Testing Convexity Properties of Tree Colorings *

Eldar Fischer † Orly Yahalom ‡

Abstract

A coloring of a graph is *convex* if it induces a partition of the vertices into connected subgraphs. Besides being an interesting property from a theoretical point of view, tests for convexity have applications in various areas involving large graphs. We study the important subcase of testing for convexity in trees. This problem is linked, among other possible applications, with the study of phylogenetic trees, which are central in genetic research, and are used in linguistics and other areas. We give a 1-sided, non-adaptive, distribution-free ϵ-test for the convexity of tree colorings. The query complexity of our test is $O\left(k/\epsilon\right)$, where k is the number of colors, and the additional computational complexity is $O(n)$. On the other hand, we prove a lower bound of $\Omega(\sqrt{k/\epsilon})$ on the query complexity of tests for convexity in the standard model, which applies even for (unweighted) paths. We also consider whether the dependency on k can be reduced in some cases, and provide an alternative testing algorithm for the case of paths. Then we investigate a variant of convexity, namely *quasi-convexity*, in which all but one of the colors are required to induce connected components. For this problem we provide a 1-sided, non-adaptive ϵ-test with query complexity $O\left(k/\epsilon^2\right)$ and time complexity $O\left(kn/\epsilon\right)$. For both our convexity and quasi-convexity tests, we show that, assuming that a query takes constant time, the time complexity can be reduced to a constant independent of n if we allow a preprocessing stage of time $O(n)$ and $O(n^2)$, respectively. Finally, we show how to test for a variation of convexity and quasi-convexity where the maximum number of connectivity classes of each color is allowed to be a constant value other than 1.

Keywords: property testing, convex coloring, sublinear algorithms, massively parameterized, phylogenetic trees, graph algorithms.

1 Introduction

1.1 Property testing

Property testing deals with the following relaxation of decision problems: Given a property $\mathcal{P}$ and an input structure S, distinguish with high probability between the case where S satisfies the

*A preliminary version appeared in the proceedings of the 24th STACS (2007).

†Computer Science Department, Technion - IIT, Haifa 32000, Israel. Email: eldar@cs.technion.ac.il.

Research supported in part by an ERC-2007-StG grant number 202405-2 and by an ISF grant number 1101/06.

‡Department of Software Engineering, ORT Braude College, P.O. Box 78, Karmiel, 21982, Israel. Phone: 972-4-9901845, Fax: 972-4-9901852, Email: oyahalom@braude.ac.il

property $\mathcal{P}$ and the case where S is "far" from satisfying $\mathcal{P}$. The power of property testing lies in the ability to design algorithms, or *property testers*, which read only a small fraction of the input structure. We assume that our algorithms access the input structure using retrieval procedures that we call *queries*. The *answer* to each query is a value in the input structure.

Property testing was initiated in the work of Blum, Luby and Rubinfeld [2], and given a general formulation by Rubinfeld and Sudan [21]. The latter were interested mainly in algebraic properties (such as linearity) of functions over finite fields and vector spaces. The study of property testing for combinatorial objects, and mainly for labelled graphs, was introduced by the seminal paper of Goldreich, Goldwasser and Ron [7]. A property in this respect is a collection of functions from a fixed combinatorial object to a finite set of labels, often $\{0,1\}$, but in our work we consider larger finite sets as well. Property testing has since become quite an active research area, see e.g. the surveys [20] and [4].

Formally, denoting our set of inputs by $\mathcal{I}$, a property $\mathcal{P}$ is a subset of $\mathcal{I}$. We have a *distance function* $\mathrm{dist} : \mathcal{I} \times \mathcal{I} \to [0,1]$, which is some fixed metric on $\mathcal{I}$. Two input structures $S, S' \in \mathcal{I}$ are said to be ϵ-*close* to each other for some $\epsilon \in [0,1]$ if $\mathrm{dist}(S,S') \leq \epsilon$. Otherwise, if $\mathrm{dist}(S,S') > \epsilon$, then we say that S and S' are ϵ-*far* from each other. We say that an input structure $S \in \mathcal{I}$ is ϵ-*close to satisfying property* $\mathcal{P}$ (or simply ϵ-*close to* $\mathcal{P}$) if there exists $S' \in \mathcal{I}$ that satisfies $\mathcal{P}$ and is ϵ-close to S. Otherwise, if every input that satisfies $\mathcal{P}$ is ϵ-far from S, then we say that S is ϵ-*far from satisfying property* $\mathcal{P}$ (or simply ϵ-*far from* $\mathcal{P}$).

Given a property $\mathcal{P} \subseteq \mathcal{I}$ and parameters $\epsilon, q > 0$, we say that a (randomized) algorithm $\mathcal{A}$ is an (ϵ, q)-*test for* $\mathcal{P}$ if the following hold for every input structure $S \in \mathcal{I}$:

1. $\mathcal{A}$ accesses S using at most q queries;
2. If S satisfies $\mathcal{P}$ then $\mathcal{A}$ accepts with probability at least $\frac{2}{3}$, and if S is ϵ-far from satisfying $\mathcal{P}$ then $\mathcal{A}$ rejects with probability at least $\frac{2}{3}$.

Depending on the context, we may also use the terms ϵ-*test* and *test* for an algorithm which satisfies Item 2 above.

If there exists an (ϵ, q)-test for a property $\mathcal{P}$ then we say that $\mathcal{P}$ is (ϵ, q)-*testable*. If a property $\mathcal{P}$ is (ϵ, q)-testable with $q = q(\epsilon)$ (i.e. q is a function of ϵ only, and is independent of n) then we say that $\mathcal{P}$ is ϵ-*testable*. If $\mathcal{P}$ is ϵ-testable for every fixed $\epsilon > 0$ then we say that $\mathcal{P}$ is *testable*. We refer to the (asymptotic) number of queries required by a given test as its *query complexity*. Property testing normally deals with very large inputs and/or costly retrieval procedures. We thus assume that the query complexity is the most limited resource, rather than the computation time.

Furthermore, a test is called *1-sided* if every input that has the property is accepted with probability 1. Otherwise, it is called *2-sided*. A test is said to be *adaptive* if some of the choices of the locations for which the input is queried may depend on the values (answers) of previous queries. Otherwise, the test is called *non-adaptive*.

1.2 Convex colorings

Given a graph $G = (V, E)$ and an integer k, a vertex coloring $c : V \to [k]$, where $[k] \stackrel{\text{def}}{=} \{1, \ldots, k\}$, is called a k-*coloring*, or simply, a *coloring* of G. Given a k-coloring c of a graph G and $i \in [k]$, let V_i be the set of vertices v in V such that $c(v) = i$. We say that c is *a convex coloring of* G if all the V_i's are connected sets (i.e., c induces connected subgraphs of G). If G is a tree, then c is

not convex if and only if there exists three distinct vertices $u, w, v \in V$ such that w is on the path between u and v and $c(u) = c(v) \neq c(w)$. Such three vertices consist a forbidden poset (*subpath*) of c.

We consider testing for convexity as defined above and several variants of this problem, on trees. A central motivation for this subcase is the study of phylogenetic (evolutionary) trees, which originated in genetics [16, 22], but appears also in other areas, such as historical linguistics (see [17]). Whether our subjects of interest are biologic species, languages, or other objects, a phylogenetic tree specifies presumed hereditary relationships, representing different features with different colors. A convex coloring is a positive indication for the reliability of a phylogenetic tree, as it shows a reasonable evolutionary behavior. Namely, a feature (color) is either inherited from a direct ancestor or appears spontaneously, in case of a mutation, but the same mutation does not normally occur in separate (i.e. disconnected) parts of the tree. We note that although phylogenetic trees are rooted trees, the convexity property does not depend on the identity of the root.

Moran and Snir [15] studied *recoloring* problems, where the input is a colored tree and one has to find a close convex coloring of the tree. They gave several positive and negative results on exact and approximate algorithms. Our results are the first, to the best of our knowledge, which approach the property testing aspect of convex colorings. Note that while property testing may be viewed as a form of approximation, our work does not relate to that of Moran and Snir, as their algorithms read the entire input while ours read only a constant fraction of it.

1.3 Variants of convexity

Given a graph $G = (V, E)$ and an integer k, a vertex coloring $c : V \to \{0, \ldots, k\}$ is called a *quasi k-coloring* of G. Note that the difference between a k-coloring and a quasi k-coloring is the use of an additional color marked as 0, whose role is explained below. Whenever the context is clear, we may refer to such a function c simply as *coloring*. Given a quasi k-coloring c of a graph G and $i \in \{0, \ldots, k\}$, let V_i be the set of vertices v in V such that $c(v) = i$. We say that c is *quasi-convex* if the color components V_i are connected for colors $i \geq 1$ (while V_0 is not necessarily connected). This property arises in various cases in which we are interested only in the connectivity of some of the color classes (where all the others may be considered as colored with 0, or simply uncolored).

In addition, we consider variants of the convexity and quasi-convexity properties, where we relax our requirement of having at most one color component from every color. A k-coloring is called *ℓ-convex* if the total number of color components that it induces is at most ℓ. Similarly, a quasi k-coloring is called *ℓ-quasi-convex* if it induces a total of at most ℓ color components for all colors $i > 0$. Similarly we discuss *list convexity* (and *list quasi-convexity*) where we have lists of upper bounds on the numbers of connected components of every color (or some of the colors).

1.4 The vertex coloring testing model

Throughout this paper we assume a fixed and known tree $T = (V, E)$ and a set $D \subseteq V$. The vertices in D are called *constraint vertices* and the vertices in $V \setminus D$ are called *free vertices*. The input is a (quasi) k-coloring of all the vertices in V, which our testers access by querying one vertex at a time. The goal is to determine whether the input is close to a (quasi) convex k-coloring of V which does not change the color of the constraint vertices. Thus, the distance between two colorings of the domain graph is a weighted sum of the free vertices in which they differ. The monotonicity

property has already been studied on a similar model (with a uniform weight function) by Fischer et. al [6], who provided efficient tests for several classes of graphs.

Formally, we assume that $\mu : V \setminus D \to \mathbb{R}$ is a fixed weight function on the free vertices of T, satisfying $\mu(v) \geq 0$ for every $v \in V \setminus D$ and $\sum_{v \in V \setminus D} \mu(v) = 1$ (namely, a distribution function). The weight function μ may represent the importance of certain vertices, the cost of modifying them, or the reliability of querying for their color. One could think of the constraint vertices as having infinite weight. Hence, we only consider colorings which preserve the color of the constraint vertices, even when we do not state so explicitly. For convenience, we define $\mu(U) = \sum_{v \in U \setminus D} \mu(v)$ for any set $U \subseteq V$. The *distance* between two colorings c_1 and c_2 of V is defined as $\mu(\Delta_{c_1,c_2})$, where $\Delta_{c_1,c_2} = \{v \in V \setminus D \mid c_1(v) \neq c_2(v)\}$.

In Section 2.2 we consider a stricter model, where the weight function over the free vertices is unknown. Such a model is called *distribution-free*, a concept that was introduced in [7] and developed by Halevy and Kushilevitz [10, 9]. A *distribution-free test* may attain a sample of the points of the domain of the input according to a fixed yet unknown distribution function μ (where each value obtained this way counts as a query). Since μ determines the weight of every vertex with respect to the distance function, the distance between two inputs is equal to the probability of obtaining a point on which they differ.

1.5 Massively parameterized models

The vertex coloring model is an example of a *massively parameterized* property testing model. In such a model, the tested property is characterized by a complex parameter which is fixed and fully known, often a graph. This is in contrast to standard models of testing graph properties, where the graph itself is the input, typically represented as an adjacency matrix [7] and/or adjacency lists [8, 19].

As massively parameterized property testing (MPPT) requires full knowledge of a large parameter, such as the structure of a graph, it may sometimes not lead to algorithms with a running time to match their low query complexity. On the other hand, MPPT has several appealing characteristics. Focusing on graph properties, the distance function in MPPT models depends heavily on the structure of the underlying graph. As a result, the study of MPPT may reveal interesting combinatorial details of the underlying graph, with graph theoretic results that could be of independent interest. Moreover, the distance function in MPPT models is usually very strict (namely, edge insertions and deletions are forbidden) and independent of representation details, which allows for a 'clean' study of graph properties.

The *orientation model* [11, 12, 3, 5], is another massively parameterized model for testing graph properties, in which the input is an orientation of a fixed and known undirected graph. The *general poset domain* model of [6] is a generic massively parameterized model for testing functions on some partially ordered domain. Another example of a massively parameterized testing model has been studied by Newman [18], where a given bounded-width branching program B is given to the tester in advance, and the input is a word given as an input to B.

1.6 Our results

Due to the definition of our model, all the constraint vertices must be queried, which contributes $O(|D|)$ to the query complexity. In the following, we omit $|D|$ for clarity.

We show that convexity of tree colorings is testable, providing a 1-sided, non-adaptive, distribution-free ϵ-test for every $\epsilon > 0$. The query complexity of our test for k-colorings is $O(k/\epsilon)$ and the additional time complexity is $O(n)$. We show that the time complexity can be reduced to be quasilinear in the query complexity (assuming that a query takes constant time) by allowing a preprocessing stage of time $O(n)$. We further provide an alternative 1-sided, non-adaptive test for the non distribution-free model where the tree is a path, with query complexity $O(\sqrt{k}/\epsilon^3)$ and additional time complexity $\widetilde{O}(\sqrt{k}/\epsilon^3)$. On the negative side, we prove a lower bound of $\Omega(\sqrt{k}/\sqrt{\epsilon})$ on the query complexity of testing convexity of paths even in the unweighted model.

We show that quasi-convexity of tree colorings is testable, giving a 1-sided, non-adaptive (but not distribution-free) ϵ-test for every $\epsilon > 0$. The query complexity of our tests for quasi k-colorings is $O(k/\epsilon^2)$ and the additional time complexity $O(kn/\epsilon)$. We show that the time complexity can be reduced to be quasilinear in the query complexity (assuming that a query takes constant time) by allowing a preprocessing stage of time $O(n^2)$.

Finally, we provide 1-sided ϵ-tests for every $\epsilon > 0$ for the relaxed convexity properties. For ℓ-convexity we give a test with query complexity $\widetilde{O}(\ell/\epsilon)$ and time complexity $O(\ell n)$. For ℓ-quasi-convexity we provide a test with query complexity $\widetilde{O}(\ell/\epsilon^2)$ and time complexity $O(\ell n)$. Given a list of integers c_i, let ℓ denote their sum. Our test for list convexity has query complexity $\widetilde{O}(\ell/\epsilon)$ and computational complexity $O(\ell n)$. For list quasi-convexity, let ℓ denote the sum of c_i's only for the colors for which they are defined. For that property we also give a test with query complexity $\widetilde{O}(\ell/\epsilon^2)$ and computational complexity $O(\ell n)$.

The rest of the paper is organized as follows: Section 2 is dedicated to testing the basic convexity property. Section 3 is dedicated to quasi-convexity. In Section 4 we consider the relaxed convexity and quasi-convexity properties. Section 5 provides a lower bound for testing both convexity and quasi-convexity. Finally, in Section 6 we give our convexity test for paths.

Throughout the paper, we make no attempt to optimize the coefficients.

2 Testing convexity

Throughout this section, our input is a k-coloring $c : V \to [k]$ of a fixed and known tree $T = (V, E)$ and $D \subseteq V$ is the set of constraint vertices. For any two distinct vertices $u, v \in V$, we denote the connected component of $V \setminus \{u\}$ that contains v by $C_u^{(v)}$. Note that if u and v are neighbors then $C_u^{(v)}$ and $C_v^{(u)}$ form a bipartition of V.

Vertices u, w, v in T form a *forbidden subpath* if w is on the (simple) path between u and v and $c(u) = c(v) \neq c(w)$. Clearly, c is a convex coloring of T if and only if it does not contain any forbidden subpath.

We say that vertices u, v, w are an *explicit witness* (for being non-convex) if they form a forbidden subpath. We say that vertices u_1, u_2, v_1, v_2 are an *implicit witness* (for being non-convex) if $c(u_1) = c(u_2) \neq c(v_1) = c(v_2)$ and the path between u_1 and v_2 crosses the path between v_1 and v_2. In such a case, the intersection vertex is a middle vertex of at least one forbidden subpath. Note that we do not need to know the color of the intersection vertex.

We say that a set $U \subseteq V$ of vertices contains a *witness* (for being non-convex) if U contains an explicit witness or an implicit witness.

Observation 2.1 *If a subset U of vertices contains a witness, then U cannot be extended into a convex k-coloring of T.*

The reverse claim is true as well.

Lemma 2.2 *Let $U \subseteq V$ be a subset of vertices that contains no witnesses for being non-convex, with respect to a coloring $c : V \to [k]$. Then there exists a convex coloring $c' : V \to [k]$ of V which agrees with c on the values of the vertices in U. In particular, if $D \subseteq U$ then c' preserve the colors of the constraint vertices.*

Proof. If $U = \emptyset$ the claim is trivially true, as any coloring which gives all the vertices the same color is convex. We thus assume that $U \neq \emptyset$.

Define c' as follows. For every vertex $u \in U$ set $c'(u) = c(u)$. Now, for every $i \in [k]$, set $c'(w) = i$ for all the vertices w which are on the simple path between two vertices u, v with $c'(u) = c'(v) = i$. Note that since U contains no witness for being non-convex, there are no two intersecting paths of different colors. Thus, by the end of this phase we obtain a partial coloring of T where for every color i, the set of i-colored vertices is connected. Denote every such set as A_i.

Now, for every $v \in V$ and for every i such that $A_i \neq \emptyset$, let $d(v, A_i)$ denote the "walking" distance on T between v and A_i, i.e. the length of the path from v to (the connected) A_i. Color every vertex v with i such that $d(v, A_i)$ is minimal, choosing the minimal index i in case of a tie.

We claim that c' is convex. First, consider two neighboring vertices u and w such that $c'(u) = i$ and $c'(w) = j$ for $i \neq j$. We show that $A_i \subseteq C_w^{(u)}$. By the definition of c', $w \notin A_i$, and since A_i is connected we have either $A_i \subseteq C_u^{(w)}$ or $A_i \in C_w^{(u)}$. Assume that $A_i \subseteq C_u^{(w)}$. Then $d(u, A_i) = d(w, A_i) + 1$ and $d(u, A_j) \leq d(w, A_j) + 1$. But from the way u and w are colored by c' we have $d(u, A_i) - d(u, A_j) < d(w, A_i) - d(w, A_j)$, a contradiction. Hence, $A_i \subseteq C_w^{(u)}$.

Now, assume that the set of vertices colored with i is not connected. Then there exist vertices u and v (by taking them as the endpoints of the shortest forbidden subpath) such that $c'(u) = c'(v) = i$ and $c'(w_1), c'(w_2) \neq i$, where w_1 is the neighbor of u on the path between u and v and w_2 is the neighbor of v on the path between u and v (w_1 could be equal to w_2). By the above, we have $A_i \subseteq C_{w_1}^{(u)}$ and $A_i \subseteq C_{w_2}^{(v)}$, a contradiction. □

Our test for convexity queries a sample set of vertices and checks whether it contains a witness for being non-convex. While the test is simple, the bulk of the proof is established by the analysis in the next subsection.

2.1 Sufficient conditions for closeness to convexity

Recall that for every color $i \in [k]$, V_i is the set of vertices $u \in V$ such that $c(u) = i$. We refer to vertices in V_i as *i-vertices* and to other vertices as *non-i-vertices*. For any subset $U \subseteq V$, let the *i-weight* of U be the total weight of all free i-vertices in U, and denote it by $\mu_i(U) \stackrel{\text{def}}{=} \mu(V_i \cap U)$.

A color $i \in [k]$ is called *abundant* if $\mu_i(V) \geq \epsilon/2k$. For an abundant color i, we say that a vertex $u \in V$ is *i-balanced* if the set $\{C_u^{(v)} | (u, v) \in E\}$ may be partitioned into two subsets, where the i-weight of the union of each subset is at least $\epsilon/8k$. We say that a free vertex v is *heavy* if $\mu(v) \geq \epsilon/8k$.

Lemma 2.3 *For every abundant color i, there exists a vertex $u \in V$ which is either i-balanced, or is a heavy i-vertex (or both).*

Proof. Assume that there exists an abundant color i such that no $u \in V$ is i-balanced and there are no heavy i-vertices. Note that in this case every $u \in V$ has a neighboring vertex v such that $C_u^{(v)}$ is of i-weight larger than $\epsilon/4k$ (as otherwise u is easily seen to be i-balanced). Consider neighboring vertices u and v such that $C_u^{(v)}$ is of minimum i-weight among those whose i-weight is larger than $\epsilon/4k$, and with a minimum number of vertices among the minimal weight $C_u^{(v)}$'s. There exists a neighbor w of v such that $C_v^{(w)}$ is of i-weight larger than $\epsilon/4k$. Due to the minimality of $C_u^{(v)}$, we must have $w = u$. Thus $C_v^{(u)}$ is of i-weight larger than $\epsilon/4k$, and, since there are no heavy i-vertices, the i-weight of $C_v^{(u)} \setminus \{u\}$ is at least $\epsilon/8k$. Therefore, both $C_u^{(v)}$ and $V \setminus \{C_u^{(v)} \cup \{u\}\}$ have i-weight of at least $\epsilon/8k$, and hence u is i-balanced. A contradiction. □

For every abundant color i, define the set B_i to be the union of i-balanced vertices and heavy i-vertices. By the lemma above, B_i is non-empty for every abundant color i. We later show that for every abundant color i, the weight of i-vertices outside B_i is relatively small.

Lemma 2.4 *B_i is a connected set for every abundant color i.*

Proof. Assume that there exist two vertices $u, v \in B_i$, and let w be on the path between u and v. Assuming that w is not a heavy i-vertex, we show that w is i-balanced. If u is a heavy i-vertex, then clearly $\mu_i(C_w^{(u)}) \geq \epsilon/8k$. Otherwise, u is i-balanced, and thus, $V \setminus \{u\}$ may be partitioned into two sets of connected components, the i-weight of each of which is at least $\epsilon/8k$. One of these sets does not contain $C_u^{(w)}$. Thus, $\mu_i(C_w^{(u)}) \geq \epsilon/8k$. Similarly, it follows that $\mu_i(C_w^{(v)}) \geq \epsilon/8k$, and hence w is i-balanced. □

Suppose that the B_i's are disjoint. For every two distinct abundant colors i and j, let the *ij-bridge* be the edge (u, v) on the path between B_i and B_j where $u \in B_i$ and $v \notin B_i$. Note that $B_i \subseteq C_v^{(u)}$ and $B_j \subseteq C_u^{(v)}$. For any abundant color i, an i-colored vertex w is called *exiled* if $w \in C_u^{(v)}$ where (u, v) is the ij-bridge for some abundant $j \neq i$.

Lemma 2.5 *The total weight of exiled vertices of all abundant colors is at most $\epsilon/2$. In other words, $\sum \mu_i(C_u^{(v)}) \leq \epsilon/2$, where the sum is over all the ij-bridges.*

Proof. Consider an abundant color i, and suppose that $\mu_i(C_u^{(v)}) \geq \epsilon/4k$. By the definition of B_i, v is not a heavy i-vertex, and thus $\mu_i(C_u^{(v)} \setminus \{v\}) > \epsilon/8k$. Since v is not i-balanced, we have $\mu_i(C_v^{(u)}) < \epsilon/8k$, but this is impossible, as u is i-balanced or heavy. We thus conclude that for every ij-bridge (u, v) we have

$$\mu_i(C_u^{(v)}) < \epsilon/4k. \tag{1}$$

Let T' be the tree created from T by contracting every set B_i into a single vertex and removing all the vertices which do not belong to a path between two B_i's. Let d_i be the degree of B_i in T'. Clearly, for every abundant color i, d_i is the number of ij-bridges. Assume, without loss of generality, that the abundant colors are numbered $i = 1, \ldots, \ell$ for $\ell \leq k$. We next show that

$$\sum_{i=i}^{\ell} d_i \leq 2(\ell - 1) \leq 2(k - 1). \tag{2}$$

We prove Equation (2) by induction on ℓ. For $\ell = 1$, T' consists of a single B_i, and the claim is trivially true. Assume that the claim is true for every $\ell' < \ell$ where $\ell > 2$. Note that by the

definition of T', all its leaves are B_i's. Now remove one of the leaves of T'. If the resulting tree has a new leaf which is not a B_i, remove it, and repeat this operation until we get a tree, T'', whose leaves are all B_i's. By the induction hypothesis, the sum of the degrees of B_i's in T'' is at most $2(\ell-2)$. Now, in creating T'' from T' we have removed one B_i of degree 1 and possibly, reduced the degree of another B_i by 1 (as after removing an edge adjacent to a B_i we stop removing leaves). Thus, $\sum_{i=i}^{\ell} d_i \leq 2(\ell-2)+2 = 2(\ell-1) \leq 2(k-1)$.

Recall that $\sum_{i=i}^{\ell} d_i$ is the number of ij-bridges in T. Thus, from Equations (1) and (2) we obtain that the total weight of exiled vertices is smaller than $2(k-1)\epsilon/4k < \frac{\epsilon}{2}$. □

The next proposition provides sufficient conditions for a coloring being ϵ-close to convexity.

Proposition 2.6 *Let $c : V \to [k]$ be a k-coloring such that*

(a) The sets B_i are disjoint;

(b) There is no constraint vertex w such that $w \in B_i$ for some $i \neq c(w)$;

Create a partial coloring of T by coloring every set B_i with color i and coloring every constraint vertex with its color in c. If the obtained partial coloring does not contain a witness for being non-convex, then there exists a k-coloring $c' : V \to [k]$ which agrees with c on D and is ϵ-close to c.

Proof. Define c' as follows. Color every B_i with i and color every constraint vertex with its color in c. This partial coloring does not contain a witness for being non-convex, and thus, by Lemma 2.2, it can be extended into a convex coloring c'. It can easily be seen that a vertex of abundant color may change its color only if it is exiled. Hence, by Lemma 2.5, the total weight of recolored vertices among those whose original color was abundant is at most $\epsilon/2$. As for vertices whose original color was non-abundant, their total weight is at most $\epsilon/2$, which completes the proof. □

2.2 A distribution-free convexity test for trees

We next provide a 1-sided test for convexity. Our test is distribution-free (see Subsection 1.4), as it uses the distribution μ as a black-box only.

Algorithm 2.7

1. a. *Query all the constraint vertices.*
   b. *Query $\lceil \frac{8k \ln 12}{\epsilon} \rceil$ vertices, where each vertex is independently chosen according to the distribution μ.*

   Let X denote the set of all vertices queried in either of the steps above.

2. *Reject the input If X contains a witness for being non-convex and accept otherwise.*

Theorem 2.8 *For every $\epsilon > 0$, Algorithm 2.7 is a 1-sided non-adaptive ϵ-test for convexity of k-colorings of trees. The query complexity of the test is $O(k/\epsilon + |D|)$ and the time complexity is $O(n)$. This can be implemented in running time $\widetilde{O}(k/\epsilon + |D|)$ using a preprocessing stage of time $O(n)$.*

It is easy to see that the query complexity is as stated. We show how detecting a witness can be performed under the time complexity requirements in Section 2.3. Clearly, a convex coloring is always accepted by Algorithm 2.7, as the set X cannot contain a witness. Thus, it remains to show that every coloring which is ϵ-far from being convex is rejected with probability at least $\frac{2}{3}$. We do so by considering all the cases in which Proposition 2.6 does not apply.

Lemma 2.9 *If there exists a vertex w such that: (a) w is either a constraint vertex or a heavy free vertex, (b) w is i-balanced for some abundant color $i \neq c(w)$, then Algorithm 2.7 rejects with probability at least* $3/4$.

Proof. We show that with probability at least $3/4$, the set X contains an explicit witness.

Since w is i-balanced, there exist two disjoint sets $W_1^i, W_2^i \subseteq V_i$, each of weight at least $\epsilon/8k$, such that every path between a pair of vertices $v_1 \in W_1^i$ and $v_2 \in W_2^i$ passes through w. Hence, it is enough for the algorithm to sample at least one vertex from each of the sets W_1^i and W_2^i as well as the vertex w in the case that it is a heavy free vertex (for if it is a constraint vertex, it is queried with probability 1). The probability for any of the sets W_1^i and W_2^i or of w to not intersect the sample set of Step 1b is at most $(1-\epsilon/8k)^{\frac{8k\ln 12}{\epsilon}} < 1/12$. Thus, by the union bound, the algorithm will fail with probability at most $3(1-\epsilon/8k)^{\frac{8k\ln 12}{\epsilon}} < 1/4$. □

Lemma 2.10 *If there exists a vertex $w \in V$ which is both i-balanced and j-balanced for some abundant colors $i \neq j$ then Algorithm 2.7 rejects with probability at least* $5/6$.

Proof. We show that with probability at least $5/6$, the set X contains an implicit witness.

There exist two disjoint sets $W_1^i, W_2^i \subseteq V_i$, each of weight at least $\epsilon/8k$, and two disjoint sets $W_1^j, W_2^j \subseteq V_j$, each of weight at least $\epsilon/8k$, where every path between a pair of vertices $u_1 \in W_1^i$ and $u_2 \in W_2^i$ passes through w and every path between a pair of vertices $v_1 \in W_1^j$ and $v_2 \in W_2^j$ passes through w. Hence, if the sample set of Step 1b contains at least one vertex from each of the sets $W_1^i, W_2^i, W_1^j, W_2^j$, then Algorithm 2.7 rejects the input. The probability for any given set of the above to not intersect the sample set is at most $(1-\epsilon/8k)^{\frac{8k\ln 12}{\epsilon}} < 1/12$. Thus, by the union bound, the algorithm will fail with probability at most $4(1-\epsilon/8k)^{8k\ln 12/\epsilon} < 1/3$. □

Lemma 2.11 *Let $c : V \to [k]$ be a k-coloring such that*

(a) The sets B_i are disjoint;

(b) There is no constraint vertex w such that $w \in B_i$ for some $i \neq c(w)$;

Create a partial coloring of T by coloring every set B_i with color i and coloring every constraint vertex with its color in c. If the obtained partial coloring contains a witness for being non-convex, then Algorithm 2.7 rejects the input with probability at least $2/3$.

Proof. First, if the set of constraint vertices contain a witness then clearly the algorithm rejects with probability 1.

Second, suppose that a set B_i is on the path between two j-colored constraint vertices u and v for some $j \neq i$. Since B_i is connected (Lemma 2.4), there exists a vertex $w \in B_i$ which is on the path between u and v. If w is a heavy i-vertex, then the probability that it is not queried is

at most $(1-\epsilon/8k)^{\frac{8k\ln 12}{\epsilon}} < 1/12$. Otherwise, if w is i-balanced, then there exist two disjoint sets $W_1^i, W_2^i \subseteq V_i$, each of weight at least $\epsilon/8k$, where every path between a pair of vertices $u_1 \in W_1^i$ and $u_2 \in W_2^i$ passes through w. The probability for any of the sets W_1^i and W_2^i or of w to not intersect the sample set of Step 1b is at most $(1-\epsilon/8k)^{\frac{8k\ln 12}{\epsilon}} < 1/12$. The constraint vertices u and v are queried with probability 1. Thus, the probability that the sample does not contain a witness is at most $2(1-\epsilon/8k)^{\frac{8k\ln 12}{\epsilon}} < 1/6$.

Third, suppose that a set B_i is on the path between a set B_j and a j-colored constraint vertex v for some $j \neq i$. Let u be the closest vertex to B_j in B_i, and let w be the closest vertex to B_i in B_j. Similarly to the previous case, the probability of not sampling two i-colored vertices such that w is on the path between them is at most $1/6$. Now, if u is heavy then the probability that it is not queried is at most $(1-\epsilon/8k)^{\frac{8k\ln 12}{\epsilon}} < 1/12$. Otherwise, since u is i-balanced and w is not i-balanced, we have $\mu_i(C_w^{(u)}) > \epsilon/8k$, and thus, the probability of not sampling an i-colored vertex in $C_w^{(u)}$ is smaller than $1/12$. As v is queried with probability 1, we obtain that the probability of not sampling an implicit witness (where w is the intersection vertex) is at most $1/6 + 1/12 = 1/3$.

Forth, if a j-colored constraint vertex w is on the path between a set B_i and an i-colored constraint vertex v, then the proof is similar to the previous case, only this time we obtain an explicit witness with high probability, as w is queried with probability 1. □

Proof of Theorem 2.8. Follows from Proposition 2.6 and Lemmas 2.9, 2.10, and 2.11. □

2.3 Finding witnesses for non-convexity

We now specify a procedure implementing Step 2 of Algorithm 2.7 in time $O(n)$, where the constants are independent of k, ϵ and D. Later we show how this procedure can be executed in time $\widetilde{O}(|X|) = \widetilde{O}(k/\epsilon + |D|)$ if we allow a preprocessing stage of time $O(n)$. For every color $i \in [k]$, let q_i be the number of vertices of color i in the query set X. Clearly, the q_i's can be computed in time $O(|X|)$. Next, we arbitrarily select a root r for T and obtain a topological ordering of the vertices using a Depth First Search from r, which can be done in time $O(n)$ (see e.g. [14]). We now consider the nodes of T in reverse topological order. This can be viewed as "trimming" leaves from the tree one by one. For each vertex v we hold a variable $m(v)$, which can receive either the value "null" or the value of a color. Initially, if $v \in X$ then $m(v)$ holds its color, and if $v \notin X$ then $m(v)$ is null. $m(v)$ will receive the value i if and only if i is the only color for which X contains i-vertices both inside and outside the subtree rooted in v. If there is more than one such color, we deduce that a forbidden subpath exists and reject. In addition, we assign for every vertex v a variable $a(v)$ which will be 0 if $m(v)$ is null, and otherwise will hold the number of vertices of color $m(v)$ in the subtree rooted in v.

Procedure 2.12 *For every v in reverse topological order, do:*

- *If $v \in X$ then set $a(v) = 1$; otherwise set $a(v) = 0$.*
- *If $v \in X$ then set $m(v) = c(v)$; otherwise set $m(v)$ to be null.*
- *For every child u of v such that $m(u)$ is not null:*
  1. *If $m(v)$ is not null and $m(v) \neq m(u)$ then reject the input and terminate.*
  2. *Otherwise, set $m(v)$ to $m(u)$ and $a(v)$ to $a(v) + a(u)$.*

- *If $m(v)$ is not null and $a(v) = q_{m(v)}$ then set $m(v)$ to be null and $a(v) = 0$.*

If the algorithm did not reject after going over all vertices, then accept.

Since for every vertex v the running time is proportional to the number of its children, the total running time is $O(n)$. We now prove that Procedure 2.12 implements Step 2 of Algorithm 2.2 correctly.

Lemma 2.13 *For every vertex v the following holds:*

1. *If Procedure 2.12 rejects in the iteration of v, then v is a middle vertex of a forbidden subpath, where if $v \in X$ then the forbidden subpath includes v as its middle vertex, and, otherwise, there exist vertices $a_1, a_2, b_1, b_2 \in X$ such that $c(a_1) = c(a_2) \neq c(b_1) = c(b_2)$ and v belongs to the path between a_1 and a_2 as well as to the path between b_1 and b_2.*

2. *If Procedure 2.12 completes the iteration of v without rejecting it, then v is not a middle vertex of a forbidden path as above.*

3. *If the processing of v is completed, then, in its end, $m(v) = i$ if and only if X includes i-vertices both inside and outside the subtree rooted in v. In such a case, $a(v)$ is equal to the number of i-vertices of X in the subtree rooted in v. If $m(v)$ is null then $a(v) = 0$.*

Proof. The claim is easily proved for the case where v is a leaf. Let v be a vertex and assume the correctness of the claim for all the children of v.

1. For the proof of the first part, notice that if $v \in X$, then the procedure rejects if and only if there exists a child u of v such that $m(u)$ is not null and $m(u) \neq c(v)$. By Part 3 of the induction hypothesis, this implies that X includes $m(u)$-colored vertices both inside and outside the subtree rooted in u. Thus, v is a middle vertex of a forbidden subpath of X. If $v \notin X$, then the procedure rejects if and only if there exist children u_1, u_2 of v such that $m(u_1)$ and $m(u_2)$ are both not null with $m(u_1) \neq m(u_2)$. By Part 3 of the induction hypothesis, this implies that there exist $a_1, a_2 \in X$ such that $c(a_1) = c(a_2) = m(u_1)$, where a_1 is a descendant of u_1 and a_2 is not. Similarly, there exist $b_1, b_2 \in X$ such that $c(b_1) = c(b_2) = m(u_2)$, where b_1 is a descendant of u_2 and b_2 is not. Clearly, v belongs both to the path between a_1 and a_2 and to the path between b_1 and b_2.

2. Suppose that v is a middle vertex in a forbidden subpath in X. Then there exist two vertices $a, b \in X$ such that $c(a) = c(b) \neq c(v)$ and v is on a simple path between a and b. It must be the case that at least one of a and b, say a, is a descendant of v. Therefore, unless the algorithm has already rejected before reaching v, for the child u of v which is an ancestor of a, we have $m(u) = c(a)$ by Part 3 of the induction hypothesis (note that b cannot be a descendant of u so $q_{m(u)} > a(u)$). However, as $m(v)$ is set to $c(v)$, we are ensured that the procedure will reject when the child u is examined (if not earlier). Similarly, suppose that $v \notin X$ and that there exist vertices $a_1, a_2, b_1, b_2 \in X$ such that $c(a_1) = c(a_2) \neq c(b_1) = c(b_2)$ and v belongs to the path between a_1 and a_2 as well as to the path between b_1 and b_2. Then at least one of a_1 and a_2 and at least one of b_1 and b_2 are descendants of v, and therefore, v has two children u_1 and u_2 such that $m(u_1) \neq m(u_2)$ (noting that $q_{m(u_1)} > a(u_1)$ and $q_{m(u_2)} > a(u_2)$). One can now see that the procedure will reject in the iteration of v.

3. If $m(v)$ is null after examining all the children of v (before checking whether $a_v = q_{m(v)}$), then $v \notin X$ and $m(u)$ is null for every child u of v. By the induction hypothesis, there exists no color i such that X contains i-vertices both inside and outside subtrees rooted in v's children. As $v \notin X$, it follows that there is no color i such that X contains i-vertices both inside and outside the subtree rooted in v. Therefore, $m(v)$ and $a(v)$ correctly attain their initial values. If after examining v's children we have $m(v) = i$ for some color i, then $m(u) = i$ for every child u of v such that $m(u)$ is not null, and if $v \in X$ then $c(v) = i$. Thus one can see that after examining v's children, $a(v)$ correctly holds the number of i-vertices in X in the subtree rooted in v. In that case, $a(v) = q_i$ if and only if there are no i-vertices in X outside the subtree rooted in v, and so the last step in the iteration provides the correct value for $m(v)$.

□

From the lemma above, it follows that Procedure 2.12 is correct, as it rejects the query set X if and only if it contains a witness.

Note that Procedure 2.12 performs significant processing only in nodes which are in X or are Least Common Ancestors (LCA's) of two or more members of X. Other nodes are just assumed to be colored as their closest descendants. This gives rise to the possibility of running the procedure over a set which includes X and the least common ancestors of vertices in X, instead of over the entire set V of vertices. Let $\widehat{X}$ be the union of X and the set of all vertices w such that $w = LCA(u, v)$ for some $u, v \in X$.

Observation 2.14 *$\widehat{X}$ is closed under the LCA operation. That is, for every $u, v \in \widehat{X}$, $LCA(u, v) \in \widehat{X}$.* □

Consider the directed tree $T_X = (\widehat{X}, \widehat{E})$, with $(u, v) \in \widehat{E}$ if and only if v is the uppermost proper descendant of u in $\widehat{X}$ with respect to the directed T. From the discussion above, it is enough to use T_X instead of T in the procedure defined earlier. We will build T_X using an algorithm which computes the LCA of two nodes in a tree in constant time, after a preprocessing stage of time $O(n)$ (see [13], [23]). After yielding the set X, we build the tree T_X using the following procedure.

Procedure 2.15

1. *Sort the vertices in X according to their preorder indexes, i.e., their indexes in a particular DFS traversal of T, computed as part of the preprocessing stage. Let us denote the vertices of X as $u_1, u_2, \ldots, u_{|X|}$ according to this order. For $i = 1, \ldots, |X| - 1$ we let $z_i = LCA(u_i, u_{i+1})$.*

2. *Compute $Y = X \cup \{z_i \mid i = 1, \ldots, |X| - 1\}$ using the constant time LCA algorithm. We shall later prove that $Y = \widehat{X}$.*

3. *Sort the vertices in Y according to their preorder indexes. Denote the vertices of Y according to this order by $v_1, v_2, \ldots, v_{|Y|}$. Set v_1 as the root of T_X.*

4. *For every $j = 1, \ldots, |Y| - 1$ do:*

   - *Compute $LCA(v_j, v_{j+1})$.*
   - *If $LCA(v_j, v_{j+1}) = v_j$ (that is, v_j is an ancestor of v_{j+1}) then add v_{j+1} to T_X as the rightmost child of v_j.*

- *Otherwise, search in T_X upwards among v_j's proper ancestors, until discovering one that is an ancestor of v_{j+1} and add v_{j+1} as its rightmost child.*

As running Procedure 2.12 on T_X would require time $O(|\widehat{X}|)$, the next lemma proves that the total running time in this case is $\widetilde{O}(|X|)$.

Lemma 2.16 *Procedure 2.15 computes the tree T_X in time $\widetilde{O}(|X|)$. Furthermore, $|\widehat{X}| \leq 2|X|$.*

Proof. We show that $Y = \widehat{X}$. Specifically, we consider the set $X_k = \{u_1, \ldots, u_k\}$ and show that $\widehat{X_k} = X_k \cup \{z_i \mid i = 1, \ldots, k-1\}$ for every $k = 1, \ldots, |X| - 1$. The claim is trivial for $k = 1$. Assuming that for a specific k we have $\widehat{X_{k-1}} = X_{k-1} \cup \{z_i \mid i = 1, \ldots, k-2\}$, we now consider $X_k \cup \{z_i \mid i = 1, \ldots, k-1\} = \widehat{X_{k-1}} \cup \{u_k, z_{k-1}\}$. The claim is proved by observing that for every $i = 1, \ldots, k-1$, if $LCA(u_i, u_k) \neq z_{k-1}$ then $LCA(u_i, u_k) = LCA(u_i, u_{k-1}) \in \widehat{X_{k-1}}$.

Clearly, $|\widehat{X}| \leq 2|X|$, by the way we have built $\widehat{X}$. To complete the proof of the lemma, we show that the total running time of all the iterations of Step 4 in Procedure 2.15 is $O(|\widehat{X}|) = O(|X|)$. Now, for a certain vertex v_{j+1}, the running time is proportional to the number of ancestors being examined. However, notice that once a vertex w has been found not to be an ancestor of a certain v_{j+1}, it will not be examined anymore, as v_{j+1} will be attached as a child of a proper ancestor of w, and the remaining examinations will be done only on its ancestors. Therefore, the total number of ancestor examinations is $O(|X|)$, and hence, this is the total running time of Procedure 2.15. □

3 Testing quasi-convexity

In this section our input is a k-quasi-coloring $c : V \to \{0, 1, \ldots, k\}$ of a fixed and known tree $T = (V, E)$ and $D \subseteq V$ is the fixed set of constraint vertices. As in Section 2, for any two distinct vertices $u, v \in V$, we denote the connected component of $V \setminus \{u\}$ that contains v by $C_u^{(v)}$.

Given an input quasi-coloring c, we define V_i to be the set of vertices v in V such that $c(v) = i$, for every $i \in \{0, \ldots, k\}$. We say that a vertex $v \in V$ is *colored* if $c(v) > 0$. Otherwise, we say that v is *uncolored*. c is said to be *quasi-convex* if V_i is connected for $i = 1, \ldots, k$. Alternatively, vertices u, w, v in T form a *forbidden subpath* if w is on the (simple) path between u and v, $c(u) = c(v) > 0$ and $c(w) \neq c(v)$. Clearly, c is quasi-convex if and only if it contains no forbidden subpaths as defined above.

We say that vertices u, v, w are an *explicit witness* (for being non-quasi-convex) if they form a forbidden subpath. We say that colored vertices u_1, u_2, v_1, v_2 are an *implicit witness* (for being non-quasi-convex) if $c(u_1) = c(u_2) \neq c(v_1) = c(v_2)$ and the path between u_1 and v_2 crosses the path between v_1 and v_2. In such a case, the intersection vertex is a middle vertex of at least one forbidden subpath. Note that we do not need to know the color of the intersection vertex.

We say that a set $U \subseteq V$ of vertices contains a *witness* (for being non-quasi-convex) if U contains an explicit witness or an implicit witness. As with witnesses for convexity, we have the following observation and lemma.

Observation 3.1 *If a subset U of vertices contains a witness, then U cannot be extended into a convex k-coloring of T.*

Lemma 3.2 *Let $U \subseteq V$ be a subset of vertices that contains no witnesses for being non-quasi-convex, with respect to a coloring $c : V \to \{0, 1, \ldots, k\}$. Then there exists a quasi-convex coloring*

$c' : V \to \{0, 1, \ldots, k\}$ of V which agrees with c on the values of the vertices in U. In particular, if $D \subseteq U$ then c' preserve the colors of the constraint vertices.

Proof. The proof is similar to that of Lemma 2.2, only we do not not set any additional vertices to be uncolored. In fact, one can view the coloring process as being performed independently in the connected components of the tree created by deleting the uncolored vertices in U. □

One could expect that testing for quasi-convexity would be as simple as testing for convexity. However, here it is not enough to sample vertices according to their weight (which is why we lose the distribution-free quality). The reason for this is that we may need to color a large weight of uncolored vertices in order to make the input quasi-convex, but unlike in the case of colored vertices, these uncolored vertices might be spread in the tree, making it difficult to discover them using a single sample. We thus need to establish more delicate knowledge of when an input is close to being quasi-convex.

3.1 Sufficient conditions for closeness to quasi-convexity

In this section we say that a color $i \in [k]$ is *abundant* if $\mu(V_i) \geq \epsilon/4k$. For an abundant color i, we say that a vertex $u \in V$ is *i-balanced* if the set $\{C_u^{(v)} | (u, v) \in E\}$ may be partitioned into two subsets, where the total i-weight of the union of each subset is at least $\epsilon/16k$. We say that a vertex v is *heavy* if $\mu(v) \geq \epsilon/16k$.

For every abundant color i, we define the set B_i as the union of i-balanced vertices and heavy i-vertices. The proof of the following lemma is identical to the proofs of Lemma 2.3 and Lemma 2.4.

Lemma 3.3 *B_i is non-empty and connected for every abundant color i.*

Suppose that the B_i's are disjoint. For every two distinct abundant colors i and j, let the *ij-bridge* be the edge (u, v) on the path between B_i and B_j where $u \in B_i$ and $v \notin B_i$. For any abundant color i, an i-colored vertex w is called *exiled* if $w \in C_u^{(v)}$ where (u, v) is the ij-bridge for some abundant $j \neq i$.

Lemma 3.4 *Suppose that the B_i's are disjoint. Then the total weight of exiled vertices of all abundant colors i is at most $\epsilon/4$. In other words, $\sum \mu_i(C_u^{(v)}) \leq \epsilon/4$, where the sum is over all the ij-bridges.*

Proof. Consider an abundant color i, and suppose that $\mu_i(C_u^{(v)}) \geq \epsilon/8k$ for some ij-bridge (u, v). By the definition of B_i, v is not a heavy i-vertex, and thus $\mu_i(C_u^{(v)} \setminus \{v\}) > \epsilon/16k$. Since v is not i-balanced, we have $\mu_i(C_v^{(u)}) < \epsilon/16k$, but this is impossible, as u is i-balanced or heavy. We thus conclude that for every ij-bridge (u, v) we have $\mu_i(C_u^{(v)}) < \epsilon/8k$. As in the proof of Lemma 2.5, we have that the number of ij-bridges is at most $2(k-1)$. Therefore, the total number of all exiled vertices is smaller than $2(k-1)\epsilon/8k < \frac{\epsilon}{4}$. □

We now assume that there is no constraint vertex $w \in B_i$ with $c(w) \neq i$, the B_i's are all disjoint, and coloring every set B_i with color i does not form a witness for non-quasi-convexity together with the constraint vertices.

Define sets S_i as follows. For every abundant color i, let S_i be the set created by augmenting every set B_i with all the i-colored constraint vertices and the vertices on the paths from B_i to them. For every non abundant color $i > 0$, let S_i be the set containing all the i-colored constraint vertices and the paths between them. Note that we do not do this for uncolored constraint vertices. We now have at most one set S_i for every color $i \in [k]$, where the sets are disjoint and every S_i contains all the i-colored constraint vertices.

We say that a vertex v is an *outsider* if it does not belong to any S_i nor to a path between two S_i's, and, moreover, none of the vertices on the path between v and its closest S_i belongs to a path between two S_i's.

Let F be the rooted forest consisting of all outsider vertices, such that the root of every tree is a vertex r adjacent to some S_i. We say that such a vertex r is *associated* with i. We also say that i is the color associated with every descendant v of such an r in F. An outsider vertex is said to be a *satellite* if it is i-colored and associated with i. Note that an outsider vertex u is a satellite if and only if u is colored with an abundant color and u is not exiled.

F is said to be *monotone* if it contains only uncolored and satellite vertices, and furthermore no uncolored vertex is an ancestor of a satellite vertex. We say that F is *good* if it can be made monotone by recoloring free vertices in F, such that the total weight of satellite and uncolored vertices being recolored is at most $\epsilon/4$. Otherwise, we say that F is *bad*.

An uncolored vertex w in F is called *good* if $\mu_i(T_w) \leq \frac{\epsilon}{16}\mu(T_w)$, where T_w is the subtree of F rooted in w and i is the color associated with w. Otherwise we say that w is *bad*. We say that a satellite vertex w in F is an *obstacle* if w has an uncolored constraint vertex in F as an ancestor.

Lemma 3.5 *If the total weight of bad uncolored vertices in F is at most $\frac{\epsilon}{16}$ and the total weight of obstacle vertices is at most $\frac{\epsilon}{8}$ then F is good.*

Proof. Define a monotone coloring of F as follows. Let U be the set of all constraint (uncolored) vertices and good uncolored vertices in F. Let U_r be the set of the topmost vertices in U. Set all the descendants of vertices in U_r to be uncolored. Color the rest of the vertices in F with the color associated with them. Clearly, in the obtained coloring, all the constraint vertices remain uncolored, all the vertices in F are either uncolored or satellites, and no uncolored vertex is an ancestor of a satellite vertex. Thus, F is monotone.

Now, the only uncolored vertices we have colored are bad ones, whose weight is at most $\epsilon/16$. As for satellite vertices, we have only changed the color of ones within subtrees of uncolored vertices whose roots are in U_r. Among these, the weight of obstacle vertices is at most $\epsilon/8$. The others are in subtrees rooted in good uncolored vertices. Since these subtrees are disjoint, and the weight of satellite vertices is a fraction of at most $\epsilon/16$ of the weight of any good subtree, the total weight of satellite vertices thus changed is at most $3\epsilon/16$. Hence, we have changed a total weight of at most $\epsilon/4$ of uncolored and satellite vertices, and therefore F is good. □

The next proposition provides sufficient conditions for a coloring being ϵ-close to convexity.

Proposition 3.6 *Let $c : V \to \{0, 1, \ldots, k\}$ be a coloring such that*

(a) *The sets B_i are disjoint;*

(b) *There is no constraint vertex w such that $w \in B_i$ for some $i \neq c(w)$;*

(c) *The set of B_i's and constraint vertices does not contain a witness for being non-convex (where the B_i's are viewed as i-colored);*

(d) *The total weight of uncolored vertices inside S_i's is at most $\epsilon/4$;*

(e) *F is good;*

then there exists a coloring $c' : V \to \{0, 1, \ldots, k\}$ such that c' agrees with c on the constraint vertices and c' is ϵ-close to c.

Proof. Define c' as follows. Color every set S_i with i and color every constraint vertex with its color in c. Then choose a monotone coloring of F which changes a minimum weight of uncolored and satellite vertices without recoloring constraint vertices. Finally, set the rest of the vertices to be uncolored. One can see that c' is quasi-convex and agrees with c on D.

We now show that c' is ϵ-close to c. First, consider vertices of abundant colors whose color has been changed. By Lemma 3.4, the total weight of exiled vertices is at most $\epsilon/4$. Other vertices of abundant colors may change their color only if they are in F (i.e. satellites). From Condition (e), the total weight of satellites and uncolored vertices in F is at most $\epsilon/4$. Other uncolored vertices may be recolored only if they are inside some S_i, and from Condition (d), the total weight of such vertices is at most $\epsilon/4$. Finally, for non-abundant colors, since they are of weight smaller than $\epsilon/4k$ each, their total weight is smaller than $\epsilon/4$. We conclude that c' is ϵ-close to c. □

3.2 A quasi-convexity test for trees

We now present our test for quasi-convexity.

Algorithm 3.7

1. a. *Query all the constraint vertices.*
   b. *Query $\lceil \frac{20k \ln 12}{\epsilon} \rceil$ vertices, where each vertex is independently chosen according to the distribution μ.*

   Let X denote the set of all vertices queried in either of the steps above.

2. *If X contains a witness for non-quasi-convextiy, reject.*

3. *Otherwise, repeat the following $\lceil 16 \ln 12/\epsilon \rceil$ times independently:*

   - *Choose a vertex w according to the distribution defined by μ and query it. If $X \cup \{w\}$ contains a witness, reject.*
   - *Otherwise, if w is colored, do nothing.*
   - *Otherwise, if w is uncolored, define a subtree T^i_w for every color i such that there are i-vertices in X, as follows. Let v_i be the neighbor of w that is on a path between w and an i-vertex in X (v_i is unique, as $X \cup \{w\}$ does not contain a witness). Now denote $T^i_w \stackrel{\text{def}}{=} C^{(w)}_{v_i}$ for every such v_i. Query $\lceil \log_{1/(1-\epsilon/16)} 6 \rceil$ vertices in each T^i_w, where each vertex is independently chosen according to the distribution defined by μ conditioned on T^i_w.*

4. If the union of X and vertices queried in Step 3 contains a witness for non-quasi-convexity, then reject. Otherwise, accept.

Theorem 3.8 *For every $\epsilon > 0$, Algorithm 3.7 is a 1-sided (adaptive) ϵ-test for quasi-convexity of quasi k-colorings of trees, with query complexity $O(k/\epsilon^2 + |D|)$. The time complexity of the test is $O(kn/\epsilon)$, and can be implemented in time $\widetilde{O}(k/\epsilon^2 + |D|)$ with a preprocessing stage of time $O(n^2)$.*

Note that for ϵ small enough, we have $\log_{1/(1-\epsilon/8)} 6 < \frac{\ln 6}{\epsilon/16}$. Thus, it is easy to see that the query complexity is as stated. In Section 3.3 we show how to detect witnesses for non-quasi-convexity, as done in Steps 2 and 4 of the algorithm, in time $O(n)$, or $\widetilde{O}(k/\epsilon^2 + |D|)$ with a preprocessing stage of time $O(n)$. In Step 3, a time of $O(n)$ is used for each subtree $T_w^i \stackrel{\text{def}}{=} C_{v_i}^{(w)}$ to compute the distribution μ conditioned on T_w^i, using a BFS traversal. We can reduce the running time by computing the conditioned distributions for all subtrees $C_{v_i}^{(w)}$ at the preprocessing stage. This requires listing the probabilities for every vertex relative to every subtree. Since every such tree is defined by an edge in the tree T and one of its vertices, there are $O(n)$ such subtrees, and hence, a preprocessing time of $O(n^2)$ would be enough to compute and store all the required distributions.

Clearly, a convex coloring is always accepted by Algorithm 3.7, as the set X cannot contain a witness. Thus, it remains to show that every coloring which is ϵ-far from being quasi-convex is rejected with probability at least $\frac{2}{3}$. We do so by considering all the cases in which Proposition 3.6 does not apply.

Lemma 3.9 *If there exists a vertex w such that: (a) w is either a constraint vertex or a heavy free vertex, (b) w is i-balanced for some abundant color $i \neq c(w)$, then Algorithm 2.7 rejects with probability at least $\frac{3}{4}$.*

Proof. Since w is i-balanced, there exist two disjoint sets $W_1^i, W_2^i \subseteq V_i$, each of weight at least $\epsilon/16k$, such that every path between a pair of vertices $v_1 \in W_1^i$ and $v_2 \in W_2^i$ passes through w. Hence, to reject the input, it is enough for the algorithm to sample at least one vertex from each of the sets W_1^i and W_2^i as well as the vertex w in the case that it is a heavy free vertex (for if it is a constraint vertex, it is queried with probability 1). The probability for any of the sets W_1^i and W_2^i or of w to not intersect the sample set of Step 1b is at most $(1 - \epsilon/16k)^{20k \ln 12/\epsilon} < \frac{1}{12}$. Thus, by the union bound, the algorithm will fail with probability at most $\frac{3}{12} = \frac{1}{4}$. □

Lemma 3.10 *If there exists a vertex $w \in V$ which is both i-balanced and j-balanced for some abundant colors $i \neq j$ then Algorithm 2.7 rejects with probability at least $\frac{2}{3}$.*

Proof. There exist two disjoint sets $W_1^i, W_2^i \subseteq V_i$, each of weight at least $\epsilon/16k$, and two disjoint sets $W_1^j, W_2^j \subseteq V_j$, each of weight at least $\epsilon/16k$, where every path between a pair of vertices $u_1 \in W_1^i$ and $u_2 \in W_2^i$ passes through w and every path between a pair of vertices $v_1 \in W_1^j$ and $v_2 \in W_2^j$ passes through w. Hence, if the sample set of Step 1b contains at least one vertex from each of the sets $W_1^i, W_2^i, W_1^j, W_2^j$, then Algorithm 2.7 rejects the input. The probability for any given set of the above to not intersect the sample set is at most $(1 - \epsilon/16k)^{20k \ln 12/\epsilon} < \frac{1}{12}$. Thus, by the union bound, the algorithm will fail with probability at most $\frac{4}{12} = \frac{1}{3}$. □

Lemma 3.11 *Let $c : V \to \{0, 1, \ldots, k\}$ be a quasi-k-coloring such that*

(a) The sets B_i are disjoint;

(b) There is no constraint vertex w such that $w \in B_i$ for some $i \neq c(w)$;

Create a partial coloring of T by coloring every set B_i with color i and coloring every constraint vertex with its color in c. If the obtained partial coloring contains a witness for being non-quasi-convex, then Algorithm 2.7 rejects the input with probability at least $\frac{3}{4}$.

Proof. First, if the set of constraint vertices contains a witness then clearly the algorithm rejects with probability 1.

Second, suppose that a set B_i is on the path between two j-colored constraint vertices u and v for some $0 < j \neq i$. Since B_i is connected (Lemma 3.3), there exists a vertex $w \in B_i$ which is on the path between u and v. If w is a heavy i-vertex, then it is the case of Lemma 3.9. Otherwise, if w is i-balanced, then there exist two disjoint sets $W_1^i, W_2^i \subseteq V_i$, each of weight at least $\epsilon/16k$, where every path between a pair of vertices $u_1 \in W_1^i$ and $u_2 \in W_2^i$ passes through w. The probability for any of the sets W_1^i and W_2^i or of w to not intersect the sample set of Step 1b is at most $(1-\epsilon/16k)^{20k\ln 12/\epsilon} < \frac{1}{12}$. The constraint vertices u and v are queried with probability 1. Thus, the algorithm will fail with probability at most $\frac{2}{12} = \frac{1}{6}$.

Third, suppose that a set B_i is on the path between a set B_j and a j-colored constraint vertex v for some $j \neq i$. Let u be the closest vertex to B_j in B_i, and let w be the closest vertex to B_i in B_j. Similarly to the previous case, the probability of not sampling two i-colored vertices such that w is on the path between them is at most $2(1-\epsilon/16k)^{20k\ln 12/\epsilon}$. Now, if u is heavy then the probability that it is not queried is at most $(1-\epsilon/16k)^{20k\ln 12/\epsilon}$. Otherwise, since u is i-balanced and w is not i-balanced, we have $\mu_i(C_w^{(u)}) > \epsilon/16k$, and thus, the probability of not sampling an i-colored vertex in $C_w^{(u)}$ is smaller than $(1-\epsilon/16k)^{20k\ln 12/\epsilon}$. As v is queried with probability 1, we obtain that the probability of not sampling an implicit witness (where w is the intersection vertex) is at most $3(1-\epsilon/16k)^{20k\ln 12/\epsilon} < \frac{3}{12} = \frac{1}{4}$.

Forth, if a j-colored constraint vertex w is on the path between a set B_i and an i-colored constraint vertex v, then the proof is similar to the previous case, only this time we obtain an explicit witness with high probability, as w is queried with probability 1. □

We henceforth assume that there is no constraint vertex $w \in B_i$ with $c(w) \neq i$, the B_i's are all disjoint, and coloring every set B_i with color i does not form a witness for non-quasi-convexity together with the constraint vertices. Define the sets S_i as in Subsection 3.1.

Lemma 3.12 *If the total weight of uncolored vertices inside the S_i's is larger than $\epsilon/4$, then Algorithm 3.7 rejects the input with probability at least $\frac{3}{4}$.*

Proof. For the analysis, we partition the set of free vertices sampled in Step 1b into two sets, a set X_1 with $\frac{4k\ln 12}{\epsilon}$ vertices, and a set X_2 with $\frac{16k\ln 12}{\epsilon}$ vertices. Note that X_1 and X_2 are independently random.

The probability that X_1 does not contain any uncolored vertex inside some S_i is at most $(1-\epsilon/4)^{\frac{4k\ln 12}{\epsilon}} < \frac{1}{12}$. Suppose that X_1 contains an uncolored vertex w inside an S_i. If w is i-balanced then there exist two disjoint sets $V_1, V_2 \subseteq V_i$, each of weight at least $\epsilon/16k$, such that every path between two vertices $v_1 \in V_1$ and $v_2 \in V_2$ passes through w. To reject the input, it is enough to sample one vertex from each of these sets. The probability that at least one of these

sets does not intersect X_2 is at most $2(1-\epsilon/16k)^{\frac{16k\ln 12}{\epsilon}} < 2\frac{1}{12}$. Thus, by the union bound, the algorithm will fail with probability at most $\frac{3}{12} = \frac{1}{4}$.

If w is not i-balanced then it is on a path between an i-balanced vertex and an i-colored constraint vertex, or between two i-colored constraint vertices, which can only increase the probability of discovering a witness and rejecting. □

Recall that F is the forest of all outsider vertices (see Subsection 3.1).

Lemma 3.13 *If F is bad then the algorithm rejects the input in Step 2 with probability at least 2/3.*

Proof. By Lemma 3.5, either the weight of obstacle vertices in F is larger than $\epsilon/8$ or the weight of bad uncolored vertices in F is larger than $\epsilon/16$.

If the weight of obstacle vertices in F is larger than $\epsilon/8$, then clearly an obstacle vertex u is sampled in Step 1b with probability larger than 2/3. Recall that there exists an uncolored constraint vertex w, which is an ancestor of u in F, that is sampled in Step 1a with probability 1. Let i be the color of u. Note that if i is a non-abundant color then w is on the path between u and an i-colored constraint vertex v. Hence, u, w, v form a forbidden subpath and thus the algorithm will certainly reject the input in Step 2. Therefore, the presence of obstacle vertices of non-abundant colors only increases the probability for rejection.

Now, assume that all the obstacle vertices are of abundant colors. For the analysis, we partition the set of free vertices queried in Step 1b into two sets, a set X_1 with $\frac{10k\ln 12}{\epsilon}$ vertices, and a set X_2 with $\frac{10k\ln 12}{\epsilon}$ vertices. Note that X_1 and X_2 are independently random. The probability that X_1 does not contain any obstacle vertex is at most $(1-\epsilon/8)^{\frac{10k\ln 12}{\epsilon}} < \frac{1}{12}$. Suppose that X_1 contains an obstacle vertex u, and let w be an uncolored constraint vertex which is an ancestor of u in F. Then, clearly, since w is an outsider vertex of a B_i of an abundant color, $\mu_i(V \setminus T_w) > \epsilon/8k$. Therefore, the probability that X_2 does not contain an i-vertex v outside T_w is at most $(1-\epsilon/8)^{\frac{10k\ln 12}{\epsilon}} < \frac{1}{12}$. Hence, the probability that a forbidden path is not discovered is at most $\frac{1}{6}$.

If the weight of bad uncolored vertices in F is larger than $\epsilon/16$, then the probability of not sampling a bad vertex w in F in Step 3 is at most $(1-\epsilon/16)^{16\ln 12/\epsilon} < \frac{1}{12}$. Suppose now that we have queried a bad vertex w in Step 3. Let i be the abundant color associated with w and let T_w be the subtree in F whose root is w. Note that $B_i \subseteq V \setminus T_w$ and hence, $\mu_i(T_w) < \frac{\epsilon}{16k}$, as otherwise w would have been i-balanced. We thus have $\mu_i(V \setminus T_w) > \frac{\epsilon}{4k} - \frac{\epsilon}{16k} > \frac{\epsilon}{8k}$, and hence the probability that X does not contain an i-vertex outside T_w is at most $(1-\epsilon/8k)^{\frac{20k\ln 12}{\epsilon}} < \frac{1}{12}$.

Suppose that X contains an i-vertex outside T_w. Then T_w is one of the trees T_w^i sampled in Step 4. As w is bad, the probability that the sample of T_w does not contain an i-vertex is at most $(1-\epsilon/16)^{\log_{1/(1-\epsilon/16)} 6} = \frac{1}{6}$.

To conclude, we can expect a bad vertex w associated with an abundant color i to be chosen in Step 3, with an i-vertex queried outside T_w in Step 1 and an i-vertex queried inside T_w in Step 3. In such a case the algorithm will detect an explicit witness and reject the input in Step 4. By the union bound, the probability of failure in this is at most $\frac{1}{12} + \frac{1}{12} + \frac{1}{6} = \frac{1}{3}$. □

Proof of Theorem 3.8. Follows from Proposition 3.6 and Lemmas 3.9, 3.10, 3.11, 3.12 and 3.13. □

3.3 Finding witnesses for non-quasi-convexity

The procedure for detecting witnesses with respect to quasi-convexity is very similar to the one presented in Section 2.3 for the convexity test. The only difference is that an uncolored vertex can only be a middle vertex in a forbidden subpath. Therefore, when considering a vertex v, we only need to check its colored children. In the following, a null value and a value of 0 for $m(v)$ are not the same. A null value of $m(v)$ means that the color of v is unknown or irrelevant, whereas $m(v) = 0$ indicates that v was queried and found to be uncolored.

Procedure 3.14 *For every v in reverse topological order, do:*

- *If $v \in X$ then set $a(v) = 1$; otherwise set $a(v) = 0$.*
- *If $v \in X$ then set $m(v) = c(v)$; otherwise set $m(v)$ to be null.*
- *For every child u of v such that $m(u)$ is not null and $m(u) > 0$:*
  1. *If $m(v)$ is not null and $m(v) \neq m(u)$ then reject the input and terminate.*
  2. *Otherwise, set $m(v) = m(u)$ and $a(v) = a(v) + a(u)$.*
- *If $m(v)$ is not null, $m(v) > 0$, and $a(v) = q_{m(v)}$, then set $m(v)$ to be null and $a(v) = 0$.*

If the algorithm did not reject after going over all vertices, then accept.

We prove the correctness of the procedure with the next lemma, whose proof is very similar to that of Lemma 2.13.

Lemma 3.15 *For every vertex v the following holds:*

1. *If Procedure 3.14 rejects in the iteration of v, then v is a middle vertex of a forbidden subpath, where if $v \in X$ then the forbidden subpath includes v as its middle vertex, and, otherwise, there exist colored vertices $a_1, a_2, b_1, b_2 \in X$ such that $c(a_1) = c(a_2) \neq c(b_1) = c(b_2)$, and v belongs to the path between a_1 and a_2 as well as to the path between b_1 and b_2.*
2. *If Procedure 3.14 completes the iteration of v without rejecting it, then v is not a middle vertex of a forbidden path as above.*
3. *If the processing of v is completed, then, in its end, $m(v) = i$ for $i > 0$ if and only if X includes i-vertices both inside and outside the subtree rooted in v. In such a case, $a(v)$ is equal to the number of i-vertices of X in the subtree rooted in v. Also, if $m(v)$ is null then $a(v) = 0$.*

As we did for Procedure 2.12, we may run Procedure 3.14 on a tree which contains only the queried vertices and all the vertices which are least common ancestors of two queried vertices. This reduces the running time of the implementation to $\widetilde{O}(k/\epsilon^2)$ (quasi-linear in the maximum sample size), if we use a preprocessing stage of time $O(n)$. See Section 2.3 for details.

4 Relaxed convexity properties

Given a tree $T = (V, E)$ and an integer $\ell > 0$, we say that a k-coloring $c : V \to [k]$ of T is *ℓ-convex* if it induces at most ℓ color components. We say that a quasi k-coloring $c : V \to \{0, \ldots, k\}$ of T is *ℓ-quasi-convex* if it induces at most ℓ components of colors $i > 0$. Given a list $L = \langle l_1, \ldots, l_k \rangle$ of integers we say that a vertex coloring of T is *convex with respect to L* if it induces at most l_i color components of every color i. If we allow some of the l_i's to be ∞, we then say that the coloring is *quasi-convex with respect to L*.

In the next subsection we present a 1-sided test for ℓ-convexity on a tree $T = (V, E)$, and later we explain how to transform it into a test for ℓ-quasi-convexity, list convexity and list quasi-convexity. Note that the query complexity and time complexity of all our tests is independent of k.

For convenience, we assume that the set of constraint vertices is empty. Our tests may be adapted to the change where there are constraint vertices by querying for their values. However, we use testing with constraints as an internal procedure. As our set of constraints is not fixed now, we use the term *(quasi) convex under D* to refer to the studied property when D is the set of constraint vertices.

4.1 ℓ-convexity of trees

This subsection is dedicated to the proof of the following theorem.

Theorem 4.1 *There exists a 1-sided test for ℓ-convexity of k-colorings of trees with query complexity $\widetilde{O}(\ell/\epsilon)$ and time complexity $O(\ell n)$.*

First, we give a few definitions to be used in the sequel. Consider the domain input tree T. A vertex w is said to be *between* the vertices u and v if it is an intermediate vertex on the (only) path between u and v. For any three distinct vertices $u, v, w \in V$, we define the *junction* $\mathrm{Junc}(u, v, w)$ of u, v, and w as follows. If any of the vertices in $\{u, v, w\}$ is between the other two, then $\mathrm{Junc}(u, v, w)$ is defined to be that vertex. Otherwise, $\mathrm{Junc}(u, v, w)$ is the unique vertex that lies in the intersection of the three simple paths between u and v, v and w, and u and w, respectively. A set U is *closed under junctions* if for any three distinct vertices $u, v, w \in U$ we also have $\mathrm{Junc}(u, v, w) \in U$. Note that if a set U is closed under junctions then all paths between pairs of adjacent vertices in U intersect each other only on members of U. We say that two vertices are *adjacent in a set U* if they are both in U and there are no other vertices in U between them.

We now characterize subtrees of T with respect to an input coloring c. A subtree is called *i-homogenous* if all its vertices are i-vertices, and *homogenous* if it is i-homogenous for some color i. We say that a subtree is *$\{i, j\}$-homogenous* if all its vertices are either i-vertices or j-vertices (if $i = j$, then clearly such a tree is i-homogenous). Given two vertices $u, v \in V$, we say that a subtree is *$\{u, v\}$-compatible* if it is both $\{c(u), c(v)\}$-homogenous and convex. Note that when saying that a subtree T' of T is close to (resp. far from) satisfying any of the above properties, we mean that that the restriction of c to T' is close to (resp. far from) satisfying it.

Before giving our algorithm for ℓ-convexity, we explain its main ideas. Throughout the algorithm we maintain a set IT of *interesting trees*, containing subtrees of T to be examined. We define the interesting trees using a set X of queried vertices. We create X in such a way that it is always closed under junctions. We ensure that the intersection between every two distinct subtrees in IT is either empty or consists of a single vertex in X. Let T_X be the spanning tree of X, that is, the

tree comprised of all the simple paths between vertices in X. We use T_X to define interesting trees of two types: A *pinned tree* is defined by two vertices u, v which are adjacent in X but not in V. It contains the path between u and v, as well as all the vertices whose nearest neighbor in T_X is between u and v (but is not u or v themselves). Equivalently, $T_{(u,v)} \stackrel{\text{def}}{=} \left(C_u^{(v)} \cap C_v^{(u)}\right) \cup \{u, v\}$. A *dangling tree* is defined by a vertex $u \in X$, and it contains u as well as all the vertices not in T_X whose nearest neighbor in T_X is u. Namely: $T_u \stackrel{\text{def}}{=} V \setminus \bigcup_{v \in X, v \neq u} C_u^{(v)}$.

Using the set X we can infer a lower bound on the number of color components in T, by assuming that every two adjacent vertices in X belong to the same color component if and only if they are colored with the same color. This could be the case because we may e.g. extend the coloring of X into a coloring of V by coloring every vertex with the color of its nearest neighbor in X. On the other hand, it can be easily seen that no coloring of V would give a smaller number of components for any of the colors.

In order to discover more color components, we examine pinned and dangling interesting subtrees of T one by one. For u, v which are adjacent in X and colored with the same color, we test $T_{(u,v)}$ for being $c(u)$-homogenous. If the test accepts, we remove $T_{(u,v)}$ from the set of interesting trees, as it is unlikely to provide us with more information on the color components in T (we shall later see that, in such a case, $T_{(u,v)}$ is "irrelevant"). Otherwise, we augment X with a witness for the non-homogeneity of $T_{(u,v)}$ (while keeping it closed under junctions), and replace $T_{(u,v)}$ in IT with its subtrees. Similarly, we test dangling subtrees T_u for $c(u)$-homogeneity. For u and v which are adjacent in X and colored with different colors, we test $T_{(u,v)}$ for being $\{u, v\}$-compatible under the constraint set $D = \{u, v\}$. That is, we test whether there is an ϵ-close convex coloring of $T_{(u,v)}$ that agrees with c on the colors of u and v. Again, if the test accepts then we discard $T_{(u,v)}$, and otherwise we proceed and divide it into smaller interesting trees. If at some point we have discovered more than ℓ color components, then the algorithm rejects the input. Otherwise, the algorithm terminates and accepts when there are no interesting trees left.

Note that, since we use the convexity testing algorithm as a subroutine for determined subtrees and accordingly sample from conditioned distributions, we lose its distribution-free quality.

We next introduce the subroutines used for testing interesting trees.

Observation 4.2 *Given a subtree T' of T, a color i and $0 < p < 1$, there exists an algorithm whose query and computational complexity are both $O(\log(1/p)\epsilon^{-1})$, such that: If T' is i-homogenous then the algorithm accepts with probability 1; and if T' is ϵ-far from being i-homogenous then, with probability at least $1 - p$, the algorithm rejects and finds a witness for its non homogeneity (i.e., a non-i-vertex).*

Proof. Given T', ϵ and p as above, query $2\ln(1/p)\epsilon^{-1}$ vertices in T' independently at random using the conditional distribution of μ to T'. If a vertex w has been found such that $c(w) \neq i$ then reject and return w as a witness, and otherwise accept. It is trivial to see that this algorithm satisfies the stated requirements. □

Lemma 4.3 *Given a pinned subtree $T_{(u,v)}$ of T with $c(u) \neq c(v)$ and $0 < p < 1$, there exists an algorithm with query complexity $O(\log(1/p)\epsilon^{-1})$ and computational complexity linear in the size of $T_{(u,v)}$ such that: If $T_{(u,v)}$ is $\{u, v\}$-compatible, then the algorithm accepts with probability 1; if $T_{(u,v)}$ is ϵ-far from being $\{u, v\}$-compatible under $\{u, v\}$, then, with probability at least $1-p$, the algorithm rejects and finds a witness for the incompatibility. Furthermore, the witness is either a vertex w*

with $c(w) \neq c(u), c(v)$ or a pair of vertices (x, w) such that x is between u and v, w has the same color as u or v, and $c(x) \neq c(w)$.

Proof. We use Algorithm 2.7 with $k = 2$ and $D = \{u, v\}$. For clarity, we first give a multi-phase algorithm, and later show how to perform it in one phase.

Given $T_{(u,v)}$ and p as above, repeat the following for $\log_3(1/p)$ times:

1. Query $8 \ln 12\epsilon^{-1}$ vertices independently uniformly at random. Let W be the union of $\{u, v\}$ and the set of queried vertices.

2. If W includes a vertex w such that $c(w) \neq c(u), c(v)$, then reject and return w.

3. Otherwise, if W includes a forbidden subpath $\langle w_1, w_2, w_3 \rangle$, then for every i we have either $c(w_i) = c(u)$ or $c(w_i) = c(v)$, as otherwise Case 2 applies.

   (a) Assume that one of the vertices in the forbidden subpath is either u or v. Since u and v are leaves and $c(u) \neq c(v)$, we can only have one of the end vertices in the subpath be u or v. Assume without loss of generality that $w_1 = u$. Let $x = \text{Junc}(u, v, w_2)$. Since u and v are not adjacent, x is between u and v. Query x.

   - If $c(x) \neq c(u), c(v)$ then set $w = x$ and return w as in Case 2.
   - Otherwise, if $c(x) = c(w_2)$ then clearly $\langle u, x, w_3 \rangle$ is a forbidden subpath and thus we set $w = w_3$.
   - Otherwise, $c(x) = c(u) \neq c(v)$, and thus $\langle v, x, w_2 \rangle$ is a forbidden subpath. We therefore set $w = w_2$.
   - Return x and w.

   (b) Otherwise, if both w_1 and w_3 are between u and v then so is w_2. Without loss of generality, assume that w_1 is between u and w_2 and w_3 is between w_2 and v.

   - If $c(w_2) = c(u)$ then $\langle u, w_1, w_2 \rangle$ is a forbidden subpath, and thus we set $x = w_1$ and $w = w_2$.
   - If $c(w_2) \neq c(u)$ then $\langle u, w_2, w_3 \rangle$ is a forbidden subpath, and thus we set $x = w_2$ and $w = w_3$.
   - Return x and w.

   (c) Now assume that both w_1 and w_3 are different from u and v and either w_1 or w_3 is not between u and v. Let $x = \text{Junc}(u, v, w_2)$. Since u and v are not adjacent, x is between u and v. Query x.

   - If $c(x) = c(w_2)$ then $c(x) \neq c(w_1) = c(w_3)$. Let w be either w_1 or w_3 such that w is not between u and v. Then either $\langle u, x, w \rangle$ or $\langle v, x, w \rangle$ is a forbidden subpath.
   - If $c(x) \neq c(w_2)$ then, in particular, $x \neq w_2$ and hence, x is both between u and w_2 and between v and w_2. Either $\langle u, x, w_2 \rangle$ or $\langle v, x, w_2 \rangle$ is a forbidden subpath. We therefore set $w = w_2$.
   - Return x and w.

4. Otherwise, if W includes vertices w_1, w_2, w_3, w_4 such that $c(w_1) = c(w_2) \neq c(w_3) = c(w_4)$ and there exists a vertex $\widetilde{w}$ which is both between w_1 and w_2 and between w_3 and w_4, then let $\widetilde{w} = \text{Junc}(w_1, w_2, w_3)$ be such a vertex. Query it.

- If $c(\widetilde{w}) \neq c(u), c(v)$ then set $w = \widetilde{w}$ and return w as in Case 2.
- Otherwise, either $\langle w_1, \widetilde{w}, w_2 \rangle$ or $\langle w_3, \widetilde{w}, w_4 \rangle$ is a forbidden subpath. Perform the same operations with the forbidden subpath as done in Case 3.

If the input has not been rejected in any of the iterations, accept. In this case $T_{(u,v)}$ is marked as "not interesting".

The above is the same as repeatedly running Algorithm 2.7 with $k = 2$ and $D = \{u, v\}$, except for the steps taken in order to find a witness of the desired form once $T_{(u,v)}$ is known not to be $\{u, v\}$-compatible. Clearly, these modifications do not change the fact that the algorithm always accepts a $\{u, v\}$-compatible input. It is easy to see that if $T_{(u,v)}$ is ϵ-far from $\{c(u), c(v)\}$-homogeneity, then it is rejected with probability at least $2/3$ in any given iteration. Assume that $T_{(u,v)}$ is ϵ-close to $\{c(u), c(v)\}$-homogeneity but ϵ-far from convexity under $\{u, v\}$. Then clearly by Theorem 2.8, $T_{(u,v)}$ is rejected with probability at least $2/3$ in any given iteration. As the iterations are independent, an ϵ-far input is rejected with probability at least $1 - p$. One can see that in Steps 3 and 4 of the algorithm, a forbidden subpath is found with either u or v as an endpoint.

Note that using a single sample of $8 \log_3(1/p) \ln 12\epsilon^{-1}$ vertices selected uniformly and independently, one can only increase the probability of finding a witness for farness, while still maintaining the 1-sidedness of the algorithm. Performing a single sample enables us to check for a forbidden subpath only once with time complexity linear in the size of $T_{(u,v)}$ (see Section 2.3). Moreover, it can be seen that the junction of every three vertices in $T_{(u,v)}$ may be computed using a naive DFS algorithm in time linear in the size of $T_{(u,v)}$. Since we compute at most two junctions, the computational upper bound hold. □

We now present our main test for ℓ-convexity.

Algorithm 4.4

- *Let $X = \{u\}$, where u is any vertex in V, and query it. Set $IT = \{T_u\} = \{T\}$. Set $CC = 1$.*
- *While $IT \neq \emptyset$ and $CC \leq \ell$, repeat:*
  1. *Consider a tree $T' \in IT$. Set $IT = IT \setminus \{T'\}$.*
  2. *Perform a test with error probability $p = 1/3\ell$ as follows:*
     - *If $T' = T_{(u,v)}$ for $u, v \in X$ such that $c(u) \neq c(v)$, perform a $\{u, v\}$-compatibility test.*
     - *Otherwise, if $T' = T_{(u,v)}$ for $u, v \in X$ such that $c(u) = c(v)$, perform a $c(u)$-homogeneity test.*
     - *Otherwise, if $T' = T_u$ for $u \in X$, perform a $c(u)$-homogeneity test.*
  3. *If the test has accepted, return to Step 1.*
  4. *Otherwise,*
     (a) *If $T' = T_{(u,v)}$ for $u, v \in X$ such that $c(u) = c(v)$ and a witness w has been found such that $c(w) \neq c(u)$:*
        - *Let $x = \mathrm{Junc}(u, v, w)$. Query x, and add x and w to X.*
        - *If $c(x) \neq c(u)$ set $CC = CC + 2$.*
        - *If $c(x) \neq c(w)$ set $CC = CC + 1$ (independently of the previous step).*
        - *If x is not a leaf, add T_x to IT.*

- *If w is not a leaf, add T_w to IT.*
- *If u and x are not adjacent in T, add $T_{(u,x)}$ to IT.*
- *If v and x are not adjacent in T, add $T_{(v,x)}$ to IT.*
- *If $w \neq x$ and w and x are not adjacent in T, add $T_{(w,x)}$ to IT.*

(b) If $T' = T_{(u,v)}$ for $u, v \in X$ such that $c(u) \neq c(v)$ and a witness w has been found such that $c(w) \neq c(u), c(v)$:

- *Let $x = \mathrm{Junc}(u, v, w)$. Query x, and add x and w to X.*
- *If $c(x) \neq c(u)$ and $c(x) \neq c(v)$ set $CC = CC + 1$.*
- *If $c(x) \neq c(w)$ set $CC = CC + 1$ (independently of the previous step).*
- *Add the new (non-degenerate) interesting trees into IT similarly to Case (a).*

(c) Otherwise, if $T' = T_{(u,v)}$ for $u, v \in X$ and witnesses x, w have been found such that x is between u and v, $c(x) \neq c(w)$ and either $c(w) = c(u)$ or $c(w) = c(v)$:

- *Add x and w to X.*
- *Set $CC = CC + 1$.*
- *If $c(x) \neq c(u)$ and $c(x) \neq c(v)$ set $CC = CC + 1$.*
- *Add the new (non-degenerate) interesting trees into IT similarly to Case (a).*

(d) Otherwise, $T' = T_u$ for $u \in X$ and a witness w has been queried such that $c(w) \neq c(u)$:

- *Add w to X.*
- *Set $CC = CC + 1$.*
- *If w is not a leaf, add T_w to IT.*
- *If u and w are not adjacent in T, add $T_{(u,w)}$ to IT.*

- *If $CC > \ell$, reject. Otherwise, if $CC \leq \ell$ and $IT = \emptyset$, accept.*

To prove the correctness of the algorithm, we need several lemmas.

Lemma 4.5 *Consider an iteration of the while loop in Algorithm 4.4. Let ℓ' be the value of CC when the iteration begins. Then:*

1. *Given X, ℓ' is the minimum number of color components in V.*
2. *Suppose that:*
   - *All the pinned trees $T_{(u,v)}$ $(u, v \in X)$ with $c(u) \neq c(v)$ are ϵ-close under $\{u, v\}$ to being $\{u, v\}$-compatible.*
   - *All the pinned trees $T_{(u,v)}$ $(u, v \in X)$ with $c(u) = c(v)$ are ϵ-close to being $c(u)$-homogenous .*
   - *All the dangling trees T_u $(u \in X)$ are ϵ-close to being $c(u)$-homogenous.*

   Then c is ϵ-close to being ℓ'-convex.

Proof. Part 1 of the lemma is easily proved by induction on ℓ'. As for Part 2, consider colorings of the pinned and dangling trees which are ϵ-close to the restriction of c to these trees and are convex or homogenous under their defining vertices. The intersections between the trees consist only of vertices in X, which are not recolored by any of theses close colorings. Therefore, we may combine them into a coloring of our entire tree T, which is ϵ-close to c and can be easily seen to be ℓ'-convex. □

Lemma 4.6 *The while loop of Algorithm 4.4 runs at most 5ℓ times.*

Proof. Since in every time the test of Step 2 rejects, CC is incremented, Step 4 can be applied at most ℓ times. Note that at most 5 new interesting trees are added in any single iteration of Step 4. Therefore, we may perform the test in Step 2 on at most 5ℓ trees. □

Proof of Theorem 4.1. By the first part of Lemma 4.5, CC is a tight lower bound for the number of color components in T, and therefore, the algorithm never rejects an ℓ-convex input.

Assume now that the input coloring c is ϵ-far from being ℓ-convex. Then, by the second part of Lemma 4.5, in any stage of the algorithm where $CC \leq \ell$, at least one of the interesting dangling trees T_u is ϵ-far from being $c(u)$-homogenous or at least one of the pinned trees $T_{(u,v)}$ is ϵ-far under $\{u, v\}$ from being $\{u, v\}$-compatible. After discovering the farness of ℓ interesting trees, the algorithm rejects. As the probability of not discovering the farness of a certain interesting tree is at most $1/3\ell$, the total failure probability is at most $1/3$. Therefore, Algorithm 4.4 is a 1-sided test for ℓ-convexity.

By Observation 4.2 and Lemma 4.3, the test in Step 2 can be implemented with query complexity $O(\log(\ell)/\epsilon)$. Therefore, the total query complexity of the algorithm is $O(\ell \log(\ell)/\epsilon)$. The computational complexity follows from Observation 4.2 and Lemmas 4.3 and 4.6. □

4.2 ℓ-quasi-convexity of trees

In this subsection we prove the following theorem.

Theorem 4.7 *There exists a 1-sided test for ℓ-quasi-convexity of quasi k-colorings of trees whose query complexity is $\widetilde{O}(\ell/\epsilon^2)$ and whose time complexity is $O(\ell n)$.*

Our test for ℓ-quasi-convexity is similar to that for ℓ-convexity. However, instead of using the test for convexity with or the test for homogeneity as a subroutine, we use the test for quasi-convexity with $k = 2$ when we have two different colors in the vertices defining the subtree, and with $k = 1$ when we have only one defining color. We only use the homogeneity test for interesting trees defined by uncolored vertices. We refer to the quasi-convexity property for $k = 1$ as *monotonicity*. In particular, we say that a dangling tree T_u is *monotone* if it is $\{c(u), 0\}$-homogenous and quasi-convex. Such a tree has only $c(u)$-colored and uncolored vertices, where no colored vertex is a descendant of an uncolored vertex. This complies with the common notion of monotonicity over trees for functions with two values, where an i-vertex is considered of "smaller" value than an uncolored vertex.

Observation 4.8 *Given a dangling subtree T_u of T and $0 < p < 1$, there exists an algorithm whose query and computational complexity are both $O(\log(1/p)\epsilon^{-2})$, such that: If T' is monotone then the algorithm accepts with probability 1; if T' is ϵ-far from being monotone then, with probability at least*

$1-p$, *the algorithm rejects and finds a witness for the lack of monotonicity. Moreover, the witness is either a vertex* w *such that* $c(w) > 0$ *and* $c(w) \neq c(u)$*, or vertices* u, x, w *such that* $\langle u, x, w \rangle$ *is a forbidden subpath.*

Proof. As in the proof of Lemma 4.3, we run Algorithm 3.7 for $k = 1$ with a sample set whose size is increased by a factor of $O(\log(1/p))$. The algorithm rejects or accepts as required, due to Theorem 3.8. Finding the required witnesses is done with techniques similar to those used in Lemma 4.3. □

We say that a subtree T' of T is $\{i, j\}$-*quasi-homogenous* if all its vertices are either uncolored or colored with either i or j. Given two distinct vertices u and v, we say that the pinned tree $T_{(u,v)}$ is $\{u, v\}$-*quasi-compatible* if it is $\{c(u), c(v)\}$-quasi-homogenous and quasi-convex (clearly, if $c(u) = c(v)$ then $T_{(u,v)}$ is monotone).

Lemma 4.9 *Given a pinned subtree* $T_{(u,v)}$ *of* T *and* $0 < p < 1$*, there exists an algorithm with query complexity* $O(\log(1/p)\epsilon^{-2})$ *and computational complexity linear in the size of* T *such that: If* $T_{(u,v)}$ *is* $\{u, v\}$*-quasi-compatible, then the algorithm accepts with probability 1; if* $T_{(u,v)}$ *is* ϵ*-far under* $\{u, v\}$ *from being* $\{u, v\}$*-quasi-compatible, then, with probability at least* $1 - p$*, the algorithm rejects and finds a witness for the incompatibility. Furthermore, a witness for the fact that* $T_{(u,v)}$ *is not* $\{u, v\}$*-quasi-compatible will be either a colored vertex* w *with* $c(w) \neq c(u)$ *and* $c(w) \neq c(v)$*, or a pair of vertices* x, w *such that* x *is between* u *and* v*,* w *is colored and has the same color as* u *or* v*, and* $c(x) \neq c(w)$*.*

Proof. Follows from Theorem 3.8, in a similar manner as Lemma 4.3 follows from Theorem 2.8. As before we select only the desired witnesses, although others may also be discovered. □

Note that here when rejecting an interesting tree, some of the witnesses may be uncolored. In such a case we do not account for the newly discovered uncolored components in our color components counter. However, we do add trees defined by uncolored vertices to our set of interesting trees, as we need to test them further in order to search for additional color components.

We are now ready to give our test for ℓ-quasi-convexity.

Algorithm 4.10 *The algorithm is the same as Algorithm 4.4, except for the following:*

- *In the initialization step, if* u *is uncolored then we set* CC *to* 0 *rather than* 1*.*
- *In Step 2 of the while loop:*
  - *For a pinned tree* $T_{(u,v)}$ *with colored* u *and* v *and* $c(u) \neq c(v)$ *we perform a test for* $\{u, v\}$*-quasi-compatibility under* $\{u, v\}$*.*
  - *For a tree defined by uncolored vertices, i.e., a pinned tree* $T_{(u,v)}$ *with uncolored* u *and* v *or a dangling tree* T_u *with* u *uncolored, we perform a* 0*-homogeneity test.*
  - *For other interesting trees (i.e., a pinned tree* $T_{(u,v)}$ *with colored* u *and* v *and* $c(u) = c(v)$*, a pinned tree* $T_{(u,v)}$ *with colored* u *and uncolored* v*, or a dangling tree* T_u *with colored* u*) we perform a monotonicity test, under the respective defining vertices as constraints.*

- *We update our counter CC differently, so that it is a lower bound for the number of color components of colored vertices only. That is, when our witness for rejecting an interesting tree includes an uncolored vertex, we do not increment CC to account for the newly discovered uncolored component, but only for components of colored vertices.*

Lemma 4.11 *Consider an iteration of the while loop in Algorithm 4.10. Let ℓ' be the value of CC when the iteration begins. Then:*

1. *Given X, ℓ' is the minimum number of color components of colored vertices in V.*
2. *Suppose that:*
   - *All the interesting trees defined by uncolored vertices are ϵ-close to being 0-homogenous.*
   - *All the dangling trees T_u ($c(u) > 0$) are ϵ-close to being monotone.*
   - *All the trees $T_{(u,v)}$ with $c(u) \neq c(v)$ and $c(u), c(v) > 0$ are ϵ-close under $\{u, v\}$ to being $\{u, v\}$-quasi-compatible.*
   - *All the other pinned trees $T_{(u,v)}$ are ϵ-close to being monotone under their respective constraints.*

   Then c is ϵ-close to being ℓ'-quasi-convex.

Proof. Similar to the proof of Lemma 4.5. □

Lemma 4.12 *The while loop of Algorithm 4.4 runs at most 5ℓ times.*

Proof. As in the case of Algorithm 4.4, at most 5 new interesting trees are added in any single iteration of Step 4, so CC is linear in the number of iterations. Moreover, CC is incremented every time the test in Step 2 rejects. To see this, note that in every rejection we either have a colored witness defining a new colored component inside an interesting tree, or a forbidden subpath with respect to quasi-convexity. In the latter case, the endpoints of the forbidden subpath originally belonged to the same presumed color component. Finding the forbidden subpath reveals that there are at least two color components instead of the original presumed one. Hence, CC is incremented after discovering the forbidden subpath.

Concluding, Step 4 may be applied only 5ℓ times before either CC exceeds ℓ or all interesting trees are exhausted and the loop ends. □

Proof of Theorem 4.7. We show that Algorithm 4.10 satisfies the stated requirements.

By the first part of Lemma 4.11, CC is a tight lower bound for the number of color components (of colored vertices) in T, and therefore, the algorithm never rejects an ℓ-quasi-convex coloring c.

Assume that c is ϵ-far from being ℓ-quasi-convex. Then, by the second part of Lemma 4.11, in any stage of the algorithm where $CC \leq \ell$, at least one of the interesting subtrees is ϵ-far from the property it is being tested for in Step 2 of the algorithm. After discovering the farness of ℓ interesting trees, the algorithm rejects. As the probability of not discovering the farness of a certain interesting tree is at most $1/3\ell$, the total failure probability is at most $1/3$. Therefore, Algorithm 4.10 is a 1-sided test for ℓ-quasi-convexity.

By Observation 4.8 and Lemma 4.9, the test in Step 2 can be implemented in query complexity $O(\log(\ell)/\epsilon^2)$. Therefore, the total query complexity of the algorithm is $O(\ell \log(\ell)/\epsilon^2)$. The computational complexity follows from Observation 4.8 and Lemmas 4.9 and 4.12. Note that computing the distribution μ conditioned on a dangling tree or on a pinned tree can be done in time $O(n)$ using an appropriate BFS traversal. □

4.3 List convexity and list quasi-convexity of trees

Theorem 4.13 *Given a list $L = \langle l_1, \ldots, l_k \rangle$ of integers, there exists a 1-sided test for convexity with respect to L on trees, with query complexity $\widetilde{O}(\ell/\epsilon)$ and computational complexity $O(\ell n)$, where $\ell = \sum_{i=1,\ldots,k} l_i$.*

Theorem 4.14 *Given a list $L = \langle l_1, \ldots, l_k \rangle$ where every l_i is either an integer or ∞, there exists a 1-sided test for quasi-convexity with respect to L on trees, with query complexity $\widetilde{O}(\ell/\epsilon^2)$ and computational complexity $O(\ell n)$, where $\ell = \sum_{1 \leq i \leq k,\ l_i < \infty} l_i$.*

The tests used to prove both theorems above are almost identical to our tests for ℓ-convexity and ℓ-quasi-convexity, respectively. The only difference is that instead of the counter CC of the total number of color components discovered, we keep a counter CC_i for every color i with $l_i < \infty$.

5 A lower bound for testing convexity on trees

In this section we provide a lower bound for testing convexity on trees, which applies also for quasi-convexity.

Theorem 5.1 *For every $0 < \epsilon < 1/8$, every (adaptive) ϵ-test for convexity of k-colorings of trees must use more than $\sqrt{3\frac{(k-1)}{64\epsilon}}$ queries in the worst case. This is specifically true for the case where T is a path, μ is a uniform distribution and there are no constraint vertices.*

Proof. Let T be a path of length n. According to Yao's theorem [24], it is enough to provide a distribution of input colorings, such that any deterministic algorithm whose inputs are chosen according to that distribution and uses $q \leq \sqrt{\frac{3(k-1)}{64\epsilon}}$ queries will fail to give the correct answer with probability larger than $\frac{1}{3}$. More precisely, we will present two distributions of inputs. D_P will be a distribution of convex k-colorings of T and D_N will be a distribution of k-colorings of T which are ϵ-far from being convex. We will prove that any deterministic algorithm using $q \leq \sqrt{\frac{3(k-1)}{64\epsilon}}$ queries has an error probability larger than $\frac{1}{3}$ when trying to distinguish between D_P and D_N.

Assume that k divides n. In both distributions we divide T into k intervals of size n/k each, such that all the vertices in each interval are colored with the same color. Without loss of generality, we can assume that the testing algorithm queries at most one vertex from every interval.

Definition 5.2 *Let D_P be the distribution of k-colorings defined by uniformly choosing a permutation of all k colors and coloring the intervals accordingly.*

Clearly, all colorings in D_P are convex. To define the distribution D_N of ϵ-far colorings, we use an auxiliary distribution $\widetilde{D_N}$ over colorings which are ϵ-far from being convex with high probability.

Definition 5.3 *Let $\widetilde{D_N}$ be the distribution of k-colorings selected by uniformly choosing $(1-8\epsilon)k$ colors to appear in one interval and $4\epsilon k$ colors to appear in two intervals. The placements of the colors are then chosen uniformly at random.*

Definition 5.4 *Let D_N be the conditional distribution of $\widetilde{D_N}$ on the event that the coloring chosen by $\widetilde{D_N}$ is ϵ-far from being convex.*

The main idea of the proof is based on the birthday problem. A test can distinguish D_P from D_N only if at least one of the query sets includes some color more than once. We show that a test that uses q queries is likely to fail in discovering a collision in a uniformly sampled set of colors, and thus cannot distinguish D_P from D_N.

Consider a (possibly adaptive) deterministic algorithm $\mathcal{A}$ that uses q queries. For any k-coloring c of T, let $\Pr_P[c]$ be the probability of c according to D_P, and let $\Pr_N[c]$ be the probability of c according to D_N. Without loss of generality, every deterministic algorithm with q queries takes the shape of a decision tree, which is a complete balanced k-ary tree of height q, where every non-leaf node corresponds to a query location with its children being labelled according to the possible outcomes of the query. Every leaf node corresponds to an answer sequence $g \in [k]^q$ with its acceptance or rejection decision. For a k-coloring c, we denote the answer sequence of our algorithm by $\mathcal{A}(c)$. For any answer sequence $g \in [k]^q$, let $\Pr_{\mathcal{A},P}[g]$ be the probability that the answer sequence is g for a coloring selected from D_P. Formally, $\Pr_{\mathcal{A},P}[g] \stackrel{\text{def}}{=} \sum_{c:\mathcal{A}(c)=g} \Pr_P[c]$. Define $\Pr_{\mathcal{A},N}[g]$ similarly as the probability that the answer sequence is g for a coloring selected from D_N, or $\Pr_{\mathcal{A},N}[g] \stackrel{\text{def}}{=} \sum_{c:\mathcal{A}(c)=g} \Pr_N[c]$. Now let a_P denote the probability that the algorithm accepts an input chosen according to D_P, and let a_N be the probability that the algorithm accepts an input chosen according to D_N. To prove the theorem, it is enough to show that $|a_P - a_N| < \frac{1}{3}$. See [4] for details.

Lemma 5.5 *A coloring chosen from $\widetilde{D_N}$ is ϵ-far from being convex with probability larger than $\frac{3}{4}$.*

Proof. For the analysis, we tag differently each appearance of colors that appear twice in a coloring chosen from $\widetilde{D_N}$. The total number of possible colorings in this distribution is $k!$. In colorings that are ϵ-close to convexity, however, at least $2\epsilon k$ colors that appear twice must appear on adjacent intervals. Otherwise, there are more than $2\epsilon k$ pairs of intervals of the same color which are separated by interval(s) of different colors. Thus, at least one interval must be recolored for every such pair to achieve a convex coloring. It is easy to see that, in the best case, changing the color of an interval from i to j can solve the problem for both colors i and j, but not for any other color $\ell \neq i, j$. Hence, if there are more than $2\epsilon k$ colors that appear each on two non-adjacent intervals, then the coloring is ϵ-far from being convex. The number of colorings in $\widetilde{D_N}$ that are ϵ-close to convexity is thus at most

$$\binom{4\epsilon k}{2\epsilon k} 2^{2\epsilon k}((1-2\epsilon)k)! = \exp(k)((1-2\epsilon)k)!$$

for choosing the $2\epsilon k$ colors which appear consecutively among those who appear twice, choosing the order of the intervals in every such pair, and then choosing the order of all intervals, where each consecutive pair is now counted as a single interval. Hence the probability of a coloring in $\widetilde{D_N}$ to be ϵ-close to convexity is at most

$$\frac{\exp(k)((1-2\epsilon)k)!}{k!} \leq \frac{\exp(k)}{((1-2\epsilon)k)^{2\epsilon k}},$$

which is smaller than $\frac{1}{4}$ for a sufficiently large k. $\square$

Let $\Pr_{\widetilde{N}}[c]$ denote the probability of a k-coloring c when chosen from the distribution $\widetilde{D_N}$ and let $\Pr_{\mathcal{A},\widetilde{N}}[g] \stackrel{\text{def}}{=} \sum_{c:\mathcal{A}(c)=g} \Pr_{\widetilde{N}}[c]$. We complete the proof by showing that the distributions D_P and D_N satisfy the required condition $|a_P - a_N| < \frac{1}{3}$. The main idea is to show that with high probability, an answer sequence of size q will not contain two appearances of the same color for both distributions, and in such a case, the algorithm will be unable to distinguish between them. We note that the proof of the farness of inputs drawn according to $\widetilde{D_N}$ also holds for quasi-convexity, and so the proof here provides a lower bound for testing quasi-convexity as well.

Observation 5.6 *For any answer sequence g we have*

$$\Pr_{\mathcal{A},N}[g] < \frac{4}{3} \Pr_{\mathcal{A},\widetilde{N}}[g].$$

Proof. Let C be the set of all the colorings which are chosen with positive probability according to $D_{\widetilde{N}}$. Let C_ϵ be the subset of C containing colorings which are ϵ-far from being convex.

By definition,

$$\Pr_{\mathcal{A},N}[g] = \sum_{c \in C_\epsilon : \mathcal{A}(c)=g} \Pr_N[c].$$

Since $D_{\widetilde{N}}$ is a uniform distribution over C and D_N is a uniform distribution over C_ϵ, and, by Lemma 5.5, $|C_\epsilon| > \frac{3}{4}|C|$, for every coloring $c \in C_\epsilon$ we have $\Pr_N[c] < \frac{4}{3} \Pr_{\widetilde{N}}[c]$. Therefore,

$$\Pr_{\mathcal{A},N}[g] = \sum_{c \in C_\epsilon : \mathcal{A}(c)=g} \Pr_N[c] < \frac{4}{3} \sum_{c \in C_\epsilon : c(\mathcal{A})=g} \Pr_{\widetilde{N}}[c] \leq \frac{4}{3} \Pr_{\mathcal{A},\widetilde{N}}[g].$$

$\square$

Let $g \in [k]^q$ be an answer sequence. We say that g is *colorful* if there exists no color that appears twice in g. Otherwise, we say that g is *degenerate*.

Lemma 5.7 *Let α_N be the probability that the answer sequence is degenerate when the input is chosen from D_N. In other words, let $\alpha_N = \sum_{g \text{ is degenerate}} \Pr_{\mathcal{A},N}[g]$. Then $\alpha_N < \frac{1}{4}$.*

Proof. We first compute $\alpha_{\widetilde{N}}$, namely, the probability that the answer sequence is degenerate when the input is chosen from $\widetilde{D_N}$. From symmetry arguments, as long as the algorithm has not queried two segments of the same color, we may perceive the querying process as choosing elements, one by one, without repetitions, from the set of colors (some of which appear twice). Therefore, the probability that at least one color is queried twice within q queries is no larger than the probability of choosing a color twice when the set of q locations is predetermined. Thus, the number of possibilities in which at least one color appears twice in the answer sequence is at most $4\epsilon kq(q-1)\binom{k-2}{q-2}(q-2)!$, as there are $4\epsilon k$ colors with two segments, and after choosing such a color and choosing its positions in the answer sequence, we are left to choose the other $(q-2)$ positions among $(k-2)$ other segments. We have that the probability of having a color appearing twice in the answer sequence is at most

$$\frac{4\epsilon kq(q-1)\binom{k-2}{q-2}(q-2)!}{\binom{k}{q}q!} = \frac{4\epsilon q(q-1)}{k-1}.$$

As $q(q-1) < q^2 \leq \frac{3(k-1)}{64\epsilon}$, we obtain $\alpha_{\widetilde{N}} < \frac{3}{16}$.

From Observation 5.6, for any answer sequence g we have $\Pr_{\mathcal{A},N}[g] < \frac{4}{3}\Pr_{\mathcal{A},\widetilde{N}}[g]$. Thus, by summing for all degenerate answer sequences, we have that $\alpha_N < \frac{4}{3}\alpha_{\widetilde{N}} < \frac{1}{4}$. □

Lemma 5.8 *For any colorful answer sequence $g \in [k]^q$,*

$$\Pr_{\mathcal{A},N}[g] \leq \Pr_{\mathcal{A},P}[g] < \frac{4}{3}\Pr_{\mathcal{A},N}[g].$$

Proof. From symmetry arguments, the probabilities of all colorful answer sequences are equal when the input is chosen from D_N, as well as when the input is chosen from D_P (for which the answer sequence is always colorful). Thus

$$\Pr_{\mathcal{A},P}[g] = \Pr_{\mathcal{A},N}[g| \text{ the answer sequence is colorful }] = \frac{\Pr_{\mathcal{A},N}[g]}{1-\alpha_N}.$$

Hence the first inequality in the statement of the lemma is trivially correct, and the second is derived from Lemma 5.7. □

We now complete the proof of Theorem 5.1 by showing that $|a_P - a_N| < \frac{1}{3}$. Let a_N^c be the probability that an input from D_N is accepted based on a colorful answer sequence. Let a_N^d be the probability that an input from D_N is accepted while a degenerate answer sequence was obtained. Thus $a_N = a_N^c + a_N^d$. From Lemma 5.8, $0 \leq \Pr_{\mathcal{A},P}[g] - \Pr_{\mathcal{A},N}[g] < \frac{1}{3}\Pr_{\mathcal{A},N}[g]$ for any colorful answer sequence. Thus, $0 \leq a_P - a_N^c < \frac{1}{3}$. In addition, from Lemma 5.7, $0 \leq a_N^d \leq \alpha_N < \frac{1}{4}$. Hence $|a_P - a_N| < \frac{1}{3}$. □

6 A convexity test for paths

We now present a standard (non distribution-free) convexity test for the special case where the tree $T = (V, E)$ is a path, whose performance is better than that of Algorithm 2.7 when the number of colors k is large enough with respect to $1/\epsilon^3$. We assume that there are no constraint vertices, but the algorithm can easily be generalized to the case where constraint vertices exist.

We note that a colored path is essentially a string. The convexity property on strings is a special case of a regular language, and thus is known to be testable by Alon et. al [1]. However, the query complexity obtained there, though polynomial in $\frac{1}{\epsilon}$, is exponential in the size of the DFA accepting the language, and in the case of the convexity of a string over k colors, it can be seen that the size of the DFA must be exponential in k. We provide a more efficient algorithm for this property. Actually, by the lower bound that we have established in Subsection 5, our algorithm is optimal up to a power of $\frac{1}{\epsilon}$.

We view one of the two leaves of T, denoted v_L, as the "leftmost" vertex and the other one, denoted v_R, as the "rightmost" vertex, thereby defining a linear left-to-right order on V. Henceforth, for every $v_1, v_2 \in V$, the closed interval $[v_1, v_2]$ denotes the subset of V which contains v_1, v_2 and all the vertices which are right of v_1 and left of v_2. The open interval (v_1, v_2) denotes the set of all the vertices which are right of v_1 and left of v_2.

Algorithm 6.1

1. *Query $q \geq \frac{1280\sqrt{k}}{\epsilon^2}$ vertices, where every vertex is chosen independently according to the weight function μ. Let $v_1, \ldots, v_r$ be the vertices queried, numbered from left to right (for some $r \leq q$).*

2. *For every $1 \leq i < r$, if there are vertices in the open interval (v_i, v_{i+1}), query $z \geq \frac{5}{\epsilon} \ln 12$ vertices in (v_i, v_{i+1}), where every vertex is chosen independently according to μ conditioned on this interval.*

3. *Reject if and only if the resulting sample contains a forbidden subpath.*

Note that Algorithm 6.1 is non-adaptive, since the distribution of the queries in Step 2 depends only on the positions of the queries of Step 1, rather than their answers. On the other hand, note that Algorithm 6.1 does not generalize directly to a distribution-free test, as the samples in Step 2 are performed in specific intervals, whose total weight might be arbitrarily small.

Theorem 6.2 *For every $\epsilon > 0$, Algorithm 6.1 is a 1-sided non-adaptive ϵ-test for convexity of k-colorings of paths, with query complexity $O(\sqrt{k}/\epsilon^3)$. The additional time complexity is $\widetilde{O}(\sqrt{k}/\epsilon^3)$ if the labels of the vertices in the path are sorted, and $O(n)$ otherwise.*

The query complexity is clearly as stated. To implement Step 3, we scan the vertices queried in Steps 1 and 2 from left to right while searching for a forbidden path (sorting the sampled vertices would take us $\widetilde{O}(\sqrt{k}/\epsilon^3)$ time if the vertices in the path are already sorted, and $O(n)$ time otherwise; these bounds also apply to sorting the vertices queried in Step 1 in order to compute the intervals sampled in Step 2). Each time we arrive at the *end* of a segment of a certain color, we add it to a list. A forbidden subpath exists in the sample if and only if we read a vertex in a color already in the list. Thus the time complexity requirement is fulfilled.

Clearly, a convex coloring of T is accepted by the algorithm with probability 1, as it does not contain any forbidden subpaths. Therefore, it remains to show that every coloring which is ϵ-far from being convex is rejected with probability at least $\frac{2}{3}$.

Recall that for every color $i \in [k]$, V_i is the set of vertices $u \in V$ such that $c(u) = i$. We refer to vertices in V_i as *i-vertices* and to other vertices as *non-i-vertices*. For any subset $U \subseteq V$, let the *i-weight* of U be the total weight of all i-vertices in U, and denote it by $\mu_i(U) \stackrel{\text{def}}{=} \mu(V_i \cap U)$. We refer to the total weight of non-i-vertices in a set U, namely $\mu(U) - \mu_i(U)$, as its *non-i-weight*.

For every color i such that $\mu_i(V) > 0$, let l_i be the leftmost vertex such that $\mu_i([v_L, l_i]) \geq \epsilon\mu_i(V)/4$. Let $L_i \stackrel{\text{def}}{=} [v_L, l_i]$ be called *i's left side*. Equivalently, let r_i be the rightmost vertex such that $\mu_i([r_i, v_R]) \geq \epsilon\mu_i(V)/4$. Let $R_i = [r_i, v_R]$ be called *i's right side*. Note that l_i and r_i are both i-vertices. We refer to the (possibly empty) open interval (l_i, r_i) as *i's middle*. We say that i is *bad* if the non-i-weight of i's middle is at least $\epsilon\mu_i(V)/4$. Otherwise, we say that i is *good*. Define *i's extended middle* to be the closed interval $M_i \stackrel{\text{def}}{=} [l_i, r_i]$. Note that due to the minimality of i's left side and right side, we have $\mu_i(M_i) > (1 - \epsilon/2)\mu_i(V)$.

We say that a color $i \in [k]$ is *abundant* if $\mu_i(V) \geq \epsilon/8k$, and otherwise we say that i is *non-abundant*. Let $\mathcal{A} \subseteq [k]$ be the set of all abundant colors, and let $\mathcal{N}_\mathcal{A} = [k] \setminus \mathcal{A}$ be the set of all non-abundant colors. Note that $\Sigma_{i \in \mathcal{N}_\mathcal{A}} \mu_i(V) < \epsilon/8$. We further denote the set of all abundant good colors with $\mathcal{A}_{good}$ and the set of all abundant bad colors with $\mathcal{A}_{bad}$.

Lemma 6.3 *If c is ϵ-far from being convex then $\sum_{i \in \mathcal{A}_{bad}} \mu_i(V) \geq \frac{\epsilon}{8}$.*

Proof. Assume on the contrary that $\sum_{i\in\mathcal{A}_{bad}} \mu_i(V) < \frac{\epsilon}{8}$. We define a convex coloring $\widetilde{c}$ of T in two phases. In Phase 1 we color all the vertices that belong to some M_i for all the colors i which are abundant and good. We do so by examining the l_i's from left to right. For every $i \in \mathcal{A}_{good}$ we color with i all the vertices in M_i that have not yet been colored in earlier stages. These vertices must belong to one consecutive interval. Otherwise, there exists a color j such that M_j is contained within M_i and is colored before M_i is. However, this is impossible, since we consider the l_i's from left to right and l_j is to the right of l_i. Hence, by the end of Phase 1 we have a partial convex coloring defined on all the M_i's of abundant good colors. In Phase 2 we extend this partial convex coloring into a complete convex coloring, by assigning to each uncolored segment one of the colors of its neighboring colored segments.

We now show that $\widetilde{c}$ is ϵ-close to c, which contradicts the assumption that c is ϵ-far from being convex. By definition, for every good color i, the non-i-weight of M_i is smaller than $\epsilon\mu_i(V)/4$. Hence, the non-i-weight of every interval colored with i in Phase 1 is smaller than $\epsilon\mu_i(V)/4$, and thus the total weight of vertices colored in Phase 1 differently than c is less than $\epsilon/4$. On the other hand, since for every good color i we have $\mu_i(M_i) > (1-\epsilon/2)\mu_i(V)$, the total weight of vertices colored in Phase 1 is

$$\sum_{i\in\mathcal{A}_{good}} \mu(M_i) \geq \sum_{i\in\mathcal{A}_{good}} \mu_i(M_i) > \sum_{i\in\mathcal{A}_{good}} (1-\epsilon/2)\mu_i(V)$$

$$= (1-\epsilon/2)\left(1 - \sum_{i\in\mathcal{A}_{bad}} \mu_i(V) - \sum_{i\in N_{\mathcal{A}}} \mu_i(V)\right) \geq (1-\epsilon/2)(1-\epsilon/4) > 1 - \frac{3\epsilon}{4}.$$

Hence, the total weight of vertices colored in Phase 2 is smaller than $3\epsilon/4$. We thus conclude that the distance between c and $\widetilde{c}$ is smaller than ϵ. □

Lemma 6.4 *Suppose that c is ϵ-far from being convex. Then with probability greater than $\frac{3}{4}$, there exists an abundant bad color i such that Step 1 of Algorithm 6.1 queries at least one i-vertex from each of i's sides.*

Proof. We first prove the claim for the case where $k < 16$. From Lemma 6.3 we have $\sum_{i\in\mathcal{A}_{bad}} \mu_i(V) \geq \frac{\epsilon}{8} > 0$. Let i be a color in $\mathcal{A}_{bad}$. By the definition of L_i, we have $\mu_i(L_i) \geq \epsilon\mu_i(V)/4$, and since i is abundant, we have $\mu_i(L_i) \geq \epsilon^2/32k > \epsilon^2/512$. Thus, the probability of not choosing an i-vertex in L_i in step 1 is smaller than

$$\left(1 - \frac{\epsilon^2}{512}\right)^{1280\sqrt{k}/\epsilon^2} < \exp\left(-\frac{1280}{512}\right) < 0.1.$$

Similarly, the probability of not choosing an i-vertex in R_i in Step 1 is smaller than 0.1. Therefore, the probability of not sampling an i-vertex from at least one of L_i and R_i is at most 0.2.

Suppose now that $k \geq 16$. Consider the half-closed interval $S = [0, 1)$ of real numbers. Suppose that we partition S into half-closed *vertex intervals* $I_{v_L}, \ldots, I_{v_R}$ of length $\mu(v)$ for each I_v according to the left to right order, that is, $I_{V_L} = [0, \mu(v_L)), \ldots, I_{v_R} = [1-\mu(v_R), 1)$. Note that uniformly selecting a point $x \in S$ and then choosing the vertex corresponding to the interval containing x is equivalent to selecting a vertex in V according to the distribution μ. We henceforth view the

sample in Step 1 of the algorithm as a sample of points in S. We say that a point $x \in S$ is an *i-point* if $x \in I_v$ for some i-vertex v.

Let τ_i be the minimum point in S such that $\mu_i([0,\tau_i]) = \epsilon\mu_i(V)/4$, and denote $Ł_i = [0,\tau_i]$. Let η_i be the maximum point in S such that $\mu_i([\eta_i,1)) = \epsilon\mu_i(V)/4$, and denote $\mathcal{R}_i = [\eta_i,1)$. Clearly, $Ł_i$ and $\mathcal{R}_i$ are disjoint intervals. Moreover, note that sampling a point in $Ł_i$ leads to selecting a vertex in L_i and sampling a point in $\mathcal{R}_i$ leads to selecting a vertex in R_i. We prove the lemma by showing that, with probability at least 3/4, there exists a color $i \in \mathcal{A}_{bad}$ such that the sample in Step 1 contains an i-point $x \in Ł_i$ and an i-point $y \in \mathcal{R}_i$.

For every $i \in \mathcal{A}_{bad}$, we consider the *color set* I_i, created from the union of the (parts of) vertex intervals of i-points in $Ł_i$. Clearly, $\mu(I_i) = \mu_i(Ł_i) = \epsilon\mu_i(V)/4$. We now define $\lfloor \frac{64\mu_i(Ł_i)k}{\epsilon^2} \rfloor$ disjoint subsets of I_i of measure $\frac{\epsilon^2}{64k}$ each, which we call *subcolor sets*. Note that a subcolor set may intersect several vertex intervals, and vice versa. However, this will not be a problem for us, as we are interested in the sampling of points.

For the sake of the analysis, we partition the points sampled in Step 1 into two disjoint sets, X_1 and X_2, where $|X_1| \geq 256\sqrt{k}/\epsilon^2$ and $|X_2| \geq 1024\sqrt{k}/\epsilon^2$. We next prove that, with high probability, X_1 contains samples from at least $\sqrt{k}$ distinct subcolor sets in $Ł_i$. For all colors $i \in \mathcal{A}_{bad}$ we have $\mu_i(Ł_i) = \frac{\epsilon}{4}\mu_i(V)$, and since these colors are abundant, we have $\mu_i(Ł_i) \geq \frac{\epsilon^2}{32k}$. From Lemma 6.3 we have $\sum_{i\in\mathcal{A}_{bad}} \mu_i(V) \geq \frac{\epsilon}{8}$, and thus $\sum_{i\in\mathcal{A}_{bad}} \mu_i(Ł_i) \geq \frac{\epsilon^2}{32}$. Note that for every $i \in \mathcal{A}_{bad}$, the part of the color set I_i that is not contained within subcolor sets is of measure smaller than $\frac{\epsilon^2}{64k}$. Hence, the total measure of subcolor sets is greater than $\frac{\epsilon^2}{32} - k\frac{\epsilon^2}{64k} = \frac{\epsilon^2}{64}$, and thus, the expected number of points in X_1 that are within subcolor sets (henceforth *subcolor points*) is greater than $4\sqrt{k}$. By the Chernoff bound, the probability that X_1 contains less than $2\sqrt{k}$ subcolor points is smaller than $p_1 = \exp(-\frac{\sqrt{k}}{2}) \leq \exp(-2) < 1/6$.

Suppose that X_1 contains at least $2\sqrt{k}$ subcolor points. Since all the subcolor sets are of equal measure, the probability of containing a subcolor point in X_1, given that event, is equal for all subcolor sets. Let $\widetilde{k}$ be the number of subcolor sets. Note that the subcolor sets cover more than half of the measure of the $Ł_i$'s, and therefore

$$\widetilde{k} > \sum_{i\in\mathcal{A}_{bad}} \frac{\mu(Ł_i)/2}{\epsilon^2/64k} = \frac{32k}{\epsilon^2} \sum_{i\in\mathcal{A}_{bad}} \frac{\epsilon\mu_i(V)}{4} \geq k,$$

where the last inequality follows from Lemma 6.3.

Now, given that X_1 contains at least $2\sqrt{k}$ subcolor points, the probability that these samples come from less than $\sqrt{k}$ subcolor sets is at most

$$p_2 = \binom{\widetilde{k}}{\sqrt{k}} \left(\frac{\sqrt{k}}{\widetilde{k}}\right)^{2\sqrt{k}} \leq \left(\frac{e\widetilde{k}}{\sqrt{k}}\right)^{\sqrt{k}} \left(\frac{\sqrt{k}}{\widetilde{k}}\right)^{2\sqrt{k}} = \left(\frac{e\sqrt{k}}{\widetilde{k}}\right)^{\sqrt{k}}.$$

As $\widetilde{k} > k$, we obtain that $p_2 < \left(\frac{e}{\sqrt{k}}\right)^{\sqrt{k}}$, which can be shown to be smaller than 1/4 for any $k \geq 16$.

Suppose now that X_1 contains samples from at least $\sqrt{k}$ subcolor sets. Let $\mathcal{C}$ be the set of colors $i \in \mathcal{A}_{bad}$ for which X_1 contains a sample in $Ł_i$. Since the measure of every subcolor set is $\frac{\epsilon^2}{64k}$, we have $\Sigma_{i\in\mathcal{C}}\mu_i(Ł_i) \geq \frac{\epsilon^2}{64\sqrt{k}}$. By definition, it follows that $\Sigma_{i\in\mathcal{C}}\mu_i(\mathcal{R}_i) \geq \frac{\epsilon^2}{64\sqrt{k}}$. Hence, the probability

that X_2 does not contain any i-point for some $i \in \mathcal{C}$ is at most

$$p_3 = \left(1 - \frac{\epsilon^2}{64\sqrt{k}}\right)^{\frac{1024\sqrt{k}}{\epsilon^2}} < \exp\left(-\frac{\epsilon^2}{64\sqrt{k}}\frac{1024\sqrt{k}}{\epsilon^2}\right) = \exp\left(-\frac{1024}{64}\right) < 10^{-6}.$$

Summing the probabilities p_1, p_2, and p_3, we obtain that the probability of not sampling i-points $x \in Ł_i$ and $y \in \mathcal{R}_i$ for any bad abundant color is smaller then $\frac{1}{4}$. □

Proof of Theorem 6.2. Assume that the input coloring c is ϵ-far from being convex. According to Lemma 6.4, with probability greater than $\frac{3}{4}$, there exists a bad abundant color i such that two i-vertices are queried in Step 1, belonging to two different i-sides. Suppose that there exists such a bad abundant color i. Fix two i-vertices which were queried from two different i-sides in Step 1, and call the interval between them *the special interval*. Clearly, if a non-i-vertex in the special interval is queried in Step 1, then the algorithm rejects. We hence assume that only i-vertices, if any, are queried in the special interval in the Step 1. Note that the special interval contains i's middle, and thus its non-i-weight is at least $\epsilon\mu_i(V)/4$ (as i is a bad color). Obviously, the i-weight of the special interval is at most $\mu_i(V)$. Therefore, the relative weight of non-i vertices in the special interval is at least $\epsilon/(4(1+\epsilon/4)) \geq \epsilon/5$. It is easy to see that even if we have queried additional i-vertices in the special interval in Step 1, then there still exists a pair of consecutive i-vertices in the sample such that the relative weight of non-i-vertices between them is at least $\epsilon/5$. Since in Step 2 we query $z \geq \frac{5}{\epsilon}\ln 12$ vertices between every pair of consecutive vertices, the probability of not discovering a non-i-vertex in such an interval is at most $1/12$.

Combining all the above, we have that an ϵ-far coloring is accepted with probability smaller than $1/4 + 1/12 = 1/3$. □

Acknowledgements

We thank Sagi Snir for introducing us to the topic of convex colorings.
We thank Ronitt Rubinfeld and the anonymous referees for their helpful comments.

References

[1] N. Alon, M. Krivelevich, Ilan Newman and M. Szegedy, Regular languages are testable with a constant number of queries, *Siam Journal on Computing* 30(6):1842–1862, 2001.

[2] M. Blum, M. Luby and R. Rubinfeld, Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences* 47:549–595, 1993 (a preliminary version appeared in *Proceedings of the* 22^{nd} *STOC*, 1990).

[3] S. Chakraborty, E. Fischer, O. Lachish, A. Matsliah and I. Newman: Testing *st*-Connectivity, *Proceedings of the* 11^{th} *RANDOM and the* 10^{th} *APPROX (2007)*: 380–394.

[4] E. Fischer, The art of uninformed decisions: A primer to property testing, *Bulletin of the European Association for Theoretical Computer Science*, 75:97–126, Section 8, 2001. Also in *Current Trends in Theoretical Computer Science: The Challenge of the New Century* (G. Paun, G. Rozenberg and A. Salomaa eds.), Vol. I 229-264, World Scientific Publishing, 2004.

[5] E. Fischer, O. Lachish, A. Matsliah, I. Newman and O. Yahalom, On the query complexity of testing orientations for being Eulerian, *Proceedings of the* 11^{th} *APPROX and* 12^{th} *RANDOM*, LNCS 5171: 402–415, Springer-Verlag, 2008.

[6] E. Fischer, E. Lehman, I. Newman, S. Raskhodnikova, R. Rubinfeld and A. Samorodnitsky, Monotonicity testing over general poset domains, *Proceedings of the 34th STOC*, pages 474–483, 2002.

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

[8] O. Goldreich and D. Ron, Property testing in bounded degree graphs, *Algorithmica*, 32, 302–343, 2002.

[9] S. Halevy and E. Kushilevitz, Distribution-free connectivity testing, In *Proceedings of the* 8^{th} *RANDOM and the* 7^{th} *APPROX*: 393–404, 2004.

[10] S. Halevy and E. Kushilevitz, Distribution-free property testing, In *Proceedings of the* 7^{th} *RANDOM and the* 6^{th} *APPROX*: 302–317, 2003.

[11] S. Halevy, O. Lachish, I. Newman and D. Tsur, Testing orientation properties, technical report, *Electronic Colloquium on Computational Complexity (ECCC)*, Report No. 153, 2005.

[12] S. Halevy, O. Lachish, I. Newman and D. Tsur, Testing properties of constraint-graphs, *Proceedings of the* 22^{nd} *IEEE Annual Conference on Computational Complexity (CCC)*: 264–277, 2007.

[13] D. Harel and R. E. Tarjan, Fast algorithms for finding nearest common ancestor, *SIAM Journal on Computing*, 13(2):338–355, 1984.

[14] D. E. Knuth, *The Art of Computer Programming*, Vol. 1: Fundamental Algorithms, Addison-Wesley, 1968. Second edition, 1973.

[15] S. Moran and S. Snir, Convex recolorings of phylogenetic trees: definitions, hardness results and algorithms, *Workshop on Algorithms and Data Structures (WADS)*:218–232, 2005.

[16] B.M.E. Moret and T. Warnow, Reconstructing optimal phylogenetic trees: A challenge in experimental algorithmics, In: *Experimental Algorithmics, Lecture Notes in Computer Science 2547*, 163–180 Springer Verlag, 2002.

[17] L. Nakhleh, T. Warnow, D. Ringe, and S.N. Evans, A comparison of phylogenetic reconstruction methods on an IE dataset, *Transactions of the Philological Society*, 3(2): 171–192, 2005.

[18] I. Newman, Testing of Function that have small width Branching Programs, *SIAM Journal on Computing* 31(5):1557–1570, 2002 (a preliminary version appeared in *Proceedings of the* 41^{st} *FOCS*, 2000).

[19] M. Parnas and D. Ron, Testing the diameter of graphs, *Random Structures and Algorithms*, 20(2):165–183, 2002.

[20] D. Ron, Property testing (a tutorial), In: *Handbook of Randomized Computing* (S. Rajasekaran, P. M. Pardalos, J. H. Reif and J. D. P. Rolim eds), Vol. II, 597–649, Kluwer Academic Publishers, 2001.

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

[22] C. Semple and M. Steel, *Phylogenetics*, Oxford University Press, 2003.

[23] B. Schieber and U. Vishkin, On finding lowest common ancestors: Simplifications and parallelization, *SIAM Journal on Computing*, 17:1253–1262, 1988.

[24] A. C. Yao, Probabilistic computation, towards a unified measure of complexity, In *Proceedings of the* 18^{th} *IEEE FOCS*: 222–227, 1977.