *Combinatorica*, 19(3):335–373, 1999.

[20] V. Guruswami, J. Håstad, and M. Sudan. Hardness of approximate hypergraph coloring. In *Proc. 41st FOCS*, pages 149–158, 2000.

[21] M. Krivelevich, R. Nathaniel, and B. Sudakov. Approximating coloring and maximum independent sets in 3-uniform hypergraphs. In *Proc. 12th SODA*, pages 327–328, 2001.

[22] M. Krivelevich and B. Sudakov. Approximate coloring of uniform hypergraphs. In *Proc. 6th ESA*, pages 477–489, 1998.

[23] L. Lovász. Coverings and colorings of hypergraphs. In *Proc. 4th Southeastern Conference on Combinatorics, Graph Theory, and Computing*, pages 3–12. 1973.

[24] C-J. Lu. Deterministic hypergraph coloring and its applications. In *Proc. 2nd RANDOM*, pages 35–46, 1998.

[25] I. Newman. Testing of function that have small width branching programs. In *Proc. 41st FOCS*, pages 251–258, 2000.

[26] M. Parnas and D. Ron. Testing metric properties. To appear in *Proc. 33rd STOC*, 2001.

[27] J. Radhakrishnan and A. Srinivasan. Improved bounds and algorithms for hypergraph two-coloring. In *Proc. 39th FOCS*, pages 684–693, 1998.

[28] V. Rödl and R. A. Duke. On graphs with small subgraphs of large chromatic number. *Graphs and Combinatorics*, 1:91–96, 1985.

[29] D. Ron. Property testing. To appear in P. M. Pardalos, S. Rajasekaran, J. Reif, and J. D. P. Rolim, editors, *Handobook of Randomized Algorithms*. Kluwer Academic Publishers, 2001.

[30] R. Rubinfeld. Robust functional equations and their applications to program testing. In *Proc. 35th FOCS*, pages 288–299, 1994.

[31] R. Rubinfeld and M. Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, April 1996.